Use rich semantics in your markup to make it machine-readable as well as human-readable

Make your websites “mashup-ready” for a new generation of web applications

Understand this fast-growing technology through real-world examples, case studies, tools, and much more

MICROFORMATS

Empowering Your Markup for Web 2.0

JOHN ALLSOPP

friendsof™
DESIGNER TO DESIGNER™
an Apress® company

Microformats: Empowering Your Markup for Web 2.0

John Allsopp

friendsof™
DESIGNER TO DESIGNER™
an Apress® company

Microformats: Empowering Your Markup for Web 2.0

ISBN-13 (pbk): 978-1-59059814-6

ISBN-10 (pbk): 1-59059-814-8

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Credits

Lead Editor:
Chris Mills

Technical Reviewer:
Brian Suda

Editorial Board:
Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Jeff Pepper, Paul Sarknas, Dominic Shakeshaft, Jim Sumser, Matt Wade

Project Manager:
Beth Christmas

Copy Edit Manager:
Nicole Flores

Assistant Production Director:
Kari Brooks-Copony

Production Editor:
Laura Cheu

Compositor:
Lynn L'Heureux

Artist:
April Milne

Proofreader:
Nancy Sixsmith

Indexer:
Toma Mulligan

Interior and Cover Designer:
Kurt Krames

Manufacturing Director:
Tom Debolski

For Sara and ZK.

CONTENTS AT A GLANCE

CONTENTS

ABOUT THE AUTHOR

John Allsopp (www.johnfallsopp.com) is a software developer and long-standing web development speaker, writer, evangelist, and self-proclaimed expert. Since 1993 he has been working with and developing for the Web. He is the head developer of the leading cross-platform CSS development tool Style Master, developer and publisher of Westciv's (www.westciv.com) renowned training courses and learning resources on CSS and standards-based development, and author of the highly regarded article, "A Dao of Web Design" (as well as dozens of other articles). He is also one of the founders of the Web Directions conference series (www.webdirections.org) and a very early member of the Web Standards Project.

When not bathed in the glow of various computer screens, John is a volunteer surf lifesaver at Bondi Beach, one of Australia's most famous beaches. A longtime Bondi resident, he now lives at the southern edge of Sydney, between the ocean and the world's second oldest national park, with his wife and young daughter.

ABOUT THE TECHNICAL REVIEWER

Brian Suda is an informatician currently residing in Reykjavík, Iceland. He has spent a good portion of each day connected to the Internet after discovering it back in 1996. He has a bachelor's degree in computer science from Saint Louis University and a master's degree in informatics from the University of Edinburgh, Scotland. One of his many projects has been to work closely with microformats.org, writing code, gardening the wiki, and hammering out FAQs and specifications. As an invited expert for the W3C's GRDDL specification, he helped to extract RDF data for the Semantic Web through the use of microformats.

Suda's own little patch of Internet can be found at `http://suda.co.uk`, where many of his past projects, publications, interviews, and crazy ideas can be found.

INTRODUCTION

Microformats, an evolutionary approach to adding richer semantics to HTML-based markup, have rapidly been gaining adoption among web publishers and service developers, large and small, over the last two or three years.

Unlike almost every other approach to adding more sophisticated semantics to the Web, such as XML, RDF, and the Semantic Web, microformats don't require developers to learn whole new technologies, throw away their existing code bases, or wait years for browser developers to catch up and actually implement support for the technology. Microformats simply use features of HTML that have been around for years and are familiar to most web developers, though sometimes in original and subtle ways. If you hand-code HTML or XHTML, and you know about the `class` attribute, then that's all you'll need to get started. (In fact, you don't even need to be familiar with the HTML `class` attribute.) Aside from this, no particular knowledge is assumed; ideas like "semantic HTML," where important, are introduced and covered in detail in this book.

Like the Web, this book is platform-agnostic. Whether you work on the Mac, Windows, Linux, or any other operating system, the examples and concepts presented will be relevant and correct. All the HTML and CSS covered here are completely standard and based on the World Wide Web Consortium's (W3C's) various "recommendations." Where relevant, some browser-specific information is provided—for example, where particular CSS features are or are not supported by given browsers. Specific content management systems (CMSs) and blogging tools, such as WordPress, Textpattern, and Drupal, are discussed, but typically briefly and in the context of their support for publishing microformatted content more easily. For coding examples you might wish to work through, any HTML, text, or (where relevant) CSS editor will work fine. (Although you might like to try out Style Master for CSS, as I develop it —this is the last reference to Style Master, I promise!)

The only thing you may need to obtain is a copy of Firefox 2. Some very interesting tools you'll look at throughout the book are extensions to Firefox. This extensions mechanism, which allows anyone with JavaScript skills to add functionality to Firefox, makes it a kind of test bed for experimenting with new browser functionality. You can get a copy of Firefox from `www.mozilla.com`.

The book has its own site, which is a section of my microformats-focused blog, Microformatique. The book's section, where you can find all the code examples for downloading, additional downloads, and more, is here: `http://microformatique.com/book`.

Thanks for checking out the book—I hope it delivers what you want it to.

Whenever I speak or publish, I try to make myself as available to people as possible. Should you wish to get in touch with me regarding anything in the book or related subjects, simply email me at `john@westciv.com`.

Layout conventions

To keep this book as clear and easy to follow as possible, the following text conventions are used throughout.

Important words or concepts are normally highlighted on the first appearance in **bold type**.

Code is presented in `fixed-width font`.

New or changed code is normally presented in **`bold fixed-width font`**.

Pseudo-code and variable input are written in *`italic fixed-width font`*.

Menu commands are written in the form Menu ➤ Submenu ➤ Submenu.

Where I want to draw your attention to something, I've highlighted it like this:

Ahem, don't say I didn't warn you.

Sometimes code won't fit on a single line in a book. Where this happens, I use an arrow like this: ➥.

```
This is a very, very long section of code that should be written all ➥
on the same line without a break.
```

PART ONE **INTRODUCING MICROFORMATS**

In this first part of the book, we'll examine what microformats are, their origins, the motivation for their development, the principles underpinning their development, and why you should be starting to use them in your web development projects. We'll also survey the current state of the art, and look at which publishers, services, and tools currently use or support microformats. But first, let's begin at the beginning with the simple question, What exactly are microformats?

1 WHAT ARE MICROFORMATS?

If you've been developing for the Web for any period of time, you'll have seen new concepts and technologies come (and many of them go) thick and fast. Some, like Cascading Style Sheets (CSS), stick and change the way the Web works. Others, like Scalable Vector Graphics (SVG), may or may not have this same impact—only time will tell. Still others simply fade away, never living up to their initial promise.

The challenge for professional developers is to work out which new technologies are worth the time and effort it takes to get up to speed with them, and which should be ignored. Keeping up with trends and developments is something professionals must do, all the while being careful not to waste valuable energy on technologies doomed to obsolescence.

One new technology in web development, **microformats**, represents a way of developing for the Web that will be both novel and familiar to most experienced web developers. Yes, microformats are a recent technology and, as such, investing a lot of time or effort in understanding them may seem something of a risk. But it's a risk many very experienced web publishers, software developers, and service developers are taking. When major companies like Yahoo and Microsoft, innovators like Technorati, and web pioneers and experts embrace a technology, you can rest assured something more than hype is involved.

By the end of this chapter, you'll understand what microformats are, what they do, and how they do it. You'll consider the principles they are built on and what motivated their development in the first place. We'll begin with the most important questions facing any new technology: Why should you bother learning to use it? What problems does it solve? After we explore these questions, I hope you'll then have a good idea about why it's worth continuing to the end of the book and, even better, you'll have a burning desire to do so.

Too much (disparate) information

At the time of this writing (late 2006), Google alone has indexed more than 20 billion items on the Web, with the vast majority of these items being HTML documents. To put this in perspective, if we were to print all these items out and stack them on top of one another, assuming just one standard-sized page per indexed item, we'd quickly get bored. It's an impressive amount of information, but what exactly can we do with it? Well, we can search to find items that match specific keywords, and then read them, just as we have been doing for well over a decade. But surely this pile of pages nearly 12,500 miles high—this extraordinary number of ideas, facts, opinions, and words—make up a collective body of knowledge that we can *Enquire Within Upon Everything*?

Let's put this issue in perspective by looking at the state of reviews on the Web today as an example, and the challenges of aggregating this type of information.

Reviews on the Web today

For instance, let's turn to the really pressing matters of our times. Does the movie *Superman Returns* suck? Do bloggers, on the whole, like the new U2 album? Where is a great pizza restaurant in downtown San Francisco near where I am staying for a couple of nights? Do people agree with Joe Clark's analysis of WCAG2? What do people think about the U.S. government's policy toward North Korea? There may be dozens, hundreds, thousands, or even millions of pages that have reviews of or opinions about any of the topics just mentioned. And yes, we can search to find these *individual* reviews. But how can we access the collective opinion of all of these sites? Traditionally, the only way to gauge the opinion of the masses (what James Surowiecki calls "the wisdom of crowds") has been to visit sites like Amazon (`www.amazon.com`), Zagat (`www.zagat.com`), or The Internet Movie Database (IMDb; `www.imdb.com`), which enable people to offer their opinions and aggregate them, or perhaps to read the comments section of a very popular blog to get an informal sense of what some people feel about issues like WCAG2 or North Korea.

But there are definite drawbacks to this centralized approach. On top of this method being highly inaccurate, sites like Amazon, Zagat, and IMDb typically require reviewers to hand over the rights to their opinions, and these sites may even retain the right to edit or delete comments for whatever reason. Users also typically need to create an account and hand over personal details, like their e-mail addresses, in order to comment. Many people are quite rightly reluctant to hand over such details and submit their opinions on those terms (or indeed on any terms), preferring to publish their thoughts on their own site or blog, a MySpace page, or elsewhere.

So, if there is so much collective wisdom (OK, at the very least, collective *opinion*) out there, why doesn't someone simply aggregate all these book reviews, restaurant reviews, and film reviews, and build a distributed equivalent of, say, Amazon, Zagat, or IMDb reviews? In fact, why aren't Amazon, Zagat, and IMDb doing it themselves?

The following section attempts to explain why it's not really happening. Yet.

The not-yet-semantic Web

The following are some examples of reviews you might find on the Web. If you looked at the code behind the following items, you might think they were very different animals, but underneath it all, they are all movie reviews of one kind or another:

- July 20 2006: I went to see the latest Woody Allen film today. It was pretty good. I'd give it a thumb and a half up :-)
- Title: *Pirates of the Caribbean 2*; Director: Gore Verbinski; Rating: 8 out of 10
- Man, the CGI in superman roxor, but the film suxor

Even if you don't speak l33t or know who Ebert and Roeper are, I bet you have a pretty good idea about what film each reviewer saw, and what he or she thought about it. But as you might guess, software finds it very difficult to read and understand even straightforward language and concepts, let alone idiomatic expressions and incomplete information.

To put this idea in other words, people are really good at gleaning meaning, or **semantics**, from written language, and software is really, really bad at it. To put it bluntly, people are smart and software is dumb.

Take just this one simple issue: ratings. Are three stars good or bad? If I give a film a rating of 10, does that mean the film is worth seeing or should be avoided? Now, any review contains similar information—what is being reviewed, whether the reviewer liked it or not, perhaps why the reviewer liked or disliked it, when the review was written, and so forth—but what is missing is some standardized way for us to mark up our web content to help *software* recognize a review when it encounters one, and help move the Web beyond the current index/search/read paradigm that we are largely currently stuck in.

In technical terms, we need a way of creating richer semantics in our web pages that doesn't break existing web content or current web browsers, and that builds on the set of skills developers already have, rather than requiring them to learn a whole new skill set.

Essentially, we aren't looking for something brand new; rather, we want a better way of doing the things web developers and content publishers already do all the time. Publishing reviews is just one example of this, and throughout the book you'll see a variety of other examples.

Enter microformats, an important, increasingly popular approach to solving the problems just mentioned.

Microformats overview

Microformats are more than simply a *technology* like CSS or XHTML—they are an approach to solving the important problem of creating rich semantic markup for today's Web. They include specific technologies (or **specifications**), which we'll devote most of our attention to throughout this book. But before we do that, let's take a look at the origin, definition, and principles of microformats. We'll then look at an example that illustrates the principles and ideas behind microformats.

Origin

The custodians of microformats are the people behind microformats.org, a community consisting of microformats pioneers, but freely open to participation of anyone with an interest in the subject. One of the founders and principal drivers of microformats is Tantek Çelik, about whose involvement in the Web at Apple, Microsoft, and now Technorati, as well as with the World Wide Web Consortium (W3C), a book could probably be written in itself. Other important early contributors are Kevin Marks and Ryan King, who also work at Technorati (an important contributor to and adopter of microformats technologies); well-known CSS and web development expert Eric Meyer; WordPress founder Matthew Mullenweg; and Brian Suda, who, in addition to being the coauthor of several microformats specifications, is the technical editor of this book.

Many others have made significant contributions to various microformats, or have made a difference by adopting microformats. You'll be introduced to a number of these people in this book, and hear why they adopted microformats and the benefits using microformats have brought them.

Legend has it that microformats originated at the very popular interactive technology conference South by Southwest (SxSW) in 2004, with what became the XHTML Friends Network (XFN), which we'll look at in detail in Chapter 5. At that time, as blogging was really beginning to take off, bloggers began annotating their blogrolls in various ways to indicate their relationships with the bloggers whose blogs they read. XFN was developed as a way to add these richer semantics in a more standardized way, and it spread rapidly through the technology blogosphere. It was the success of XFN that paved the way for further microformats innovation.

Definition

Microformats are, in the words of microformats.org,

Designed for humans first and machines second, microformats are a set of simple, open data formats built upon existing and widely adopted standards. Instead of throwing away what works today, microformats intend to solve simpler problems first by adapting to current behaviors and usage patterns (e.g., XHTML, blogging).

Or, more succinctly,

. . . a set of simple open data format standards that many are actively developing and implementing for more/better structured blogging and web microcontent publishing in general.

You'll delve into some microformats examples in great detail throughout the book, including the following:

- **hCard**: For marking up contact information for people or organizations
- **hCalendar**: For marking up information about events such as seminars, conferences, meetings, concerts, and parties
- **XFN**: For marking up the relationships between people (e.g., who has met whom, who is friends with whom, who is a colleague of whom)
- **hReview**: To help solve the reviews on the Web problem previously discussed

Principles

Underpinning every microformat is a set of principles that help to guide the outcome of developing new microformats. As you work through the practical aspects of microformats and specific microformats, you'll see these principles in action. Microformats

- Solve a specific problem
- Start as simply as possible
- Are designed for humans first, machines second
- Reuse building blocks from widely adopted standards
- Are modular and embeddable
- Enable and encourage decentralized development, content, and services

Let's take a look at a simple practical example to help bring this discussion together.

Microformats example

If someone's address or contact information is presented on a web page, we can usually work this out from just glancing at the page (or doing a quick search of it), because as we know, people are smart. But because computers aren't smart (sorry, laptop, you are many things, but sadly smart is not one of them), it's much harder for them to do this. One of the microformats principles from the previous section is *solve a specific problem*. The specific problem just identified is that we lack a common format for marking up addresses/contact information in web pages, something that must appear on millions of web pages, at least.

To solve this problem the microformats way, we want to *start as simply as possible*, and we want to *reuse building blocks from widely adopted standards*. Unfortunately, there doesn't happen to be a widely adopted standard for marking up addresses in X/HTML, so can we find one elsewhere?

Yes, you with your hand up at the back. HTML does have an `<address>` *element, so why not use that, I hear you ask. This is where we have to pay careful attention to the HTML specification. On close reading, we find that for whatever reason, the* `<address>` *element is for the contact information of the author of the page—so it doesn't fit the bill for a more general standard for marking up addresses.*

It may come as something of a surprise to those who work with the Web to know that the World Wide Web Consortium, or W3C, is not the only source of widely adopted standards for the Web and, in particular, the Internet. Another common source of standards is the Internet Engineering Task Force (IETF). If you've ever seen a reference to a Request for Comment or a specific RFC, then IETF is very likely its source. An RFC is in effect a standard. One commonly used IETF standard, which you've almost certainly used even if you didn't know it, is vCard (RFC 2426). A great many applications use or support the vCard format for storing contact details, among them Outlook, Apple's Address Book application, and Evolution, the popular Linux contact information management application.

Because it's widely adopted, vCard is an ideal candidate for solving our problem of finding a standard on which to base a microformat for addresses. vCard allows us to mark up all kinds of contact information beyond simply an address, so if we want to *start as simply as possible*, we find there is a subset of vCard specifically for addresses called adr. Adr allows us to mark up the following address information:

- `post-office-box`
- `extended-address`
- `street-address`
- `locality`
- `region`
- `postal-code`
- `country-name`

The adr subset would appear to be an ideal solution to our problem of how to best mark up addresses on the Web.

Let's recap the process. We identified a problem: the need to mark up address information semantically. We identified that HTML itself does not have the appropriate elements or attributes to do this, so we looked elsewhere for existing formats and schemas to help us solve the problem. We recognize that the IETF standard vCard (in fact, a subset of vCard: adr) provides exactly the format we need, and now we want to adopt it for HTML to fully solve our problem.

So, as web developers, how can we use adr? We examine how adr and many other microformats work later in the book, including how and why we translate the "properties" of vCard into particular HTML constructs (elements like `<div>` and `<span>`, and attributes like `class`, `title`, etc.), but for now, let's take a look at what our finished adr microformat might look like in code:

```
<div class="adr">
  <abbr class="type" title="dom">U.S.</abbr>
  <span class="type">home</span> address, for
  <abbr class="type" title="postal">mail</abbr> and
  <abbr class="type" title="parcel">shipments</abbr>:
  <div class="street-address">123 Main Street</div>
  <span class="locality">Any Town</span>,
  <span class="region">CA</span>
  <span class="postal-code">91921-1234</span>
</div>
```

You can see a couple of other microformats principles in action here. As clean semantic HTML, it's *designed for humans* [well, at least web developers] *first, machines second*. Compare the preceding HTML with the following vCard equivalent:

```
ADR;TYPE=dom,home,postal,parcel:;;123
Main Street;Any Town;CA;91921-1234;
```

That's a little more compact, but it's clearly designed for machines first.

This is where the microformat really wins out: the code is far more human-readable. If you render it in a browser, even with no style sheet, you have a fully readable address, as shown in Figure 1-1.

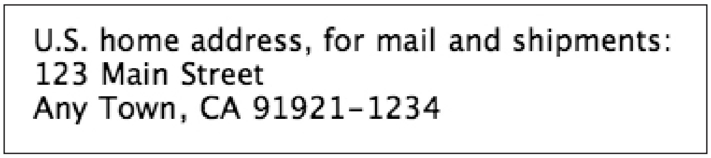

U.S. home address, for mail and shipments:
123 Main Street
Any Town, CA 91921-1234

Figure 1-1. Microformats are easily human-readable when rendered in a browser.

You can also see that this HTML is both modular and embeddable. You could validly and easily put this code on *any web page*, and as you'll see a little later in the book, the adr microformat is part of the more complex hCard microformat, and it could, of course, be used with other microformats where addresses might be needed (e.g., in restaurant reviews).

We'll now move on to look at the last of the microformats principles, *enable and encourage decentralized development, content, and services*, because that's where I think microformats get really exciting. Let's consider how adr might do this.

Imagine trying to build a directory of cinemas, bookstores, butchers—indeed, any directory of businesses—by searching the Web for address information. Software recognition of addresses is a surprisingly hard problem to solve, but if addresses were marked up using adr, it would be much, much easier. If you throw in telephone numbers (and, as you'll see a little later, the vCard microformat has telephone numbers as well as adr-based address information), you can start imagining a distributed Yellow Pages or White Pages, where you publish and update your own details at your own site, and aggregators routinely check pages for changes, and then update their indexes of addresses and other contact details. No more centralized system required for registering and updating information.

And it's just a little step from there to the ideas discussed at the beginning of the chapter: distributed film, book, and restaurant reviews; distributed bookmarking services similar to del.icio.us; and even decentralized classified ads. In fact, some of these products and services are already in development with the help of microformats, and we'll discuss them later in the book.

The core microformats principles of solving real, existing problems; starting simply; building on existing standards; being embeddable and modular; and enabling decentralized development, content, and services recur throughout the book. If you grasp these principles—the underlying philosophies of microformats—then the technology itself will make a lot of sense to you, and the various ways you can use microformats will jump out at you. You'll start seeing opportunities for working with microformats everywhere.

Benefits of microformats

You might be thinking, "I'm a web developer, and I get by fine without microformats. Why should I bother learning something new? I've only just gotten the hang of CSS, I need to get up to speed with Ajax, and my manager says we must be Web 2.0–compliant as soon as possible. I'm too busy as it is!"

If you go back a few years, you would have heard exactly the same thing said about CSS (trust me, I did). But it is important to remember that the Web is a relatively new technology and medium, and it is still evolving. Fortunately, microformats are deliberately *evolutionary*, unlike the radically new Semantic Web or even CSS, which require new tools, new browsers, and new developer skills. Microformats build on the existing skills of web developers, because they are just valid, semantic HTML or XHTML. They work with existing tools, and they work in existing browsers. They are just a more structured, more semantic way of using HTML. And structured, semantic HTML is definitely a trend that has been growing for the last few years. Increasingly, developers have been using lists for navigation and other places where lists are appropriate, using heading elements for the headings in their content, and using CSS rather than presentational HTML such as font elements or tables for layouts. Microformats just take this trend forward another step.

As a bottom-line value proposition, here's what you can get out of using microformats right now, in practical terms:

- By providing a standardized way of marking up content that developers are already marking up in many different ways, microformats help us code more efficiently and it becomes easier to maintain code.
- By making commonly published data available in standard formats, microformats help enable distributed software services such as aggregation and indexing, which would otherwise be extremely difficult or unfeasible to implement.
- By using existing schemas where possible, microformats enable seamless interoperability between web-based content and desktop applications, such as Outlook and iCal for calendaring information, or Evolution, Address Book, and many others for vCard.

But perhaps the best way to convince yourself of the value of learning to use microformats is to consider what people are saying and, more important, doing with microformats already.

For example, in March 2006, during the keynote at Microsoft MIX06, none other than Bill Gates said, "We need microformats and to get people to agree on them. It is going to bootstrap exchanging data on the Web . . . we need them for things like contact cards, events, directions."

You can watch the keynote just mentioned if you like, via the magic of YouTube: `www.youtube.com/watch?v=Z9X-vHJ_Z-I`.

Whatever you might think about William H. Gates III, there is little doubt that if something crosses his radar, and he takes enough notice to mention it explicitly at such a high-profile event, then it is worth investigating further.

Staying with Microsoft for just a moment, Ray Ozzie, legendary developer of Lotus Notes and now Chief Technology Officer at Microsoft, wrote the following in his blog (`http://rayozzie.spaces.live.com/Blog/cns!FB3017FBB9B2E142!377.entry`): "Microformats are ideal for representing structured data in HTML; we use them extensively in our demos."

To have such significant, influential members of our industry recognizing and indeed using microformats should underscore just how much potential they have to become very important very quickly.

Summary

This chapter introduced the core concepts, principles, and philosophies behind microformats. You saw a sampling of the kinds of practical, current problems they are designed to solve, and how they set about solving these problems without breaking existing browsers, tools, and markup, and without developers needing to learn whole new technologies, as required with CSS and XML.

In the next chapter, you'll take a look at how microformats are being used today by some of the biggest names on the Web. After that, you'll be ready to jump in and start working with microformats.

2 THE STATE OF THE ART IN MICROFORMATS

New technologies often face a "chicken or egg" moment. No matter how much promise a technology has, it must be adopted and word of mouth about it must spread, or it will simply fade away, much like "push" technologies and Smell-O-Vision.

Well, maybe Smell-O-Vision wasn't such a promising idea, but it was indeed an actual, used invention—go look it up (`http://en.wikipedia.org/wiki/Smell-o-vision`*).*

Let's take the example of the fax machine. By itself, a fax machine is pretty useless. You need a "network effect" of other adopters. The more fax machines there are, the more useful every single one becomes.

A more topical example is Microsoft's new Zune (currently on sale for less than a week as I write this). The feature Microsoft hopes will offer a killer alternative to the iPod is its wifi-based "squirting," which allows you to transfer a song to another Zune user. One challenge Zune faces is that to start with, there aren't that many Zunes out there, making the feature less useful than if Zunes were as ubiquitous as mobile phones. But Microsoft's strategic decision to allow a song to be transferred from one Zune to another only once (this second Zune cannot then transfer the song on) bafflingly hobbles the network effect required to make the feature's use reach a tipping point and become a must-have social technology.

We forget that the Web was once like this, too. The early 1990s saw a great deal more Gopher content, newsgroup content, mailing list content, and content available in "walled garden" networks like CompuServe and AOL, not to mention thousands of dial-up bulletin boards, than was then on the Web. Developing web-based content was consequently a risk—maybe it would be better to develop content to be delivered by one of these other channels. Developing browsers, publishing tools, search engines, and other web-based services was also a risk—maybe all the content would end up elsewhere, and the effort developing these services be wasted, and opportunities would be lost developing for other services.

Web pioneers like David Filo and Jerry Yang (who started Yahoo as simply a list of sites they were interested in, kind of like del.icio.us with just two users), or Marc Andreesen (who was, along with Eric Bina, the developer of Mosaic, and whose incorporation of inline images into a web browser is widely credited with the explosion of interest in the Web, and who went on to found Netscape Communications with Bina) are some of those early adopters, who benefited rather significantly from taking the risk on the Web.

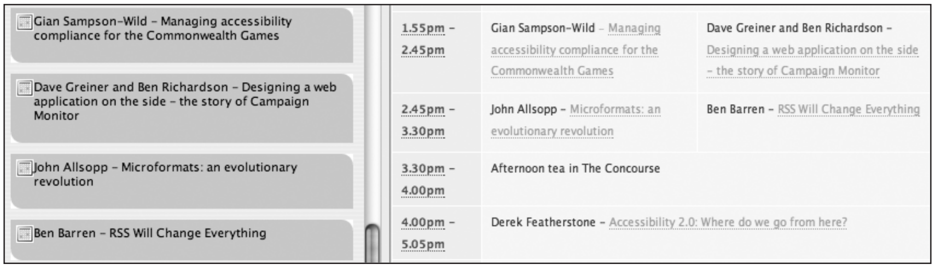
In the case of microformats, as a content developer you might fairly ask, "If there are no services that take advantage of microformatted data, why should I use microformats?" Service and software developers might similarly ask, "If there is no microformatted data, why should I develop services in the hope that there might one day be that data available?"

This chapter aims to demonstrate that these fears are no longer as well founded as they might have been even a year ago, when you might have argued that microformats were a great idea, yet to be proven. Today, all manner of publishers, publishing and content development tools, and online services are supporting microformats, from big, well-known names like Yahoo to innovative startups like Cork'd. To get some inspiration before we venture into the nitty-gritty of the technology, let's take a look at some of what's going on in the microformats world.

The future of browsers

Imagine if publishers could mark up contact details in a way that browsers knew they were there and could display them to users. What if when you visited a television program listing site or a movie listing site, your browser could find all the program times or showing times, and present them to you? What if you could simply subscribe to these listings with a single click?

Right now, if the page is marked up using the hCard microformat for contact detail, and you use Firefox with an extension called Tails Export installed, you can call up a list of the contacts on that page or the list of events published in the hCalendar microformat, as shown in Figure 2-1.



Gian Sampson-Wild – Managing accessibility compliance for the Commonwealth Games

Dave Greiner and Ben Richardson – Designing a web application on the side – the story of Campaign Monitor

John Allsopp – Microformats: an evolutionary revolution

Ben Barren – RSS Will Change Everything

1.55pm – 2.45pm	Gian Sampson-Wild – Managing accessibility compliance for the Commonwealth Games	Dave Greiner and Ben Richardson – Designing a web application on the side – the story of Campaign Monitor
2.45pm – 3.30pm	John Allsopp – Microformats: an evolutionary revolution	Ben Barren – RSS Will Change Everything
3.30pm – 4.00pm	Afternoon tea in The Concourse	
4.00pm – 5.05pm	Derek Featherstone – Accessibility 2.0: Where do we go from here?	

Figure 2-1. Calling up a list of hCard contacts on a web page using the Tails Export Firefox extension

The left pane shows all of the events on the page, and on the right is the page itself. It's a great example of how browsers will very likely make the microformatted metadata in a page visible, should users wish to see it. We'll take a deeper look at Tails Export and the closely related Tails extension for Firefox in Chapters 7 and 8.

But it's not just about viewing the data. Microformats make web-based data much more interoperable with other applications. If you use the Mac OS X iCal application, or the open source Sunbird calendaring application from the Mozilla foundation (or indeed any software that supports the iCalendar format), and a page has its event details marked up using the hCalendar microformat, then using either X2V or Technorati Events Feed Service, both of which you'll explore in detail in Chapters 7 and 8, you can subscribe to any web page of events, and iCal or Sunbird will remain synchronized if the events on the page change or are updated.

While as yet these aren't mainstream use cases, indications are that upcoming versions of widely used browsers such as Internet Explorer and Firefox 3 will include built-in support for microformats.

Internet Explorer platform architect (and long-time Internet Explorer, and prior to that Mosaic developer) Chris Wilson has, along with others at Microsoft (all the way to the top: Ray Ozzie and Bill Gates himself), shown considerable interest in microformats. Speculation is strong that future versions of Internet Explorer will have native support for viewing microformats. In a similar vein, Alex Faaborg, user experience designer on Firefox 3, has been considering how structured data and specifically microformatted content might be supported directly inside browsers. Faaborg concludes, "it is time for Web browsers to provide the user with a clean, consistent, and simplistic user interface"[1] for structured content generally, and specifically microformats.

And released literally as this chapter was being finalized is Operator by IBM's Michael Kaply. Another Firefox extension (the ability to extend Firefox in this way really does make it a test bed for future browser functionality), Operator gives users a toolbar of actions that they can easily perform on the microformatted content in a page, as shown in Figure 2-2.

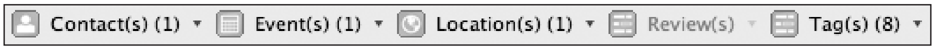


Figure 2-2. The Operator Firefox extension in action

For example, this is the toolbar generated when a user visits a page with one hCard (the microformat for contact details), one hCalendar (for events), one location marked up with the GEO microformat, and eight tags (marked up using rel-tag). Operator presents the user with a number of options for each kind of content. Users can add contacts to their desktop address book application or online address books such as Yahoo contacts. I predict that the Operator extension is a lot like what future native support for microformats will look like in browsers.

1. See http://blog.mozilla.com/faaborg/author/afaaborg@mozilla.com/feed.

Tools to help publish microformatted content

As you'll see throughout the book, microformats are usually very easy to hand-code using nothing more than a text editor. That said, there is already a high degree of support for developing and publishing microformatted content in new and existing development and publishing tools, as outlined in the sections that follow.

Online creators

The Microformats community has developed a number of creators for hCard, hCalendar, and hReview. These provide a simple way to start publishing microformatted content and are available from the following URLs:

- `http://microformats.org/code/hcard/creator`
- `http://microformats.org/code/hcalendar/creator`
- `http://microformats.org/code/hreview/creator`

Dreamweaver

The Web Standards Project, via Drew McLellans's Dreamweaver Microformats toolbar, makes developing with microformats as easy as filling in a form for Dreamweaver users. Visit `www.webstandards.org/action/dwtf/microformats` for more information.

Blogging tools and content management systems

Because of the open way in which many blogging tools and content management systems (CMSs) are developed, and their extensible nature via plug-ins, there is significant support for microformats publishing with systems such as WordPress (which even has built-in support for some microformats) and Texpattern, while there is strong interest from other similar projects, like Drupal, in supporting microformat publishing. The Structured Blogging project (`http://structuredblogging.org`) has plug-ins for both WordPress and Movable Type to help publish microformatted content such as reviews, contact details, and events.

This book covers solutions for specific platforms and particular microformats, when individual microformats are covered in detail.

Publishers using microformats

Perhaps the most exciting and vibrant aspect of microformats right now is the variety and number of publishers using them to publish all kinds of information. This section presents just a few of those publishers and examples of their microformats usage.

Yahoo

Let's start with Yahoo, one of the most widely used, widely recognized sites on the Web. Over the last couple of years, Yahoo has been demonstrating considerable innovation and community-mindedness when it comes to its development strategies. Yahoo Developer Network (`http://developer.yahoo.com`) is the focal point for Yahoo's interaction with web developers, and it includes the Yahoo Design Pattern Library (`http://developer.yahoo.com/ypatterns`), an open and growing set of design patterns for web user interfaces. While these design patterns are not directly related to microformats, the concepts are in many ways similar.

One measure of Yahoo's innovative spirit is their broad adoption of microformats. Later in the book, we'll devote some time to an in-depth look at some of the company's efforts, but for now, let's quickly overview how microformats are in use across the family of Yahoo sites.

Two of Yahoo's review-focused sites, Yahoo Local (`http://local.yahoo.com`) and Yahoo Tech (`http://tech.yahoo.com`), use the hReview format for publishing reviews. They also use the hCard format for reviewer details and, in the case of Yahoo Local, for contact details for the businesses being reviewed.

The increasingly popular Flickr photo sharing service (`http://flickr.com`), which is a Yahoo company, uses microformats in a couple of ways. On its profile page for a user, details are published using hCard. Flickr also recently added **geotagging**—that is, the ability to say where a photo was taken using longitude and latitude. This information (and there are currently millions of photos that have been geotagged in this way) is published using the GEO microformat.

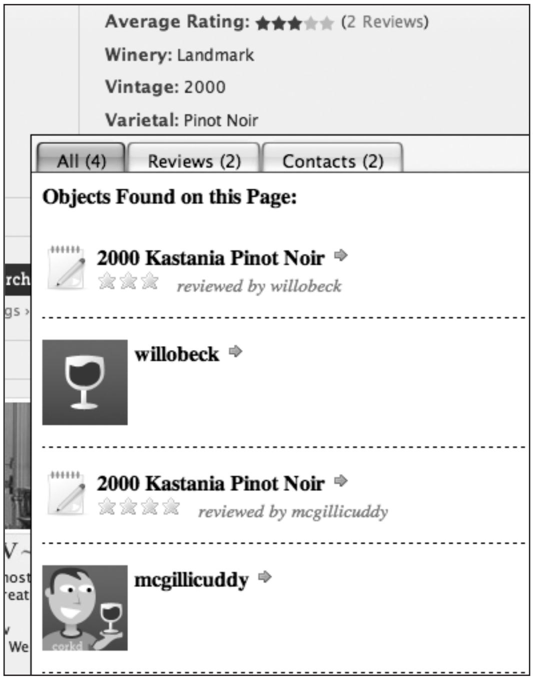
Recent Yahoo acquisition Upcoming (`http://upcoming.org`), a site for publishing and tracking events, uses the hCalendar format for publishing event details. As you'll see later in this book, use of this microformat makes it possible to easily subscribe to all the events in an area or all events that have been given a specific tag by other Upcoming users, such as "design," using iCal or Sunbird.

We'll revisit these Yahoo sites in Chapter 12, where we focus on how Yahoo uses microformats on quite a number of its sites.

Cork'd

Yahoo isn't the only one using microformats—they've been adopted by a range of innovative publishers and developers. Cork'd (`http://corkd.com`), brainchild of noted web designer and developer Dan Cederholm and Hivelogic's Dan Benjamin, is a site for wine lovers (from amateur to expert) to review wine and for anyone to get wine recommendations and reviews. We'll revisit Cork'd in depth later, as it's a great example of how small teams and startup sites can benefit from microformats as much as the big guys. Cork'd uses the hReview format for reviews, hCard for marking up reviewers names, and XFN for marking up the relationships between reviewers (you can find and mark up members of Cork'd who are your friends, colleagues, etc.).

Figure 2-3 shows the X-ray power of Tails in action on Cork'd—lurking beneath the lovely design is well-structured, semantic, microformatted content.



Average Rating: ★★★☆☆ (2 Reviews)
Winery: Landmark
Vintage: 2000
Varietal: Pinot Noir

Objects Found on this Page:

2000 Kastania Pinot Noir ➪
reviewed by willobeck

willobeck ➪

2000 Kastania Pinot Noir ➪
reviewed by mcgillicuddy

mcgillicuddy ➪

Figure 2-3. Using the Tails extension to reveal microformatted content on Cork'd

Eventful

Eventful (`http://eventful.com`) is in many ways similar to Upcoming: it's a way of broadcasting events you are organizing, or keeping track of and discovering events that are coming up. Eventful uses hCalendar for events (not surprisingly), hCard for event addresses, and geotagging for event locations.

Online magazines

Digital Web magazine (`www.digital-web.com`), the venerable online publication for web designers and developers, publishes a calendar of upcoming web-related events in hCalendar. *Vitamin* magazine (`www.thinkvitamin.com`), another design- and development-focused online publication, as well as the UK division of Ziff Davis technology publications (`www.zdnet.co.uk`) use hCard for contact details.

Apple

Often—in fact, typically—microformats are used for public-facing information, but that's not necessarily how they have to be used. Apple's .Mac online mail application (`www.apple.com/dotmac`) is one use of microformats you'll see only if you have an account with them. On top of being chock-full with Ajaxie goodness, users' address book details are in fact marked up using hCard.

edgeio

One last example we'll take a look at for now is in some ways quite different from the others we've seen, and it embodies one of the key promises of microformats: enabling "decentralized development, content, services." edgeio (`www.edgeio.com`), a recent entrant in the online classifieds world, aggregates and publishes classified listings from all over the Web. edgeio uses the hListing draft microformat to publish, and it also aggregates listings posted in hListing format. For example, if you post a listing in this format on your blog, and then let edgeio know about it using a form on its front page, they'll aggregate and publish your listing.

Services using microformats

Over time, the distinction between publishers and services will in many cases become hazy. After all, is edgeio a publisher or a service? For that matter, is a search engine a publisher or service? Depending on how you use a search engine, it can be both (a bit like light can be both a wave and a particle, depending how you look at it. Well, nothing like that except by analogy). For now, though, let's keep the distinction, and look at some of the services using microformats in the sections that follow.

Technorati

A name you'll encounter often in this book, Technorati (`http://technorati.com`), whose employees Tantek Çelik, Ryan King, and Kevin Marks all play significant roles in the microformats community, has several services associated with microformats.

The most widely used Technorati service is no doubt its tag search, which uses author-defined tags (using the rel-tag microformat) instead of keywords to find content on the Web.

The Technorati Events Feed Service (`http://technorati.com/events`) allows you to download or subscribe to online events marked up using the hCalendar format. Similarly, the Technorati Contacts Feed Service (`http://technorati.com/contacts`) allows users to grab contact details marked up using hCard on any page, and add them to any address book application that supports the (almost universally supported) vCard format.

At the Technorati Kitchen (`http://kitchen.technorati.com/search`), you can find a search engine specifically for microformatted content. With this tool, users can search for contact details, events, or reviews marked up using the appropriate microformat.

Pingerati

You've seen a number of times that a significant goal of microformats is to enable "decentralized development, content, services." Aggregating and indexing is difficult and expensive, and yet it's absolutely vital for decentralized services. One service that provides big shortcuts to this process is Pingerati (a second such service, Alexa, is discussed next).

Websites and applications can "ping" (i.e., inform) the Pingerati service (`http://pingerati.net`) about new or changed content containing microformats. Some of the publishers discussed previously, like edgeio, Eventful, Upcoming, and Yahoo Tech, send pings to Pingerati when they have new or updated microformatted content. Even more important, services can automatically receive notification of new and changed content. Technorati and Eventful are two such services at present, but any service can do this.

Alexa

Alexa, a subsidiary of Amazon, is a search engine that opens its index to developers as a search platform. Most people will probably have heard of Alexa via its traffic rankings, or the Alexa Toolbar, but the fact that developers can use Alexa's data to build search applications opens significant opportunities for interesting, "vertical" search engines.

One such example is an hCard search engine (`www.alexa.com/site/devcorner/samples?page=hcard`) that looks through Alexa's index for hCard-formatted information. While it's not yet a production-level tool, it demonstrates the kind of vertical, microformat-based search service that Alexa could enable.

Summary

The examples presented in this chapter do not represent an exhaustive list of what big and small publishers, service providers, and tool developers are doing with microformats, but they do serve to show the breadth of adoption of a technology still in many ways in its infancy. Throughout the book, you'll explore these companies, sites, and examples in more detail.

In many ways, this really is the perfect time to begin adopting microformats, if you've not yet done so. Others have taken the early adopter risks and demonstrated the technical feasibility and business cases for using microformats, and they are not yet ubiquitous. There are still opportunities for novel uses and for novel services based on microformats.

Now it's time to get practical and sharpen your semantic HTML skills (you'll use them time and again throughout the book) in the next chapter. After that, you'll dive headlong into microformats.

PART TWO **USING MICROFORMATS**

The first part of the book covered the concepts, philosophies, and principles associated with microformats. We looked at some of the problems microformats help solve and some of the interesting, powerful uses they are being put to by publishers (big and small), aggregators, indexers, and other web services developers. Hopefully now you want to join their ranks.

In this part of the book, you'll get your hands dirty. You'll begin by looking at the foundations underpinning all microformats: the correct, structural, and semantic use of HTML and XHTML. You'll then examine the main microformats in turn and learn what problems they were designed to solve, how they work, and how you can integrate them into your web development more fully—through web services that may work with a particular microformat, or through the use of CSS to style microformatted content.

3 STRUCTURAL AND SEMANTIC HTML

The history of HTML is convoluted. HTML began life as a simple, structural, semantic markup language for publishing scientific documents. As the popularity of the Web grew, more and more presentational aspects (elements like `<font>` and attributes like `bgcolor`) were added by browser developers (I'm looking at you, Netscape; what were you thinking with `<blink>`?), until HTML morphed into a mishmash of ad hoc "innovations" that strayed far from the original intent of its developer, Tim Berners-Lee.

In this chapter, you'll see how the damage done to HTML has been reversed over the last decade or so, through the standardization process of the W3C. You'll also gain an understanding of what **semantic HTML** *really* means, and you'll learn how to use HTML and XHTML as they were intended and designed. Because microformats explicitly build on the semantic foundations of HTML, it's essential that you have an understanding of semantic HTML; without this understanding, microformats will make little sense. In the process, you'll examine what HTML isn't and is, cover bad practices that you need to shy away from, and see examples of correctly marked-up HTML. Finally, you'll explore the limitations of HTML that led to microformats being developed in the first place.

Even if you have been developing HTML for years by hand, do take the time to read this chapter. The ideas and themes considered here will recur throughout the remainder of the book.

The bad old days of HTML

Chances are that whether you started developing for the Web months or years ago, at some point you've done something like this:

```
<h1><font face="Times New Roman" size="5">This is a heading</font></h1>
```

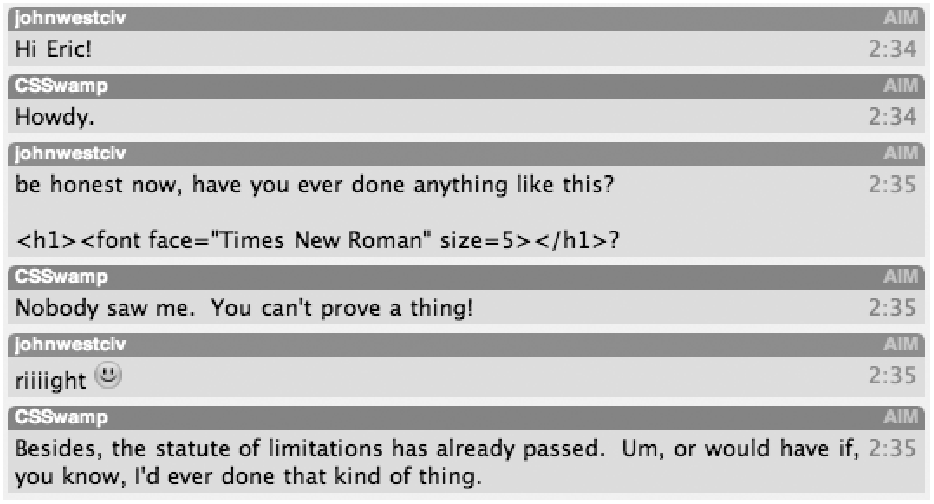
I admit it: I have. In fact, I guarantee that even Eric Meyer has probably done something like this at some point. Don't believe me? Let's ask him, shall we? (See Figure 3-1.)

OK, so Mr. Meyer is being a bit coy.

Or perhaps you've done something like this:

```
<b>this is very important</b>
```

Chances are good that you know you shouldn't do this, but maybe you aren't *really* sure why.



johnwestclv AIM
Hi Eric! 2:34

CSSwamp AIM
Howdy. 2:34

johnwestclv AIM
be honest now, have you ever done anything like this? 2:35

<h1><font face="Times New Roman" size=5></h1>?

CSSwamp AIM
Nobody saw me. You can't prove a thing! 2:35

johnwestclv AIM
riiiight 2:35

CSSwamp AIM
Besides, the statute of limitations has already passed. Um, or would have if, you know, I'd ever done that kind of thing. 2:35

Figure 3-1. Even Eric Meyer isn't completely absolved!

Web developers used to do these sorts of things because before CSS, if we wanted to add style to web pages, *we had no other choice*. There was no other way to style web pages.

Now, what you might not know (unless you are really old like me) is that there was a time when there was no `<font>` element. Introduced by Netscape Navigator, with considerable controversy, the `<font>` element was not part of the original HTML language. This original HTML was purely a structural markup language, not the presentational markup language it slowly became from about 1993 onward with the addition of presentational elements like `<font>` and attributes like `bgcolor`.

Many people would consider (and a good number even argued so at the time) the whole sidetrack into presentational markup to be a mistake. Indeed, the last decade, beginning with the introduction of CSS in 1996, then the strict version of HTML 4.0, and finally XHTML 1.1 (which no longer permits presentational markup in valid HTML documents), has seen the long, painful process of putting this genie back into its bottle. But like all genies, it won't go quietly.

The key to modern web development is understanding that HTML is for marking up content structurally, not describing the presentation of pages. Microformats build on top of the correct, structural, and semantic use of HTML, so using microformats to their full potential depends on understanding how HTML should properly be used.

HTML isn't a presentational markup language

Sometimes it's easier to start with what is going wrong. I've already said it, but just for emphasis:

HTML is not a presentational markup language.

The best use of HTML—the appropriate, current use of HTML—is for structural and semantic markup.

Many people will argue that you can write valid XHTML using the `<font>` element, and that you can use tables for layout with valid XHTML 1.0. So who says HTML is not a presentational language?

While presentational aspects of HTML are valid HTML 4.01 (in the Transitional doctype) and even XHTML 1.0 (again, in the Transitional doctype), they are officially deprecated in these specifications and obsolete in the most recent HTML specification, XHTML 1.1. When an aspect of a language is deprecated, this indicates that it will be obsolete in a future version (and, as we have seen, presentational aspects of HTML are obsolete in XHTML 1.1).

So the W3C says HTML is not a presentational language. That's who.

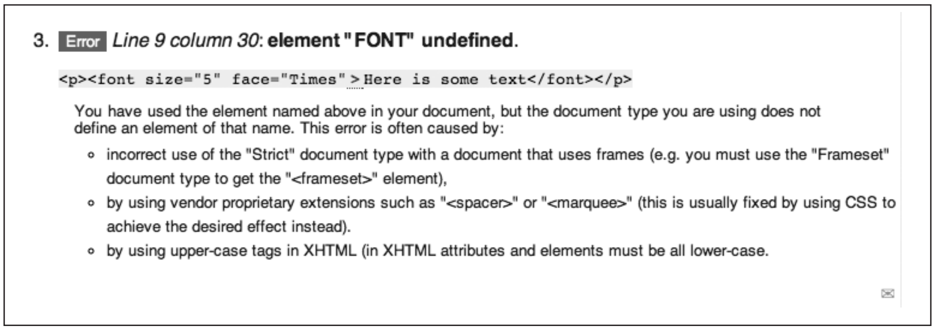
Uncovering the deprecated aspects of HTML

In the HTML specifications, nowhere is it explicitly spelled out that "these are the presentational elements and attributes of the language." However, it is largely a matter of common sense. If an element (like `<font>`) or attribute (like `bgcolor`) clearly exists to bring style to a page, it is very likely deprecated. You can easily verify this by validating your pages against a strict doctype, or by looking up the attribute or element in the XHTML 1.0 specification to determine whether it has been deprecated.

For example, if I try to validate a document with the following HTML using the strict version of HTML 4.01:

```
<p><font size="5" face="Times">Here is some text</font></p>
```

Figure 3-2 shows what the W3C's HTML validator tells me.



3. Error *Line 9 column 30*: **element "FONT" undefined.**

```
<p><font size="5" face="Times">Here is some text</font></p>
```

You have used the element named above in your document, but the document type you are using does not define an element of that name. This error is often caused by:

- incorrect use of the "Strict" document type with a document that uses frames (e.g. you must use the "Frameset" document type to get the "<frameset>" element),
- by using vendor proprietary extensions such as "<spacer>" or "<marquee>" (this is usually fixed by using CSS to achieve the desired effect instead).
- by using upper-case tags in XHTML (in XHTML attributes and elements must be all lower-case.

Figure 3-2. The W3C validator lets you know what's not acceptable in your HTML under the doctype you are using.

Ensuring that your HTML is syntactically correct—that is, that it follows all the rules of the HTML language—can be either extremely complex if you try to do it all "in your head" or straightforward if you just use a validator.

A validator won't ensure that your HTML is perfect in terms of using the right element for the right reason, but it will ensure that you haven't broken the grammatical rules of HTML. For example, a validator will ensure that all your start and end tags are balanced correctly (every start tag must have an end tag, and every end tag must have a start tag); that you don't break containment rules, such as having block elements inside inline elements or paragraphs; and that you have the required attributes for elements, such as the `alt` *attribute for* `<img>` *elements, and have not used incorrect attributes, such as proprietary browser extensions or obsolete attributes.*

Choosing a strict doctype to validate against goes a step further, as the validator will then flag any use of presentational elements like `<font>` *or presentational attributes like* `bgcolor`*.*

Validating an HTML document

The first step in validating an HTML document is to ensure that your document declares a **doctype**—that is, it states which version of HTML or XHTML it is. In order for an HTML document to be considered valid, a doctype *must be included*.

To do this, you add a **doctype declaration** (DTD) to the top of the HTML file, outside the `<html>` element. For example, the following DTD says that the document that follows is HTML 4.01 Strict:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" ➥
  "http://www.w3.org/TR/html4/strict.dtd">
```

This DTD says the document is XHTML 1.1:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" ➥
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Remember, there is no Transitional version of XHTML 1.1.

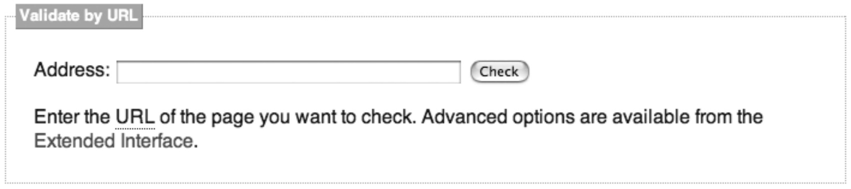
Notice that a URL appears in each of these DTDs. The URL is not required, but when a doctype doesn't include it, something special happens in all modern browsers: they draw or render pages in what's commonly called **quirks mode**. In other words, they effectively emulate how older browsers would render the page. If the doctype includes a URL, the page is rendered in **standards mode** or **standards-compliant mode**, meaning according to the latest W3C specifications for HTML and CSS.

The reason for this difference is that going back to the time when decent CSS support first appeared in browsers, generally considered to be Internet Explorer 5 for the Mac in 2000, if this and later browsers started rendering every web page strictly according to the HTML and CSS specifications, many millions of pages on the Web would be broken, because they had been authored to take advantage of current browser bugs. The assumption behind rendering documents with a full doctype including the URL in conformance with standards is that the browser treats these as an assertion that the page is fully standards-compliant, whereas the absence of a doctype altogether indicates the very real likelihood that the developer did not design the page with standards compliance in mind. The halfway position of allowing developers to include a doctype to allow validation, but leave off the URL to indicate that the page is designed to render using older, less conformant rendering models, has been adopted by all contemporary browsers.

The most important thing to take away from this discussion is that pages will often render quite differently in exactly the same browser, depending on whether you add or leave off a doctype, or leave in or omit the URL. If you have added a doctype to your pages for the first time to validate them, and they suddenly appear "broken" when rendered by a browser, then leave off the URL and all should be fine—your pages will validate, and yet not break when rendered in older browsers.

Several online validators and many web development applications include a validator, but I recommend going to the source and using the W3C's own validator. You'll find it at `http://validator.w3.org`, and there are three ways to use it:

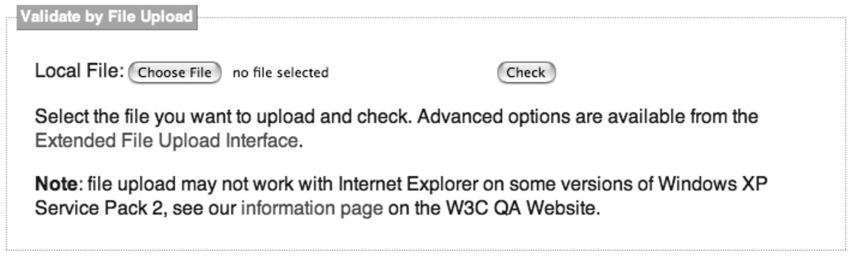
- If your page is already online, you can just enter your page's URL into the validator, as shown in Figure 3-3.



Enter the URL of the page you want to check. Advanced options are available from the Extended Interface.

Figure 3-3. You can enter the URL of a page into the W3C validator to check it.

- You can upload your HTML file directly from your hard disk, as shown in Figure 3-4.

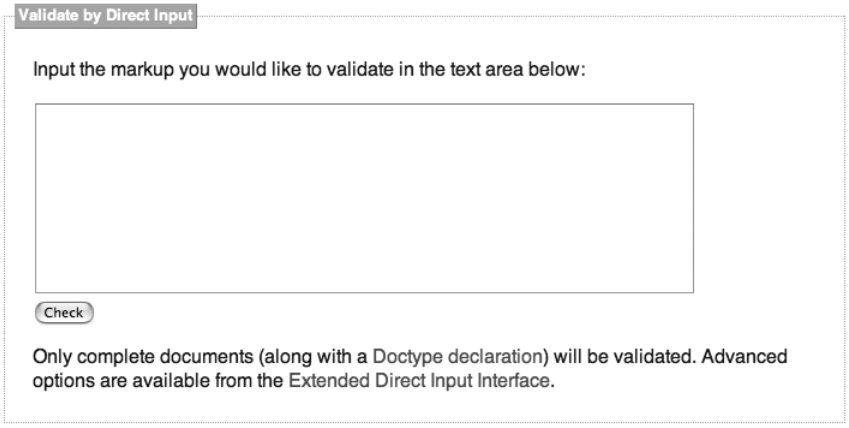


Select the file you want to upload and check. Advanced options are available from the Extended File Upload Interface.

Note: file upload may not work with Internet Explorer on some versions of Windows XP Service Pack 2, see our information page on the W3C QA Website.

Figure 3-4. You can upload a page from your hard disk to the W3C validator.

- You can simply paste your HTML directly into a form field, as shown in Figure 3-5. Be aware that some browsers may have problems with large HTML documents if you do this.

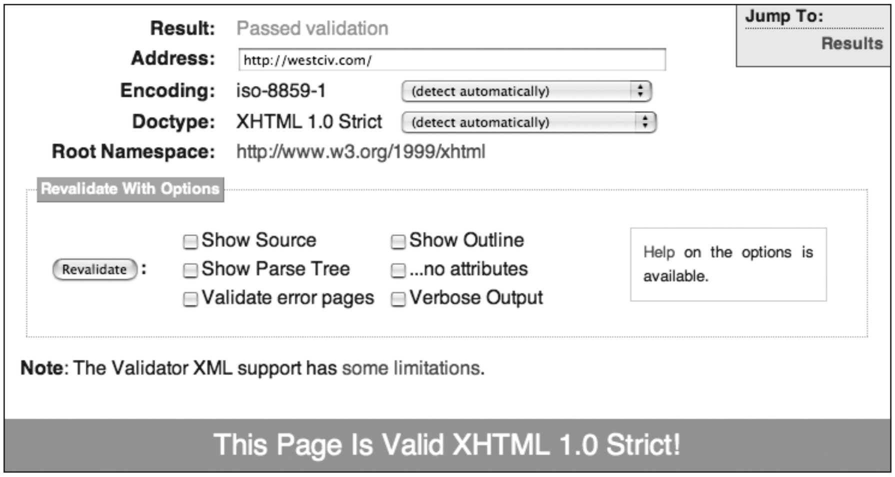

Validate by Direct Input

Input the markup you would like to validate in the text area below:

Only complete documents (along with a Doctype declaration) will be validated. Advanced options are available from the Extended Direct Input Interface.

Figure 3-5. You can paste code that you want checked directly into the W3C validator.

Whichever method you use, the W3C validator will return the results of checking your document against the version of HTML declared by your DTD. Figure 3-6 shows what you'll hopefully see.



Result:	Passed validation
Address:	http://westciv.com/
Encoding:	iso-8859-1 (detect automatically)
Doctype:	XHTML 1.0 Strict (detect automatically)
Root Namespace:	http://www.w3.org/1999/xhtml

Revalidate With Options

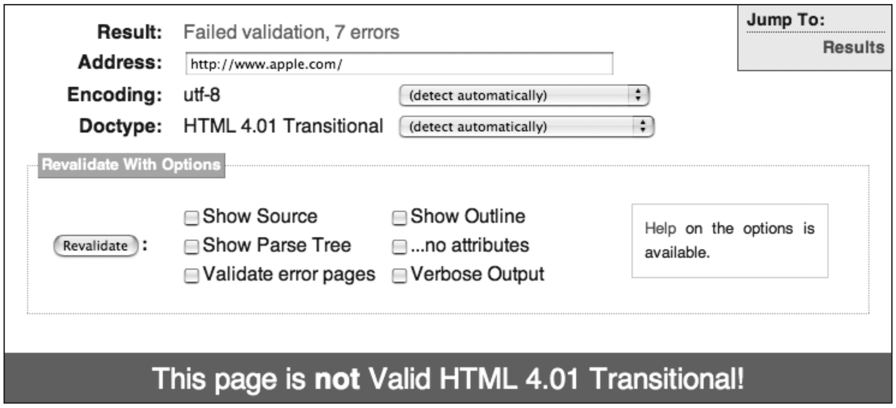
Help on the options is available.

Note: The Validator XML support has some limitations.

This Page Is Valid XHTML 1.0 Strict!

Figure 3-6. Output from the W3C validator when a page has passed its checks

Figure 3-7 shows what you might well see if this is the first time you've tried validating your HTML.



Result: Failed validation, 7 errors
Address: http://www.apple.com/
Encoding: utf-8 (detect automatically)
Doctype: HTML 4.01 Transitional (detect automatically)

Help on the options is available.

This page is **not** Valid HTML 4.01 Transitional!

Figure 3-7. Output from the W3C validator when a page has failed its checks

I don't have the space to go into how to use the results, but you should know that the validator will report every error, its line number, the cause of the error, and often even links to further reading. Don't despair if you get a lot of errors initially. It's likely that the same smallish set of errors is repeated throughout your document, and you can rapidly get the number of errors down to a reasonable number, and then fully validate your document.

See my article "The State of the Art in Australian Web Development" for more information on the common errors I have found on major websites: `http://westciv.com/style_master/house/good_oil/best_practices/index.html`.

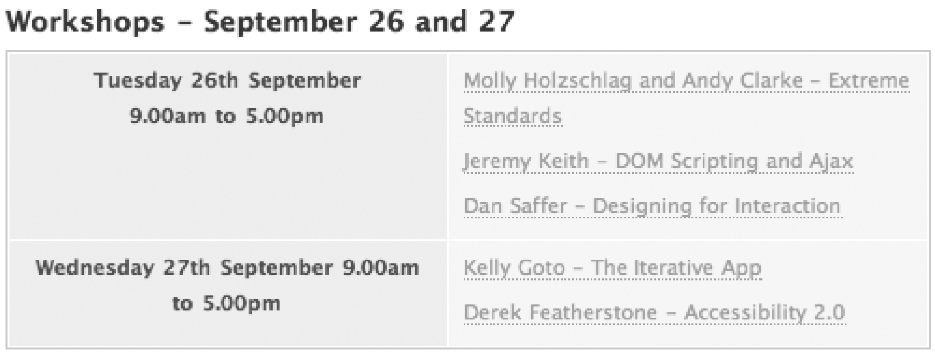
There are a couple of gray areas and apparent exceptions to the rule of thumb that presentational HTML is deprecated. One regards the use of tables for marking up page layout, which we'll turn to in just a moment. Another is that both the bold and italic elements (but not the underline element) are still valid, nondeprecated parts of HTML. Exactly why this is the case is perhaps a little too time consuming to explain here, and the topic is not without some controversy. In keeping with our rule of thumb, though, using `<b>` or `<i>` is to be avoided. Very often the emphasis `<em>` or strong emphasis `<strong>` element is appropriate to use here, although in English, the use of italics may indicate not emphasis; rather, it may indicate the use of a foreign-language word not commonly used in English. For example, if you were to use the German word "schadenfreude" on your page, you would italicize it to indicate that it is a borrowed word. `<em> schadenfreude </em>` would not be an appropriate use of the `<em>` element, because you are not emphasizing this word. Perhaps `<span class="borrowed">schadenfreude</span>` would be more appropriate; here, you create a class to use any time we use a word borrowed from a language other than English.

Tables are bad—m'kay?

The use of tables helps illuminate the difference between structural and presentational markup. It demonstrates that using HTML properly is not merely about inflexibly following a set of rules (such as "tables are bad"), but understanding the philosophy or spirit of correct markup and using the elements of HTML within that spirit.

Tables are valid HTML 4.01 Strict, XHTML 1.0 Strict, and XHTML 1.1, so you may create a perfectly *valid* page that uses 40 nested tables for the layout. But the purpose of tables is to allow the *markup of tabular data*. A timetable is a perfect example of what tables in HTML are designed for.

Figure 3-8 shows an entirely appropriate use of a table in HTML. (This table also has some sophisticated microformatting, which you'll learn about in a later chapter.)

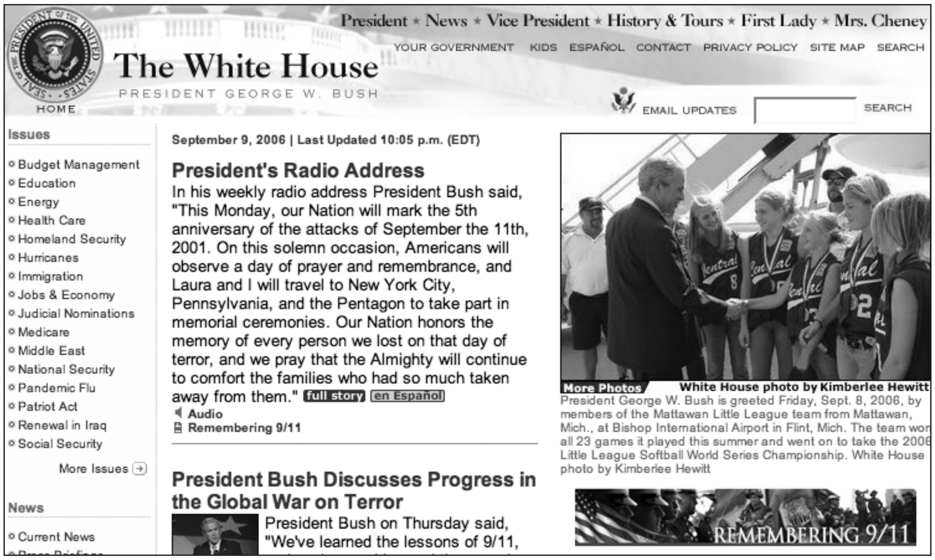


Workshops – September 26 and 27

Tuesday 26th September 9.00am to 5.00pm	Molly Holzschlag and Andy Clarke – Extreme Standards Jeremy Keith – DOM Scripting and Ajax Dan Saffer – Designing for Interaction
Wednesday 27th September 9.00am to 5.00pm	Kelly Goto – The Iterative App Derek Featherstone – Accessibility 2.0

Figure 3-8. An acceptable use of an HTML table

Figure 3-9 shows the kind of thing tables are *not* designed for.



President ★ News ★ Vice President ★ History & Tours ★ First Lady ★ Mrs. Cheney

YOUR GOVERNMENT KIDS ESPAÑOL CONTACT PRIVACY POLICY SITE MAP SEARCH

The White House

PRESIDENT GEORGE W. BUSH

HOME

EMAIL UPDATES SEARCH

Issues

- Budget Management
- Education
- Energy
- Health Care
- Homeland Security
- Hurricanes
- Immigration
- Jobs & Economy
- Judicial Nominations
- Medicare
- Middle East
- National Security
- Pandemic Flu
- Patriot Act
- Renewal in Iraq
- Social Security

More Issues

News

- Current News

September 9, 2006 | Last Updated 10:05 p.m. (EDT)

President's Radio Address

In his weekly radio address President Bush said, "This Monday, our Nation will mark the 5th anniversary of the attacks of September the 11th, 2001. On this solemn occasion, Americans will observe a day of prayer and remembrance, and Laura and I will travel to New York City, Pennsylvania, and the Pentagon to take part in memorial ceremonies. Our Nation honors the memory of every person we lost on that day of terror, and we pray that the Almighty will continue to comfort the families who had so much taken away from them." full story en Español

Audio

Remembering 9/11

President Bush Discusses Progress in the Global War on Terror

President Bush on Thursday said, "We've learned the lessons of 9/11,

More Photos **White House photo by Kimberlee Hewitt**

President George W. Bush is greeted Friday, Sept. 8, 2006, by members of the Mattawan Little League team from Mattawan, Mich., at Bishop International Airport in Flint, Mich. The team won all 23 games it played this summer and went on to take the 2006 Little League Softball World Series Championship. White House photo by Kimberlee Hewitt

REMEMBERING 9/11

Figure 3-9. This type of layout is not what you use HTML tables for!

Almost all the layout at The White House site (`www.whitehouse.gov`) is done with tables. Why is this bad? The HTML 4.01 specification states the following (see `www.w3.org/TR/PR-html40-971107/struct/tables.html`):

The HTML table model allows authors to arrange data—text, preformatted text, images, links, forms, form fields, other tables, etc.—into rows and columns of cells.

Tables should not be used purely as a means to layout [sic] *document content as this may present problems when rendering to non-visual media. Additionally, when used with graphics, these tables may force users to scroll horizontally to view a table designed on a system with a larger display. To minimize these problems, authors should use style sheets to control layout rather than tables.*

So while using tables for layout is, strictly speaking, valid (i.e., the validator won't report an error), it is not good contemporary structural and semantic markup practice.

Now that you have a handle on what HTML is *not* meant to do, let's next take a look at what it *is* designed for.

HTML is a structural and semantic markup language

One of the most intimidating words for many web developers is "semantics." Sometimes it seems that everyone knows what it means, yet it's not clearly defined anywhere. If you do a web search for "semantics," or even "HTML semantics," you probably won't find a lot of help clarifying what it means. This is ironic, because if you turn to a dictionary, you'll find that the term is defined as "the study of meanings."

So, at the risk of being wrong (which has never stopped me in the past!), let me try a working definition:

Semantic HTML *is the use of the most appropriate HTML elements and attributes for the content we are marking up.*

Perhaps the best way of thinking about HTML in this context is to consider the elements and attributes of HTML as building blocks, or LEGOs, for creating meaningful documents. HTML provides special-purpose components, some of which all developers will be very familiar with, such as headings, lists, and paragraphs, and others of which we might have heard of or even use, but less frequently, such as `<em>` and `<strong>`. Still others we may never have heard of, but can be very useful, such as quotations and citations. Then, of course, there are the elements very few people know about, which show HTML's origins, such as `<kbd>` for marking up keyboard input, `<var>` for variables, and `<samp>` for samples. In the building-block/LEGO analogy, these components are akin to LEGO pieces designed for specific purposes, such as specially molded heads for people. You could, of course, use a plain square block for a head, and you could even use a pen to draw a face on the block, but it's really not as appropriate as using the piece designed for this job.

In addition to all those specialized pieces, there are generic LEGO pieces that you can use for all kinds of building—from walls, to aircraft, to trees, to cows. HTML also provides mechanisms for creating our own purpose-built blocks, using the `<div>` and `<span>` elements, along with the `class` and `id` attributes. These components can be very useful, and indeed, microformats make extensive use of this aspect of HTML to create rich document semantics. However, they are often overused, which leads to the condition many of us suffer from when we begin using HTML in this more "sophisticated" way: the excessive use of classes and divs, even where not necessary (referred to as **classitis** and **divitis**, respectively).

Let's look at a classic example of classitis, from the CSS Zen Garden of all places. The Zen Garden HTML was originally designed to be very stylable with CSS, and so class is used extensively as hooks for CSS. Now, using this as an example might be seen as being critical of Dave Shea for his code, but if we keep in mind what he was trying to attempt with the Zen Garden—very stylable HTML at a time when the support for CSS in a lot of browsers was not fantastic (well, it was mostly rubbish, actually)—I think his choices were entirely valid. But nonetheless, it shows us what classitis looks like.

Unfortunately, too, I think an unintended side effect of the success of the CSS Zen Garden was that it exposed a lot of developers to one way of using CSS—with class*—and encouraged this rather bad habit. Later in the book, I cover techniques for reducing reliance on classes using descendent selectors attribute selectors, and other powerful, still underused CSS features.*

```
<div id="container">
  <div id="intro">
    <div id="pageHeader">
      <h1><span>css Zen Garden</span></h1>
      <h2><span>The Beauty of
 <acronym title="Cascading Style Sheets">CSS</acronym>
 Design</span></h2>
    </div>
    <div id="quickSummary">
      <p class="p1">
        <span>
          A demonstration of what can be accomplished visually ➥
          through <acronym title="Cascading Style Sheets">CSS ➥
          </acronym>-based design. Select any style sheet from the ➥
          list to load it into this page.
        </span>
      </p>
      <p class="p2">
        <span>
          Download the sample <a href="/zengarden-sample.html"➥
          title="This page's source HTML code, not to be modified."> ➥
          html file</a> and <a href="/zengarden-sample.css"➥
          title="This page's sample CSS, the file you may modify."> ➥
          css file</a>
        </span>
      </p>
    </div>
    <div id="preamble">
      <h3><span>The Road to Enlightenment</span></h3>
      <p class="p1">
        <span>
          Littering a dark and dreary road lay the past relics of ➥
          browser-specific tags, incompatible <acronym title= ➥
          "Document Object Model">DOM</acronym>s, and broken ➥
          <acronym title="Cascading Style Sheets">CSS</acronym> ➥
          support.
        </span>
      </p>
      <p class="p2">
        <span>
          Today, we must clear the mind of past practices. Web ➥
          enlightenment has been achieved thanks to the tireless ➥
          efforts of folk like the <acronym title="World Wide Web ➥
          Consortium">W3C</acronym>, <acronym title="Web Standards ➥
          Project">WaSP</acronym> and the major browser creators.
        </span>
      </p>
      <p class="p3">
        <span>
          The css Zen Garden invites you to relax and meditate on ➥
          the important lessons of the masters. Begin to see with ➥
          clarity. Learn to use the (yet to be) time-honored ➥
          techniques in new and invigorating fashion. Become one ➥
          with the web. ➥
        </span>
      </p>
    </div>
  </div>
</div>
```

As you can see, just about every element in the document has a class or id value!

Document structure

Before worrying all that much about more advanced aspects of HTML, let's focus on the fundamentals: document structure. Using paragraphs, lists, headings, and other key building blocks of HTML appropriately provides our documents with solid structural and semantic foundations. Without them, adding more sophisticated HTML and microformats is simply building on sand.

Headings and paragraphs

Not too long ago, you would often find markup like this on the Web (and sadly, if you look under the hood of many sites, it persists even today):

```
<p><font face="Times" size=7 >What HTML Is</font></p>
```

Clearly the contents of the paragraph are meant to be a heading! We should, of course, simply use the following:

```
<h1>What HTML is</h1>
```

That is, we should use the elements provided by HTML in the appropriate way.

Here's another common problem:

```
<div class="header">What HTML is</div>
```

This is an example of classitis, where `<div>` and `<span>` elements are used with classes, instead of the appropriate HTML elements. In many respects, it's not really any better than using presentational HTML like the `<font>` element.

Similarly, you can find developers using headings when they want "big and bold" text, even for content that isn't a heading. But the simple lesson is, HTML has headings, so use them when your content has "a title, subtitle, or topic that stands at the top or beginning, as of a paragraph, letter, or chapter."[1]

Using headings appropriately brings practical benefits. It's quite probable that search engines use heading markup to look for more significant content on a page (though search engine developers keep to themselves just how their search engines work), and we do know that screen-reader software (i.e., software that reads web pages to its users, who typically suffer from some kind of visual disability), such as JAWS, uses the headings on a page to create navigation for its users. It's a common misconception that screen reader users simply "listen to" web pages. Just as we know most sighted users will skim content, looking at significant parts of a page like headings, a screen reader allows its users to do likewise. If you use HTML appropriately, one of the many benefits is that you help people listen to your web pages more effectively.

Lists

One of the most flexible and perhaps most underutilized semantic features of HTML is its various list types. HTML features unordered lists, ordered lists, and definition lists. An excellent example of the increasing semantic sophistication of web development is the now quite common use of unordered lists for navigation.

Here's an example from my own site:

1. From *The American Heritage Dictionary of the English Language* (Houghton Mifflin, 2000).

```
<ul id="site-sections">
  <li><a href="index.html" id="current-section"
>home<span></span></a></li>
  <li><a href="style_master/index.html">style master</a></li>
  <li><a href="style_master/house/index.html">learning</a></li>
  <li><a href="westciv/faq.html">support</a></li>
  <li><a href="westciv/downloads.html">downloads</a></li>
  <li><a href="https://order.kagi.com/?WC4">store</a></li>
</ul>
```

Figure 3-10 shows what this list looks like with just a little CSS styling.

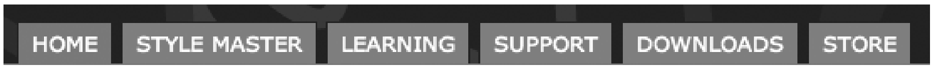


Figure 3-10. An elegant navigation menu, created with nothing more than an unordered list and some CSS

One perhaps not particularly obvious advantage of this list is that if a reader turns CSS off or uses a text-based browser, the logic of the document is still immediately apparent. Figure 3-11 shows the same list with no CSS applied.

Why is a list appropriate for marking up site navigation? Well, that's what navigation is after all: a list of links to other parts of a site.

Another common and sensible use of lists is for instructions. Here, an ordered list may make the most sense, as instructions typically involve steps that must be followed in a strict sequence. Here is an example from `www.cacaoweb.net`:

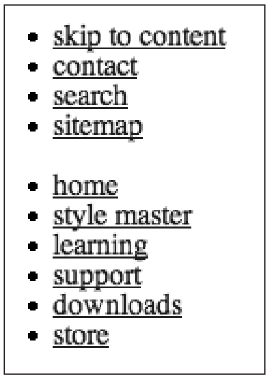


Figure 3-11. The unordered list navigation menu still makes perfect sense without CSS.

```
<ol>
  <li>Preheat oven to 350 deg F (Gas mark 4 or 180 deg C).</li>
  <li>Line a 13 x 9 in (33 x 23 cm) cake tin ... tin.</li>
  <li>Melt the butter.</li>
  <li>Beat eggs with sugar, and ... butter.</li>
  <li>Add chopped nuts.</li>
  <li>Bake at 350 degrees until ... minutes.</li>
  <li>Cool the cake. Dust with powdered sugar ...</li>
</ol>
```

Using lists like this has an advantage similar to the accessibility benefit we saw with headings. Screen readers typically read out the number of items in a list before reading out each item. So lists provide context for screen reader users, and help them make decisions about what content to read and what content to skip.

Lists such as these might have been marked up using a table, or even with a paragraph and line breaks, not too long ago. (And indeed lists are marked up in this fashion to this day by those less enlightened than ourselves.) Using lists appropriately (and using HTML in a structural, semantic way) makes styling our sites typically much easier than older approaches, such as using tables or paragraphs with line breaks. We'll take a look at a number of simple yet powerful CSS techniques utilizing the nature of well-structured documents in upcoming chapters.

Beyond the <p> tag

Most of us have learned HTML on the job. Typically, we've acquired the skills we needed to solve problems at hand, and as that wonderful expression goes, "To a man with a hammer, everything looks like a nail." Once we learn how to use a particular technique, we'll often reuse that technique over and over again, because it "works," despite there potentially being something better out there.

In my experience, many web developers acquire a set of HTML skills and reuse these. But HTML probably has many more features than most of us are familiar with. Let's delve into some of the less commonly used but still very useful elements and attributes of HTML, as many of these will come in handy when you use microformats.

Yes, you in the back, who pointed out the address element of HTML—this time, you are right, `<p>` *is in fact an element, not a tag. Well, there are both a* `<p>` *element and a* `<p>` *tag, but typically people mean element when they say tag. What? Here's one for the pedants among us.*

This is an element: `<p>This is a paragraph</p>`*. It has both a start tag* `<p>` *and an end tag* `</p>`*. I've been referring to* `<elements>` *in this book, including the angle brackets to make them stand out more on the page, even if that is not quite correct, strictly speaking. But at least you notice them better.*

Perhaps the most egregious error many of us commonly make along these lines is to refer to the "alt tag" (don't be embarrassed if you use that expression—most of us do, and it's a hard habit to break). But not only is alt not a tag, it's not even an element. It's an attribute. So just say "alt text," and that should cover it nicely.

Citations, quotations, and more

At the heart of the Web is the link (or "hyperlink" as it was once called, although somehow "hyperlink" sounds quaint to me nowadays, a little like the way my grandmother called the radio a "wireless." Ironic that "wireless" is a contemporary term once more). And a link is often a reference to what someone else has said—that is, a **citation**. When we link to something we are often citing, and HTML has the underused `<cite>` element, which enables us to semantically mark up the reference for what it is, a citation.

Similar to but not to be confused with a citation is a **quote**. HTML provides not just one, but two elements for quoting. For short quotes there is the `<q>` element (an inline element), and for longer quotes there is the `<blockquote>` element (not surprisingly, a block element). If you are not quite sure of the difference, I'll discuss both inline and block elements in a moment. Of course, when quoting, we often cite the source of our quote, so citations and quotes naturally go together.

For example, referring to Lincoln's address at Gettysburg, among the most famous of all speeches, we could mark it up as follows:

```
<p>Ironically, <cite>Lincoln's Gettysburg address</cite> through ➥
its brevity and power proved to contradict itself.➥
<q cite="http://en.wikipedia.org/wiki/Gettysberg_address">
The world will little note, nor long remember ➥
what we say here</q> for indeed the ➥
world has ever since noted this speech.</p>
```

If you pay careful attention, you'll see that the quote element contains a `cite` attribute, which points to a source for the quotation, using a URL. For our purposes, only the `<q>` and `<blockquote>` elements can take the `cite` attribute.

If you burrow even more deeply into HTML, you'll find other very useful elements and attributes of HTML you may well not have heard of, but which you'll soon see microformats cleverly use to create sophisticated valid HTML constructs.

For example, the `<abbr>` element indicates "an abbreviated form," according to the HTML specification. Obviously, you can use this element to mark up abbreviations like `<abbr>W3C</abbr>`, `<abbr>WHO</abbr>` or `<abbr>WIPO</abbr>`. WIPO . . . what's that, you ask? Well, here you can add a little more HTML, for example, the `title` attribute (yes, the `title` attribute is not just for links) to "provide the full or expanded form of the expression." So, you would have `<abbr title="World Intellectual Property Organization"> WIPO</abbr>` to mark up the abbreviation and its expansion.

Browsers like Firefox and other Mozilla-based browsers, and Opera (but sadly neither Safari 2.0 nor Internet Explorer 6 or older on Windows), show the contents of the title in a tooltip. While the HTML specification doesn't require the `title` attribute's value to be displayed in a tooltip, it's still a very helpful feature of those browsers that provide it. We could use CSS to provide a tooltiplike expansion of the `title` attribute value when the mouse hovers over the `<abbr>` element (at least in Safari, which supports the generated content aspects of CSS2), but I'll leave that for you to play around with. Even the just-released Internet Explorer 7 as well as older versions of Internet Explorer don't support this aspect of CSS. We'll look at generated content in some detail in later chapters.

As you'll see when you start looking at microformats in detail, the `<abbr>` element and `title` attribute (among other HTML elements and attributes for marking up content) are very powerful mechanisms for marking up content, and they are used extensively.

XHTML compounds

By now you might be asking, "Why haven't you covered divs and spans, `id`s and `class`es? These are all-purpose tools for adding semantics to HTML, and I see and use them all the time." It's a fair question, and there is a specific reason why I've left off discussing these commonly used aspects of HTML so far (and, in fact, will for a little while longer). But I'll get to them, I promise.

Tantek Çelik, among others, has proposed that before we simply revert to divs, spans, ids, and classes, we look first to both existing aspects of HTML and what he calls *HTML compounds* (by analogy, with chemistry we treat HTML elements like, well, elements, so by combining them, we get compounds).

An HTML compound is "two or more XHTML Elements used in combination to express new, perhaps more specific, semantics."[2] Once more, let's look at a practical example, this one based on an example by Tantek Çelik. Suppose you are marking up a bibliography—a list of books cited in an article. You already have an inkling of the HTML that might go into making the compound bibliography: lists and cite elements. Here is the unmarked-up text of the bibliography (from the CSS2.1 specification's list of normative references, if you were wondering):

- "Colorimetry, Second Edition", CIE Publication 15.2-1986, ISBN 3-900-734-00-3, http://www.cie.co.at/publ/abst/15-2-86.html.
- "Cascading Style Sheets, level 1", H. W. Lie and B. Bos, 17 December 1996, revised 11 January 1999. The latest version is available at http://www.w3.org/TR/REC-CSS1.
- "Cascading Style Sheets, level 2, CSS2 Specification", B. Bos, H. W. Lie, C. Lilley and I. Jacobs, 12 May 1998. The latest version is available at http://www.w3.org/TR/REC-CSS2.

Each of these entries is a citation, so you'll mark them up as such:

```
<cite>"Colorimetry, Second Edition", CIE Publication 15.2-1986, ➥
  ISBN 3-900-734-00-3.
  http://www.cie.co.at/publ/abst/15-2-86.html</cite>
  <cite>"Cascading Style Sheets, level 1", H. W. Lie and B. Bos, 17 ➥
  December 1996, revised 11 January 1999 The ➥
latest version is available at http://www.w3.org/TR/REC-CSS1
</cite>
<cite>"Cascading Style Sheets, level 2, CSS2 Specification", B. Bos, ➥
  H. W. Lie, C. Lilley and I. Jacobs, 12 May 1998, The latest ➥
  version is available at http://www.w3.org/TR/REC-CSS2</cite>
```

While we are at it, let's make the online references into links:

```
<cite><a href="http://www.cie.co.at/publ/abst/15-2-86.html"> ➥
  "Colorimetry, Second Edition", CIE Publication 15.2-1986, ➥
  ISBN 3-900-734-00-3</cite>
<cite>"<a href="http://www.w3.org/TR/REC-CSS1">Cascading ➥
  Style Sheets, level 1", H. W. Lie and B. Bos, 17 December 1996, ➥
  revised 11 January 1999</a>
</cite>
<cite><a href="http://www.w3.org/TR/REC-CSS2">"Cascading Style ➥
  Sheets, level 2, CSS2 Specification", B. Bos, H. W. Lie, ➥
  C. Lilley and I. Jacobs, 12 May 1998</a></cite>
```

2. See http://tantek.com/presentations/2005/09/elements-of-xhtml.

Now this is a list, typically ordered alphabetically, so we'll mark it up as an ordered list:

```
<ol>
  <li><cite><a href="http://www.cie.co.at/publ/abst/15-2-86.html"> ➥
    "Colorimetry, Second Edition", CIE Publication 15.2-1986, ➥
    ISBN 3-900-734-00-3</cite></li>
  <li><cite>"<a href="http://www.w3.org/TR/REC-CSS1">Cascading ➥
    Style Sheets, level 1", H. W. Lie and B. Bos, 17 December 1996, ➥
    revised 11 January 1999</a></li>
  <li><cite><a href="http://www.w3.org/TR/REC-CSS2">"Cascading Style ➥
    Sheets, level 2, CSS2 Specification", B. Bos, H. W. Lie, ➥
    C. Lilley and I. Jacobs, 12 May 1998</a></cite></li>
</ol>
```

Figure 3-12 shows what the list will look like in a browser, without any CSS. (Of course, you could style the list in any number of ways with CSS.)

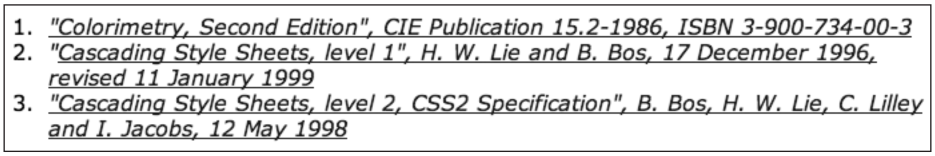

1. "Colorimetry, Second Edition", CIE Publication 15.2-1986, ISBN 3-900-734-00-3
2. "Cascading Style Sheets, level 1", H. W. Lie and B. Bos, 17 December 1996, revised 11 January 1999
3. "Cascading Style Sheets, level 2, CSS2 Specification", B. Bos, H. W. Lie, C. Lilley and I. Jacobs, 12 May 1998

Figure 3-12. Semantic markup made easy, with an ordered list this time

And so you have semantic markup for the bibliography, using off-the-shelf HTML elements.

If you're familiar with citations, you know they are more complex than this example demonstrates—there is the issue of authors, serial numbers, publications dates, and so on. So how would you mark up these items semantically? There is often going to be a limit to what you can mark up using just the building blocks of HTML elements and attributes. To really create semantically rich constructs for marking up complex things like citations, you need to go beyond these basic HTML building blocks.

This is the point at which we can turn to the generic semantic markup of HTML: divs and spans, classes and ids.

divs, spans, classes, and ids

Precisely because HTML has so few building blocks, its developers introduced the <div> and <span> elements to provide, in their words, a "generic mechanism for adding structure to documents."

What's the difference between a div and span? Technically, the <div> element is a generic block element, while the <span> element is a generic inline element. The terms "block" and "inline" are a little misleading. After all, I've just spent quite some time emphasizing that HTML is for marking up structured content, not presentation, and yet "block" and "inline" sound suspiciously presentational, do they not?

The terms are left over from a time when this separation of presentation and structure in HTML was less rigidly enforced. But even now, there is a difference between block and inline elements, and it doesn't relate just to presentation. In fact, it has nothing to do with presentation at all. You are probably aware that in HTML, a heading may contain an <a> element, but a an <a> element must not contain a heading. A document with this markup would be flagged as invalid by an HTML validator, because it breaks an HTML containment rule. The HTML containment rules are (with some complications) that block elements may contain other block elements as well as inline elements, but inline elements may only contain other inline elements.

There are some exceptions to this generalization. For example, although the <p> element is a block element, it cannot contain other block elements, only inline elements. When we discuss individual microformats in detail in upcoming chapters, these containment rules will become very relevant.

As noted already, the class and id attributes of HTML are increasingly commonly used by web designers and developers—though not always correctly. This is frequently because developers use class and, to a lesser extent, id to simply create "hooks" for styling with CSS, as you saw a little earlier in this chapter with the CSS Zen Garden example. When this technique is overused, you'll find developers adding class values to just about everything that moves on their web pages, to make styling with CSS "easier"—a prime example of classitis.

If you think about it for a moment, even though such developers are using valid strict HTML, they are violating the spirit of the principle of contemporary web development: separating the presentation of a page from the structured content of that page. Does that mean developers shouldn't use class, id, <div>, and <span>? Not at all, but developers should use them appropriately. Use them to mark up the logical structure of your pages, not to tag the bits of the pages you want to style later on.

Let's look again at some code we saw a moment ago, to illustrate the difference between these two approaches. The following is how many developers would mark up the previous recipe example:

```
<h2 class="instruction-heading">Instructions</h2>
<ol class="instructions">
  <li class="instruction-step1">Preheat oven to 350 deg F (Gas mark ➥
    4 or 180 deg C).</li>
  <li class="instruction-step2">Line a 13 x 9 in ... tin ➥
    with grease-proof paper and grease the tin.</li>
  <li class="instruction-step3">Melt the butter.</li>
  <li class="instruction-step4">Beat eggs with ... vanilla, ➥
    flour, cocoa, baking powder, salt (optional) and melted
butter.</li>
  <li class="instruction-step5">Add chopped nuts.</li>
  <li class="instruction-step6">Bake at 350 degrees until a wooden ➥
    pick inserted stands up. Should take 20-30 minutes.</li>
  <li class="instruction-step7">Cool the cake. Dust with powdered ➥
    sugar.</li>
</ol>
```

And no, that is not an exaggeration! A perfect example of classitis is the use of a CSS class selector throughout a style sheet instead of more appropriate selectors (it's so common that it even has its own Wikipedia entry: `http://en.wikipedia.org/wiki/Classitis`).

Now, let's think about what we are *really* trying to mark up here with all these class values. We have a couple of related elements: the heading and the list. Since these are related, we can group them together using a generic element. We'll have to use a div because it will contain block elements, and the HTML containment rules specify that block elements can only be contained within other block elements.

So, let's wrap the heading and list in a `<div>` element (the generic block element):

```
<div>
  <h2 class="instruction-heading">Instructions</h2>
  <ol class="instructions">
    <li class="instruction-step1">Preheat oven to 350 deg F (Gas ➥
      mark 4 or 180 deg C).</li>
    <li class="instruction-step2">Line a 13 x 9 in ... tin ➥
      with grease-proof paper and grease the tin.</li>
    <li class="instruction-step3">Melt the butter.</li>
    <li class="instruction-step4">Beat eggs with ... vanilla, ➥
      flour, cocoa, baking powder, salt (optional) and melted
butter.</li>
    <li class="instruction-step5">Add chopped nuts.</li>
    <li class="instruction-step6">Bake at 350 degrees until a wooden ➥
      pick inserted stands up. Should take 20-30 minutes.</li>
    <li class="instruction-step7">Cool the cake. Dust with powdered ➥
      sugar.</li>
  </ol>
</div>
```

Now, because the div contains a lot of related information, it makes sense to give it a `class`, and as the information it contains is instructions, it makes sense for the `class` value to be `instructions`. Why not use an `id` here? Well, it's quite possible that a single page might contain several recipes, so we'll have quite a few blocks of instructions on the page. The difference between `id`s and `class`es is that an `id` value can appear on a particular page only once. Using the same `id` value twice on a page makes the page invalid, so we use `class` for this `<div>` element instead.

Once we give our div a `class` value of `instructions`, something magical happens: we can make all the other `class`es we added disappear.

```
<div class="instructions">
  <h2 class="instruction-heading">Instructions</h2>
  <ol>
    <li>Preheat oven to 350 deg F (Gas mark 4 or 180 deg C).</li>
    <li>Line a 13 x 9 in (33 x 23 cm) cake tin with grease-proof ➥
      paper and grease the tin.</li>
    <li>Melt the butter.</li>
    <li>Beat eggs with sugar, and add vanilla, flour, cocoa, baking ➥
      powder, salt (optional) and melted butter.</li>
    <li>Add chopped nuts.</li>
    <li>Bake at 350 degrees until a wooden pick inserted stands up. ➥
      Should take 20-30 minutes.</li>
    <li>Cool the cake. Dust with powdered sugar.</li>
  </ol>
</div>
```

The div creates a context on the page—in essence, it's like a root element for our `instructions` compound. Typically, the only reason developers would have added all those classes we saw in the earlier example was for styling with CSS. But once we create this context, we no longer need to give `class` values to the list items, because instead of styling the list items as follows:

```
li.instruction-step {list-style-type: decimal}
```

we can style them like this, using the descendent selector of CSS:

```
.instructions li {list-style-type: decimal}
```

The limits of HTML: Why we need microformats

In a sense, div and span, in conjunction with the `class` and `id` attributes, enable developers to create their own semantic building blocks. This is, in fact, how many developers use them. It's a very powerful approach, but it also has drawbacks.

The biggest problem is that there is essentially no way of standardizing the `class` and `id` values used, so there is not much real value in doing this, because across sites there will be little or no consistency in the use of `class` and `id` values. In a technical sense, we could say that we aren't developing consistent semantics. Very often, too, if we look under the hood at the `class` and `id` values people are using, they are things like `BigBlue` or `leftCol`—that is, presentational names. It makes much more sense to name an element with `class` and `id` values according to what it is rather than what it looks like. For instance, is the `leftCol` actually site-level navigation? If so, let's give it a name that reflects that.

A detailed survey I did in late 2005 of over 1,000 websites shows what is happening with the use of `class` and `id` attributes. The survey found nearly 5,000 distinct `class` and `id` values in those 1,200 or so sites (so on average, each site invented four or more unique `class` or `id` values that were used on no other site). Only a handful of `class` or `id` values recurred more than about 5% of the time, among them `header`, `footer`, `content`, and `search`. If you look more deeply, developers are using synonymous terms, such as `header`, `logo`, `title`, `main`, and `banner`, which I suspect all refer to the same basic construct: a main title block at the top of the page. All of which emphasizes that without some form of standardization, the value of using `class` and `id` attributes to provide consistent semantics is very limited.

You can find the results of my survey at `http://westciv.typepad.com/dog_or_higher/2005/11/real_world_sema.html`. *Related results from a later Google study are here:* `http://code.google.com/webstats/2005-12/classes.html`.

But where is any consistency going to come from? It certainly doesn't appear to be emerging from common practice; the surveys just mentioned show that well enough. Do we need to wait for some standards body like the W3C to devise a standard set of `class` and `id` values? We might be waiting for a long time. While the HTML specification does have, for example, a list of possible values for the `rel` attribute, and XHTML 2.0 has a limited vocabulary associated with the `role` attribute, that's as far as it goes. In reality, is a body like the W3C going to be able to provide `class` and `id` values for every possible situation we might need? Clearly the answer is no.

Enter microformats. Microformats aim to solve the consistency problem, one step at a time, from the bottom up. Anyone can be involved in proposing or developing a microformat, and the success or otherwise of a new format doesn't depend on browser support or the standards process, but on the marketplace of ideas. If there is a need for a particular microformat, it will be adopted. The more the microformat is adopted, the more of a network effect it will have associated with it. In other words, the more content in that format, the more incentive for software developers to create tools to work with that content; the more tools there are, the more incentive web developers have to create content in that format.

Summary

I hope you can see precisely what motivates the development of microformats: filling a need that HTML by itself cannot. You should also see by now that microformats are *evolutionary*, not *revolutionary*. They utilize only valid current HTML and only use HTML appropriately, according to the HTML specification.

In this chapter, you've gone right to the foundations of modern web development and taken a long, hard look at the following topics:

- What HTML isn't (a presentational markup language)
- What HTML, underneath it all, is (a structural semantic markup language)
- How HTML is commonly misused as a presentational language, and the overuse of "correct" features like `class` and `id` attributes, and `<div>` and `<span>` elements
- How to ensure our HTML documents are valid
- Some of the more sophisticated, often overlooked features of HTML

You've also seen that eventually, HTML in itself is not enough to provide all the tools you need to publish semantically rich content for today's Web, which is where microformats come in.

Now you're ready to examine specific, widely used microformats. The principles, techniques, and features of HTML this chapter covered will stand you in very good stead as you come to grips with the nitty-gritty details of microformats.

4 LINK-BASED MICROFORMATS: REL-LICENSE, REL-TAG, REL-NOFOLLOW, AND VOTELINKS

By now I'm sure you're itching to get your hands on some actual microformats. The previous chapters covered a lot of reasonably theoretical aspects of HTML that you'll need to understand to really come to grips with microformats in practice. Now it's time to get your hands dirty.

In this chapter, you'll look at a family of microformats based on the link (`<a href="...">`) element and two particular attributes, `rel` and `rev`. All the microformats presented in this chapter are constructed simply by adding one of a set of values to the `rel` or `rev` attribute of HTML. But with just this simple pattern, you'll see that you can add quite a lot of **metadata**, or additional information about the page (e.g., license or copyright information for the page, or "tags" that describe what the page is about). You'll also take a look at how to add some style to these microformatted links using some less well-known and often underutilized features of CSS.

Let's begin by taking a look at the specific features of HTML that make all these particular microformats possible: the `rel` and `rev` attributes of the `<a>` element.

The rel and rev attributes

Both the link and anchor elements may have a `rel` and/or a `rev` attribute. These attributes were a source of some initial confusion among the members of the microformats community, who were among the first to really utilize them in a way that extends their basic use as spelled out in the HTML specification.

The `rel` attribute, according to the HTML specification, "describes the relationship *from* the current document *to* the anchor specified by the `href` attribute." If, however, we take a look at the examples in the specification, we'll find the following possible values for the `rel` attribute:

- `stylesheet`
- `next`
- `copyright`
- `bookmark`

So, it might be better to restate the `rel` attribute definition as "describes the relationship of the anchor specified by the `href` attribute to the current document." After all, `rel="stylesheet"` does not mean that the current document is a style sheet for the anchor specified by the `href` attribute—indeed, quite the opposite. Similarly, `rel="next"` does not mean that this document is the next document after the one being pointed to in a sequence, nor does `rel="copyright"` mean that this document is a copyright statement for the document the `href` points to.

What about `rev`? Well, according to the HTML specification, it describes "a reverse link from the anchor specified by the `href` attribute to the current document." Given that the two attributes are in effect defined together, `rev` is generally taken to mean the symmetric reverse of `rel`. That is, `rev` defines the *reverse* relationship to `rel`.

HTML provides a dozen or so link types or values for `rel` and `rev`, but leaves open the possibility of defining our own link types. Link microformats take advantage of this feature of HTML. Does that mean we can just "make up" values for `rel` and `rev`? Indeed we can. The HTML specification states,

Authors may wish to define additional link types not described in this specification. If they do so, they should use a profile to cite the conventions used to define the link types.

The term "should" tells us the following, according to Network Working Group RFC 2119 (see `www.ietf.org/rfc/rfc2119.txt` for more information):

There may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

So, the best way to add our own `rel` and `rev` values is to link to a profile that defines the values we wish to use. In general, however, this is considered overkill. These `rel` and `rev` microformats do illustrate just how "micro" a microformat can be. And, in the case of rel-license, the microformat consists of a single attribute, `rel`, and a single possible value, `license`.

Now it's time to delve into this family of microformats, starting with rel-license.

The rel-license microformat

In general, copyright law in most countries grants the creator (or other owner) of any material an exclusive set of rights, which can be granted to others through a license. The rise of open source software and other associated open source content, exemplified by but not restricted to the Creative Commons movement, has seen a significant increase in people licensing the use of their material to others, often without cost and with very little restriction on the ways in which that material can be used.

If you publish on the Web, it would make sense if you could somehow mark up how the material could be used by others. Enter the rel-license microformat. But before we examine rel-license, we'll first look at how to create a license for copyrighted work through Creative Commons.

Creating a Creative Commons license

If you publish material—to a blog; at presentations, conferences, or other groups; and/or on a website—you may want to make it widely usable by others. You may want to do this without restriction, meaning people can use your material in any way they wish. Alternatively, you may want to do this with some restrictions—for example, you may want attribution to appear where your work is used. Or you may even want to restrict the use of your material to noncommercial uses, and if someone wants to republish your material in a commercial product or work, he or she needs to negotiate with you for that right. The Creative Commons organization makes it very easy to create licenses for any of these situations. Creative Commons uses the slogan "some rights reserved" (as opposed to the traditional copyright notice "all rights reserved") to capture the intent of this licensing approach (see `http://creativecommons.org` for more information).

Just why you might want to license in this way, and how the different levels of licensing work, goes beyond the scope of this chapter and book, but you'll find detailed discussions of these topics at the Creative Commons site. For now, let's quickly step through the process of creating a license at Creative Commons for you to use.

When you get to the Creative Commons site, click the Publish image, as shown in Figure 4-1.

Figure 4-1. The Creative Commons Publish icon

Image used in accordance with the Creative Commons Attribution 2.5 license (http://creativecommons.org/licenses/by/2.5/).

You can then choose one of a number of common licenses, as shown in Figure 4-2.

Or you can create a customized license by selecting options from a form, as shown in Figure 4-3.

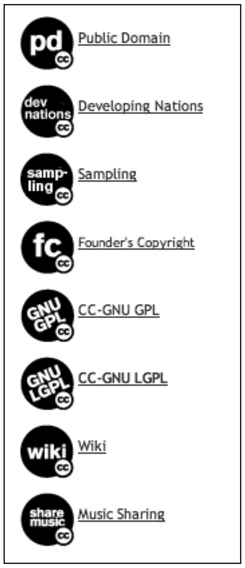
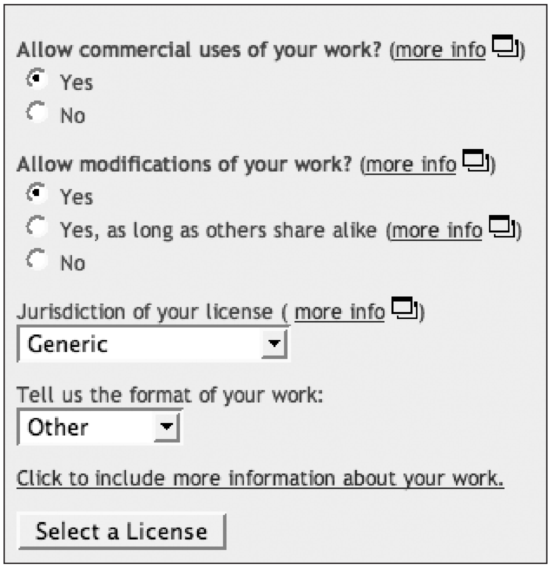


Image used in accordance with the Creative Commons Attribution 2.5 license (http://creativecommons.org/licenses/by/2.5/).

Figure 4-2. The different standard license options available on the Creative Commons site



Allow commercial uses of your work? (more info)
[✓ Yes] No

Allow modifications of your work? (more info)
[✓ Yes] Yes, as long as others share alike (more info) No

Jurisdiction of your license (more info)
Generic

Tell us the format of your work:
Other

Click to include more information about your work.

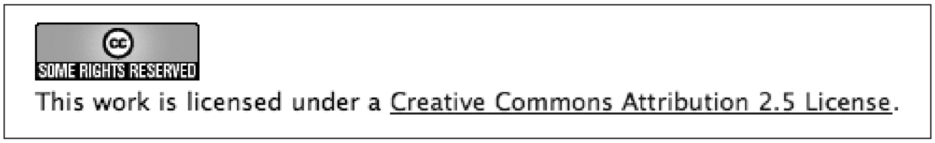
Figure 4-3. You can even create a custom license using the available form.

Image used in accordance with the Creative Commons Attribution 2.5 license (http://creativecommons.org/licenses/by/2.5/).

Click `Select a License`, and Creative Commons creates some HTML for you to add to your site, for example:

```
<!--Creative Commons License-->
<a rel="license" href="http://creativecommons.org/licenses/by/2.5/">
  <img alt="Creative Commons License" style="border-width: 0"
src="http://creativecommons.org/images/public/somerights20.png"/>
</a><br/>This work is licensed under a <a rel="license"
href="http://creativecommons.org/licenses/by/2.5/">
Creative Commons Attribution 2.5 License</a>.
<!--/Creative Commons License-->
<!-- <rdf:RDF xmlns="http://web.resource.org/cc/"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <Work rdf:about="">
    <license
rdf:resource="http://creativecommons.org/licenses/by/2.5/" />
  </Work>
  <License
rdf:about="http://creativecommons.org/licenses/by/2.5/"><permits
rdf:resource="http://web.resource.org/cc/Reproduction"/><permits
rdf:resource="http://web.resource.org/cc/Distribution"/><requires
rdf:resource="http://web.resource.org/cc/Notice"/><requires
rdf:resource="http://web.resource.org/cc/Attribution"/><permits
rdf:resource="http://web.resource.org/cc/DerivativeWorks"/></License>
</rdf:RDF> -->
```

Paste this code into the `<body>` of your document, and you are done! The result in a browser is shown in Figure 4-4.



This work is licensed under a Creative Commons Attribution 2.5 License.

Figure 4-4. The standard Creative Commons license attribution

Image used in accordance with the Creative Commons Attribution 2.5 license (`http://creativecommons.org/licenses/by/2.5/`).

Of course, you can style this attribution with CSS to make it look a little nicer. More on that later in the chapter.

Using rel-license

Let's start with a sample rel-license link. As you can see by looking at the source code in the preceding section, Creative Commons has created the link for us:

Note that you can use the rel-license microformat with any kind of license—it doesn't have to be a Creative Commons one.

```
<a href="http://creativecommons.org/licenses/by/2.5/" rel="license">
  Creative Commons Attribution version 2.5
</a>
```

The most important thing here is the `rel="license"` attribute. This attribute means (in rather tortuous language) that "the document this link points to has the relationship of being a license for this document." Or, put a little more nicely, "this link points to the license for this document."

Next, we have the URL for the license:

```
href="http://creativecommons.org/licenses/by/2.5/"
```

The text for the link is simply there for our readers. It's a description of the link, but what make this a rel-license microformat are the `rel="license"` and the `href`.

So, to add any particular license to a page, you need to know the URL for that license, and you need to use the following form:

```
<a href="url for the license" rel="license">
  A description of the license for your readers
</a>
```

If you find this approach too daunting, an extension for Dreamweaver is available that makes it even easier to create rel-licenses for your content. You can find this extension at `www.webstandards.org/action/dwtf/microformats`*.*

There's also a plug-in for the blogging tool Textpattern, pnh_mf, by HTML guru Chris Casciano, for creating rel-licenses and many other kinds of microformatted content. You can find it at `http://placenamehere.com/TXP/pnh_mf`*.*

Typically when you add a license link to a page, you are indicating that the license for the whole page is the linked license document. Some more complex microformats like hReview, which you'll see in later chapters, include rel-license as a possible component. When a review microformat includes a rel-license link inside it, then the linked license applies to the specific review, not the page as a whole. This way, you might have several reviews on the same page, each with a different license. How might this happen? It can occur if, for instance, you are aggregating reviews from different reviewers, and each has chosen a different license.

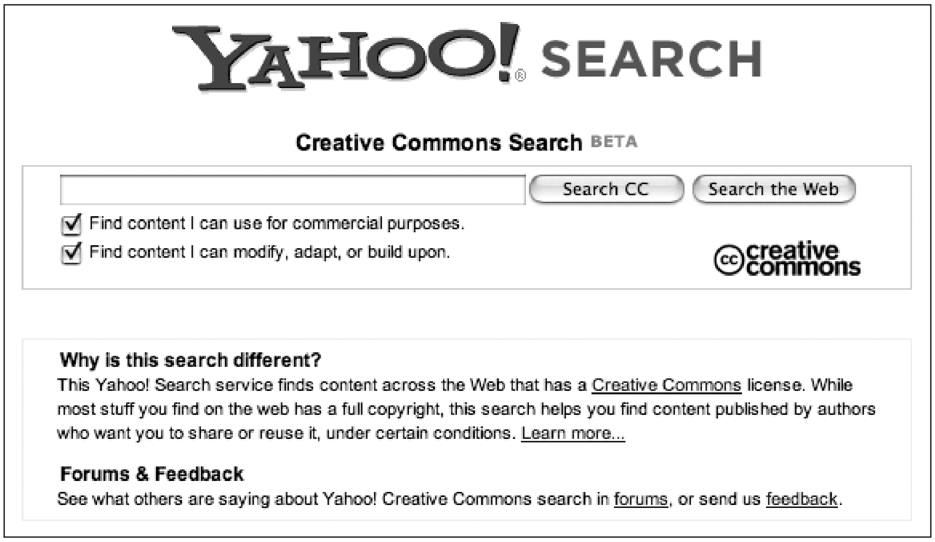
At this point you may be thinking, "What? Is it really that simple?" Yes, microformats can be and often are that simple. Going back to the discussion of the underlying principles of microformats from the first chapter, you can see that even such a simple format as rel-license embodies these principles. It's simple. It solves a specific problem. It's embeddable. Now let's see how it enables decentralized services.

Enabling decentralized services with rel-license

Now you might be thinking that the rel-license microformat is a kinda cool, in a geeky way, but really, there isn't much practical benefit for you as a developer (or for anyone else, for that matter) in using this microformat to add licensing information to your content. Or maybe you're thinking that rel-license is a nice way to make a statement about copyright, and to give a little back to the Web, but there's not much more to it than that.

Well, a couple of little companies like Google and Yahoo might disagree.

Both of these web behemoths have implemented searching for content based on its license type, as specified by the rel-license microformat. Yahoo has a dedicated Creative Commons Search feature (see Figure 4-5), which you can find at `http://search.yahoo.com/cc`.



☑ Find content I can use for commercial purposes.
☑ Find content I can modify, adapt, or build upon.

Why is this search different?
This Yahoo! Search service finds content across the Web that has a Creative Commons license. While most stuff you find on the web has a full copyright, this search helps you find content published by authors who want you to share or reuse it, under certain conditions. Learn more...

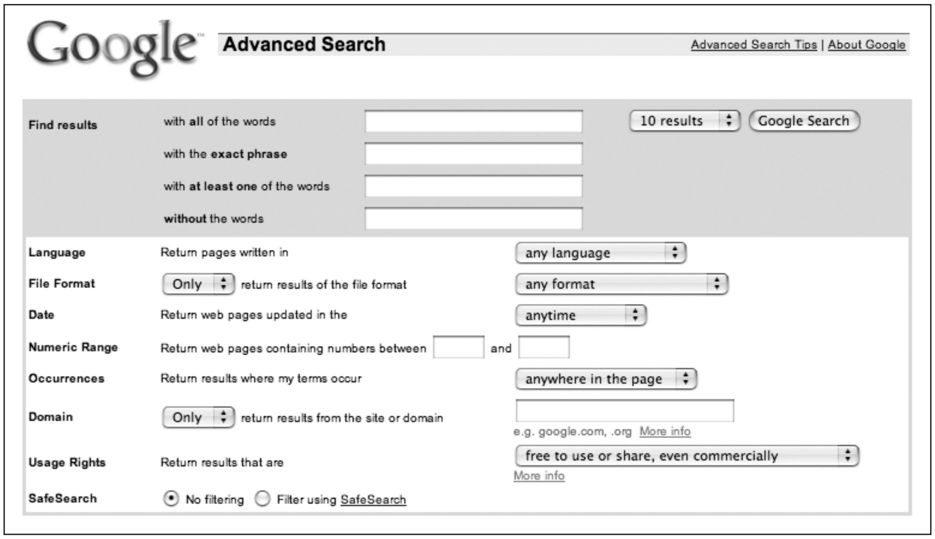
Forums & Feedback
See what others are saying about Yahoo! Creative Commons search in forums, or send us feedback.

Figure 4-5. Yahoo's Creative Commons Search page

And Google lets you specify a license type in its Advanced Search functionality (see Figure 4-6), which you can find at `www.google.com/advanced_search`.

If you develop content and then license it so that people may use, share, or modify it in various ways, by simply adding this single microformatted link, you'll be making your content known and available to a far wider audience.

This is an excellent example of how even a simple microformat can enable decentralized services. There's no need for you to register your licensed material with anyone to let others easily know that it is available, and under what conditions. You simply add the link with the `rel="license"` value to your site and let license developers like Creative Commons and search engines like Yahoo and Google take care of the rest.


Google **Advanced Search** Advanced Search Tips | About Google

Find results	with **all** of the words	[ ] 10 results · Google Search
	with the **exact phrase**	[ ]
	with **at least one** of the words	[ ]
	without the words	[ ]
Language	Return pages written in	any language
File Format	Only · return results of the file format	any format
Date	Return web pages updated in the	anytime
Numeric Range	Return web pages containing numbers between [ ] and [ ]	
Occurrences	Return results where my terms occur	anywhere in the page
Domain	Only · return results from the site or domain	[ ] e.g. google.com, .org More info
Usage Rights	Return results that are	free to use or share, even commercially · More info
SafeSearch	[✓ No filtering] Filter using SafeSearch	

Figure 4-6. Google's Advanced Search functionality allows you to specify a license type when searching for pages

Styling rel-license content with CSS

So far in this book I haven't made much mention of styling websites. While much of the focus of microformats is, of course, on content, by intelligently structuring our content, we get the ability to easily add a little style, which may aid the usability and attractiveness of the content we publish.

As the saying goes, a picture is worth a thousand words. So adding a small icon that indicates how your content is licensed will add some visual communication to the written word. Now, the obvious way to do this would be to add an image element to the HTML of your link. But you can, in fact, take advantage of CSS to do this.

If you don't know a great deal about CSS, there are many places online and lots of useful books to help you get started or to build on your existing knowledge. For nearly a decade, I've been publishing articles, tutorials, a comprehensive guide, and more about CSS (along with the Style Master CSS editor). You can find all this at my company's website: `http://westciv.com`.

In addition, two great books are available from the same publisher as this book: Simon Collison's Beginning CSS Web Development: From Novice to Professional *(Apress, ISBN: 1-59059-689-7), and Andy Budd's* CSS Mastery: Advanced Web Standards Solutions *(friends of ED, ISBN: 1-59059-614-5). You'll find these resources very useful if your taste runs to printed material.*

To demonstrate, let's add a background image to the link and use a bit of padding to make it appear that this image is part of the link. In fact, it will be part of the link, so it will be clickable. First, we'll need a selector to select only links with `rel="license"`. The standard way for designers to do this is to add a class value to the link—something like `class="license"`. But this only styles the link, and it's redundant, because we've already used the `rel` attribute to denote this as a license link.

This is a classic example of classitis—using `class` attributes when they aren't needed—because CSS provides a selector for selecting elements based on their attribute values. It's supported in modern browsers (including Internet Explorer 7), though not in Internet Explorer 6 or older for Windows. You might argue that this makes bothering to style rel-license links this way reasonably worthless, because a good many people won't be able to see the style we add, so our pages won't look the same in all browsers. But this demonstrates an important concept when working with browsers that have varying degrees of support for CSS: **progressive enhancement**. If we design with progressive enhancement in mind, users of newer browsers that support more features have a richer experience than those with older browsers. It gets us away from the outdated idea that when we design for the Web, all pages should look the same in all browsers. The important thing is that we provide all the basic information to all browsers, and it's just the enhancements like this image that some users won't see. By adopting this strategy for styling our pages, we avoid as much as possible complex hacks and redundant CSS and HTML.

The attribute selector of CSS is among the trickiest to use. It can be used in a number of different ways, but we'll focus on a specific technique here. We can use the attribute selector to select any element with *a particular value for a particular attribute*. In this case, we want a selector that will select any anchor element when it has a `rel` attribute value of `license`. We specify this using the CSS selector `a[rel="license"]`.

So now we have a selector for our license links (yes, you can have more than one license link, and this statement will select all of them). Next, we want to add a background image. For the sake of simplicity, let's suppose our image is in a directory called "images" in the same parent directory as our style sheet, and it's called `cc-logo.png`. To add it as a background image to our statement, we use the `background-image` property:

```
a[rel="license"] {
   background-image: url(images/cc-logo.png)
}
```

Currently, the background image will repeat across and down the background of the element, as shown in Figure 4-7.

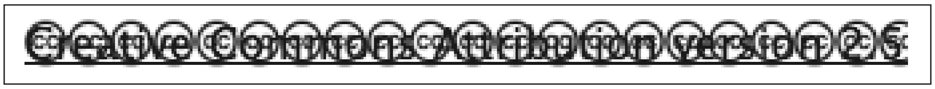


Figure 4-7. Our rel-license CSS styling doesn't look great at first.

To remedy this, we need to set the `background-repeat` to `no-repeat`, which gives us the element shown in Figure 4-8.

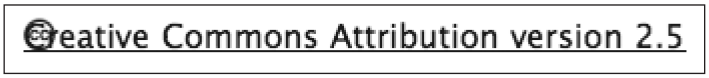


Figure 4-8. Using no-repeat, the background image displays only once, which is better, but not great.

It's common for visual cues like our icon (e.g., images that indicate a download is a PDF, a link is external to a site, etc.) to be placed to the right of the link. To relocate our icon, we'll need to set the `background-position` of the image to `right`, giving us the result in Figure 4-9.

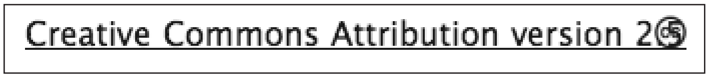


Figure 4-9. Now we've used background-position: right to make the background graphic shift over to the right . . . but it's still not quite there.

Finally, we need to ensure that the image is not covered up by the text of the element. To do this, we add `padding` to the right of the element (we use `padding` rather than `margin` because `padding` is between the content of an element and its edge, while `margin` is between the edge of an element and the edge of an adjacent element). The image is 16 pixels wide, so we'll use a `padding` of `20px` to allow for some extra whitespace between the image and the edge of the text. Our finished link is shown in Figure 4-10.

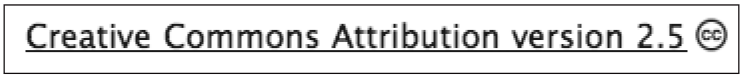


Figure 4-10. Finally, a dash of padding on the right side of our link sorts out the positioning issues—nice!

Our final statement is as follows:

```
a[rel="license"] {
  background-image: url(images/cc-logo.png);
  background-repeat: no-repeat;
  background-position: right center;
  padding-right: 20px;
}
```

All the code examples in this book are available online. The HTML and CSS for this example are available at `http://microformatique.com/book/chapter4/rel-license.zip`*.*

We'll use techniques like this throughout the book to add additional, meaningful visual cues to the information already provided within the document. It's important that the visual cues alone don't convey the information; rather, they should augment the information. This ensures we don't discriminate against certain readers, whether they have visual disabilities or use devices that don't support the features of CSS we are using here (or don't support images at all).

The rel-tag microformat

The last two or three years have seen an explosion of **tagging** on the Web. The idea behind tagging is that users or publishers spontaneously label things like images or web pages according to their personal or idiosyncratic decisions as to what those images or pages are about. Two of the best-known tag-based systems are Flickr and del.icio.us.

One of the photo-sharing service Flickr's (`www.flickr.com`) most powerful features is that as the owner of a photo, or as one of the photo owner's contacts, you can tag that photo with a label or labels that describe the image. Figure 4-11 shows an example from a recent conference I was involved with. In the case of Flickr, the tags are labels for the photo.

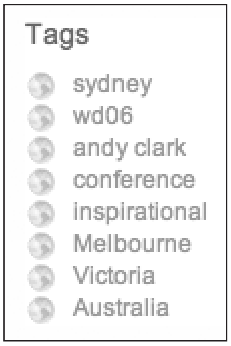


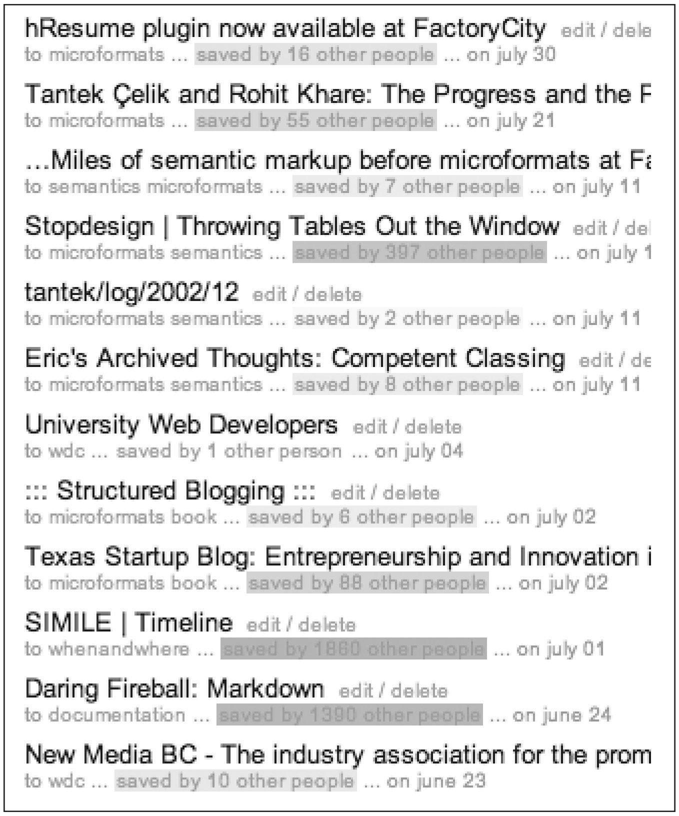
Figure 4-11. Tags for one of my images on Flickr

Reproduced with permission of Yahoo! Inc. © 2007 by Yahoo! Inc. YAHOO! and the YAHOO! logo are trademarks of Yahoo! Inc.

del.icio.us (`http://del.icio.us`), a social bookmarking site where anyone can bookmark a page and add their own labels, or tags, to classify these pages, uses a different tagging model: anyone can tag external resources, typically web pages like blog posts. For example, Figure 4-12 shows some pages I've tagged in my del.icio.us account.

The second line of each entry (starting with "to") lists the tags I have given this resource and how many others have also bookmarked it (but not necessarily with the same tag).

As you've discovered, one of the important goals of microformats is to help enable decentralized services. Flickr and del.icio.us are both centralized tagging services—the tags are entered and maintained at centralized sites. As you might guess, the rel-tag microformat exists to enable decentralized tagging for web-based content. Next, we'll take a look at how to use rel-tag microformats to tag content and then see how this works on the Web.



hResume plugin now available at FactoryCity edit / dele
to microformats ... saved by 16 other people ... on july 30

Tantek Çelik and Rohit Khare: The Progress and the F
to microformats ... saved by 55 other people ... on july 21

...Miles of semantic markup before microformats at Fa
to semantics microformats ... saved by 7 other people ... on july 11

Stopdesign | Throwing Tables Out the Window edit / de
to microformats semantics ... saved by 397 other people ... on july 1

tantek/log/2002/12 edit / delete
to microformats semantics ... saved by 2 other people ... on july 11

Eric's Archived Thoughts: Competent Classing edit / de
to microformats semantics ... saved by 8 other people ... on july 11

University Web Developers edit / delete
to wdc ... saved by 1 other person ... on july 04

::: Structured Blogging ::: edit / delete
to microformats book ... saved by 6 other people ... on july 02

Texas Startup Blog: Entrepreneurship and Innovation i
to microformats book ... saved by 88 other people ... on july 02

SIMILE | Timeline edit / delete
to whenandwhere ... saved by 1860 other people ... on july 01

Daring Fireball: Markdown edit / delete
to documentation ... saved by 1390 other people ... on june 24

New Media BC - The industry association for the prom
to wdc ... saved by 10 other people ... on june 23

Figure 4-12. Some of the pages I've tagged in my del.icio.us account

Reproduced with permission of Yahoo! Inc. © 2007 by Yahoo! Inc.
YAHOO! and the YAHOO! logo are trademarks of Yahoo! Inc.

Using rel-tag

If you think a bit about Flickr and del.icio.us, you'll realize that even though they both use the term "tagging" to describe how users label content, the process of tagging on those sites works in quite different ways. With Flickr, a tag is a label a photograph owner (typically) uses to describe his or her photo. The tag is associated with the photo, and it is added and displayed on the page where the photo appears. With del.icio.us, a tag is a link from another page, created by anyone, that describes the page linked to.

rel-tag is Flickr-like tagging—a microformat tag is used by the author of a page to describe that page. Like rel-license, rel-tag is a `rel`-based link microformat, and so you code it very similarly. You simply have a link with a URL and a `rel` value of `tag`. For example, here is how you would tag a page as being about microformats:

```
<a href="http://technorati.com/tag/microformats" rel="tag"> ➥
  microformats
</a>
```

This link says about the page containing it, “I am labeling this page with the tag ‘microformats’.” In fact, to be more specific, the tag may apply to a page or part of a page. So, if you have several blog posts on a page, each may have its own tag or tags.

Tagging with rel-tag, then, is different from tagging in del.icio.us. Tagging systems like del.icio.us let anyone tag a page on the Web. In del.icio.us, users tag other resources, in effect saying, “I am labeling the page found at the URL `http://microformatique` with the tag ‘microformats.’”

> *There is, in fact, a microformat called xFolk that enables tagging of the destination of a link, much like del.icio.us tagging, but you use it to do so on your own pages, not at a centralized site. xFolk is not yet widely adopted, so I don’t cover it in detail in this book, but you can find more information about it here:* `http://microformats.org/wiki/xfolk`.

Let’s take a closer look at the URL value of the `href` attribute. What does it actually point to? Is that URL significant? The destination of the link is termed a **tag space**, and the final part of the link after the last forward slash (/) in the URL is the **tag value**, which makes it very easy to extract the value of the tag from the URL. The destination of the link *should* exist (in technical terms, this means that it does not necessarily have to, but there should be a good reason if it doesn’t).

Note that the actual text of the link is not the value of the rel-tag microformat, and while it makes sense that it should usually be the same as the value as the microformat, this is not required. Why not require the text of the link to match the value of the tag? One reason is accessibility. Screen readers help people read pages by presenting the content of links to the reader in a list. A single word value is often not nearly as useful as a phrase. Another reason for the link text not necessarily matching the tag value is for internationalization purposes. If you are tagging a page in a language other than English, you can use the same tag space, despite the language being used, and have the content of the link in the relevant language. Here’s a simple example:

```
<a href="/tagspace/house" rel="tag" lang="es">casa</a>
```

Here the page is tagged with “house,” but because the page is in Spanish, the Spanish word for house, “casa,” is used.

A moment ago, I said that the destination of the link is a “tag space”—but what exactly is that? The rel-tag specification describes a tag space as “a place that collates or defines tags.” There is no requirement to point to a particular tag space, such as Technorati’s. You may create your own tag space or use, say, Wikipedia as a tag space (so when the link is followed, the reader gets further reading on the related subject). It doesn’t matter that people use different tag spaces because again, the specification of rel-tag makes it clear that it is only the tag value (the part after the / in the `href` value) that matters. So even if different developers point to different tag spaces, tag-based search engines can identify pages tagged with the same label.

rel-tag vs. the meta element

"Hold on," I hear some readers say. "HTML provides a framework for adding keywords that describe the content of a page, using the meta element and its `keyword` attribute. Shouldn't we be using this feature of HTML instead of inventing a new way? How does using the rel-tag differ from simply using the meta element to add keywords for the page like this?"

```
<meta name="keywords" content="microformats">
```

There are a several significant problems with the meta element. First, it's hidden—when you visit a page, you don't see its keyword values unless you inspect the source of the page. This has two consequences. The first is that the value of those keywords may become incorrect over time, as pages are edited or changed. The second is that there is, in effect, no accountability. The meta element can easily be used to make false assertions about the content of the page, which readers don't see but search engines do.

In fact, the second consequence just noted has happened so routinely with people trying to game search engines that major search engines do not take meta element keywords into account when indexing pages. See the article "Death of a Meta Tag" by Danny Sullivan at `http://searchenginewatch.com/sereport/article.php/2165061` *for more information about this.*

Meta elements also suffer from being page-specific, so that for pages with multiple pieces of content about quite different subject areas, meta elements aren't nearly as useful as rel-tag links; unlike meta keywords, rel-tag links can tag parts of a page. For blog posts this is particularly relevant, as different posts on a blog often cover a wide range of subjects, and unless each post appears on a single page, meta keywords would have to be updated for each post.

Ultimately, rel-tag illustrates an important aspect of microformats. Microformats try to capture emergent behavior—that is, *what developers are actually doing*. Meta keywords, through both their abuse and lack of use, are in effect dead on the Web (though they are of considerably greater use on intranets and with internal search engines). Yet we still need to add meta-level descriptions about content on the Web, as evidenced by the popularity of tagging in services like del.icio.us and Flickr. rel-tag exists to enable this tagging of content in a distributed fashion. And as you'll see in a moment, it's extraordinarily popular.

Benefits of using rel-tag: Technorati tags

Web developers, and in particular bloggers, are amazingly practical people. They tend to do things that work and have perceived value, and ignore things that they view as having little practical application in their lives, despite how theoretically useful these things may be. They also tend to be a bit faddish, too, but we'll forgive them that. At times this focus on the practical has drawbacks—it has been a long, hard road to convince many developers to adopt development practices that make web content more accessible, because by and large developers are young, able-bodied people with little exposure to or familiarity with disabilities.

When it comes to tagging using the rel-tag, adoption has been extremely—indeed astonishingly—rapid. Introduced only at the beginning of 2005, Technorati was, within 18 months, tracking 100 million tagged blog posts. Clearly this is something a lot of people are doing. What's particularly interesting is that by doing so, web developers and bloggers have developed a bottom-up way of classifying information, termed **folksonomies** by Thomas Vander Wal, which is distinct from the traditional top-down approach of taxonomy.

Unfortunately, I don't have space here to go into detail about folksonomies, but for a great introduction and overview, see Bruce Sterling's Wired Magazine *article titled "Order Out of Chaos" at* `www.wired.com/wired/archive/13.04/view.html?pg=4`.

But why are millions of people adding tags to their web content? Let's take a look at the benefits to developers of tagging their content with Technorati's tag search feature, and keep in mind that it's surely only a matter of time before the big-name search engines start taking advantage of this extra information as part of their indexing process.

Technorati, in addition to providing search features based on keywords (the kind of searching we've been doing for years on the Web), provides searching by tags, using the rel-tag values that developers have added to their pages to return matching pages. Technorati even lets you subscribe to results using RSS, so users can track pages that have a particular tag from all over the Web. As the number of blogs and other "real-time" information sources (such as online news outlets) increases, the idea of subscribing to these sources individually makes increasingly less sense. Subscribing to particular tags, regardless of which sites or blogs they appear in, makes increasingly more sense, as it keeps us abreast of issues, events, and people (typically ourselves) of interest, regardless of where that information is published and even if we have never heard of the publishers before.

As developers, we've seen that costs are very low for tagging content—all we need to do is add rel-tagged links and let the Technorati search engine know about the new content (it's not just for blog posts either; any HTML-based web resource can be tagged). Then the content will be available to anyone who uses Technorati tag searching, or who subscribes to a tag via Technorati.

Tools for tagging

Several blogging tools have plug-in support for rel-tag. This can help take any lingering tedium out of the task of creating the link to a particular tag space.

For WordPress:

- SimpleTags: www.broobles.com/scripts/simpletags/
- Bunny's Technorati Tags: http://dev.wp-plugins.org/wiki/BunnysTechnoratiTags

For Textpattern:

- pnh_mf: http://placenamehere.com/TXP/pnh_mf (mentioned earlier in the rel-license discussion)
- tru_tags: www.truist.com/blog/493/trutags

For Bloxsom:

- Tagging plug-in from Axel Beckert: http://noone.org/blog/tags/Tagging

For Dreamweaver:

- Drew McLellan's WaSP Dreamweaver microformats extension allows you to easily add tags to any web page you are working on: www.webstandards.org/action/dwtf/microformats

Styling rel-tag content with CSS

Typically, developers add tags at the end of a blog post or page, in lists that look something like this:

Tags: microformats folksonomy tags technorati

Each word in the list is a rel-tag link. It's quite possible to have your links as part of the main content of a page, using a rel-tag link at the first occurrence of your tag word.

We might even use a similar technique to the one we used with rel-license links, and add an icon as a background image to the link. Recently, Chris Messina (`http://factoryjoe.com/blog`) proposed a number of icons for various microformats, including one for rel-tag content. We'll style our tagged links with this.

To add an image to our rel-tag links, we'll use a very similar selector and set of properties for our style sheet. The selector has the same pattern as our rel-license selector, but now we are selecting a link with the `rel` attribute value of tag, so our selector will be

```
a[rel="tag"]
```

Again, we'll add a background image, this time with the name `tag-icon.png`, but all the other properties are more or less identical:

```
a[rel="tag"] {
  background-image: url(images/tag-icon.png);
  background-repeat: no-repeat;
  background-position: center left;
  padding-left: 32px;
}
```

We make sure the image doesn't repeat and is aligned to the left of the background of our link, and we add some padding to make sure that the text doesn't cover up the background image. Figure 4-13 shows how our microformats link will look.

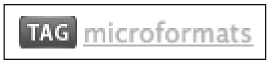


Figure 4-13. Our styled tag, complete with background TAG tag

If we have more than one tag, they will look like Figure 4-14, where our two tag links are as follows:

```
<a href="http://technorati.com/tag/microformats" rel="tag">
microformats</a>,
<a href="http://technorati.com/tag/semantics" rel="tag">semantics</a>
```

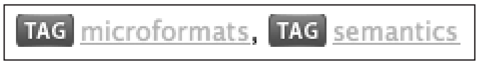


Figure 4-14. A couple of tags in a row

The last thing we'll do is take a cue from Tantek Çelik, who suggests that tag links be green as a visual cue (see `http://tantek.com/log/2005/06.html#d03t2359`). We simply need to add a `color` value of `green` (or, to closely match the microformats icon, `#a8c90b`):

```
a[rel="tag"] {
  background-image: url(images/tag-icon.png);
  background-repeat: no-repeat;
  background-position: center left;
  padding-left: 32px;
  color: #a8c90b;
}
```

Something else we might want to do to use structured semantic HTML even more appropriately is identify that this is in fact a list. We can make the HTML a list like so:

```
<ul>
  <li><a href="http://technorati.com/tag/microformats" rel="tag">
microformats</a></li>
  <li><a href="http://technorati.com/tag/semantics" rel="tag">
semantics</a></li>
</ul>
```

Of course, we'll need to add a little CSS trickery to make these items display in a line, rather than displaying as list items, as they will by default. To do this, we change the `display` type of the `<li>` elements to `inline`, like so:

```
li {display: inline}
```

which gives us more or less what we want. The only other problem with this approach is that if other list items are on the page, these too will appear inline, which may well not be what we want. Unfortunately, CSS doesn't provide a "parent" selector, or we could select list items only when they were the parent of links with a `rel` value of `tag`. So here is one place where, if we want to use styling, we'll have to add to our HTML. We could add a class to each list item here, but it would be better to add a single class value to the parent ordered list. So we add a class of tags to this <ul> element:

```
<ul class="tags">
  <li><a href="http://technorati.com/tag/microformats" rel="tag">
microformats</a></li>
  <li><a href="http://technorati.com/tag/semantics" rel="tag">
semantics</a></li>
</ul>
```

Then we use a descendent selector to style only the list items inside lists with this class:

```
ul.tags li {display: inline}
```

You'll see later in the chapter how we could even add in commas between the tags by using the `content` property of CSS.

> *You can grab the HTML, CSS, and image for this design from* `http://microformatique.com/book/chapter4/rel-tag.zip`*.*

So far we've examined two straightforward, widely used, and useful `rel`-based link microformats. Next we'll take a look at a new link microformat that uses the `rev` attribute and can take more than a single value: VoteLinks, along with a related, but simpler (and somewhat controversial) microformat, rel-nofollow.

The rel-nofollow and VoteLinks microformats

The revolutionary system that Google employs to make its results so relevant, **PageRank**, uses links to a site as an indicator of the site's popularity. The more links to a site (and the relative popularity of the sites those links come from), the higher the ranking of that site.

But there is a drawback to this approach: what if you link to a site that you violently disagree with—perhaps a site that rips off a design of yours? Google treats this as a sign of popularity. So we want a way of linking to such sites that does not give them "Google juice."

Two microformats address this issue—one very simply (rel-nofollow) and one in a way that leads to other interesting possibilities (VoteLinks). Both are discussed in the sections that follow.

rel-nofollow

rel-nofollow was introduced by Google to address precisely the problem that any link to a site, even a link that is highly critical of that site, is considered by Google to be an endorsement and adds to that site's PageRank. When rel-nofollow was introduced, it was immediately criticized, in terms of both implementation and implication. The main criticism was that indiscriminate use of rel-nofollow (as was common in many blogging engines, because comment spammers could leave comments with links to their sites to gain PageRank) meant that legitimate sites lost PageRank credit they might otherwise have received, creating potentially a two-tier system that further entrenched the currently popular sites. "rel-nofollow" is also something of a misnomer, as it does not tell a search engine not to follow a link, but rather not to give any PageRank credit for the link. The term "nofollow" comes from the possible values for the robots' meta element, and so was used by analogy. Ironically, the value `noindex` would arguably have been more accurate.

For more on the rel-nofollow issues, see the Wikipedia entries on rel-nofollow and PageRank at `http://en.wikipedia.org/wiki/Nofollow` *and* `http://en.wikipedia.org/wiki/PageRank`*, respectively. And for a particularly strong criticism of rel-nofollow, visit the NoNoFollow website at* `www.nonofollow.net`*.*

As with the other `rel`-based link microformats discussed in this book, all we need to do is add a single `rel` value to our links, `rel="nofollow"`, and search engines like Google will no longer count this link as going toward the PageRank of the linked page. For example, to add a link to a page critical of an article I wrote, while not giving it any extra PageRank credit, I'd link like this:

```
<a href="http://westciv.typepad.com/dog_or_higher/ ➥
  2006/09/free_trade_agre.html" rel="nofollow">
    John Allsopp's illogical rant about free trade
</a>.
```

VoteLinks

Links to other sites are often endorsements (votes for) or criticisms (votes against) of those pages or the product, person, philosophy, or whatever those pages represent. While humans can readily determine from reading a web page whether the author of that page might be critical or approving of something, it's much more difficult for software to determine this.

But what if we could encode in our links whether they were an endorsement or criticism of the destination? Not only could search engines weigh the value of these links differently (perhaps subtracting some link weight for votes against, or giving no weight to negative vote links), but also it would be much easier for search engines to determine whether a particular movie, person, idea, or product was popular or unpopular on the Web—the very idea discussed at the beginning of Chapter 1.

VoteLinks are, according to Brian Suda, "Romanesque." The gladiatorial battles of classical Rome, despite popular belief, did not typically end in the death of combatants, who were often, after all, valuable slaves. When a gladiator was at the mercy of his or her opponent (there were women gladiators), the crowd would signify by thumbs up or down whether the felled gladiator should be spared. Many voting systems also work in this way (e.g., the "first past the post"/"winner take all" voting systems of the United Kingdom, Canada, and the United States, among other countries), and VoteLinks exist to mark up precisely this kind of voting on the Web.

Using VoteLinks

Earlier you learned about the `rel` and `rev` attributes of HTML, and how they describe the relationships between web documents via link or anchor elements. You saw that `rel` "describes the relationship of the anchor specified by the `href` attribute to the current document," while `rev` describes the reverse of this (i.e., the relationship of the current document to the linked document).

Now, if you are going to use a link to vote for or against a page you link to, you are describing the relationship of this page (or part of it) to the destination. So the `rev` attribute, rather than the `rel` attribute, is appropriate for this kind of link. In the original VoteLinks specification, the `rel` attribute was used, but this has subsequently been changed.

VoteLinks is different in still another way from the other `rel`-based microformats: there are three possible values for the `rev` attribute, rather than the single `rel` values of rel-license and rel-tag.

- rev="vote-for"
- rev="vote-against"
- rev="vote-abstain"

So, to say you are in favor of the rel-nofollow microformat, use the following link:

```
<a href="http://microformats.org/wiki/rel-nofollow" rev="vote-for">
  the nofollow microformat roxor
</a>
```

To vote against the idea, use the following link:

```
<a href="http://microformats.org/wiki/rel-nofollow" rev="vote-against">
  the nofollow microformat suxor
</a>
```

And to abstain from voting (or sit on the fence), use this link:

```
<a href="http://microformats.org/wiki/rel-nofollow" rev="vote-abstain">
  I can see pros and cons with the no-follow microformat
</a>
```

Benefits of using VoteLinks

As you saw with rel-tag and rel-license, search engines and other services are already using these microformats, so there are some really good reasons to start using them right now. VoteLinks is not widely adopted by search engines and other web services, so there is not quite the same immediate benefit. Why adopt this microformat, then?

Adopting a particular microformat can be a classic "chicken or egg" situation. If few people use a microformat like VoteLinks on their sites, then there is little or no incentive for search engines or other services to look for votes online. If no one is looking for the content, then why mark it up as a VoteLink?

I'd suggest that if VoteLinks is a relevant semantic way of marking up content that is already on your sites, then given that the cost of using the microformat is so low (simply adding a `rev` value), there really is no reason not to implement it. It seems such a simple way to make opinions easily known to software that I'd be surprised if it isn't ultimately widely adopted.

Styling VoteLinks content with CSS

One way the VoteLinks microformat is useful right now is in enabling styling of content, much like you saw with rel-license and rel-tag. In this section's example, we'll add thumbs-up and thumbs-down icons to vote for and vote against links. We need two different statements, one for each `rev` value, so we'll use these two different selectors:

```
a[rev="vote-for"]
a[rev="vote-against"]
```

This time we achieve a similar effect as with rel-license, but use a very different technique. This technique is supported in modern browsers (Firefox and other Mozilla-based browsers, Safari, and Opera, though sadly not Internet Explorer 7). The best bit is that the browsers that don't support this technique also don't support the attribute selector, so we don't need to worry about those browsers at all. The link will be still visible as a link, but the extra styling we add will be visible only to the users of contemporary browsers—another example of progressive enhancement.

Instead of adding a background image, we use the **generated content** feature of CSS2 to add an image after an element. Generated content works with a special kind of selector. We append `:before` or `:after` (much like the `:link` or `:hover` pseudoclass selectors) to any other selector. So if we want to add something to the end of an element with a `rev` value of `vote-for`, we append `:after` to the selector for that element, giving us this:

```
a[rev="vote-for"]:after
```

Similarly, to add content after a link with a `rev` value of `vote-against`, we use this selector:

```
a[rev="vote-against"]:after
```

Next, we add the image as content. To do this, we use the `content` property of CSS2. This property can get quite complicated, but here we'll use it quite simply. We can add a `URL` value to the `content` property to indicate an external resource (typically an image) to be displayed after the element. So the CSS rule

```
a[rev="vote-for"]:after {
  content : url("images/vote-for-logo.gif")
}
```

will add the image after links with a `rev` value of `vote-for`. The image is actually added as part of the link element, so the underline or any other styling for the link element also affects the image, and the image is clickable. So, for example, if I am voting for Buffy the Vampire Slayer over Neo from *The Matrix* in a death match, I use this markup:

```
a href="http://en.wikipedia.org/wiki/Buffy_the_vampire_slayer"
 rev="vote-for">Buffy the Vampire Slayer</a>
```

The result looks something like Figure 4-15 when styled with CSS.

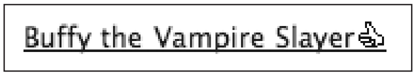


Figure 4-15. Our initial VoteLinks `vote-for` styling.
I've given Buffy the Vampire Slayer a big thumbs up!

We can also style the generated image "element" using the selector—for example, we can give it a different background color from the link element, or even give it a background image. In this case, if we give the link element itself some padding, then it will in fact be added to the right of the added image, so we'll have to actually add `margin` to the added image to create some spacing between the image and the link text. Instead of using pixels to account for the width of the image, we can use the more recommended `em` unit. This way, as the size of the font of the link text increases, the space between the image and the text grows and shrinks in proportion to that text size. This is in contrast to the previous examples, where we used pixels to create this whitespace, so the space between the text and the image would remain the same, regardless of how large or small the text was zoomed.

```
a[rev="vote-for"]:after {
content : url(images/vote-for-logo.gif);
margin-left: .5em
}
```

The result is shown in Figure 4-16.

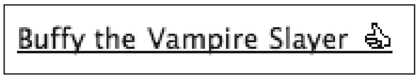


Figure 4-16. Using a bit of `margin-left` to
space out the link a bit makes it look better.

In Figure 4-16, we've added spacing between the left edge of the added image element and the right edge of the link element of `.5em`, or roughly half the width of a letter "m" at the current font size for the link element. We could also use `padding-left` on the added image or `padding-right` on the link element. The difference between these is as follows:

- When using `margin` between the elements, the background color you see between the text and the image is the background color of the first ancestor element with an explicit background color set. This may cause a visual separation between the link text and the image, breaking down the value of the image to communicate additional information about the link.
- When using `padding`, the background color, which shows between the elements, is the background color of the element that has the padding on it.

So, as a picture tells ~2^10 words, Figure 4-17 shows this in action.

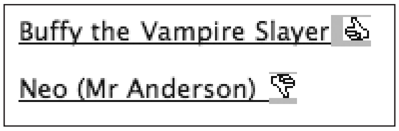


Figure 4-17. Our completed vote-for and vote-against VoteLinks. I knew Buffy would fare better in a fight against Neo.

Both the links are in paragraphs, like this:

```
<p>
  <a href="http://en.wikipedia.org/wiki/Buffy_the_vampire_slayer"
rev="vote-for">Buffy the Vampire Slayer
</a>
</p>
<p>
  <a href="http://en.wikipedia.org/wiki/Neo_%28The_Matrix%29"
 rev="vote-against">Neo (Mr Anderson)
</a>
</p>
```

We give paragraphs a white background in our CSS:

```
p {background-color: #fff}
```

Now, for the vote-for link, we have a padding-left on the image, which we added after the link:

```
a[rev="vote-for"]:after {
  content : url(images/thumbs-up.png);
  padding-left: .5em;
  background-color: #bbb;
}
```

Because padding is between the edge of an element and its content, this shows as the gray background color.

For the vote-against link, we have a margin-left on the image that we added after the link:

```
a[rev="vote-against"]:after {
  content : url(images/thumbs-down.png);
  margin-left: .5em;
  background-color: #bbb;
}
```

Because the margin of an element is between its edge and the edge of the adjacent element, in this case the paragraph, the gray background doesn't show; rather, the white background of the adjacent paragraph shows.

You can download the source HTML, CSS, and images for this example from `http://microformatique.com/book/chapter4/votelinks.zip`*.*

Summary

In this chapter, you came to grips with the rel-license, rel-tag, rel-nofollow, and VoteLinks microformats, and you saw how they all simply use the `rel` and `rev` attributes of the anchor element. We refer to these as **link** microformats. These link microformats together with a number of other simple microformats (some of them link-based, like XFN, which you'll see in the next chapter, and others like XOXO) built from other standard HTML elements are collectively referred to as **elemental** microformats. Later in the book we'll turn our attention to more complex microformats, often referred to as **compound** microformats.

Though all the microformats you've encountered in this chapter are conceptually simple and easy to use, you've seen that they can bring immediate significant benefits to web-based content. If you cast your mind back to Chapter 1 and the microformats principles, you might recall that one of those principles was that microformats be modular and embeddable. You'll see some of these kinds of microformats (in particular, rel-tag) being used again as parts of other, more complex microformats, demonstrating this modularity and embeddability in action.

But before turning to these more complex compound microformats, in the next chapter you'll look at the original microformat, XFN (XHTML Friends Network). XFN, like the microformats you've examined in this chapter, is an elemental link format, but it is more complex again than any of the formats covered so far.

5 MICROFORMAT TO DESCRIBE RELATIONSHIPS BETWEEN PEOPLE: XFN

As you progress through the book, the microformats you encounter will become increasingly more complex. In the last chapter, you learned about link microformats, and their use of the `rel` and `rev` attributes of HTML. The chapter ended with a discussion of VoteLinks, which, unlike the previous formats you saw, allows multiple values on the `rev` attribute.

This chapter looks at another, even more complex link microformat: XFN, or XHTML Friends Network. I start out with an overview of XFN before moving on to examine how to use it. You'll then see some examples of XFN in action and how to style XFN content with CSS.

XFN overview

XHTML Friends Network (XFN) is a microformat that in fact predates microformats. It is the original microformat, legendarily conjured up by Eric Meyer, Matt Mullenwegg, and Tantek Çelik in the corridors at the South by Southwest Interactive (SxSWi) conference in 2004.

XFN grew out of the observation that blogrolls (the list of other blogs that bloggers read frequently) are in fact a list of all kinds of relationships with the people publishing those other blogs. XFN was developed as a simple way of expressing in HTML the relationships between people—for example, friendships, romantic relationships, and work or kinship relationships.

While more complex than the formats you've seen so far in this book, XFN is still a very simple format, using only a small number of permitted `rel` values to express the relationship between the person represented by the page linked to and the person represented by the page linked from.

For example, Brian Suda is the technical reviewer of this book. That makes him my colleague. Though we've never (quite) met in person, we chat frequently about technical stuff and about more lighthearted things, too. In my book, this makes Brian a friend (you'll see soon that XFN defines the relationships you can encode using it). So, when I link to Brian's site, I want to be able to mark up that he is a friend and a colleague. In XFN, I'd do that like this:

```
<a href="http://suda.co.uk/" rel="friend colleague">Brian Suda</a>
```

The editor of this book is Chris Mills. I've met Chris and consider him a friend, and of course he is a colleague, so I'd mark up a link to his site like this:

```
<a href="http://www.friendsofed.com/bloggED/" ➥
rel="friend met colleague">Chris Mills</a>
```

By the way, if you are scratching your head about `rel="friend met colleague"`, then rest assured, it's quite acceptable to have space-separated words as the value of the `rel`, as well as `rev`, `class`, and other HTML attributes. This feature will soon come in handy when using the `class` attribute.

Conceptually, Figure 5-1 shows what my XFN links are saying.

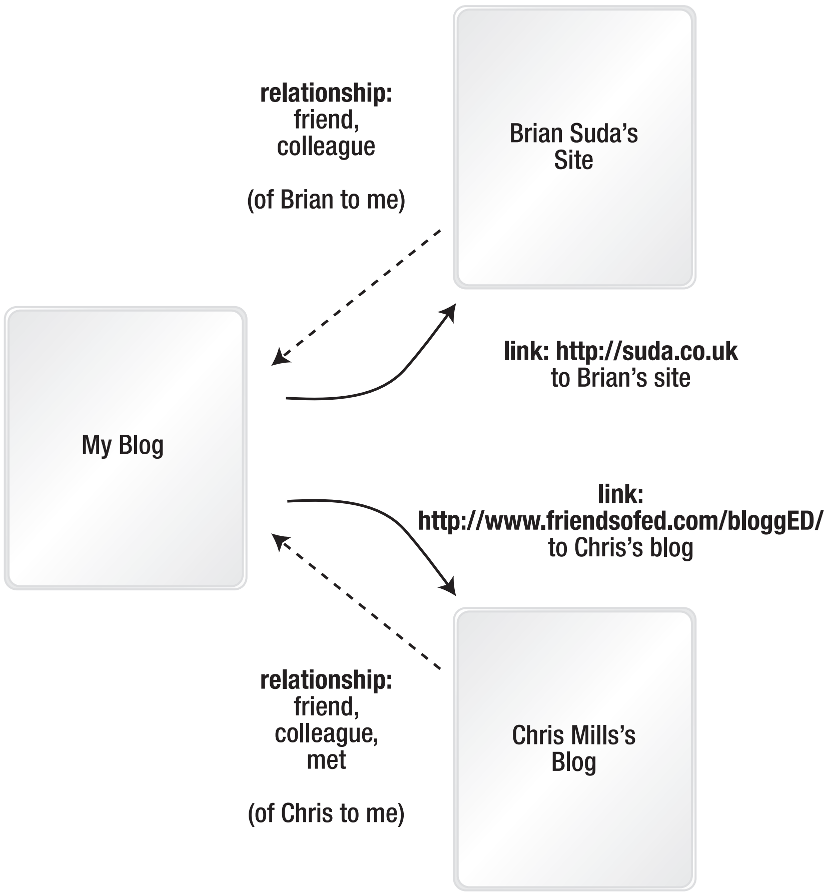

relationship: friend, colleague
(of Brian to me)
Brian Suda's Site
link: http://suda.co.uk to Brian's site
My Blog
link: http://www.friendsofed.com/bloggED/ to Chris's blog
relationship: friend, colleague, met
(of Chris to me)
Chris Mills's Blog

Figure 5-1. A summary of the XFN relationships I have defined between Brian, Chris, and me

The solid lines represent the links from my blog (representing me) to the sites representing Chris and Brian. The dotted lines represent the relationships between Chris and me, and Brian and me, as marked up using the `rel` attribute and XFN values. If you find yourself getting the direction of `rev` and `rel` mixed up (trust me, almost everyone does), then remember it is `rel="stylesheet"`, and it's the destination of the link that is a style sheet for the page with the link, not the other way around.

Recall from the original microformats principles that one important principle when developing a microformat is to *solve a specific problem*. XFN does precisely this. At the time XFN was developed, many bloggers were adding specific formatting to their blogrolls. Two or three years ago, for example, a blogger might have added an asterisk (*) to indicate he or she had met someone. Given that this practice was in existence, it made sense to formalize it, which is something microformats focuses on doing.

XFN also follows the principle of reusing existing practices by utilizing the HTML `rel` attribute, which, as you've already seen, expresses the relationship between pages. In this case, we extend the idea to have pages represent the people who develop them. There have been some criticisms of XFN along the lines that since `rel` is about the relationship between pages, it doesn't make sense to have it represent the relationship between people, even if these people are closely related to the pages themselves. However, this understanding of a page representing a person was already in use in the blogging world, so XFN, rather than inventing the concept, simply formalized it and provided a means of marking it up in HTML.

XFN relationships

As with the other `rel`-based microformats you've seen, XFN uses a specific set of values (or **controlled vocabulary**) for the `rel` attribute. The possible values fall into seven categories, with definitions as follows:

- **Friendship**
  - **Friend**: Someone you are a friend to. (I have a feeling the developers of XFN kept this one vague, as what constitutes a "friend" will vary perhaps significantly from person to person.)
  - **Acquaintance**: Someone you have exchanged greetings with and not much (if anything) more—maybe a short conversation or two.
  - **Contact**: Someone you know how to get in touch with.
- **Physical**
  - **Met**: Someone you have actually met in person.
- **Professional**
  - **Co-worker**: Someone you work with or who works at the same organization as you.
  - **Colleague**: Someone in the your same field of study/activity.
- **Geographical**
  - **Co-resident**: Someone you share a street address with.
  - **Neighbor**: Someone who lives nearby, perhaps only at an adjacent street address or doorway. (Like "friend," "neighbor" is a more fluid concept, varying from context to context.)
- **Family**
  - **Child**: Your genetic offspring, or someone you have adopted and take care of.
  - **Parent**: Your biological (or adopted) mother or father.
  - **Sibling**: Someone you share a parent with.
  - **Spouse**: Someone you are married to.
  - **Kin**: A relative, or someone you consider part of your extended family who doesn't fit into any of the preceding criteria.

- **Romantic**
  - **Muse**: Someone who brings you inspiration.
  - **Crush**: Someone you have a crush on.
  - **Date**: Someone you are dating.
  - **Sweetheart**: Someone with whom you are intimate and at least somewhat committed.
- **Identity**
  - **Me**: A link to yourself at a different URL.

And by the way, for all you social networking junkies, you haven't physically met the people you know in Second Life or World of Warcraft, or on LinkedIn, unless you have met them in the physical world, too. `rel="met"` *is only for people you have physically met.*

For in-depth details of each possible value, see Appendix A. For now, let's take a look at a couple of inherent XFN design philosophies.

First, notice that there is no "enemy" or other negative relationship; these types of relationships were deliberately omitted by the designers of XFN. The XFN designers remark in the specification that "there is enough hatred in the world. We should work to eliminate hatred, not to spread it," which is indeed laudable. But it does bring to mind the suggestions that the fylfot character (a character similar to the swastika, widely found in Eastern texts—often Buddhist ones) be omitted from Unicode, because of its negative association. In this case, would it make more sense to have people expressly state their negative as well as positive relationships with people? Would this transparency have positive as well as negative implications and effects? And perhaps more important, to what extent should values be embedded in technologies, particularly information technologies? Should the developers of various microformats preclude by license their use by particular governments, particular companies, or even specific people? How about restricting particular uses? These are, to me at least, important and very interesting issues, but of course, I've long since gone beyond the scope of this book.

Criticisms have been leveled at XFN that, in essence, the relationships XFN allows us to model are such a small sample of the broad range of human relationships, that it's an exercise of little lasting value. Specific criticisms have been aimed at particular values, such as "date," with the objection that the term "date" does not describe someone you are dating. Similarly, "spouse" is defined in terms of "marriage," relegating nonmarriage relationships to the status of "sweetheart," itself defined in terms of a lesser commitment than spouse ("at least somewhat"), ignoring same-sex relationships, which in many parts of the world cannot be "strengthened" by marriage, and also ignoring relationships between people who for whatever reason actively choose not to have a marriage. There is, of course, no reason why, over time, XFN might not grow to encompass other forms of relationships.

You might also notice that there are no gender-specific descriptions. XFN uses "sibling," not "brother" or "sister"; "spouse," not "wife" or "husband"; and so on. The specification observes that the description of the relationships is based on the perspective of the person making the description, and describes the relationship, *not the person*—hence there is no reference to gender, for example. If XFN introduced descriptions of the people involved in these relationships, as well as the relationships themselves, it would become a considerably more complex entity. People also often ask, "Why is there no 'ex-spouse,' 'ex-friend,' or similar possible values?" XFN is designed to express information about relationships at the time it is published. So be careful about what you say, and be ready to edit when you marry your sweetheart or fall out with a friend.

One last thing to observe here, which we'll look into shortly, is `rel="me"`. This is different from all the other possible values, which describe relationships with other people. Why would we need a way of saying "I am me"? `rel="me"` has a particular use: it can be used to establish that two or more sites or pages (e.g., your profile page at claimID or a social software site like LinkedIn) are associated with the same person. A number of online services use `rel="me"` to allow you to claim things like the blog you write.

XFN in action

Now that you've seen other link microformats in the previous chapter, new link microformats like XFN become very straightforward to understand and use. To use XFN, you simply add one or more of the `rel` values listed previously as the value of a `rel` attribute in a link to another web page. You can use more than one value, for example:

```
<a href="http://marxandmarzipan.com" rel="friend met colleague ➥
  neighbor">Maxine, the co-founder of westciv</a>
```

This indicates that this person is, well, a friend, someone I've met, a colleague, and a neighbor.

While the markup of XFN is really quite straightforward, remembering and entering various values can get tedious. In this section, we look at XFN-compatible tools and how to make XFN connections more efficiently than by hand-coding, and we also cover how XFN is being used in search engines, products, and applications.

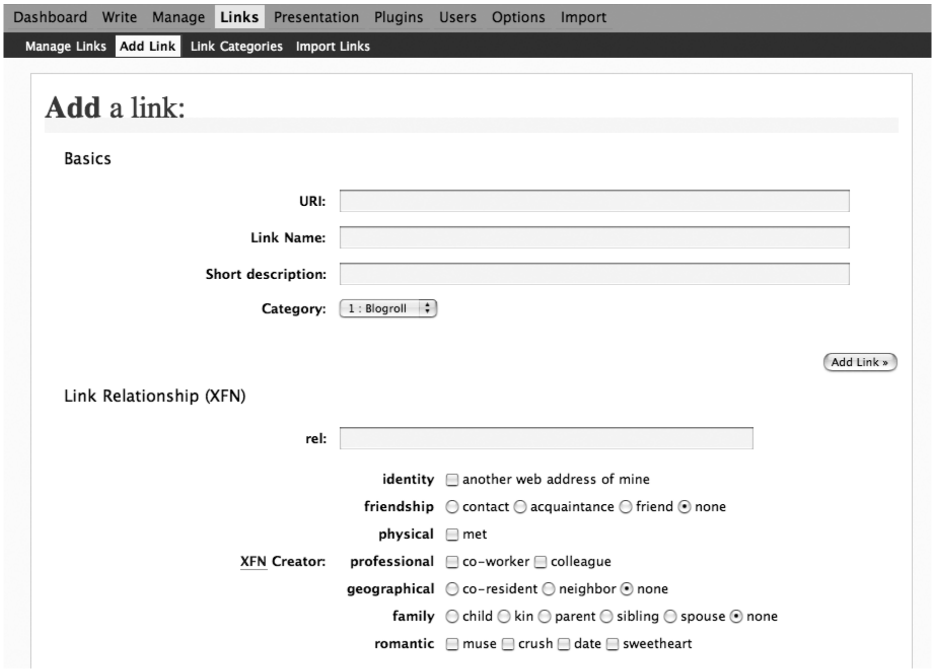
Tools

Even though hand-coding XFN is very straightforward, there is a fair degree of support for XFN in blogging and development tools, as you'll see in the sections that follow.

WordPress

If you've used WordPress (`http://wordpress.org`) and its built-in feature for editing blog rolls (see Figure 5-2), you'll probably have guessed that it has built-in support for XFN.

This is the standard "Add a link" feature of WordPress 2, which automatically creates the HTML for a blogroll. When you use the Link Relationship (XFN) form, WordPress will automatically create the relevant XFN for each blogroll entry.



Add a link:

Basics

URI:	
Link Name:	
Short description:	
Category:	1 : Blogroll

Link Relationship (XFN)

rel:		
XFN Creator:	identity	☐ another web address of mine
	friendship	○ contact ○ acquaintance ○ friend [✓ none]
	physical	☐ met
	professional	☐ co-worker ☐ colleague
	geographical	○ co-resident ○ neighbor [✓ none]
	family	○ child ○ kin ○ parent ○ sibling ○ spouse [✓ none]
	romantic	☐ muse ☐ crush ☐ date ☐ sweetheart

Figure 5-2. The blogroll editor in WordPress, including an XFN Creator feature

Given that Matt Mullenwegg was one of the early developers of XFN, and he is one of the main forces behind WordPress, this high level of support should come as no surprise. For more information on XFN in WordPress, see the WordPress documentation at `http://codex.wordpress.org/Defining_Relationships_with_XFN`.

Bloxsom

Autoxfn is a plug-in for the Bloxsom blogging platform by Buzz Anderson. You can find it at `http://weblog.scifihifi.com/2004/02/08/autoxfn-10`.

Moveable Type

Design guru D. Keith Robinson explains how to set up an XFN link list for Moveable Type at his widely read blog, Asterisk. See `www.7nights.com/asterisk/archive/2004/01/xfn-friendly-link-list-with-movable-type` for details.

XFN Creator

Originally created by Matt Mullenwegg and now maintained by Tantek Çelik, XFN Creator is an online form-based editor for creating XFN for your links. You can find it at `http://gmpg.org/xfn/creator`.

Existing pages

If you already have a site full of links (and who doesn't?), then Exefen, by the prodigious Matt Mullenwegg, is an online tool that grabs all the links of a page and helps you create XFN for any or all of them at once. See `http://photomatt.net/tools/exefen`.

Search engines, services, and applications

Being the earliest microformat meant that XFN saw some experimental services, notably rubhub and XHTML Friends. Interestingly, these services have not evolved much from their early incarnations, which showed considerable promise. However, XFN is still being used in quite interesting ways.

As the number of services that work in conjunction with websites and allow people to claim these grows, it remains a challenge for these services to enable people to easily but correctly make these claims. What's to stop me from claiming `www.microsoft.com` (or any other site) as my own? What's to stop someone else from claiming my sites?

Technorati, for instance, has a number of possible systems. One involves checking your credentials by asking for the username and password to log into your blog and claim it. Of course, I trust Technorati with this information, and I trust its claim that it won't keep this information, but this approach would not work for an arbitrary service that I didn't already know and trust. Sounds a bit too much like phishing for my liking.

Another approach is to add some code to your blog, to demonstrate that you are indeed the owner of the blog. The clever way that Technorati enables this is by using reciprocal `rel="me"` links. The theory is that if you can add a link to a site, with a `rel="me"` XFN value, then it demonstrates you own the site. If you create reciprocal links, then you can claim one site from the other.

More generally, `rel="me"` helps deal with a problem that will only increase as more people get online, with photosharing accounts, blogs, accounts with services like del.icio.us and Ma.gnolia, and other online presences: how can you tell whether a site is associated with a particular person? A name—even a reasonably unusual name like John Allsopp—is no guarantee of unique identity. For example, this is not me:

`http://johnallsopp.com`

nor is this:

`http://www.johnallsopp.co.uk` (despite this John Allsopp being a web developer)

Yet, this is me:

`http://www.flickr.com/photos/johnfallsopp`

and this is also me:

`http://westciv.typepad.com/dog_or_higher` and this is as well:

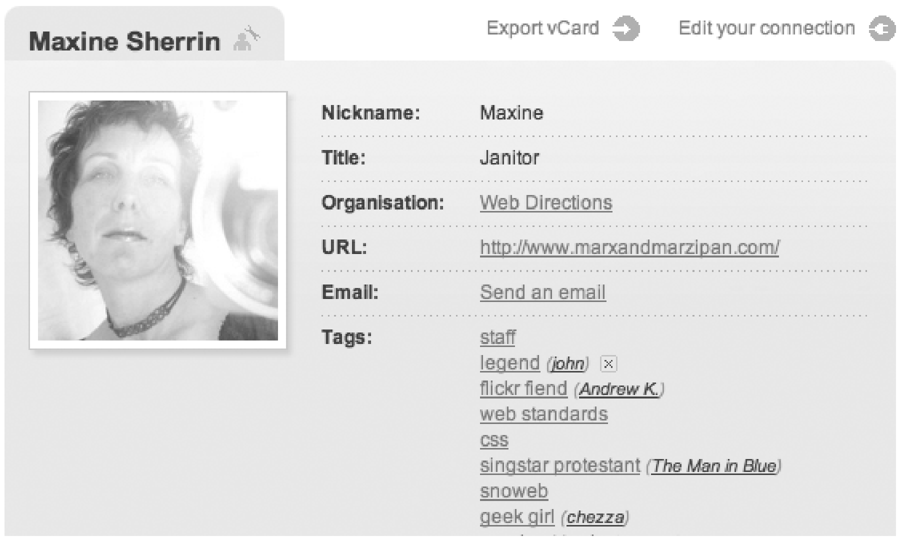
`http://microformatique.com`

(There are more, but that ought to do for now.) Using a `rel="me"` link from one particular site to all the others, I can establish this association between all of these sites and me. As I add sites and other kinds of online presence, I can simply add an XFN link to them, too.

Making connections

One site that has made extensive use of XFN is Web Connections (`http://connections.webdirections.org`), a simple social networking engine first developed for the Web Directions South conference by Tim Lucas and Cameron Adams, to help conference-goers develop, maintain, and grow their relationships with one another, particularly after the conference ended. Web Connections has a simple, form-based approach to "making connections," not unlike WordPress, and it uses XFN to mark up these connections.

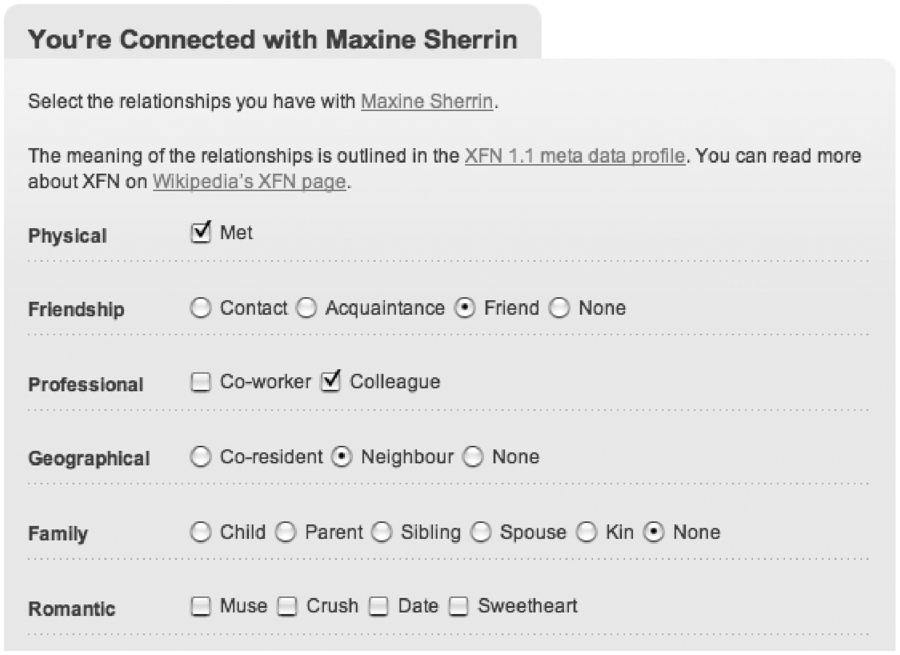
The first step in using Web Connections is to locate the profile of the person you wish to make a connection with. As shown in Figure 5-3, I've chosen my business partner, the very glamorous Maxine Sherrin.



Nickname:	Maxine
Title:	Janitor
Organisation:	Web Directions
URL:	http://www.marxandmarzipan.com/
Email:	Send an email
Tags:	staff legend (john) flickr fiend (Andrew K.) web standards css singstar protestant (The Man in Blue) snoweb geek girl (chezza)

Figure 5-3. Maxine's profile in Web Connections

Next, you edit your connection, as shown in Figure 5-4.



You're Connected with Maxine Sherrin

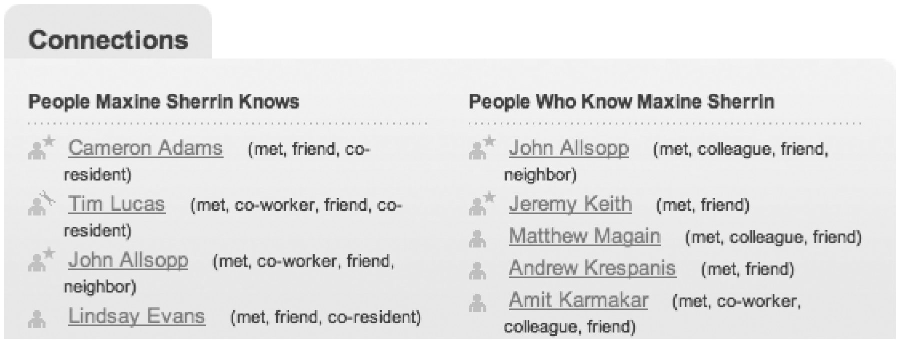
Select the relationships you have with Maxine Sherrin.

The meaning of the relationships is outlined in the XFN 1.1 meta data profile. You can read more about XFN on Wikipedia's XFN page.

Physical	[✓ Met]
Friendship	Contact Acquaintance [✓ Friend] None
Professional	Co-worker [✓ Colleague]
Geographical	Co-resident [✓ Neighbour] None
Family	Child Parent Sibling Spouse Kin [✓ None]
Romantic	Muse Crush Date Sweetheart

Figure 5-4. Editing the details of the connection I have with Maxine

If you take a look at Maxine's profile, you'll find her connections with their relationships listed and marked up using XFN, as shown in Figure 5-5.



Connections

People Maxine Sherrin Knows	People Who Know Maxine Sherrin
Cameron Adams (met, friend, co-resident)	John Allsopp (met, colleague, friend, neighbor)
Tim Lucas (met, co-worker, friend, co-resident)	Jeremy Keith (met, friend)
John Allsopp (met, co-worker, friend, neighbor)	Matthew Magain (met, colleague, friend)
Lindsay Evans (met, friend, co-resident)	Andrew Krespanis (met, friend)
	Amit Karmakar (met, co-worker, colleague, friend)

Figure 5-5. Maxine's XFN relationships on Web Connections

In the left column, Maxine's connection to me is third in the list, while the right column shows other people's relationships as they have marked them up using XFN, including mine, which I just created, at the top.

Although it's a simple idea, XFN provides an easy-to-use, commonly understood vocabulary for marking up connections. You could imagine all kinds of data representations and retrieval from even this simple system. You could, for example, decide to find all of your friends' friends who are not your friends. Or you could have a rule that only showed your email address, phone number, or other such details to people who you have called a friend. The important thing, particularly if we want such a solution to scale beyond a single system like Web Connections, is to have some agreed-upon standard for marking up things like relationships. And that's precisely what microformats aim to help provide—in this case, with XFN, it's a controlled vocabulary for relationships.

Styling XFN content with CSS

Because XFN is based on the value of an attribute, `rel`, to style XFN links in such a way that they will appear similar in all browsers currently in wide use, we need to add a `class` value. The alternative and better approach is to use the attribute selector, which you saw in the last chapter, but which as you know isn't supported in Internet Explorer 6 or older for Windows. Internet Explorer 7, however, does support this selector.

In keeping with our approach of "progressive enhancement" and "keeping our code simple," rather than adding class values such as `class="friend met"` to enable styling in any browser, we'll stick to using just the `rel` values for XFN, and we'll use attribute selectors to style our content. This way, we are enhancing the experience of users of more contemporary browsers, while maintaining the best possible code. Meanwhile, users of older browsers won't know what they are missing (kind of like vegetarians).

You might imagine a number of different icons for friends, colleagues, and so on, and you could use either the background image or `content` property techniques you saw in the last chapter to add these as icons on your links. Chris Messina (who you met in the last chapter) and Wolfgang Bartelme have designed a number of XFN icons that could be used for precisely this purpose, as shown in Table 5-1.

Table 5-1. XFN icons

Icon	Description
👤	Me
👥	Friend
👥✓	Friend you've met
👥	Sweetheart
👥✓	Sweetheart you've met (I'm not thinking about the kind you haven't met too much)
👥	Colleague
👥✓	Colleague you've met

You can grab all the icons shown in Table 5-1 from `http://microformats.org/wiki/icons`.

You already know how to add background images using CSS, so rather than repeat the technique, let's look at something new. In this chapter, you're going to use an old-school technique for styling blogrolls (or other XFN links) with asterisks if you've met that particular person. I call this technique "old-school" because it's something some bloggers used to do to signify precisely this, which led in part to the design of XFN, and in some ways, ultimately to microformats.

You'll use the `:after` selector and add `content`, in this case a string, after links that have a `rel` attribute whose value includes the value `met`. Now, you might be thinking this is just like what you learned in the previous chapter. The selector will be

```
a[rel="met"]:after
```

which will work fine if you have the following link:

```
<a href="..." rel="met">
```

However, it will not work if you have this:

```
<a href="..." rel="met friend">
```

In this case, the value of the `rel` attribute is not equal to `met`, but includes `met` as one of its values. Fortunately, CSS provides a selector for precisely this use (as you'll see later, this selector also comes in handy for other situations, such as multiple class values on the same element, which commonly occurs with more complex microformats). The syntax of this selector is a little different from the one just shown. You add a tilde (~) like so:

```
a[rel~="met"]:after {}
```

This selects any link (`<a>` element) with a `rel` attribute value that includes `met`. Now, you might be wondering if it also selects links like the following:

```
<a href="..." rel="metropolis">
```

It doesn't, because what CSS means by "includes" is "the element's attribute value is a space-separated list of 'words,' one of which is exactly the value." This doesn't include parts of words.

Now, you just need to add an asterisk after these elements, which you do using the `content` property, like when you added an image after the element in the previous chapter. All you need to do is add the following property:

```
content: " *"
```

giving you this:

```
a[rel~="met"]:after {
  content: " *"
}
```

(Note how I got lazy and instead of adding a left margin or padding, I just used a space as well as the *.)

So, if you had a link to Eric Meyer in your blogroll, and you had met him, you'd have some HTML like this:

```
<a href="http://meyerweb.com/" rel="colleague friend met">Eric Meyer</a>
```

In a browser, this link will look something like Figure 5-6.

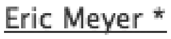

Eric Meyer *

Figure 5-6. The final styled XFN link

The HTML and CSS for this example is available online at `http://microformatique.com/book/chapter5/XFN.zip`.

So you've now used the XFN attribute values as "hooks" for adding meaningful style to the page. You haven't amended the HTML to accommodate styling, which is a principle you should try to keep in mind time and again throughout the book.

Once upon a time, designers were obsessed with pages looking identical in all browsers, and so they spent a lot of time and effort making their markup much more complicated than required in order to achieve this goal. The reality is that users' platforms vary dramatically, while users themselves are capable of making all kinds of changes to their viewing setup—it's one of the strengths and beauties of the Web as a medium. By letting go of the desire to control the pages our readers see, we make our lives as designers and developers much easier, and in fact enhance our users' experience.

Summary

In Chapter 4, you saw a number of link-based "elemental" microformats. This chapter showed a more complex link-based microformat: XHTML Friends Network (XFN).

XFN allows us to model the relationships between ourselves and others on the Web, using links between our sites and sites developed by other people. Using XFN, we can say whether we have met someone, and whether that person is a friend, acquaintance, relative, or possibly some combination of these. Why bother? When I think of XFN, I often recall this quote from Tim Berners-Lee, inventor of the World Wide Web:

The dream behind the Web is of a common information space in which we communicate by sharing information. There was a second part of the dream, too . . . That was that ***once the state of our interactions was on line****, we could then use computers to help us analyse it, make sense of what we are doing, where we individually fit in, and how we can better work together.*

XFN simply but elegantly allows us to mark up this "state of our interactions," and so helps this dream of the Web come one step closer to being realized.

But there's much more to cover—you've only just started on this journey. Now that you've seen a number of elemental microformats, let's next turn to some more complex microformats, sometimes referred to as **compound** microformats.

6 LOCATION MICROFORMATS: GEO AND ADR

Google Maps, along with Yahoo Maps, Microsoft's mapping products, and a significant number of smaller, dynamic mapping companies like Multimap and ZoomIn, coupled with increasingly inexpensive Global Positioning System (GPS) devices (now you can even get cameras that know where they are using GPS!), have all been part of the explosion of geographical and address data on the Web over the last couple of years. But standardized formats for marking up addresses or geographical locations using longitude and latitude have been missing from the equation. Each service typically has its own way of marking up data—for example, Google uses Keyhole Markup Language (KML), an XML-based format, while Yahoo uses GeoRSS, a variation on RSS. In the meantime, this lack of uniformity hasn't stopped developers from adopting all kinds of conventions for marking up location-related data, such as visible geotags by Flickr users (you'll see how Flickr now provides a way of geotagging using microformats later in the chapter).

This chapter takes a look at how microformats address the clear need to mark up addresses and geographical data. The chapter starts off with an overview of location microformats, and then delves into two specific location microformats: geo and adr.

Location microformats overview

As you've already seen, addresses are very difficult for software to extract from unstructured text. For example, here's a challenge you'll find quite straightforward, but imagine how difficult it might be for a computer. There are three locations in the following paragraph. Can you identify them?

If you want to visit our offices, they are at Suite 5 Level 18 Clarence Street Sydney, on the corner of Erskine. Send any packages to P.O. Box 1189 Sydney 2000, but the best way to find us is at the Fox and Hounds on George Street, opposite the Sheraton Hotel just near The Rocks. We are there every Thursday night playing Trivia.

Getting software to extract structured data from text is difficult, even if it conforms to expected patterns, like the physical address in the preceding paragraph, but by the time you get to an "anecdotal" location like a pub on a particular street, or a location opposite a familiar landmark, it's getting nigh on impossible. Yes, well-resourced teams may be able to develop software to extract information like this meaningfully, but ideally the Web is a relatively level playing field, where innovative solutions shouldn't require teams of the very best computer scientists in the world and large sums of money to develop.

Now, if we had way of encoding locations in our markup, then we'd have human-accessible data like "opposite the Sheraton Hotel on George Street just near The Rocks" (which, let me tell you as a Sydney-sider is a lot more meaningful than "13 George Street"—George Street runs several kilometers, but everyone knows where "The Rocks" is, and from there you can easily find the Sheraton Hotel, even if you've never been there), and we'd also have a format that software could easily identify. As you can probably guess, microformats can come to our aid to provide a solution.

Even if your chemistry education stopped in high school, you should be able to recall the fundamental concepts of elements, like hydrogen or oxygen, and compounds, which are combinations of two or more elements. By analogy, the microformats you've seen so far are called ***elemental*** *microformats. This type of microformat is, according to microformats.org, "a minimal solution to a single problem, built from standard XHTML elements." More-complex microformats that are built from elemental microformats and often also other HTML elements are, unsurprisingly, referred to as* ***compound*** *microformats.*

Now that you have a solid understanding of microformat concepts and principles, and you have number of elemental formats under your belt, you'll start building and using more complex formats.

There are two common ways of pinpointing your location uniquely on earth: one that will work anywhere and one that will work anywhere you have a postal system. The first uses what's loosely termed **geodata** (e.g., longitude and latitude), and the other uses a postal address. Both of these methods are hundreds of years old, and they are very well understood. There are microformats that address each of them: the geo and the adr microformats. In the following sections, we'll look at both formats, how you can use them, and how they are already being used by some significant websites.

The geo microformat

Any place on earth can be uniquely described using longitude and latitude. If you are a computer geek, you can think of a longitude and latitude pair as a globally unique identifier (GUID) for a location, whether it's the South Pole, the café around the corner, or your office (well, it will identify your office building; if you also want to specify that your office is on the 15th floor, you'll need altitude as well to differentiate it from all the other offices at the same longitude and latitude). The geo (pronounced "gee-oh") microformat allows you to encode longitude and latitude in your web content.

You've seen that an important microformats principle is to "reuse building blocks from widely adopted standards." So to develop a microformat for geodata, again we look for an existing standard that addresses this issue. We find it in an IETF RFC mentioned in Chapter 1 that we'll look at in detail in the next chapter: vCard. The role of vCard is for encoding "contact information . . . for people, companies, and organizations," and one small part of it (remember the "keep it simple" principle) is the geo field, for encoding—surprise, surprise—geodata.

Now, you should be aware that there is an important restriction on the use of the geo microformat. The geo microformat specification states the following:

If the publisher knows and is publishing the name of the location in addition to its GEO lat/long, then the publisher MUST use hCard instead of just GEO to publish the name and GEO lat/long of the location.

If the publisher knows and is publishing the address of the location, OR if the address of the location was what was actually entered by a human, and the publisher simply turned that into lat/long using some sort of a service, then the publisher SHOULD use ADR to publish the actual human entered address information since it communicates far more semantic information than a simple GEO lat/long coordinate.[1]

So, if we have the name of a hotel, such as "The Sheraton, Sydney," or "The White House," or some other named location, we must use the hCard microformat (more information on this microformat in the next chapter). If we know the address of the location, we should use the adr microformat, rather than just the longitude or latitude (more information on adr is coming up later in this chapter).

When might it be sensible to use the geo microformat by itself? Well, many places don't have an address—for example, the summit of Mount Everest (longitude 86° 56' 40" E, latitude 27° 59' 16" N), or the splashdown site of Apollo 11 after returning to earth (13° 19' N, 169° 9' W). Both of these locations would be perfect candidates for the geo microformat.

Similarly, tourist information (whether personal travel diaries, blog posts, or commercial travel sites) is a perfect candidate for the use of geo, as is web content on historical events, where location is often very important (e.g., places where battles took place or important documents were signed). Place is a profoundly important piece of information, and the phenomenal rise in mapping applications like Google Maps and Yahoo Maps, along with map-based "mashups" demonstrates the importance of some form of standardized way of marking up geographic information.

A common use for geodata that's already in use by bloggers and others on the Web is tagging the location a photo was taken. You an imagine a search engine that looks for geo information associated with images and presents it on a world map (perhaps mashed up using Yahoo Maps or Google Maps), or that enables you to find photos of nearby spots, photos of a particular location, and so on.

Recently, Flickr, responding to the common practice of tagging such photo locations with Flickr tags, implemented a feature that allows you to geotag photos on Flickr using a map interface. Overnight, more than 1 million photos were geotagged in this way. And exciting news for those associated with the microformats community was that Flickr chose to use the geo microformat to mark up this information.

Let's take a quick look at how this geotagging works and the code that Flickr uses to mark up locations. If you have a Flickr account, start up the organizer, choose the Map tab, and then move the map to the location you took your photo, as shown in Figure 6-1.

1. See `http://microformats.org/wiki/geo#Draft_Specification`.

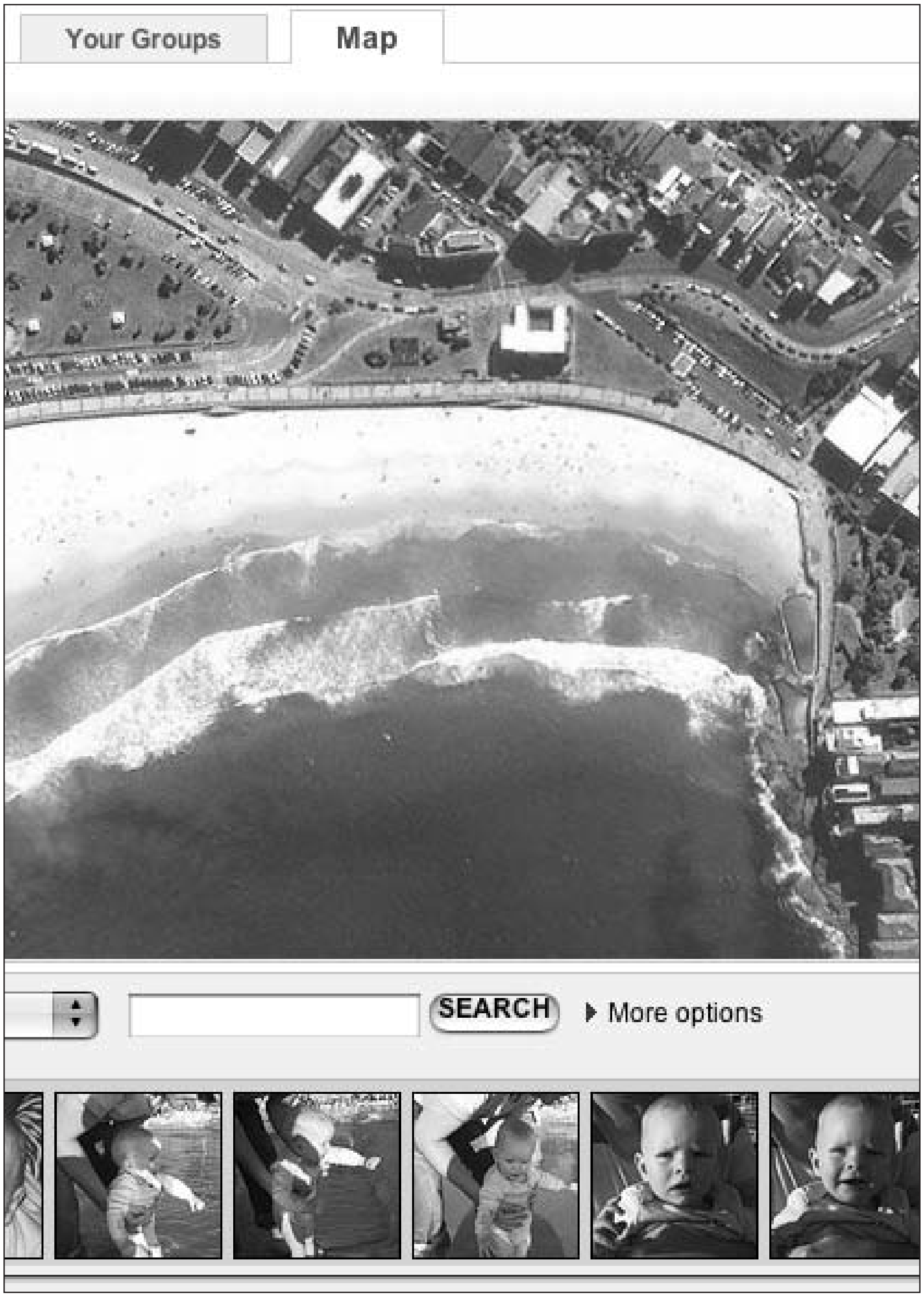


Figure 6-1. The Flickr organizer Map tab

Reproduced with permission of Yahoo! Inc. © 2007 by Yahoo! Inc.
YAHOO! and the YAHOO! logo are trademarks of Yahoo! Inc.

Now drag one of your photos to the place it was taken, as shown in Figure 6-2.

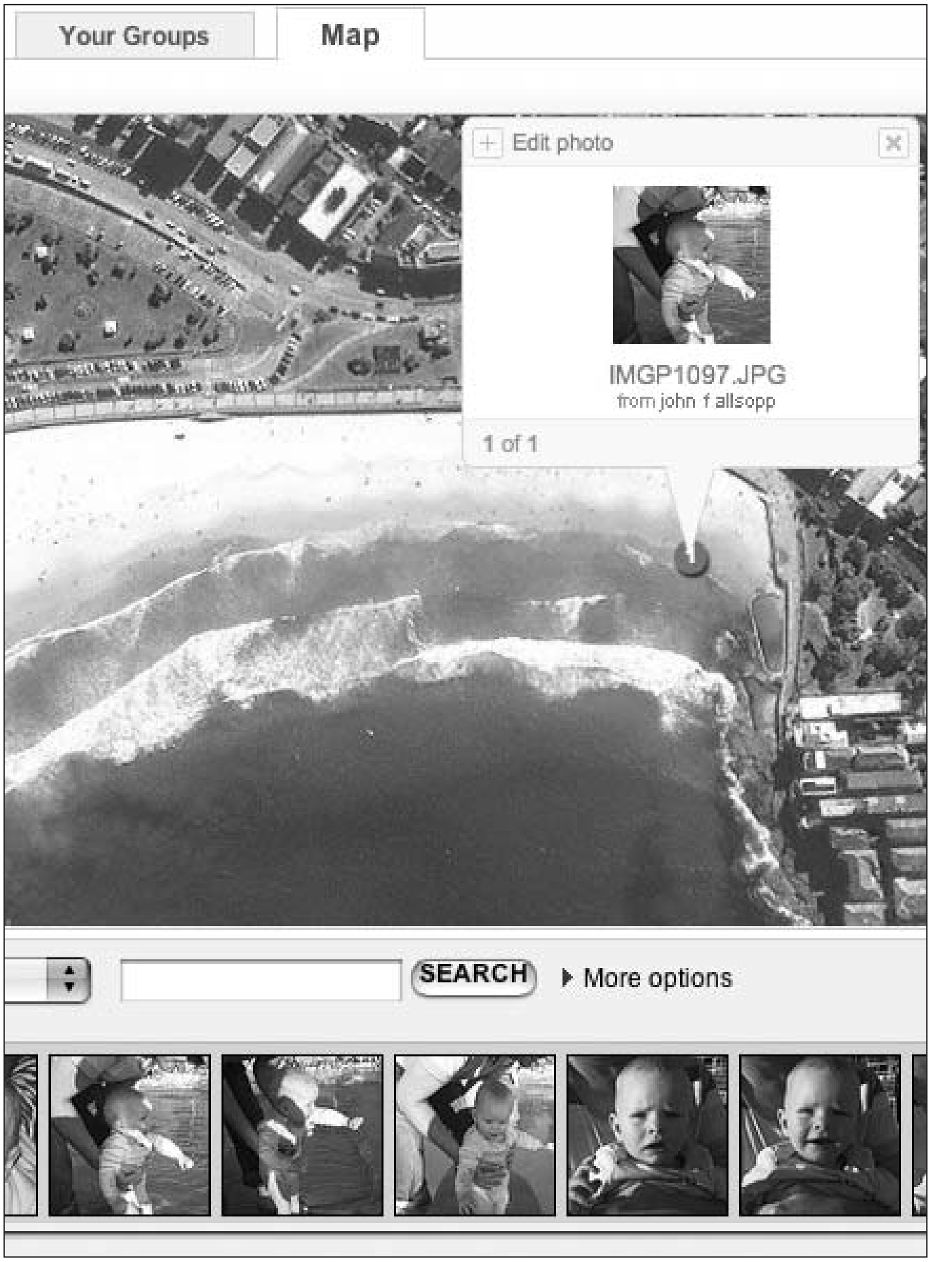


Figure 6-2. Dragging a photo to the location it was taken

If you look at the photo in its original location and click “map” in the list of properties, you get a pop-up map, similar to the one shown in Figure 6-3.

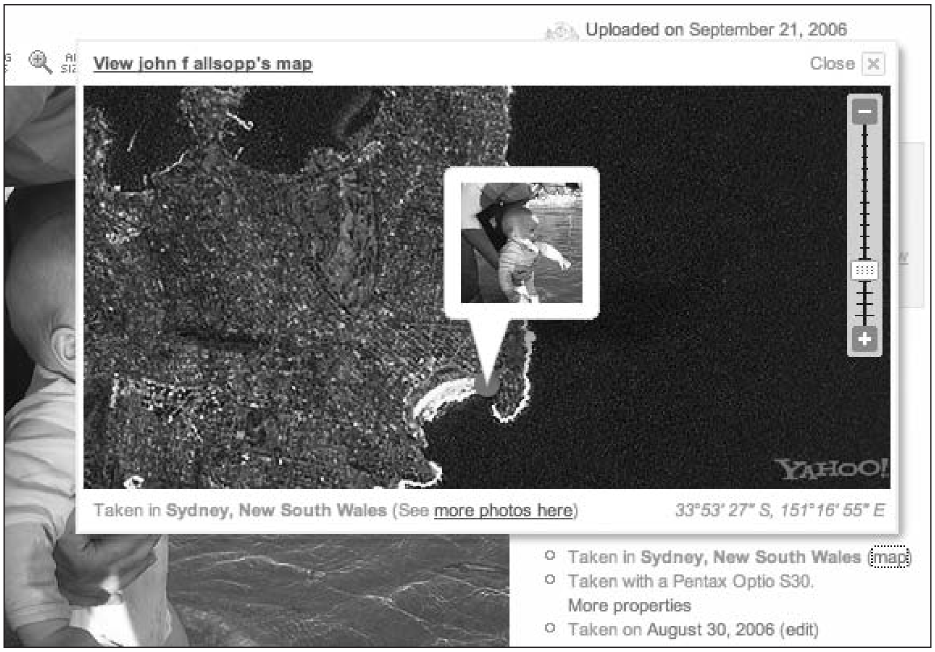

Uploaded on September 21, 2006
View john f allsopp's map
Close
YAHOO!
Taken in Sydney, New South Wales (See more photos here)
33°53' 27" S, 151°16' 55" E
Taken in Sydney, New South Wales (map)
Taken with a Pentax Optio S30.
More properties
Taken on August 30, 2006 (edit)

Figure 6-3. For all images that you've assigned a location in Flickr in this manner, you can get a pop-up map to show where they were taken.

Reproduced with permission of Yahoo! Inc. © 2007 by Yahoo! Inc.
YAHOO! and the YAHOO! logo are trademarks of Yahoo! Inc.

Now if you take a look at the code for the longitude and latitude in the bottom-right area of the pop-up map, you'll see something like this:

```
<div style="position: absolute; right:7px; bottom:7px; ➥
font-family: Arial; font-size: 11px; color: #999"><i> ➥
33&#176;53' 27" S, 151&#176;16' 55" E<span class='geo' ➥
style='display:none'><span class='latitude'>-33.890937</span> ➥
<span class='longitude'>151.281985</span></span></i></div>
```

Leave aside the fact that Flickr has mixed in inline CSS, as well as presentational HTML (yes, I know, `<i>` is valid strict HTML and XHTML, but this is still a presentational use of the `<i>` element). As you'll see in just a moment, this is, in fact, the geo microformat in action.

Using geo

The microformats presented so far in the book have been associated with single HTML elements. From this chapter on, the formats covered are compounds of two or more HTML elements. All compound microformats of two or more elements are contained inside what is termed the **root element**. Root elements typically have a `class` value that indicates the microformat values they contain. In the case of the geo microformat, the root element has a `class` value of geo.

A geo root element may look like this, for example:

```
<span class="geo">...</span>
```

In geo, the root element contains two properties, each of them HTML elements: one for longitude and one for latitude. Here's a very simple example:

```
<span class="geo">
  <span class="latitude">27.976628</span>,
  <span class="longitude">86.933302</span>
</span>
```

In the previous Flickr example, we had this geo microformat (which I've slightly reformatted to make it more readable):

```
<span class='geo' style='display:none'>
  <span class='latitude'>-33.890937</span>
  <span class='longitude'>151.281985</span>
</span>
```

(Don't worry about those large numbers right now—we'll get to those in a moment.)

The root element in both cases is a `<span>` element with a `class` value of `geo`. This contains geo's two properties, longitude and latitude, themselves also `<span>` elements, with `class` values of `longitude` and `latitude`, respectively. The actual longitude and latitude values are the text (or **content**) of the respective elements.

This example demonstrates something really important in microformats: the **class design pattern**, which is used frequently for constructing a microformat. The class design pattern uses the `class` attribute value on HTML elements to add more meaning (or "richer semantics") than is inherently available in HTML. For a microformat, it's actually the attribute values of elements, rather than the elements themselves, that are important. So the same class value might be used with different kinds of elements, and it will still be a properly formatted geo microformat. For example, instead of the span elements we have here, we could have the following:

```
<dl class="geo">
  <dt>Lat:</dt>
  <dd class="latitude">27.976628 </dd>
  <dt>Lon:</dt>
  <dd class="longitude">86.933302</dd>
</dl>
```

which is completely equivalent from a microformat perspective to the same construction using span elements. The definition list approach is actually how the mapping service Multimap reports location.

But what are those large numbers, like “27.976628”? These are longitude and latitude in a decimal format, or **decimal degrees**. Traditionally, longitude and latitude are reported as degrees, minutes, and seconds, such as 57 degrees, 21 minutes, 8 seconds (or 57° 21' 8"). But computers like the decimal format (actually, the decimal format makes it much easier for programmers to do calculations), so it is commonly used, particularly by mapping software.

How do you convert from traditional to decimal mapping values? It’s probably rare that you’ll have to do this by hand, but if you do, then you’re in luck—the formula you need is of the following form:

```
decimal degrees = degrees . (minutes+(seconds/60))/60)
```

Taking our example of 57 degrees, 21 minutes, 8 seconds from earlier, the decimal degrees value would be 57.(21+8/60)/60 or 57.352222 (typically, decimal degrees are reported to six decimal places).

For the lazy among us, here’s a site that will convert in either direction: `www.fcc.gov/mb/audio/bickel/DDDMMSS-decimal.html`.

Getting location data

“But wait,” I hear you say. “How do I get these values—whether decimal or traditional—in the first place?” Well, if you can convince your boss or financers to let you travel around the world with a GPS device, that’s one way, but if time and money are an issue, several online mapping services can help you here. The following sections describe some possible options.

Google Maps

You can use Google Maps to get location values, although it is a bit of a hack, because Google Maps doesn’t actually display the longitude and latitude on the page. It is there, though—you just need to know how to find it:

1. Find the location on Google Maps by searching or browsing.
2. Double-click the location you want the geo code for.
3. From the “Link to this page” link near the top-right corner (see Figure 6-4), get the URL value, which will look something like this:

```
http://maps.google.com/maps?f=q&hl=en&sll=37.795678, ➥
122.400699&sspn=0.021195,0.026178&q=Embarcadero&ie= ➥
UTF8&ll=37.793915,-122.403145&spn=0.010598,0.013089&om=1
```

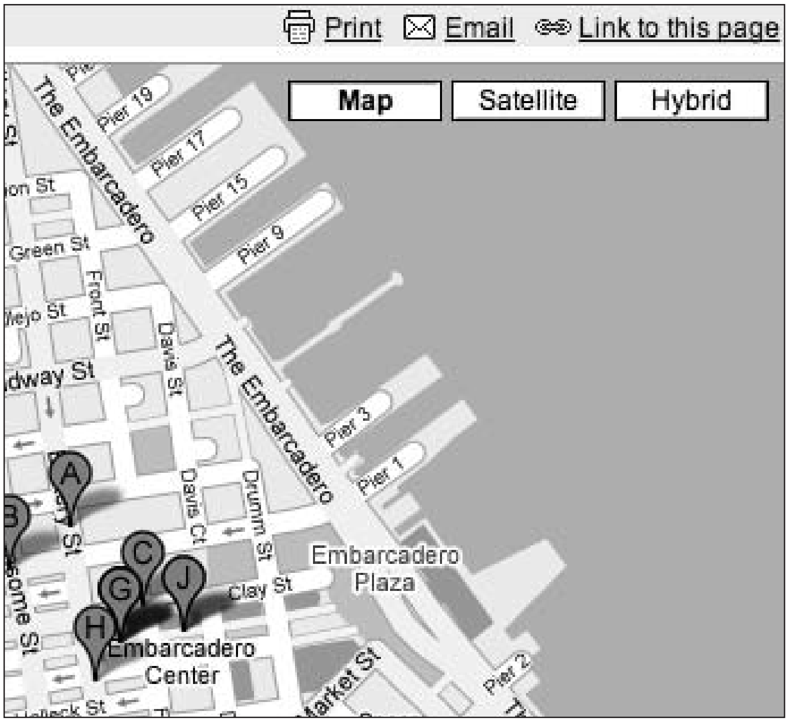


Figure 6-4. Google Maps shows geo information via the "Link to this page" link.
Google Maps™ mapping service/NAVTEQ™. Used with permission.

The longitude and latitude are 37.795678 and –122.400699. Like I said, it's a bit of a hack, but there are other possibilities, too.

Multimap

Multimap (`www.multimap.com`) makes it very easy to get location data, in both traditional and decimal degree formats. It even uses the geo microformat for this information. Just search for or browse to a location. You'll find the location information under the Map Information heading, as shown in Figure 6-5.

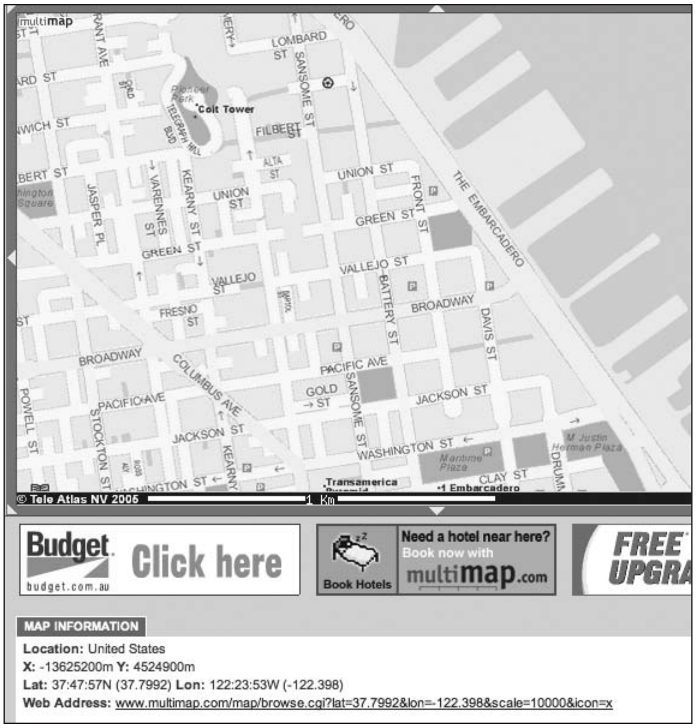

MAP INFORMATION
Location: United States
X: -13625200m Y: 4524900m
Lat: 37:47:57N (37.7992) Lon: 122:23:53W (-122.398)
Web Address: www.multimap.com/map/browse.cgi?lat=37.7992&lon=-122.398&scale=10000&icon=x

Figure 6-5. It's easy to find geo information in Multimap.

Yahoo Maps

The process of getting longitude and latitude values from Yahoo Maps is almost identical to the Google Maps method.

1. Search for or browse to a location in Yahoo Maps.
2. In the top-right corner of the map is a link for linking to the map, as shown in Figure 6-6.

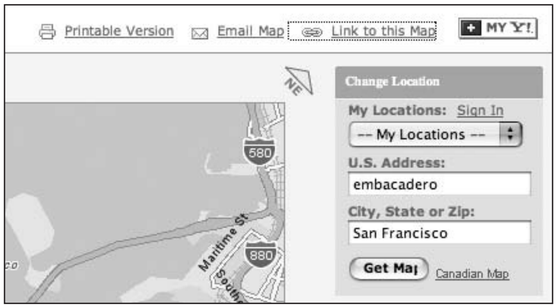


Figure 6-6. Yahoo Maps includes a Link to this Map link, which contains geo data.

3. Inside the URL the link is pointing to, which looks something like this:

```
http://maps.yahoo.com/linkmap?addr=&csz= ➥
San+Francisco%2C+CA&state=CA&uzip=94103&ds= ➥
n&name=&desc=&lat=37.7742&lon=-122.417068&mlt= ➥
37.7742&mln=-122.417068&zoomin=yes&BFKey=&mag=4
```

we find `lat=37.7742` and `lon=-122.417068`, which is marginally friendlier than Google Maps (where you have to remember that latitude comes first).

Getting geo data with an address

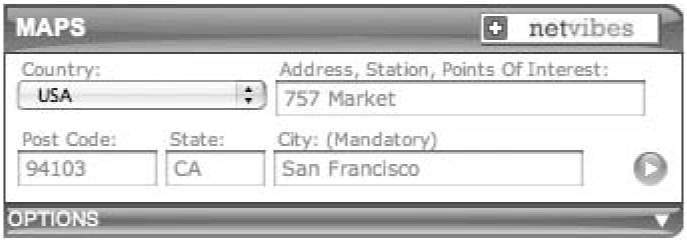
A special case is where you have an address and want to convert it to a geo location. A number of services will do this, in addition to Google Maps, which as you just saw is able to convert an address in a convoluted fashion. Most of these services are specific to a single country, although Maporama (`http://maporama.com`) is able to search for an address in a great many countries, and return longitude and latitude with the results.

Keep in mind that if you have an address, and you also want to add the geo data, then it is recommended you add the address using the adr microformat, which we'll look at in a moment.

Imagine I'm staying in San Francisco for a few days, and because I am a very geeky kind of guy, I want to publish the geographic location of the hotel I am staying at. I know the address:

```
757 Market Street
San Francisco
CA 94103
```

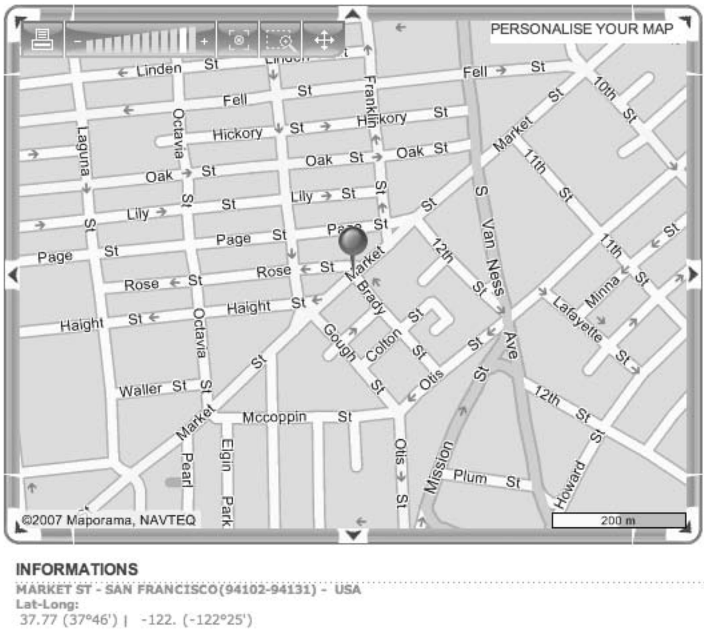
First, I need to look up the geographic location for that address. Maporama provides a simple search interface for this. I enter the address, as shown in Figure 6-7.



Country: USA		Address, Station, Points Of Interest: 757 Market
Post Code: 94103	State: CA	City: (Mandatory) San Francisco

Figure 6-7. Searching for a location by address in Maporama

Maporama returns a map, with the longitude and latitude displayed, as shown in Figure 6-8.



INFORMATIONS
MARKET ST - SAN FRANCISCO(94102-94131) - USA
Lat-Long:
37.77 (37°46') | -122. (-122°25')

Figure 6-8. Maporama displaying the location, along with the longitude and latitude

Sadly, Maporama doesn't use the geo microformat for this.

If you are manually marking up locations and have addresses, Maporama and Multimap are good services for getting the geo data you need. But what if you are looking to grab this kind of data programmatically? While the specifics of how you would do this are beyond the scope of this book, Yahoo has a Geocoding API that facilitates looking up geo-data from addresses: `http://developer.yahoo.com/maps/rest/V1/geocode.html`. And for addresses within the United States, Geocoder.us (`http://geocoder.us`) is a free service based on U.S. Census data.

So go ahead and use this data in all kinds of creative ways—just make sure that if you publish longitude and latitude, you use the geo microformat to do so.

The abbr design pattern

So far you've seen how to mark up longitude and latitude when these values are visible on the page. But what if you are referring to a location, such as the splashdown point for Apollo 11, and you want to add geo data, but not have this visible as part of the content? For example, you might want to add this information to a page as additional data for software to use, like in a mashup of the locations of all the splashdown locations of Apollo command modules. Here you can use another microformats design pattern: abbr.

According to microformats.org, the abbr design pattern is designed to "make text that is human readable also formally machine readable." You'll see in later chapters that other microformats besides geo make use of this design pattern.

To understand how the abbr design pattern works, think back to Chapter 4, where you examined the `<abbr>` element. The `<abbr>` element indicates "an abbreviated form" according to the HTML specification. Obviously, you can use this element to mark up abbreviations like `<abbr>W3C</abbr>`, `<abbr>WHO</abbr>`, or `<abbr>WIPO</abbr>`. But what's WIPO, you ask? Well, here you can add a little more HTML, the `title` attribute (yes, the `title` attribute is not just for links), to "provide the full or expanded form of the expression."

In this case, we'll mark up our splashdown location like this:

```
Apollo 11 splashed down <abbr title="13.150000;169.150000" ➥
class="geo"> 400 miles (640 km) South West of Wake Island, ➥
in the North Pacific Ocean</abbr>
```

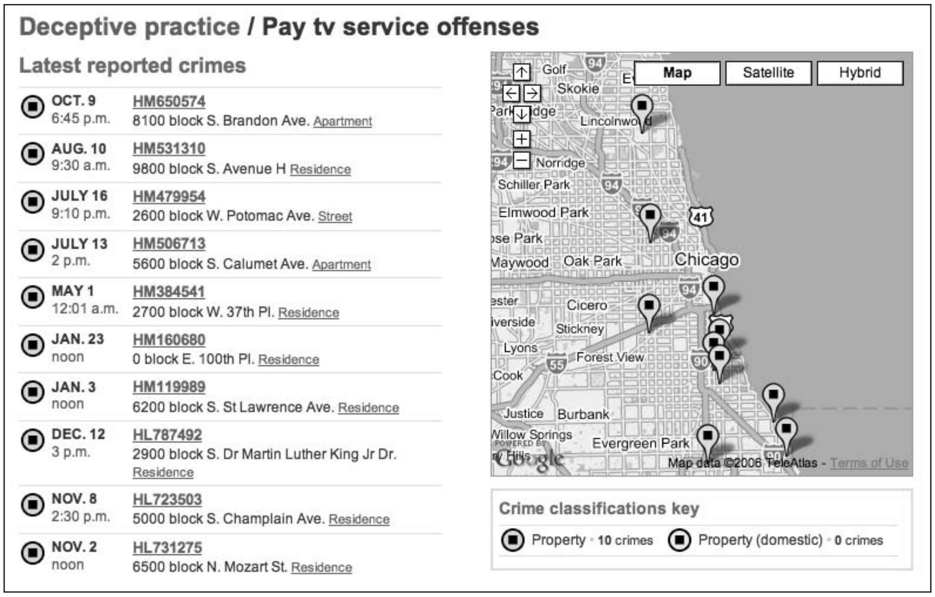
Notice that the latitude and longitude are the value of the `title` attribute, separated by just a semicolon, and latitude precedes longitude. With the abbr design pattern, we can, according to the abbr design pattern specification, "enclose the human-friendly text that you want to make machine readable with `<abbr>`" and "add a `title` attribute to the `abbr` element with the machine-readable data as the value." So, our human-readable text is "400 miles (640 km) South West of Wake Island, in the North Pacific Ocean," and the machine-readable part is "13.150000;169.150000."

Again, one of the important aspects of microformats is to use HTML correctly and appropriately. Here, we've taken an existing feature of HTML and used it semantically. Because this is an approach to markup we can use time and again, we formalize it as a *pattern*. You'll see the abbr design pattern used frequently in other microformats in the same way: to encode machine-readable and human-readable data.

Benefits of using geo

You've no doubt seen Google Maps or other mapping system–based mashups. One of the more famous mashups is Adrian Holovaty's chicagocrime.org mashup (`www.chicagocrime.org`), which takes publicly available data about crimes in Chicago and "mashes it up" with Google Maps. As an example, Figure 6-9 shows some of the heinous pay TV service offenses committed in Chicago (sadly, I couldn't find a list of really serious offenses, like illegal music downloads).

The key to mashups is having readily machine-readable information to mash up with a mapping service. At present, there are a number of different, though somewhat similar online geodata formats, including Google Earth's KML (an XML format) and GeoRSS (an extension to RSS for encoding geodata), but there is no widely adopted web/HTML-friendly format for marking up geodata. The geo microformat's simplicity means not only that is it very easy for content developers to adopt, but also that using geodata in map mashups is straightforward for application developers.



Deceptive practice / Pay tv service offenses

Latest reported crimes

●	OCT. 9 6:45 p.m.	HM650574 8100 block S. Brandon Ave. Apartment
●	AUG. 10 9:30 a.m.	HM531310 9800 block S. Avenue H Residence
●	JULY 16 9:10 p.m.	HM479954 2600 block W. Potomac Ave. Street
●	JULY 13 2 p.m.	HM506713 5600 block S. Calumet Ave. Apartment
●	MAY 1 12:01 a.m.	HM384541 2700 block W. 37th Pl. Residence
●	JAN. 23 noon	HM160680 0 block E. 100th Pl. Residence
●	JAN. 3 noon	HM119989 6200 block S. St Lawrence Ave. Residence
●	DEC. 12 3 p.m.	HL787492 2900 block S. Dr Martin Luther King Jr Dr. Residence
●	NOV. 8 2:30 p.m.	HL723503 5000 block S. Champlain Ave. Residence
●	NOV. 2 noon	HL731275 6500 block N. Mozart St. Residence

Crime classifications key

● Property · 10 crimes ● Property (domestic) · 0 crimes

Figure 6-9. Mashup example: chicagocrime.org

Google Maps™ mapping service/TeleAtlas. Used with permission.

Already some experimental tools have been created to mash up geodata with Google Maps, like the one from Brian Suda (whose very cool X2V service you'll see when we look at the hCard and hCalendar microformats). This service, which at the time of this writing doesn't have a name, finds geodata on a page, converts it to Google's KML, and then passes this to Google Maps via its API, which plots the geocoded data on a Google Map. The mashup even takes event or location names from other microformats like hCard and hCalendar, and adds these to the map mashup as well. This probably sounds a little complicated, but the great part is that once you as a developer have used the geo microformat, you have nothing else to worry about.

Here's an example from the wild. At the site for a conference I help run, we mark up all the data we can with microformats. For locations, we use geo, and our information looks something like this:

```
<abbr class="geo" title="-37.831107;144.962325">
  Bell's Hotel and Brewery, 157 Moray St, ➥
CNR with Coventry, South Melbourne
</abbr>
```

I've simplified this example, because as you saw a short time ago, we should be using the adr microformat for the address. At the site, we actually do so.

We use the abbr pattern to show people human-readable information, and we use the title attribute to encode the machine-readable latitude and longitude. Let's see how Brian Suda's web-based service uses this.

At http://suda.co.uk/projects/microformats/geo, we just add the URL of our site. We can choose to convert it to either Google Maps KML format or GeoRSS (Yahoo Maps supports GeoRSS). These files can then easily be used with these mapping systems, with no need to learn the different mapping formats or APIs, or separate files can be written for different mapping systems.

You can even automatically create a Google Maps mashup from your geo-encoded data—here's how:

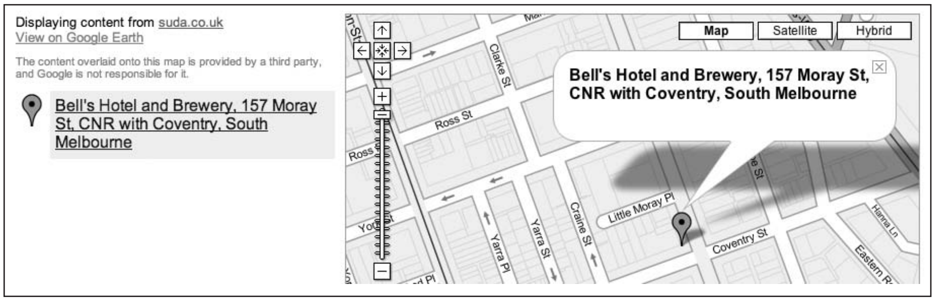
1. Take this URL at Google Maps: http://maps.google.com/maps?q=.
2. Escape the URL for Brian's converter service, and then append it to the Google Maps URL:

   ```
   http://maps.google.com/maps?q=http%3A//suda.co.uk/projects/ ➥
   microformats/geo/get-geo.php%3Ftype%3Dkml%26uri%3D
   ```

3. Finally, append the URL for your page (or any other page) that contains the geo-data you want to mash up with, and add the URL of the page you want to create a mashup for (either escaped or not), which should give you this:

   ```
   http://maps.google.com/maps?q=http%3A//suda.co.uk/projects/ ➥
   microformats/geo/get-geo.php%3Ftype%3Dkml%26uri% ➥
   http://microformatique.com/book/chapter6/geo.html
   ```

If you put this into your browser's address field, you'll get something like the page shown in Figure 6-10.


Displaying content from suda.co.uk
View on Google Earth
The content overlaid onto this map is provided by a third party, and Google is not responsible for it.
Bell's Hotel and Brewery, 157 Moray St, CNR with Coventry, South Melbourne

Bell's Hotel and Brewery, 157 Moray St, CNR with Coventry, South Melbourne

Figure 6-10. Creating an ad hoc mashup with Google Maps and Brian Suda's geo service
Google Maps™ mapping service/MapData Sciences Pty Ltd., PSMA. Used with permission.

Not only do you have the location extracted and mapped, but you also have the human-readable address extracted from the <abbr> element. No need for a Google Maps API key or any JavaScript coding. By comparison, here's what you'd have to add to your HTML to get a similar effect:

```
<head>
  <script type="text/javascript">
  //<![CDATA[
    function load() {
      if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map"));
        map.setCenter(new GLatLng(-37.831192, 144.962561), 15);
        map.addControl(new GSmallMapControl());
        map.addControl(new GMapTypeControl());
        var point = new GLatLng(-37.831192, 144.962561)
        //map.addOverlay(new GMarker(point));
        var marker=new GMarker(point);
        map.addOverlay(marker);
        marker.openInfoWindowHtml("<p style='font-size: small; ➥
        width=10em'><div class='vevent' style='font-size: small'> ➥
        <span class='summary'> Web Directions presents ➥
        <div class='vcard' style='display: inline'> <span class='fn'> ➥
        Ben Barren<\/span><\/span><br \/> <span><abbr class='dtstart' ➥
        title='20060810T1800+1000'> 6.00pm Thursday August 10 2006 ➥
        <\/abbr><\/span><br \/> <span class='location'>Bell's Hotel ➥
        and Brewery, 157 Moray St <br \/>South Melbourne<\/span><br \/> ➥
        <\/div><\/p>");
      }
    }
  //]]>
  </script>
</head>
<body onload="load()" onunload="GUnload()">
  <div id="map" style="width: 500px; height: 300px"></div>
</body>
```

Mashups are already very big news. And the more geodata there is on the Web, the more interesting applications along these lines we'll see, including mashups using *your* data. By using the geo microformat to mark up location information, you'll be giving your data a much better chance of ending up playing a part in as-yet-unthought-of map mashups.

Styling geo content with CSS

Unlike with rel-based microformats, which require the use of the poorly supported attribute selector to style with CSS, because the geo microformat is class based, we can style our geo content in just about any browser, using the class selector.

We might, for instance, want to indicate locations that are geo encoded, not unlike how we used CSS to style license or tag links in a previous chapter. To do so, we start with a selector for any element with a class value of geo:

```
.geo {}
```

Then we use either of the techniques from the previous chapter for adding an image after the text of the element with a class of geo: using background-image or using the :after selector to add generated content. Using the former technique, we have a statement like this:

```
.geo {
background-image: url(geo-logo.png);
background-repeat: no-repeat;
background-position: right center;
padding-right: 36px;
}
```

Using generated content, we get this:

```
.geo:after {
  content: url(geo-logo.png);
  padding-left: .5em
}
```

We'll use this technique to style a photo on a site that is also geotagged using the geo microformat:

```
<p>Photo taken at <abbr title="-33.890937;151.281985" class="geo">
North Bondi</abbr></p>
```

The result looks like Figure 6-11 when viewed in a browser.

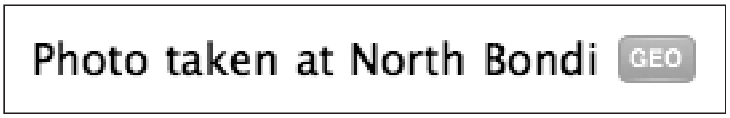


Photo taken at North Bondi

Figure 6-11. An image caption marked up with the geo microformat and including a geo icon

The first technique, as you saw previously, works in any browser from Internet Explorer 4 and Netscape 4 upward (this time, because we are using the class selector, even these browsers will display the style). The second technique works only in contemporary browsers (Safari, recent versions of Opera, and Firefox and other Mozilla-based browsers).

You can download the HTML and CSS for this section's example from http://microformatique.com/book/chapter6/geo.zip.

The adr microformat

While geographic data is quite new to the Web, I suspect adding physical addresses to web pages goes back to some of the earliest pages, and there must be many millions of addresses online. But HTML doesn't provide a standardized way of marking up addresses (the `<address>` element, as you saw, is restricted to "supply[ing] contact information for a document or a major part of a document").

So, using the microformats approach, we look for a simple, specific, existing standard for address information. Almost every one of us uses a well-established format for this all the time, in our address books in applications like Entourage, or in the Mac OS X Address Book. Typically, address book applications use the vCard format, which you encountered briefly earlier in the book, and a subset of the vCard format is adr for—you guessed it—addresses. In Chapter 1, you were introduced to adr and saw that it has the following fields:

- `post-office-box`
- `extended-address`
- `street-address`
- `locality`
- `region`
- `postal-code`
- `country-name`

The adr subset of vCard is an ideal candidate for our microformat for addresses, as it's simple, modular, and widely used. The following sections describe how to use adr, the benefits of doing so, and how to style adr content with CSS.

Using adr

adr is reasonably simple to hand-code, and some tools are available to help you add it to your websites. The following sections discuss the available options, starting with how to hand-code adr content.

Hand-coding adr content

How do we translate the existing vCard format into HTML markup? After all, as demonstrated in Chapter 1, it's not a pretty sight. Here is the adr component of a simple vCard:

```
ADR;TYPE=dom,home,postal,parcel:;; ➥
123 Main Street;Any Town;CA;91921-1234;
```

Again, we'll implement the class design pattern and use class attribute values to add these fields to standard HTML elements, giving them more semantics than HTML alone. Just as with geo, we'll have a root element, which contains all the address information. We'll use adr as the class value for this root element, which, as you saw, can be any HTML element but will often be a `<div>` or `<span>` element.

Using the class design pattern, the locality field in adr becomes `<span class="locality">`, the country-name field becomes `<div class="country-name">`, and so on, giving us an HTML fragment like this:

```
<div class="adr">
  <div class="street-address">2560 Ninth Street </div>
  <div class="extended-address">Suite 219</div>
  <span class="locality">Berkeley</span>,
  <span class="region">CA</span>
  <span class="postal-code">94710</span>
  <div class="country-name">USA</div>
</div>
```

which is displayed in HTML as follows:

```
2560 Ninth Street
Suite 219
Berkeley, CA 94710
USA
```

You can see that each adr property is mapped into an HTML element, using the class design pattern to give each element a class value of the name of the associated vCard adr field.

You might be wondering why street-address, extended-address, and country-name are `<div>` elements, while the other properties of adr are `<span>` elements. Remember, using the class pattern, it doesn't matter what elements you use, provided they are valid HTML (so, if you used `<span>` for the root element, the HTML containment rules presented in Chapter 3 require all of the elements contained in the root element to be inline elements). We've in fact chosen these elements to be divs because they'll appear on their own line in the browser—recall that divs are block elements, so by default that's how they'll appear, while spans are inline elements, and so will appear by default inline, not on a new line. We could just as readily have made all of the elements inside the root element spans, but then we'd need to use some extra CSS if we wanted the address to appear in the browser in the conventional way. We'll take a look at styling adr with CSS shortly.

Tools for coding adr

If you don't want to hand-code adr, there are quite a few tools available to do the heavy lifting for you:

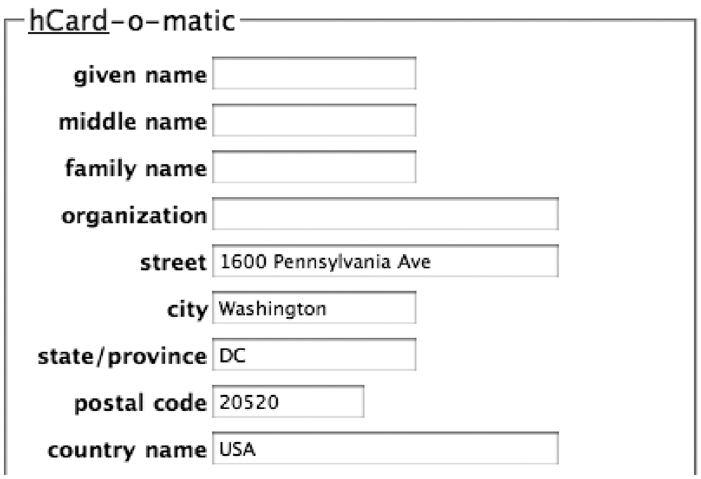
- **hCard Creator** (`http://microformats.org/code/hcard/creator`): You can use the hCard Creator from microformats.org and only take the adr part it generates (remember, adr is a subset of hCard).
- **Dreamweaver Extensions Suite** (`www.webstandards.org/action/dwtf/microformats`): This suite, available, from the Web Standards Project, provides tools for adding various microformats, including hCard, to your HTML.
- **Textpattern Microformats Plug-in** (`http://placenamehere.com/TXP/pnh_mf`): The very geekily named pnh_mf plug-in provides a simple way for adding various microformats, again including hCard, to Textpattern-based sites.
- **Structured Blogging** (`http://structuredblogging.org`): This site offers plug-ins for WordPress and Moveable Type that allow easy creation of microformatted content and support hCard.

Let's take a closer look at the first of these tools, hCard Creator.

hCard Creator (or "hCard-o-matic"), is a form-driven tool for creating hCard microformatted content.

You'll take a detailed look at hCard in the next chapter, but as you've already seen, adr is a subset of hCard.

To use the tool, you simply fill in the relevant fields in the hCard-o-matic form, as shown in Figure 6-12.


hCard-o-matic

given name	
middle name	
family name	
organization	
street	1600 Pennsylvania Ave
city	Washington
state/province	DC
postal code	20520
country name	USA

Figure 6-12. hCard-o-matic from microformats.org allows you to create hCards automatically.
The hCard-o-matic is copyright © 2005 Tantek Çelik. Some rights reserved (`http://creativecommons.org/licenses/by/2.0/`).

hCard-o-matic creates the HTML code for you, like so:

```
<div class="vcard">
  <span class="fn"> </span>
  <div class="adr">
    <div class="street-address">1600 Pennsylvania Ave</div>
    <span class="locality">Washington</span>, ➥
    <span class="region">DC</span>,
    <span class="postal-code">20520</span>
    <span class="country-name">USA</span>
  </div>
</div>
```

For now, let's simply take the adr part of the hCard code it creates, giving us the following adr microformat:

```
<div class="adr">
  <div class="street-address">1600 Pennsylvania Ave</div>
  <span class="locality">Washington</span>,
 <span class="region">DC</span>,
  <span class="postal-code">20520</span>
  <span class="country-name">USA</span>
</div>
```

Without any CSS styling, the result looks like Figure 6-13 in most browsers.

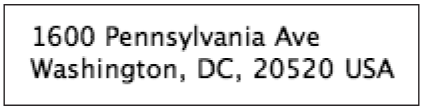

1600 Pennsylvania Ave
Washington, DC, 20520 USA

Figure 6-13. Example adr microformat viewed in a browser

In a moment, we'll look at styling adr content and see some techniques that will come in handy for styling many other kinds of compound microformats as well.

Benefits of using adr

Why go to the trouble of adding this extra markup for your address information? By now the importance of the promise of standardized markup conventions for common web-based information has likely dawned on you, as it has quite a few people and organizations who have built services associated with the hCard format, from which adr is derived.

Brian Suda, whom you met in the "The geo microformat" section of this chapter, has a service called X2V, which takes a URL page with hCard (and other microformatted) information and converts it to vCard, allowing you to save it to your hard disk and open it in an address book application or other vCard-aware application. There are also extensions to Firefox, like Smartzilla and Tails, which will extract hCard and other microformatted data, presenting it in floating windows and also enabling you to download it to your hard disk. We'll look at these services in more detail in later chapters.

But perhaps the most exciting developments in this area are those that enable distributed services, such as Technorati's microformats search, as well as a number of open source libraries and toolkits for Ruby, PHP, and other development tools. These tools can enable tens, or even hundreds of thousands, of developers to easily add microformat searching and extraction features to existing and new online applications. Do you want your location (and other) data left out of this revolution?

We'll take a look at a number of these services in the next chapter, where we cover hCard in detail and we also take a closer look at what Technorati already offers by way of microformats searching, which your site can benefit from in a matter of hours, or even less.

About Technorati

Founded by David Sifry, Technorati is a search engine focusing on the blogosphere and real-time web searching. Chief Technologist Tantek Çelik, frequently mentioned in this book, is one of the most important members of the microformats community, and along with Ryan King and Kevin Marks, he has been involved with developing a number of microformats and guiding the community. Technorati has promoted and adopted numerous microformats, including the use of rel-tag (as of late 2006, Technorati was indexing more than 100 million tagged posts), hCard, and xFolk.

In May 2006, Technorati introduced its first public version of a search engine specifically for microformatted data (see `http://kitchen.technorati.com`*), along with a "distributor" for microformatted data called Pingerati (see* `http://pingerati.net`*).*

Let's start with the second of these two innovations. You've seen that one promise of microformats is distributed services—for example, an aggregator of event information published at sites all over the Web. A significant impediment to the reality of such services is the difficulty of developing a search engine to trawl the Web for microformatted content. With Pingerati, Technorati has created a service that publishers can "ping" (a geeky way of saying "inform") to let it know about new microformatted data. Obviously, this is very useful for Technorati, but Technorati also sends updates of this data to services interested in indexing them. At present, Pingerati knows about pages that contain contacts (hCard), events (hCalendar), classified listings (hListing), reviews (hReview), and link tags (xFolk.)

Technorati microformats search uses the information indexed by Pingerati to enable searching for this kind of information. But while this service works a lot like traditional index/search services, some of the services using Pingerati, such as Yahoo Tech, demonstrate the possibilities of a new age of distributed, rather than centralized, data. You learned about some of these possibilities in Chapter 2, and you'll see others pop up throughout the rest of the book, when we look at the specific microformats that help drive them.

Styling adr content with CSS

As with geo, because adr is class based, you can style its content easily with the CSS class selector. Again, one simple, useful idea is to add visual cues for address information, using either background images (which, along with the class selector, are widely supported in browsers) or the `content` property (which is less supported in browsers; Internet Explorer 7 still does not support generated content).

An adr is the first block of content you've marked up with microformats so far, so it gives us an opportunity to look at some new styling techniques. At present, it's very common to style blocks of content with rounded-corner boxes. There are a number of ways of doing this, some much cleaner than others. We'll take a look at a couple of them here.

Using the border-radius property

First we'll consider a theoretical solution, with minimal support in browsers (it has some support in Firefox 1.5 and newer). You might wonder why we would even touch on something that barely has any support, but it demonstrates the progressive enhancement principle of CSS-based design, where rather than aiming for pages that look identical in all browsers, our goal is good design, with some enhancements for browsers that support newer features. It's very important to note that using this technique, users of browsers with more limited CSS support should get access to all the content and services of a page—users of newer browsers simply have an enhanced experience. One benefit of this approach is you don't need to frequently update your site design as newer versions of browsers add support for more-advanced CSS features.

CSS3 features several new border properties associated with rounded corners on borders. The `border-radius` property allows us to add a rounded corner of a given radius to all edges of an element at once, or alternatively to individual corners. Right now, this is only supported in Gecko-based browsers, like Firefox (1.5 and higher), Mozilla (1.7 and higher), and less well-known Gecko-based browsers like Flock and Camino. Keep in mind that, for the `border-radius` property to have an effect, we need to give the element a border in the first place!

To add a border with a 1em rounded corner on each edge to our adr elements, we use the following statement:

```
.adr {
  border: thin solid black;
}
```

This statement styles our address from the previous example as shown in Figure 6-14.

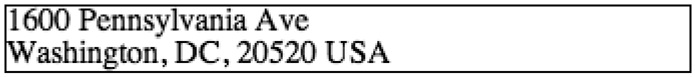
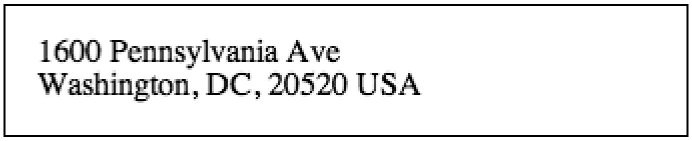

1600 Pennsylvania Ave
Washington, DC, 20520 USA

Figure 6-14. Adding a border to our adr microformat

Let's add a little padding to create some space between the border and the content:

```
.adr {
  border: thin solid black;
  padding: 1em
}
```

The result is something like Figure 6-15.



1600 Pennsylvania Ave
Washington, DC, 20520 USA

Figure 6-15. Adding some padding to our adr microformat

Now it's time to add the rounded corners. In theory, we add the CSS3 rounded corners like this:

```
border-radius: 1em
```

But as mentioned earlier, as of the time of this writing, the `border-radius` property is supported only by Mozilla-based browsers as an experimental property (and a beta version of WebKit, the rendering engine of Apple's Safari, which is also used in projects like Nokia's S60 mobile device browser and Apple's recently announced iPhone).

Now, a CSS convention for browser developers is that when implementing experimental properties, you prefix them with an identifier for that browser. Gecko-based browsers use the prefix `-moz-`, WebKit uses `-webkit-`, and so on. So it won't surprise you that the property for Mozilla-based browsers is as follows:

```
-moz-border-radius
```

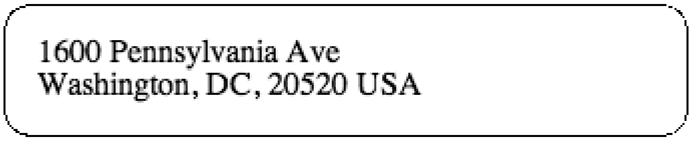
And in WebKit-based browsers (like Safari) it's this:

```
-webkit-border-radius
```

So, putting these all together, we have the following:

```
.adr {
  border: thin solid black;
  -webkit-border-radius: 1em;
  -moz-border-radius: 1em;
  border-radius: 1em;
  padding: 1em
}
```

When a future version of Mozilla supports the CSS3 `border-radius` property natively, then the `border-radius` property will override the `-moz-border-radius` property (because `border-radius` comes after `-moz-border-radius` in the style sheet cascade). In essence, we are future proofing our site. In a browser that supports rounded corners, we get the display shown in Figure 6-16.



1600 Pennsylvania Ave
Washington, DC, 20520 USA

Figure 6-16. Adding rounded corners to our adr microformat border using the CSS3 `border-radius` property

This is not yet perfect, but it's a very big step in the right direction.

Using background images

Another widely used technique for adding rounded corners with CSS uses background images on various elements to give the *appearance* of rounded corners. One problem with this technique is that developers frequently add HTML elements to their content simply to enable the technique to work, which is problematic in terms of separating content from appearance. But using your HTML intelligently can enable this technique to work in many situations without you having to add extraneous HTML simply for the sake of the technique.

There are two forms of this technique. In one, the box with the rounded corners will have a *fixed width*, requiring only two background images. In the other, we want the width of the box to grow and shrink, perhaps based on the page width (making the width of the box a percentage of the width of its parent element), or perhaps as a function of the size of the text (giving roughly the same line length in terms of characters regardless of the user's referred font size). This second **fluid**, or **elastic**, technique requires background images on four elements, and so it can be harder to pull off without adding elements simply for achieving the effect.

The secret to implementing this technique correctly is using the HTML you *already have* intelligently for styling. Let's take a different, slightly more complex adr microformatted address and use the technique to style it with a rounded corner box.

```
<div class="adr">
  <div class="street-address">2560 Ninth Street </div>
  <div class="extended-address">Suite 219</div>
  <span class="locality">Berkeley</span>,
  <span class="region">CA</span>
  <span class="postal-code">94710</span>
  <div class="country-name">USA</div>
</div>
```

This is the address of this book's publisher in the United States.[2] If we want a fixed-width (in terms of pixels) box with rounded corners, we can achieve it reasonably simply. Let's give the containing `<div>` of class `adr` a width in pixels:

```
.adr {
  width: 312px
}
```

With this, you might have already noticed a problem—the Internet Explorer box model bug. If you aren't familiar with this bug, then very simply, Internet Explorer 5, 5.5, and 6 for Windows (6 only in quirks mode, which you'll remember from Chapter 3) treat the width of an element as shown in Figure 6-17.

While this treatment seems logical, it's actually not how the CSS specification says width should be calculated, which is as shown in Figure 6-18.

2. Please send notes of praise, large sums of money, and gifts of an automotive nature, by way of thanks to the author, to this address.

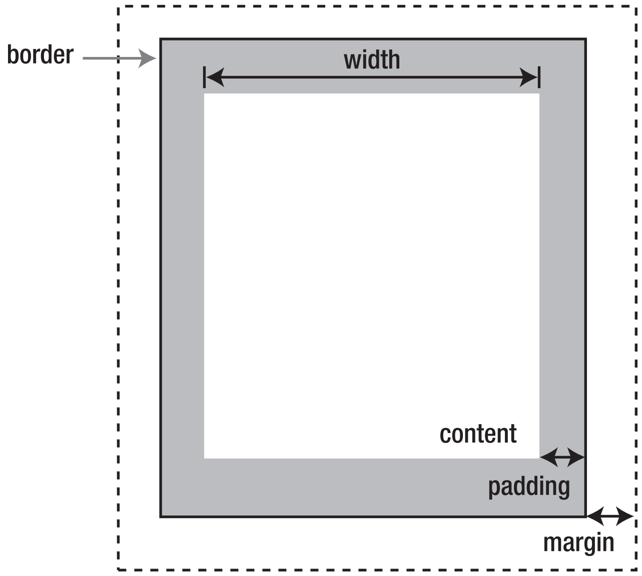


Figure 6-17. This is how older versions of Internet Explorer interpret the box model in quirks mode.

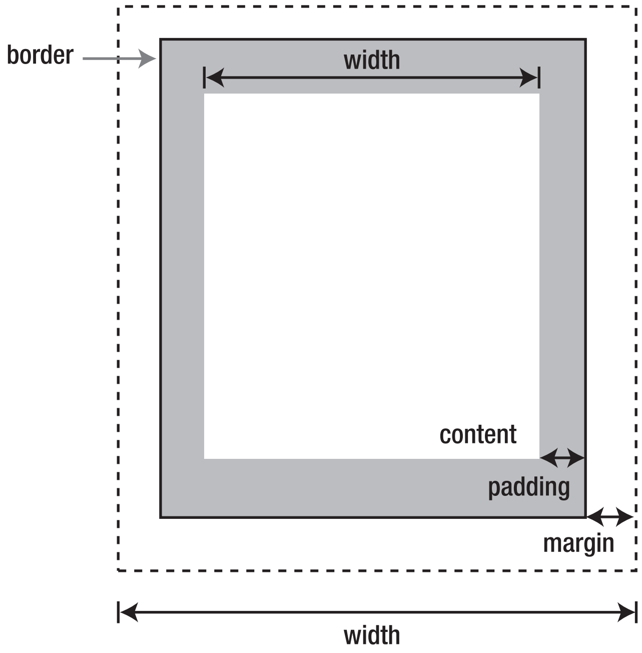


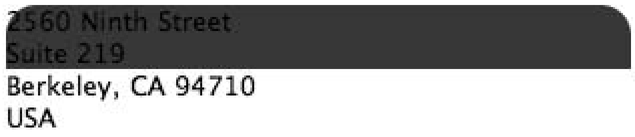
Figure 6-18. This is how the box model should be interpreted correctly, according to the W3C specification.

The problem is, if you set an explicit width as well as padding and/or margin on an element using CSS, then in older versions of Internet Explorer, the width of the element will be different from that in newer browsers. The secret here is to not set margin or padding on the element you are adding the background image to. You'll see later that if you use the fluid technique, this is not an issue.

So, let's now add a background image to the adr element, which will be the top of our rounded corner "box." We'll also stop it from repeating in either direction, as that is obviously not what we want.

```
.adr {
  width: 312px;
  background-image: url(images/fixedroundtop.png);
  background-repeat: no-repeat;
}
```

Our adr now looks like Figure 6-19.



2560 Ninth Street
Suite 219
Berkeley, CA 94710
USA

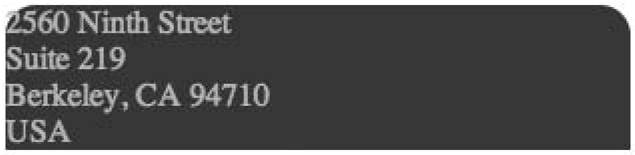
Figure 6-19. Our adr is not looking very stylish just yet, but we can fix it.

Don't panic—we aren't finished yet.

Next, we'll add some background color to the adr element, matching the color of our box's top rounded edge and giving our text some color to make it readable:

```
.adr {
  width: 312px;
  background-image: url(images/fixedroundtop.png);
  background-repeat: no-repeat;
  background-color: #333333;
  color: #b5b5b5;
}
```

Our full adr element now looks like Figure 6-20.



2560 Ninth Street
Suite 219
Berkeley, CA 94710
USA

Figure 6-20. Better, but we still have more work to do.

Obviously, we need to add some whitespace, but this is where we need to keep in mind the Internet Explorer box model bug, and actually add the whitespace to the other elements. We'll start by adding it to the `<div>` of class `street-address`.

We'll add padding to the top and left of the element like so:

```
.street-address {
  padding-top: 1em;
  padding-left: 1em;
}
```

This gives us the display shown in Figure 6-21.

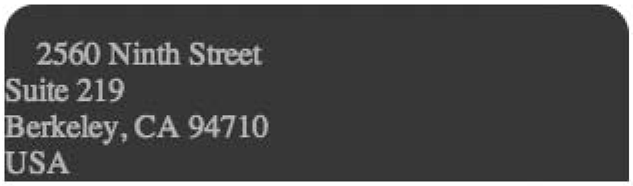

2560 Ninth Street
Suite 219
Berkeley, CA 94710
USA

Figure 6-21. Using padding to give the address some space away from the edge of the container

We similarly add padding to the left of the other elements like this:

```
.extended-address {
  padding-left: 1em;
 }
.locality {
  padding-left: 1em;
}
.country-name {
  padding-left: 1em;
}
```

This gives us the output shown in Figure 6-22.

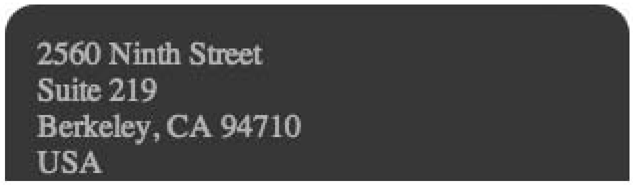

2560 Ninth Street
Suite 219
Berkeley, CA 94710
USA

Figure 6-22. After we give all our different adr fields some padding, the element looks a whole lot better.

Now it's time to add the bottom edge. We can't use the adr element, as we've already added a background image to it (in CSS3, you can add multiple background images, but this is not something supported in browsers other than Safari yet). So we need to look for another element to add this image to. Luckily for us, the country-name element provides us with a hook for our style. We'll add a nonrepeating background image to it, like we did to the top image, as follows:

```
.country-name {
  padding-left: 1em;
  background-image: url(images/fixedroundbottom.png);
  background-repeat: no-repeat;
}
```

If we take a look at the browser output now, something looks off. Where's the bottom image? It's there, but we just can't see it, because right now, the background image starts in the default position: the top-left corner of the element it is attached to. If we change the background position to bottom, here's what we get:

```
.country-name {
  padding-left: 1em;
  background-image: url(images/fixedroundbottom.png);
  background-repeat: no-repeat;
  background-position: bottom;
}
```

We're now able to see the bottom image, as shown in Figure 6-23.

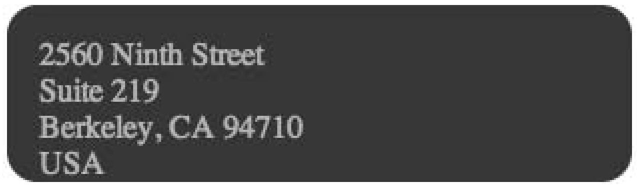

2560 Ninth Street
Suite 219
Berkeley, CA 94710
USA

Figure 6-23. Now we can see our image's bottom rounded corners.

To balance the top and bottom whitespace on the whole element, we add some padding to the bottom of the country-name element, like so:

```
.country-name {
  padding-left: 1em;
  background-image: url(images/fixedroundbottom.png);
  background-repeat: no-repeat;
  background-position: bottom;
  padding-bottom: 1em;
}
```

This gives us our nice, complete rounded-corner box, as shown in Figure 6-24.

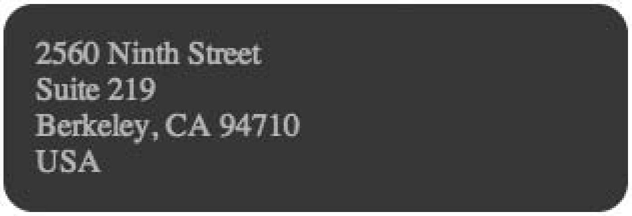

2560 Ninth Street
Suite 219
Berkeley, CA 94710
USA

Figure 6-24. Some bottom padding as a final touch gives us a nicely displayed adr microformat.

You can find the HTML, CSS, and images for this example at `http://microformatique.com/book/chapter6/adr.zip`.

In this example, we used the structure of our HTML to facilitate styling with CSS. As other adr elements will typically have a very similar structure, particularly within a single site, microformats provide us the benefit of more streamlined style sheets (in addition to their other benefits). If we set up our CSS like this and use the adr microformat for any addresses at our site, then all our addresses get a similar appearance, with essentially no more effort.

This example suffers a little from requiring the adr element to have a fixed width. In a later chapter, we'll take a look at developing fluid rounded-edge boxes using only the HTML associated with the hCard microformat.

Summary

Throughout this book, the complexity of the microformats you've been working with has steadily increased. I've been incrementally introducing concepts like the use of attributes such as `rel`; classes; and microformats design patterns, like the abbr and class design patterns, as they become appropriate.

This chapter focused on two conceptually related microformats, both of which are subsets of the more complex hCard microformat we'll turn to in the next chapter. The geo microformat allows you to mark up geodata, longitude, and latitude semantically, using HTML appropriately. You learned about the abbr design pattern, which is used to "make text that is human readable also formally machine readable," and the class design pattern, for adding the semantics of other formats like hCard to HTML.

You were also introduced to the adr microformat, a compound microformat. adr maps part of the vCard standard onto HTML, using the class design pattern.

Now it's time to start looking at some increasingly complex compound microformats, which use all of the concepts and techniques presented in this chapter. The next chapter covers hCard, which you should be quite familiar with by now—it's been used in a number of examples in this chapter, as geo and adr are subsets of it. Things are about to get really interesting.

7 CONTACT INFORMATION MICROFORMAT: HCARD

Above all else, and despite the wishes of traditional media companies, the Web is a medium for two-way communication. Technology sites like Slashdot and Digg, and political sites like Daily Kos are as much about the discussion as they are about the articles. Rare is the site that does not prominently feature a Contact link on its home page. And with blogging in particular, the two-way conversation of trackbacks and comments are essential to the medium.

But precisely how these almost ubiquitous contact details are marked up from site to site varies dramatically. This makes a seemingly simple task like building a white pages–style directory from freely published information on the Web a daunting challenge, as observed previously.

In this chapter, we'll take a close look at a microformat that aims to provide this missing, vital, common format for contact details online: hCard.

hCard overview

As you learned in Chapter 1, microformats are a whole new approach to solving the problem of how to best mark up certain types of commonly used content on the Web. This approach includes a set of underlying principles, of which two of the most important are "solve a specific problem" and "reuse building blocks from widely adopted standards."

Now, of all the kinds of content on the Web, surely one of the most common must be contact information for people and/or organizations. This information might be as simple as a link to a contact form or an email address, all the way up to a detailed set of addresses, telephone numbers, and other forms of contact. But as demonstrated in the last chapter, where we looked at a couple of subsets of this data (addresses and geographic location), HTML does not provide any built-in mechanism for marking up contact details.

Here are a couple of examples of some famous addresses and how they are marked up at their official pages:

```
<FONT FACE="Arial, Helvetica, Sans Serif" SIZE="2">
  The White House<BR>
  1600 Pennsylvania Avenue NW<BR>
  Washington, DC 20500
  <br><br>
</font>

<td valign="top" width="50%" rowspan="2">
  <b>Address</b><br>
  Please include the recipient's first and last name,
  or the department name.<br>
  Microsoft Corporation<br>
  One Microsoft Way<br>
  Redmond, WA 98052-6399</td>
```

I've included these just to show how little consistency there is in marking up contact information—that is, apart from using `<br>` elements to separate lines.

Enter hCard. The hCard microformat solves the specific problem of marking up contact information for people or organizations, reusing building blocks from a widely existing standard: vCard. But in fact, hCard goes even further, providing us with a more semantic way of marking up people, organizations, and places, not just in the context of contact information. So, whenever we refer to a person, place, or organization, we can mark these up using hCard.

I've touched on vCard, the existing standard that hCard is based on, a number of times in this book, but let's take a closer look at it here. vCard is a standard from IETF, an organization responsible for a wide range of standards used on the Internet. IETF's stated goal is "to make the Internet work better." The IETF standards are published as Requests for Comment (RFCs), one of which, RFC2426 (`www.ietf.org/rfc/rfc2426.txt`), is vCard, described as "directory information for a white–pages person object." It's also often described as an "electronic business card" format.

vCard is widely used in desktop address book–style applications such as Microsoft Outlook and Mac OS X's Address Book. Figure 7-1 shows Apple's vCard, as seen in Address Book.

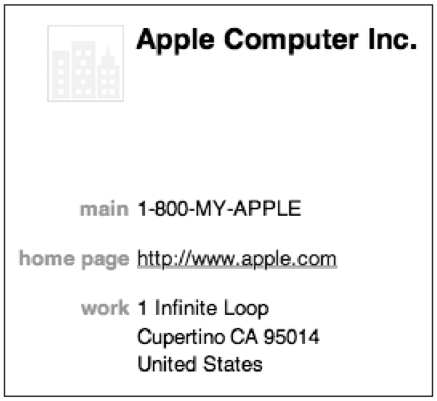


Figure 7-1. A vCard in Apple's Address Book application

Here's what the vCard code looks like:

```
BEGIN:VCARD
VERSION:3.0
N:;;;;
FN:Apple Computer Inc.
ORG:Apple Computer Inc.;
TEL;type=MAIN;type=pref:1-800-MY-APPLE
item1.ADR;type=WORK;type=pref:;;
1 Infinite Loop;Cupertino;CA;95014;United States
```

```
item1.X-ABADR:us
item2.URL;type=pref:http\://www.apple.com
item2.X-ABLabel:_$!<HomePage>!$_
X-ABShowAs:COMPANY
X-ABUID:B4DB7A3B-3E21-4E5A-9014-3745ECAC3CCD\:ABPerson
END:VCARD
```

vCard provides both a schema and the semantics for publishing structured contact information. It just doesn't provide a particularly human-friendly format for doing so. hCard explicitly adopts both this schema and the semantics of vCard as they pertain to its possible values, or "fields," but it loosens the association by allowing hCard to be used for marking up people, organizations, and places, not simply in the context of contact details.

vCard has a rich set of properties that enable publishers to mark up the following, among other things:

- Identification details, such as name, nickname, photos, date of birth, and so on
- Postal contact details
- Electronic contact details, such as email address(es) and telephone number(s)
- Place and time details, such as geographic location and time zone
- Information associated with an organization, such as the organization someone is a member of or works for, and his or her role and title

hCard, being modeled explicitly on vCard, allows us to mark up any or all of this information, though typically only a small subset is used in most hCards.

Whenever we publish information about people or organizations, particularly in the context of contacting them, hCard is the ideal markup microformat.

Using hCard

As with other compound microformats you've seen (and indeed all compound microformats), hCard has a root element that contains all of the properties for that hCard. This can be any HTML element (though typically it will be a span or a div) with a `class` value of `vcard` (note that we don't use `hcard` as the `class` value).

We then use the class design pattern to give HTML elements inside this root element richer semantics taken from the vCard specification. You saw this in action a couple of times in the previous chapter, as both the adr and geo microformats are a subset of hCard.

In the sections that follow, we'll examine ways of using hCard to mark up various types of contact information. We'll start by looking how to use hCard to mark up the names of people.

Names of people

vCard has a number of name-related fields. These aren't all particularly human-friendly, but they are easy enough to get the hang of. There are three name elements (or technically "types") in vCard: FN, N, and nickname. Here's how they work and what they do:

- FN, or "formatted name," is for how a name should appear for the hCard. My FN would be John Allsopp.
- N, or "structured name," is a name with prefixes, suffixes, middle names, and so on. For example, I might have an N value of Mr. John Francis Allsopp. You'll see shortly how to actually encode these in HTML.
- Nickname is for a shorthand name for the hCard. In my case, the nickname might be "Jack" (which my brothers use, but no one else) or "Sweep" (which a particular set of friends uses).

Very often, all that will be required for an hCard is an FN value, so we'll start with that. We know the root element is an element with a `class` value of `vcard`. Because we aren't sure yet what elements it will contain, let's make this a `<div>`, as it can then contain both block and inline elements.

```
<div class="vcard"></div>
```

Next, we need to add an element with a `class` value of `fn` . If we think about this, it's most likely to be displayed as an inline element, so let's make it a `<span>`:

```
<div class="vcard">
  <span class="fn">Buffy Summers</span>
</div>
```

You'll notice that throughout this chapter I present some vCard properties like FN in capital letters, but I've just marked up this property using lowercase letters for the class value. Because XHTML is case sensitive, by convention we use lowercase for class and other attribute values in hCard and other microformats, but in vCard, property names are defined in capital letters.

That's all we need for an hCard for this person, and in many contexts, this would be quite useful. For example, we could mark up the speakers at a conference with hCards just like this (in fact, many conferences do exactly this). But, of course, we can take hCards much further by using the schema of vCard.

To recap, FN is for the formatted name of our person, organization, or place (we'll get to entities other than people shortly), but sometimes the "display name" is different from the actual name. For a person's actual name, we have the N property, which is used in conjunction with several subproperties. When you use the N property, you *must* use one or more subproperties with it. These subproperties, like all other properties of hCard, come from vCard and are as follows:

- `family-name`
- `given-name`
- `additional-names`
- `honorific-prefixes`
- `honorific-suffixes`

If we want the full, structured name of this person, we'd have the following:

```
<div class="vcard">
  <span class="n">
    <span class="honorific-prefixes">Ms.</span>
    <span class="given-name">Buffy</span>
    <span class="additional-names">Anne</span>
    <span class="family-name">Summers</span>
  </span>
  <span class="honorific-suffixes">V.S.</span>
</div>
```

Note the difference with the FN property: whereas FN does not have subproperties, N does (and must). So, while the following is valid hCard:

```
<span class="fn">Buffy Summers</span>
```

this is not:

```
<span class="n">Ms. Buffy Anne Summers V.S.</span>
```

The hCard *must* be written as follows:

```
<span class="n">
  <span class="honorific-prefixes">Ms.</span>
  <span class="given-name">Buffy</span>
  <span class="additional-names">Anne</span>
  <span class="family-name">Summers</span>
</span>
```

If this is also to be the FN (i.e., how the full structured name is to be displayed), then rather than duplicating it all with a second span with a class of `fn`, we just give the `n` span a second class, `fn`, resulting in the following:

```
<div class=vcard>
  <span class="fn n">
    <span class="honorific-prefixes">Ms.</span>
    <span class="given-name">Buffy</span>
    <span class="additional-names">Anne</span>

    <span class="family-name">Summers</span>
    <span class="honorific-suffixes">V.S.</span>
  </span>
</div>
```

The vCard specification requires that all vCards have both N and FN values. However, since this may be overkill in many circumstances, we can leave out the N value in the case of people (we'll look at the special case of organizations shortly). If we have one of the following situations for a person, we can use FN alone, and the value for N is inferred from the FN value.

If the value of the FN property is exactly two words (separated by whitespace), and there is no explicit N property, then the N property is inferred from the FN property as follows:

- The content of the FN is broken into two "words" separated by whitespace.
- The first word of the FN is interpreted as the `given-name` for the N property.
- The second/last word of the FN is interpreted as the `family-name` for the N property.
- However, if the first word ends in a comma (`,`) *or* if the second word is a single character (optionally followed by a period, `.`), then the first word (minus the comma at the end, if any) is interpreted as the `family-name` and the second word is interpreted as the `given-name`.

While seemingly complicated, these are very commonsense rules, with the result that we often do not have to have an N value for our hCards. If N values were required, hCard would be a much more cumbersome construct to use in many situations.

In practice, the following:

```
<span class="fn">Buffy Summers</span>
```

is semantically equivalent to this:

```
<span class="n">
  <span class="given-name">Buffy</span>
  <span class="family-name">Summers</span>
</span>
```

And the following is equivalent to the preceding examples:

```
<span class="fn">Summers, Buffy</span>
```

whereas the following:

```
<span class="fn">Summers, B</span>
```

is equivalent to this:

```
<span class="n">
  <span class="given-name">B</span>
  <span class="family-name">Summers</span>
</span>
```

In a great many cases, constructs such as `<span class="fn">Buffy Summers</span>` will suffice. Only in more formal circumstances, such as an address book entry or a white pages–style directory entry, will the N property be required.

I should touch on the nickname property , too, while considering the issue of names. People use nicknames, or “handles,” commonly on the Web. vCard, and so hCard, has an explicit property for this, which we could use like this:

```
<span class="nickname">Bono</span>
```

But remember, an FN is required for hCard, so we could add it as follows:

```
<span class="fn nickname">Bono</span>
```

There is one more “optimization” possible with hCard: if an FN has a single-word value, we imply that this is a nickname. So we could optimize the preceding as follows:

```
<span class="fn">Bono</span>
```

To state this technically, if the value of the FN property is one word, we treat it as a nickname, and we consider that there is an N value with no content. So, for our Buffy hCard, we would now have this:

```
<div class=vcard>
  <span class="fn n">

    <span class="honorific-prefixes">Ms.</span>
    <span class="given-name nickname">Buffy</span>
    <span class="additional-names">Anne</span>
    <span class="family-name">Summers</span>
    <span class="honorific-suffixes">V.S.</span>
  </span>
</div>
```

Note that I’ve reused the given name and nickname value by adding the nickname class value to the given-name subproperty of N.

URLs

Not surprisingly, vCard includes a property for a URL. The most obvious way for us to add a URL to an hCard is to add a separate field like this:

```
<span class="url">http://en.wikipedia.org/wiki/Buffy_Summers</span>
```

But, of course, this doesn’t really make the link all that human-readable. What we as humans want from a link is to be able to follow it. So, why not make the hCard’s FN also a link? Let’s go back to the original hCard we were developing:

```
<div class="vcard">
  <span class="fn">Buffy Summers</span>
</div>
```

If we change the inner span to a link and give it a class of url as well as an FN, we have this:

```
<div class="vcard">
  <a href="http://en.wikipedia.org/wiki/Buffy_Summers" class="fn url">
Buffy Summers</a>
</div>
```

Here we have a human-readable link (we can see it is a link, and we can also follow it), as well as a machine-readable URL. And we've relied on our N optimization rules to keep the hCard as simple as possible.

You might be wondering why you actually need the url class value. After all, even a machine can see it is a URL. The issue here is that you may want more than one link in your hCard—for example, an email link, or links to an organization, a map for an address, and so on. By giving this link the class url you are saying that this link is *a URL* for this hCard. And you may want to have multiple URLs for a single person (which is permitted in vCard, and so hCard). This "trick" is used to provide ways of encoding instant messenger addresses, which you'll see in a moment.

Back to our growing hCard for Buffy, we now have the following:

```
<div class="vcard">
  <a class="fn n url"
     href="http://en.wikipedia.org/wiki/Buffy_Summers">
    <span class="honorific-prefixes">Ms.</span>
    <span class="given-name nickname">Buffy</span>
    <span class="additional-names">Anne</span>
    <span class="family-name">Summers</span>
    <span class="honorific-suffixes">V.S.</span>
  </a>
</div>
```

We've taken care of the basics of an hCard for a person (and in fact have a perfectly valid hCard). Now let's start extending it.

Date of birth

What self-respecting address book entry would be complete without a date of birth? vCard has a date of birth field named BDAY.[1]

1. The French might find this a little amusing.

If you are wondering why vCard has all these far from human-friendly field names like N, FN, and BDAY, it's because back when vCard was invented, people still used to worry about things like the size of data structures, due to low-bandwidth network connections and considerably smaller disk sizes. vCard began life in 1992, when a 50MB hard drive was probably on the large side, 10MB of RAM nothing to be sneezed at, and a 14Kb (yes, kilobit) modem was state of the art. Back then, we worried about a few bytes here and there.

What I don't quite understand is why occasionally field names get very profligate. I mean, "nickname" is surely at least twice as big as it has to be—wouldn't "NN" suffice? That's a story for another day.

The obvious way for us to add a date of birth is something like the following:

```
<span class="bday">5 11 1981</span>
```

But it's clear we already have a problem. Well, several problems. Is this May or November? Is this 1981 BCE or CE, and are we going by the Jewish, Buddhist, Gregorian, or Julian calendar? These sound like pedantic issues, because we humans can typically deduce the meaning from the context. But we also know that software is not nearly so smart as us. We need to spell out everything for our silicon-powered friends.

Humans can reasonably easily disambiguate something like this:

```
<span class="bday">November 5 1981</span>
```

Given that the date is the birthday of someone in the United States, even if he or she is a vampire slayer, we can reasonably guess that the date is according to the Gregorian calendar (the commonly used calendar in the Western world). But of course, machines aren't quite so clever at working this sort of thing out, and we need to mark up this information in a way to help them.

We've already seen with microformats a way for presenting information that is both human- and machine-readable, using the abbr design pattern introduced with the geo microformat in the previous chapter. With this design pattern, we use an `<abbr>` element; the human-readable part is the *content* of the `<abbr>` element, while the machine-readable part is stored in the `title` of the `<abbr>` element.

Back to our example, Buffy was born on January 19, 1981, so that gives us the following basic HTML:

```
<abbr title="" class="bday"> Jan 19 1981</abbr>
```

It just remains for us to put the date in a machine-readable format. I don't want to go anywhere near the controversy this problem has caused over the years, but suffice it to say that the International Organization for Standardization (ISO) has a standard format for date and time interchange:

```
YYYYMMDDThh:mm:ss
```

And here is the (slightly) more readable "extended" format:

```
YYYY-MM-DDThh:mm:ss
```

This format uses the Gregorian calendar, and all dates must be the same length (so single-digit dates and times are padded with zeros: 4:30 a.m. becomes 04:30, and 5 November is 1105). Why does the month come before the day? Well, the cynical might say it's because that's how they do it in the United States, but the format has its logic: it goes from the most significant to the least significant values, from years through to seconds.

The time part can be left off, which for birth dates makes sense. Our BDAY element becomes

```
<abbr title="19810191" class="bday"> Jan 19 1981</abbr>
```

So, we have a human-friendly and yet machine-readable date, using the <abbr> design pattern for dates.

You'll see quite a bit more on dates in the next chapter during the discussion of hCalendar, so don't fret if you don't have the concept down perfectly yet.

Building on our existing hCard, we get this:

```
<div class="vcard">
  <a class="fn n url"
     href="http://en.wikipedia.org/wiki/Buffy_Summers">
    <span class="honorific-prefixes">Ms.</span>
    <span class="given-name nickname">Buffy</span>
    <span class="additional-names">Anne</span>
    <span class="family-name">Summers</span>
    <span class="honorific-suffixes">V.S.</span>
  </a>
  <p><abbr title="19810191" class="bday"> Jan 19 1981</abbr></p>
</div>
```

Photos

We can add a photo for the person whose hCard this is using the PHOTO property of vCard. A photo is an image, and as we are discussing content on the Web, it makes sense to link to a web-based image for this property. We'll have an image element with a class of photo and an src value of the URL to this image. And, of course, we'll have some alt text to make it accessible. This makes our ever-growing hCard for Buffy look like this:

```
<div class=vcard>
  <a class="fn n url"
href="http://en.wikipedia.org/wiki/Buffy_Summers">
    <span class="honorific-prefixes">Ms.</span>
    <span class="given-name nickname">Buffy</span>
    <span class="additional-names">Anne</span>
    <span class="family-name">Summers</span>
    <span class="honorific-suffixes">V.S.</span>
  </a>
  <img class="photo" src="http://en.wikipedia.org/wiki/
Image:Buffy_Summers.jpg" alt="Buffy Summers" />
  <p><abbr title="19810191" class="bday"> Jan 19 1981</abbr></p>
</div>
```

Organizations

There are two ways in which an organization like a company might be associated with an hCard. One is in the hCard for the organization, and the other relates to the role that the person for whom this is the hCard plays at that organization (i.e., as an employee or in some other capacity). In this section, we'll look at the hCard for a person at an organization first, and then we'll look at the hCard for an organization itself.

hCard for a person at an organization

For individuals, we can specify the organization (or organizations) in which that person plays a role, the title the person holds at the organization, and even a logo for the organization. In most cases, it's very straightforward—we simply add new elements for the properties we want to add to our hCard.

First, we can add the organization someone works for (or owns, or has some other association with). The vCard property for this is ORG. Here's our example, using the class design pattern:

```
<span class="org">Scooby Gang</span>
```

I'm not quite sure what title Buffy has in this gang, but she certainly seems to be some kind of leader, so let's make her the CEO (people seem to be able to be the CEO of just about anything these days):

```
<span class="title">CEO</span>
```

But Buffy's *role* is not the same—she's a vampire slayer, so we give her this role in addition to the title:

```
<span class="role">Vampire Slayer</span>
```

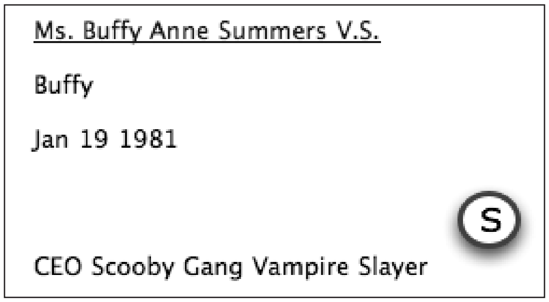
If we want to add the logo for the organization, we can do that, too. It makes sense that we'll link to an image file for the logo, so it will be an `<img>` element, with a class of `logo` and an `href` containing the URL of the image file. As far as I am aware, the Scooby Gang, being a secretive sort of outfit, doesn't have a logo (though that hardly stopped the Thunderbirds, or SMERSH, for that matter), so we'll have to pretend they have one for the sake of the example:

```
<img class="logo" src="http://www.scoobygang.com/logo.gif"
alt="Scooby Gang secret logo" />
```

Our increasingly complex hCard now looks like this:

```
<div class=vcard>
  <a class="fn n url"
href="http://en.wikipedia.org/wiki/Buffy_Summers">
    <span class="honorific-prefixes">Ms.</span>
    <span class="given-name">Buffy</span>
    <span class="additional-names">Anne</span>
    <span class="family-name">Summers</span>
    <span class="honorific-suffixes">V.S.</span>
  </a>
  <p class="nickname">Buffy</p>
  <p><abbr title="19810191" class="bday"> Jan 19 1981</abbr></p>
  <span class="title">CEO</span>
  <span class="org">Scooby Gang</span>
  <span class="role">Vampire Slayer</span>
  <img class="logo" src="http://www.scoobygang.com/logo.gif"
alt="Scooby Gang secret logo" />
</div>
```

If we view this in a browser (see Figure 7-2), it doesn't look all that nice, but we'll take care of that a bit later with some CSS.


Ms. Buffy Anne Summers V.S.

Buffy

Jan 19 1981

CEO Scooby Gang Vampire Slayer

Figure 7-2. Our completed Buffy hCard, as viewed in a browser

I told you it wouldn't be pretty. (And what's with that logo? Look, the Scooby Gang, despite its ironic pretensions, is a really lame name, so the logo befits the name.)

hCard for an organization

For the most part, we can think of organizations as being like people—they have names, addresses, phone numbers, and so on. Some properties, like nicknames or photos, are perhaps less relevant to companies, and so are less likely to be used in an organization hCard. And other properties, like role and title, probably don't make sense at all for organizations. But on the whole, we can think of organizations as being very similar to people when it comes to hCards.

The biggest challenge is determining when an hCard is for a person and when it is for an organization. To create an hCard for a company (or other organization), all we need to do is set the value of the FN and ORG properties to the same thing, which typically we do by having one element with the two class values set on it, like this:

```
<span class="fn org">Scooby Gang</span>
```

It's also important that no value for the N property is set in the hCard for an organization, but that's really about all we have to address directly for an hCard for an organization as opposed to one for a person. We mark up other properties, such as addresses, URLs, and so on, exactly as we would for a person.

Addresses

In the last chapter, when examining the adr and geo microformats, you saw that these are, in fact, parts of the vCard specification, and thus the hCard microformat. You just used them in isolation, reinforcing the microformats principles of modularity and embeddability discussed in the first chapter. As such, there's no need to go into great detail about them here.

Here's our example of an address from the previous chapter:

```
<div class="adr">
  <div class="street-address">2560 Ninth Street </div>
  <div class="extended-address">Suite 219</div>
  <span class="locality">Berkeley</span>,
  <span class="region">CA</span>
  <span class="postal-code">94710</span>
  <div class="country-name">USA</div>
</div>
```

All we need to do is embed this in an hCard to give the person or organization an address. In Buffy's case, the address is as follows:

```
<div class="adr">
  <div class="street-address">1630 Revello Drive</div>
  <span class="locality">Sunnydale</span>,
  <span class="region">CA</span>
  <div class="country-name">USA</div>
</div>
```

But it's not always quite as simple as that all the time. Many of us have more than one address—for the place we live, for the place we work, and perhaps for another, separate postal address as well (e.g., P.O. box).

To deal with this in vCard, addresses, email addresses, and telephone numbers have an optional TYPE property, which takes a value from a list of possible values. For addresses, the possible values are as follows:

- `intl`: An international delivery address
- `postal`: A postal delivery address
- `parcel`: A parcel delivery address

- work: A delivery address for a place of work
- dom: A domestic delivery address
- home: A delivery address for a residence
- pref: The preferred delivery address when more than one address is specified

Of these, we are most likely to use home, work, and postal. So how do we mark up our postal addresses if we have two or more addresses for our hCard—say, postal and work addresses? To add the type value, we add a <span> element with a class of type, and the element's content is the value. For example, to denote our postal address we use the following:

```
<div class="adr">
  <span class="type">postal</span>:

  ...

</div>
```

Also, if we have more than one address, we simply create separate adr divs for each. For example, if we have a postal address and a work address, our markup is along these lines:

```
<div class="adr">
  <div class="type">Work</div>:
  <div class="street-address">2560 Ninth Street </div>
  <div class="extended-address">Suite 219</div>
  <span class="locality">Berkeley</span>,
  <span class="region">CA</span>
  <span class="postal-code">94710</span>
  <div class="country-name">USA</div>
</div>

<div class="adr">
  <div class="type">Postal</div>:
  <div class="street-address">1855 Haight Street </div>
  <span class="locality">San Francisco</span>,
  <span class="region">CA</span>
  <span class="postal-code">94117</span>
  <div class="country-name">USA</div>
</div>
```

Places

In the previous chapter, you learned about the geo microformat, which is for adding information about a geographic location using longitude and latitude. To use geo in an hCard, you just take the geo content and embed it in your hCard. Using Multimap, it's easy to discover that the preceding work address translates to a latitude of 37.8688 and a longitude of –122.2976. Now, this is most likely something we would provide for machine-readable rather than human-readable purposes, so we'll use the abbr design pattern like so:

```
<abbr class="geo" title="37.8688; -122.2976">Work</abbr>
```

which gives us an address element like this:

```
<div class="adr">
  <div class="type">
    <abbr class="geo" title="37.8688; -122.2976">Work</abbr>
  </div>:
  <div class="street-address">2560 Ninth Street </div>
  <div class="extended-address">Suite 219</div>
  <span class="locality">Berkeley</span>,
  <span class="region">CA</span>
  <span class="postal-code">94710</span>
  <div class="country-name">USA</div>
</div>
```

Telephone numbers

Do people still call one another in this age of instant messaging, text messaging, email, and other forms of electronic communication? I guess they still do so enough that telephone information is important for contact details. Adding a single phone number is, like most things in hCard, straightforward. An element with a class value of `tel` contains our phone number, and the number itself is the value of this field, for example:

```
<div class="tel">+61 2 9130 1731</div>
```

If, as is often the case, you have more than one phone number, you might want to label the type of each phone number. As with addresses, vCard allows a number of specified types for telephone numbers:

- `voice`: The default type
- `home`: Telephone number associated with a residence
- `msg`: Telephone number that has voice messaging support
- `work`: Telephone number associated with a place of work
- `pref`: Preferred-use telephone number
- `fax`: Facsimile telephone number
- `cell`: Cellular telephone number
- `video`: Video conferencing telephone number
- `pager`: Paging device telephone number
- `bbs`: Bulletin board system telephone number
- `modem`: A modem-connected telephone number
- `car`: Car phone telephone number

- isdn: An ISDN service telephone number
- pcs: Personal communication services telephone number

To add both a work and cell phone number, we use the following:

```
<div class="tel"><span class="type">Work</span>:
 <span class="value">+61 2 9130 1731</a></div>
<div class="tel"><span class="type">Cell</span>:
 <span class="value">+61 504 149 597</a></div>
```

Notice that we add a special element, in this case a span, with the class of value, for the value of the telephone property, along with the span of class type, whose content is the type of the phone number taken from the list of telephone number types defined for vCard and enumerated previously. This is similar to how we marked up an address with a type and value.

Email addresses

It should come as no surprise that email addresses work in a similar way to phone numbers in vCard and thus hCard. We can simply add an email address like so:

```
<a class="email" href="mailto:john@westciv.com">john@westciv.com</a>
```

Again, if there is more than one email address, we can mark them up similarly to telephone numbers, once more using the type property. But here is one place that vCard shows its age to an extent: with the set of possible email types. vCard (and so hCard, because hCard takes its semantics from vCard) has the following possible email types:

- INTERNET: An Internet addressing type (the default, and almost certainly the only type people will typically use)
- x400: An X.400 addressing type
- pref: A preferred-use email address when more than one is specified
- Other IANA registered address types

Here the origins of vCard, which predates the widespread adoption of the Internet, become obvious. It seems no one imagined I might have a work and home email address, just as I have work and home phone numbers. In effect, this means that unlike with telephone numbers and addresses, email addresses have limited scope to be marked up more semantically in hCard, using the schema from vCard. We can still have multiple email addresses in an hCard with no problem, but perhaps the best we can do is to mark them up like this:

```
<ul>
  <li>
    <a class="email" href="mailto:john@westciv.com">
      john@westciv.com
    </a>
  </li>
  <li>
    <a class="email" href="mailto:john@webdirections.org">
      john@webdirections.org
    </a>
  </li>
</ul>
```

Representing new interweb technologies with hCard

As previously mentioned, vCard, which hCard is based on, has been around for some time. While this means vCard is tried, tested, and widely adopted, it also means that some of the newer Net-based ways of communicating, like instant messaging, were simply unimagined at the time of its development, and so are not catered for explicitly in vCard. So how can we mark up various chat addresses, Skype handles, and other Voice over IP (VoIP) handles and so forth in hCard?

You might think this would require extensions to vCard, but by using the URL property intelligently and creatively, and by looking at current practices among web developers, we can use protocol prefixes like `aim:` (much like `mailto:`) to indicate AOL Instant Messenger (AIM) and similarly other, newer protocols.

For example, an AIM ID could be represented like this:

```
<a class="url" href="aim:goim?screenname=someguy">Say hi on AIM</a>
```

or a Yahoo ID like this:

```
<a class="url" href="ymsgr:sendIM?SomeGuy">
Say hi on Yahoo instant messenger</a>
```

or a Skype handle like this:

```
<a class="url" href="callto:myskypehandle">skype me</a>
```

We haven't *invented* these uses of protocols; rather, we've *adapted* them from current practice, again adhering to the principles of microformats. And if you have these (and other) protocols set up in your browser with the applications you want to use to handle them, simply clicking these links will launch or bring to the front the right handler application and initiate the call, chat session, or other process.

There is, in fact, a current proposal to add instant messaging to vCard, but such processes can take a considerable amount of time. Should instant messaging be adopted in vCard, I've little doubt that future versions of hCard would adopt this new aspect of the vCard schema.

Tools for working with hCard

Now that I've taken you through all the details of how to put hCards together on your web pages, let's have a look at what tools are available to make working with hCards easier, both in terms of harvesting them and creating them.

Web-based tools/extensions for harvesting hCards

Because hCard is a kind of metadata (it's not always apparent that contact details, for instance, are marked up as hCard), a couple of interesting web-based tools are available to help locate and extract hCard content (and other microformatted content) from the pages you are reading.

The following two closely related (and confusingly similarly named) Firefox extensions can extract hCard, hCalendar, and hReview content as you read a page, and allow you to view it in a separate pop-up window within the browser window. They're also great for testing whether your hCards are working properly.

- **Tails** (by Calvin Yu): `http://blog.codeeg.com/tails-firefox-extension-03`
- **Tails Export** (by Robert DeBruin): `https://addons.mozilla.org/firefox/2240`

Another tool that works in a similar fashion is Brian Suda's X2V (`http://suda.co.uk/projects/X2V`), which extracts hCards and other microformatted content from web pages and saves them as vCards.

In the following sections, we'll look at the Tails Export extension and X2V, and in the next chapter, we'll delve into the Tails extension.

Tails Export extension

If you have Tails Export installed, when you visit pages that have microformatted content, the icon in Figure 7-3 appears in the bottom-right corner of your browser.

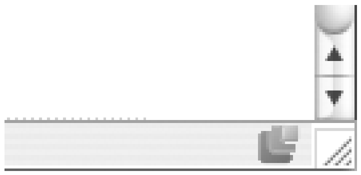
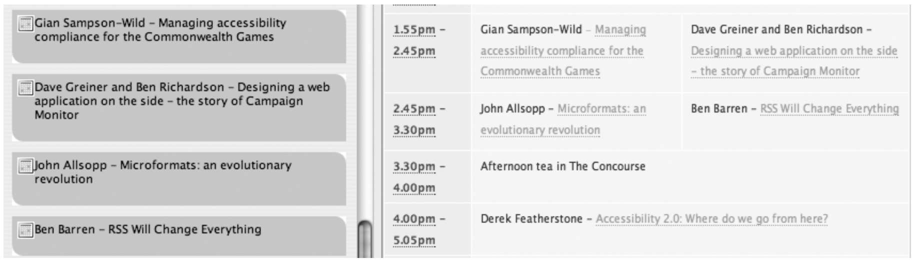


Figure 7-3. Icon indicating that microformats are present on the web page being viewed

Click the icon, and a summary of all of the microformatted content in the page is displayed in a sidebar, as shown in Figure 7-4 (if you leave this sidebar open, it is updated when you visit other pages).



Gian Sampson-Wild – Managing accessibility compliance for the Commonwealth Games

Dave Greiner and Ben Richardson – Designing a web application on the side – the story of Campaign Monitor

John Allsopp – Microformats: an evolutionary revolution

Ben Barren – RSS Will Change Everything

1.55pm – 2.45pm	Gian Sampson-Wild – Managing accessibility compliance for the Commonwealth Games	Dave Greiner and Ben Richardson – Designing a web application on the side – the story of Campaign Monitor
2.45pm – 3.30pm	John Allsopp – Microformats: an evolutionary revolution	Ben Barren – RSS Will Change Everything
3.30pm – 4.00pm	Afternoon tea in The Concourse	
4.00pm – 5.05pm	Derek Featherstone – Accessibility 2.0: Where do we go from here?	

Figure 7-4. Tails Export in action, pulling up the microformats it has found on the current web page

On the right of Figure 7-4 is the page itself, with the program for the conference marked up using the hCalendar microformat. On the left, the microformatted content is displayed.

A similar extension for Firefox is Smartzilla (`www.stripytshirt.co.uk/features/firefox/smartzilla`*), which enables you to extract microformatted content from a page. In addition, Chris Casciano has a script for NetNewsWire (the Mac OS X RSS application), which allows users to extract hCard (and hCalendar) content as vCards and iCalendar files (see* `http://placenamehere.com/mf/nnwextract`*).*

Just released as this book was being finalized is the Operator extension for Firefox, by Michael Kaply of IBM (`https://addons.mozilla.org/firefox/4106`*). While I don't have the space to go into this extension and all its functionality in detail, it has already caused a good deal of excitement and is considered by many to point the way for future incorporation of microformats support directly inside the next generation of web browsers, such as Firefox 3.*

X2V

As previously mentioned, Brian Suda's online X2V tool extracts hCards and other microformatted content from web pages and saves them as vCards. X2V is open source, so you can incorporate it into online services that need to parse such content.

What's particularly cool about X2V is that you can use it to add a downloadable vCard (or, as you'll see in the next chapter, an iCalendar file, as supported by Apple's iCal, Microsoft Outlook, and other calendaring applications). Of course, you can readily make a `.vcf` vCard file available online for downloading, but this approach actually converts the content published on your page when it's requested by the user. With X2V, there's no need to keep two file versions synchronized and no possibility that the linked `.vcf` file might get out of date.

Here's an example of X2V in action. At my conference company's site, we have contact information published using hCard (you'll see this in more detail shortly when we look at styling hCards using CSS). If I want to make a downloadable version of this information, here's what I need to do.

I start by creating a link:

```
<a href="">Download contact details as a vCard</a>
```

Next, I create the URL for the X2V service, beginning with the URL to X2V itself:

```
http://suda.co.uk/projects/X2V/get-vcal.php?uri=
```

Then I just append the URL for the page with the hCard on it:

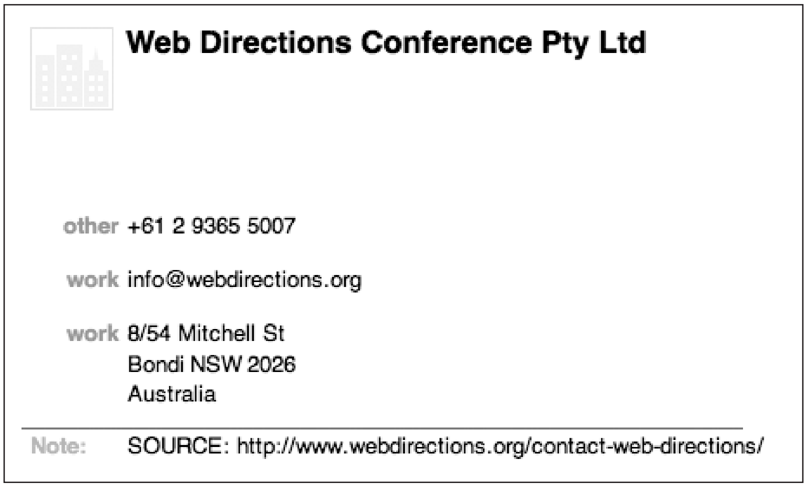
```
http://webdirections.org/contact-web-directions/
```

which gives me the following:

```
<a href="http://suda.co.uk/projects/X2V/get-
vcard.php?uri=http://www.webdirections.org/contact-web-directions/">
Download contact details as a vCard</a>
```

To test this, just follow the link (or paste the URL into the address field of your browser). At this point, depending on your operating system and browser, a number of things will happen.

On Mac OS X with Safari, the vCard is downloaded and automatically added to Address Book, displaying an entry like the one shown in Figure 7-5.


Web Directions Conference Pty Ltd

other +61 2 9365 5007

work info@webdirections.org

work 8/54 Mitchell St
Bondi NSW 2026
Australia

Note: SOURCE: http://www.webdirections.org/contact-web-directions/

Figure 7-5. The vCard created from the Web Directions contact hCard using X2V

With Firefox and other Mozilla browsers, and on Windows, you may have to manually open the file, but other than that, the process is basically the same. The only thing to keep in mind is that if you are likely to have a lot of downloads, it's probably best to grab the code and run the service from your own servers. The code is published under a W3C open source license (see `www.w3.org/Consortium/Legal/copyright-software-19980720`).

Tools for creating hCards

For publishing and working with hCard content, a number of very helpful tools are available. I describe some of these in the sections that follow.

Publishing

For Dreamweaver users, Drew McLellan has developed a toolbar that easily enables creating hCard and other microformatted content. It's available from the Web Standards Project at `www.webstandards.org/action/dwtf/microformats`. Once installed, the Insert bar for Dreamweaver has a new set of features added to it, labeled "Microformats," as shown in Figure 7-6.

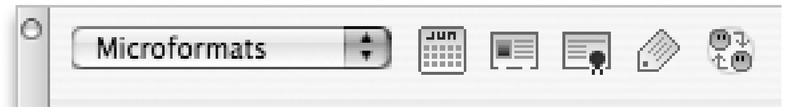
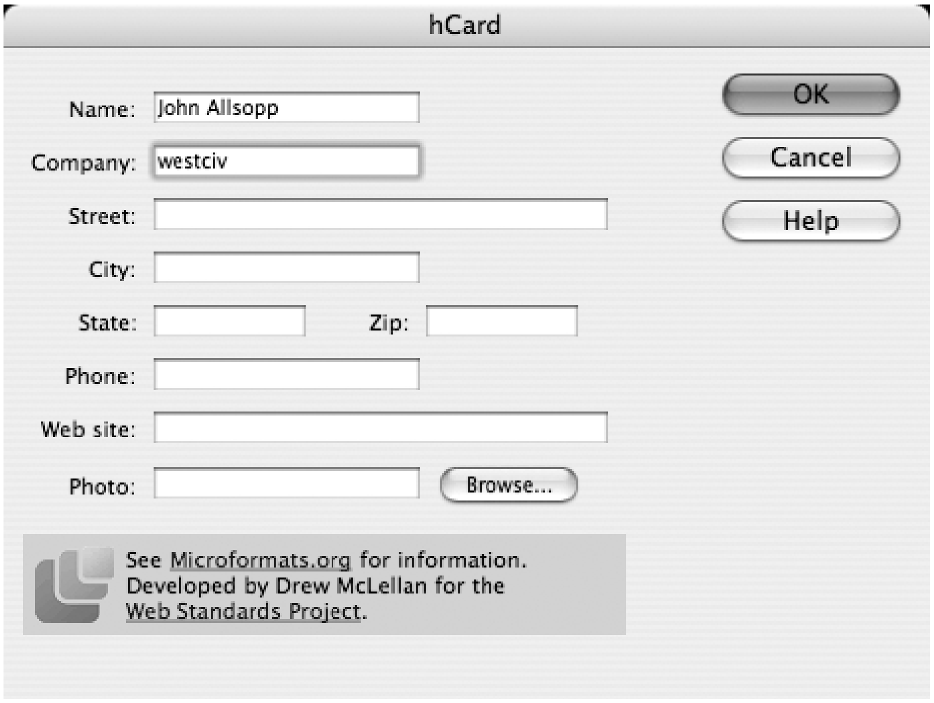


Figure 7-6. Drew McLellan's Microformats toolbar adds a special set of microformats options to the Dreamweaver Insert bar.

Click the second icon from the left to open the hCard editor, which is shown in Figure 7-7.


hCard

Name: John Allsopp
Company: westciv
Street:
City:
State: Zip:
Phone:
Web site:
Photo:

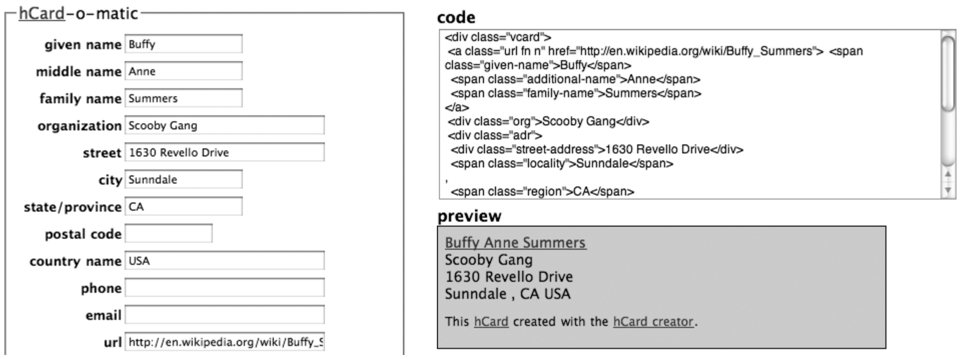
See Microformats.org for information.
Developed by Drew McLellan for the
Web Standards Project.

Figure 7-7. The hCard editor of the Dreamweaver Microformats toolbar

Simply fill in the details, and it creates your hCard for you.

Next up, we'll look at hCard Creator from microformats.org (`http://microformats.org/code/hcard/creator`). This is one of several easy-to-use, web-based, form-driven creators for microformatted content. It generates the HTML for you, based on your form input.

Figure 7-8 shows hCard Creator in action with our Buffy hCard.



hCard-o-matic

given name	Buffy
middle name	Anne
family name	Summers
organization	Scooby Gang
street	1630 Revello Drive
city	Sunndale
state/province	CA
postal code	
country name	USA
phone	
email	
url	http://en.wikipedia.org/wiki/Buffy_S

code

```
<div class="vcard">
 <a class="url fn n" href="http://en.wikipedia.org/wiki/Buffy_Summers"> <span
class="given-name">Buffy</span>
  <span class="additional-name">Anne</span>
  <span class="family-name">Summers</span>
</a>
 <div class="org">Scooby Gang</div>
 <div class="adr">
  <div class="street-address">1630 Revello Drive</div>
  <span class="locality">Sunndale</span>
,
  <span class="region">CA</span>
```

preview

Buffy Anne Summers
Scooby Gang
1630 Revello Drive
Sunndale , CA USA

This hCard created with the hCard creator.

Figure 7-8. microformats.org's hCard Creator in action

Not every last feature of hCard is available using hCard Creator. For instance, nickname isn't supported, you may want or need to change some of the elements hCard Creator uses to mark up the content, and the formatting depends on how addresses are formatted in a particular country. But it is a simple, useful way to start creating hCards.

For those of you who use the Textpattern blogging system/CMS, Chris Casciano has created a plug-in called pnh_mf for easily marking up microformatted content, including hCard. You can obtain pnh_mf from `http://placenamehere.com/TXP/pnh_mf`.

Libraries

A number of libraries are available for developers, to help accelerate development of services that consume or process hCard content:

- **hKit Microformats Toolkit for PHP5** (`http://allinthehead.com/hkit`): Drew McLellan, developer of the Dreamweaver Microformats toolbar, has also developed hKit, a PHP5 toolkit for extracting common microformats, including hCard.
- **Microformat Parser for Ruby** (`http://blog.labnotes.org/2005/11/20/microformat-parser-for-ruby`): Assaf Arkin developed this microformat parser for Ruby and Rails developers.
- **XV2** (`http://suda.co.uk/projects/X2V`): You've already seen it in action in this chapter, but I thought I'd mention XV2 again, because it's so cool. X2V allows for easy extraction of hCards and hCalendars (converting them to vCards and iCalendars, respectively). Remember that XV2 is open source, so you can use it in your own projects—it is actually a set of XSLT files. In addition to handling the simple process of converting an hCard to a vCard, XV2 could be very useful for building all kinds of applications that need to extract data from hCards.

Aggregators and indexers

At the start of the book, you learned that a major motivation for microformats is to enable "decentralized development, content, services." It's in this area of aggregators, indexers, and other decentralized services that the true promise of microformats lies.

In the last chapter, you were introduced to Technorati's Microformats search (`http://kitchen.technorati.com`) and the Pingerati service (`http://pingerati.net`). With these services, the foundations are in place for smarter, more focused, and more useful search engines, and services that aggregate microformatted content.

If you publish hCards, or any other kind of microformatted content, and you let Pingerati know about them, then all kinds of services that you may never have heard of (and that may not even exist yet) can find your content without you having to do anything more. You can manually ping by visiting the Pingerati site and entering the URL of the page you want Pingerati to crawl in search of microformatted content.

Automatically letting Pingerati know about updated content is as simple as sending an HTTP GET with this format:

> http://pingerati.net/ping/[url of update page]

For example, to inform Pingerati of content updated on `http://webdirections.org`, I just send HTTP GET to

> http://pingerati.net/ping/http://webdirections.org

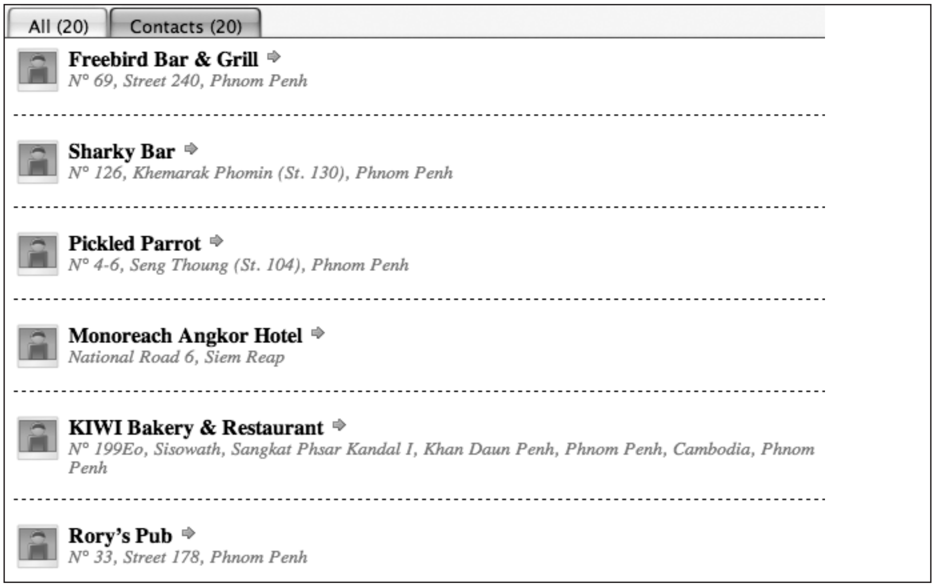
Services publishing with hCard

Because microformats are markup, it probably won't be immediately obvious that a page you're reading contains them. One tool you can use to determine whether microformats are being used on a page is the Tails Export extension, if you use Firefox (as discussed in the "Tails Export extension" section earlier).

It might surprise you that hCard and other microformats are widely used by some very big publishers, as well as some smaller, innovative publishers:

- **Avon** (`www.avon.com`), a large cosmetics manufacturer, publishes the contact details of more than 40,000 representatives using hCard.
- **Yahoo Local** (`http://local.yahoo.com`), Yahoo's recommendation and review service for tens of thousands of businesses all across North America, publishes the contact details of these businesses using hCard.
- Popular photo-sharing site **Flickr** (`http://flickr.com`) uses hCard to mark up profile details on its profile pages. Flickr also makes use of geo, as demonstrated in the last chapter, and XFN for marking up "contacts."

- **Yahoo Tech** (`http://tech.yahoo.com`), Yahoo's technology review site, also uses hCard for reviewer details.
- **Cork'd** (`http://corkd.com`), the recently launched wine review site, uses hCard to mark up reviewer details. The site also uses the hReview microformats for reviews, as you'll see in our case study later in the book, in Chapter 11.
- One of my favorite uses of hCard is the **Cambodian Yellow Pages** (`www.yellowpages-cambodia.com`).[2] Figure 7-9 shows the Bars and Pubs section, which I investigated using Tails (another Firefox extension for extracting microformatted content from pages, which we'll look at more in the next chapter).



Freebird Bar & Grill
N° 69, Street 240, Phnom Penh

Sharky Bar
N° 126, Khemarak Phomin (St. 130), Phnom Penh

Pickled Parrot
N° 4-6, Seng Thoung (St. 104), Phnom Penh

Monoreach Angkor Hotel
National Road 6, Siem Reap

KIWI Bakery & Restaurant
N° 199Eo, Sisowath, Sangkat Phsar Kandal I, Khan Daun Penh, Phnom Penh, Cambodia, Phnom Penh

Rory's Pub
N° 33, Street 178, Phnom Penh

Figure 7-9. The Bars and Pubs section of the Cambodian Yellow Pages

2. I kid you not, it's time to put aside any stereotypical notions we might have of Southeast Asia's once- war-ravaged countries.

It turns out that every single business contact published in the Cambodian Yellow Pages—and there must be tens of thousands—is published as an hCard!

That such a broad range of sites, some of which are extremely popular, have adopted hCard (and other microformats) as ways of publishing structured content should give you confidence that using microformats is not a fad, but an increasingly established best practice in web development.

Styling hCard content with CSS

Being rich, quite complex structures, hCards provide designers with both a bit more of a challenge than you've seen in previous chapters and sophisticated scaffolding for styling them. A recent example of styling hCards that plays on the business card metaphor is by Andy Hume, at `http://thedredge.org/2005/06/using-hcards-in-your-blog`. While his approach of using fixed-width cards (similar in technique to the rounded corner boxes from the last chapter) is a pretty good one, I thought I'd try something different here. Let's take a look at how we might style a variable-width business card style for our hCards.

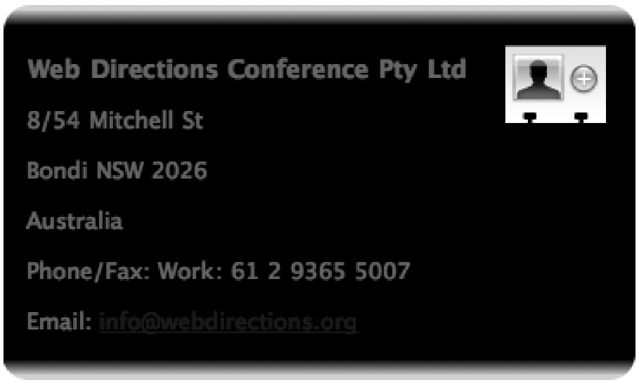
If you want to play with the HTML, CSS, and images for this design and use them to follow along with the example in this section, you can download them from `http://microformatique.com/book/chapter7/hcard.zip`.

Let's take a common hCard, which includes address, telephone, and email details:

```
<div class="vcard">
  <p class="fn org">Web Directions Conference Pty Ltd
  <a href="http://suda.co.uk/projects/X2V/get-vcard.php?
uri=http://microformatique.com/book/chapter7/index.html">
  <img src="images/vcard-add.png" alt="download vcard icon"></a></p>

  <div class="adr">
    <p class="street-address">8/54 Mitchell St</p>
    <p>
      <span class="locality">Bondi</span>
      <span class="region">NSW</span>
      <span class="postal-code">2026</span>
    </p>
    <p class="country-name">Australia</p>
  </div>
```

```
    <div class="telecommunications">
      <p class="tel">Phone/Fax: <span class="tel">
        <span class="type">Work</span>:
   <span class="value">61 2 9365 5007</span>
      </p>
      <p class="email">Email:
        <a class="value" href="mailto:info@webdirections.org">
          info@webdirections.org
        </a>
      </p>
    </div>
  </div>
```

We'll use a variation on the now well-established "sliding doors" technique (if you create a CSS technique, it's very important to give it a memorable name) created by Douglas Bowman and enhanced by Scott Schiller (see `www.schillmania.com/projects/dialog`), which will give us a final design that looks like Figure 7-10.



Web Directions Conference Pty Ltd

8/54 Mitchell St

Bondi NSW 2026

Australia

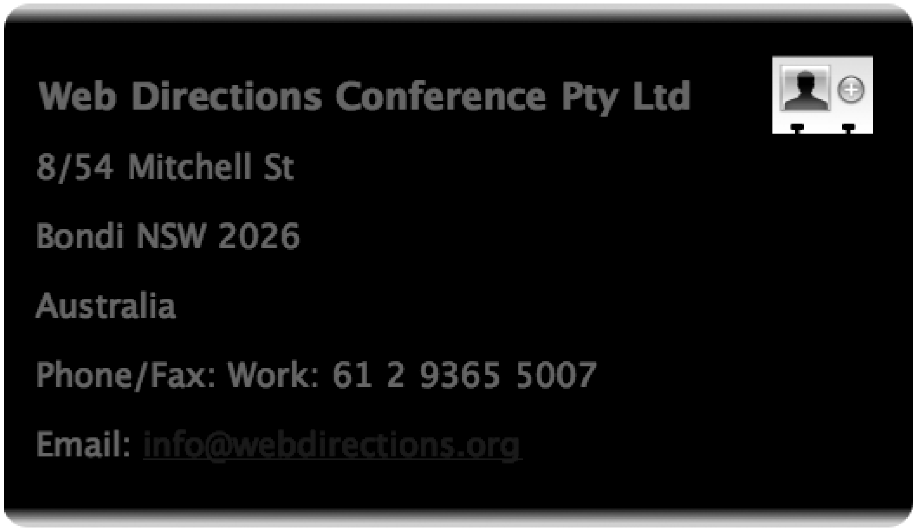
Phone/Fax: Work: 61 2 9365 5007

Email: info@webdirections.org

Figure 7-10. The example's final design

The technique, in a nutshell, uses background images on four elements, two at the top, and two at the bottom, to add each rounded corner.

We're going to make this design "fluid" in the sense that it grows and shrinks in proportion to the font size of the element's text. This is sometimes referred to as an **em-driven design** (you'll discover why in a moment). To see how this works in practice, Figure 7-11 shows the same design with the text's size increased.



Web Directions Conference Pty Ltd

8/54 Mitchell St

Bondi NSW 2026

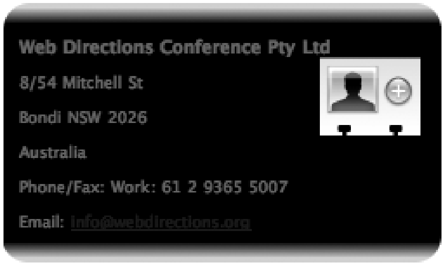
Australia

Phone/Fax: Work: 61 2 9365 5007

Email: info@webdirections.org

Figure 7-11. Because of the sliding doors technique, our hCard still works just fine with the text's size increased.

Figure 7-12 shows the same design again, with the text's size decreased.



Web Directions Conference Pty Ltd

8/54 Mitchell St

Bondi NSW 2026

Australia

Phone/Fax: Work: 61 2 9365 5007

Email: info@webdirections.org

Figure 7-12. And it's fine with the text's size decreased as well!

Compare this with our design for the address in the last chapter, where the width of the element was fixed, regardless of how wide or narrow the page, or how big or small the text.

The example's hCard image comes from Chris Messina. You can download it and other microformat icons from `http://microformats.org/wiki/icons`.

With CSS3, the entire task would be considerably easier, because we can add multiple background images to an element and border *images* for each edge of an element. Safari (version 1.3 and higher) actually supports multiple background images, but sadly, they're not supported in Firefox 1.5, or even Firefox 2.0 (let's not even mention Internet Explorer 7, eh?).

This technique is probably not supported enough to use now. Instead, we'll use a technique that involves only CSS2 and works in pretty much any browser.

As you saw with the rounded corner adr box example in the last chapter, very often developers add div or span elements as containers for background images. In fact, if you visit Scott Shiller's site, that's what he has done there. But if at all possible you shouldn't add any HTML simply for presentational purposes, even if the presentation is done via CSS. Rather, you should aim to use the HTML you have already, as much as possible, to add the style you want. This approach can take some creative thinking, but once you get the hang of it, it becomes a more natural way of using HTML compared with simply adding divs and spans at will as hooks for style. Of course, this technique isn't always simple, and in fact, sometimes it's simply not possible, requiring you to add just a little HTML to provide the hooks for your style.

The first step is to add a background image to the whole vCard element, as shown in Figure 7-13.

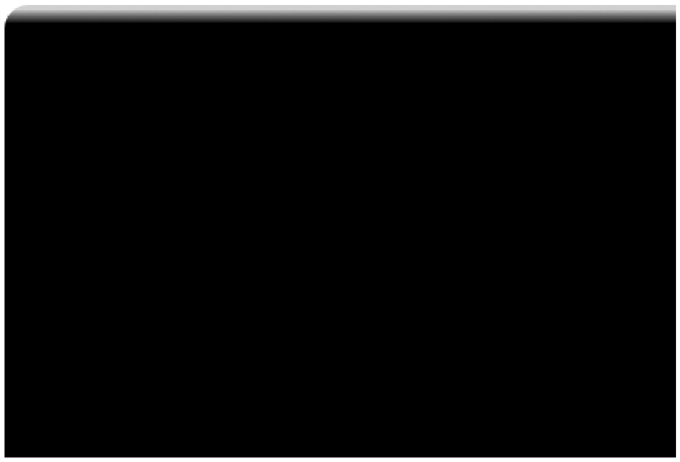


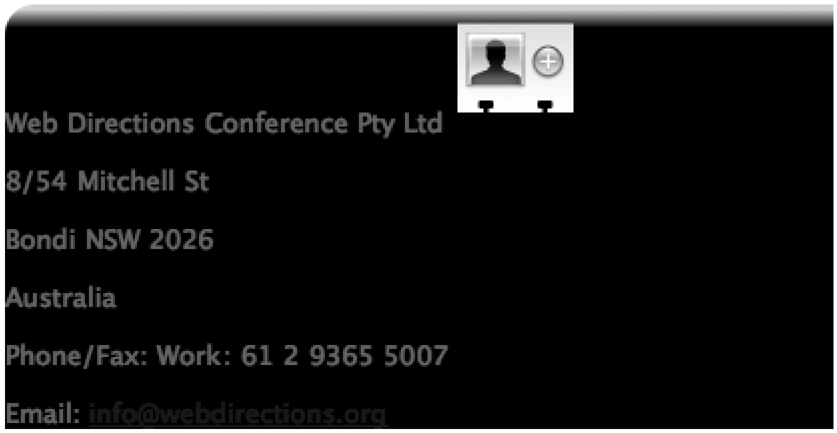
Figure 7-13. The example's background image

We make this wide enough (e.g., 1,000 or more pixels) and tall enough so that no matter how large the content of the vCard grows, it will never overflow this area. We can't simply repeat the image, because the top-left corner will show when the image repeats.

We add this as the background image of the vCard element using CSS. While we're at it, let's give the text a sans serif font and some color so that it will be visible, and let's stop the image from repeating.

```
.vcard {
  background-image: url(images/vcardfill.png);
  background-repeat: no-repeat;
  color: #666;
  font-family: "Lucida Grande", Verdana, Helvetica, Arial, sans-serif;
}
```

In a browser, our work so far will look something like Figure 7-14.



Web Directions Conference Pty Ltd

8/54 Mitchell St

Bondi NSW 2026

Australia

Phone/Fax: Work: 61 2 9365 5007

Email: info@webdirections.org

Figure 7-14. The first background border image added to our hCard

Next, we need to add the top-right corner of the vCard. In keeping with our aim of not adding HTML simply for styling purposes, we want to use the existing structure of the page where possible. Here, we'll use the paragraph of class fn and org, which is the first child element of the vCard element.

```
<p class="fn org">Web Directions Conference Pty Ltd ➥
  <img src="images/vcard-add.png" alt="download vcard icon"></p>
```

Here's our CSS:

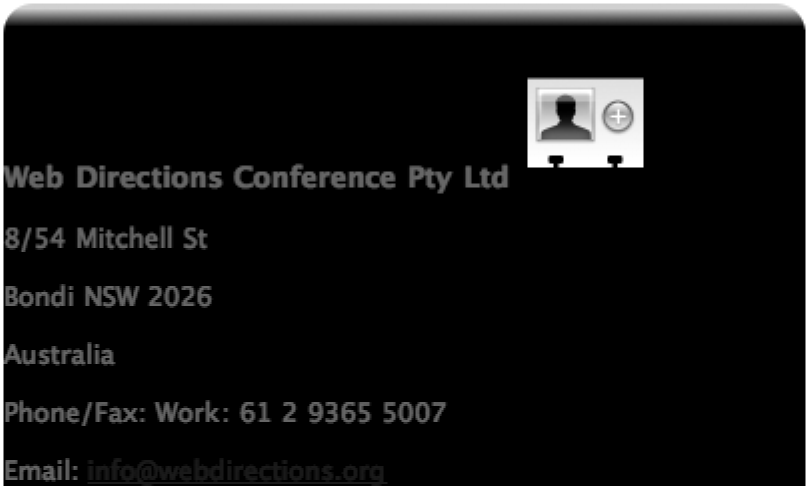
```
.fn {
  background-image: url(images/topright.png);
  background-repeat: no-repeat;
  background-position: top right;
  padding-top: 2em;
  font-weight: bold;
  font-size: 1.1em;
}
```

Again, we don't want the image to repeat, but this time, we've specified a background position for the image. This will make the background image start from the top, but its right edge will be located at the right edge of the element. We also make the font size a little bigger and the weight bold, to differentiate it from the rest of the text in the hCard.

Figure 7-15 shows the image we are adding as the background to this element.

Figure 7-15. Our top-right rounded corner image

Putting our two CSS rules so far together gives us something like Figure 7-16.


Web Directions Conference Pty Ltd

8/54 Mitchell St

Bondi NSW 2026

Australia

Phone/Fax: Work: 61 2 9365 5007

Email: info@webdirections.org

Figure 7-16. The completed top border

We specify a `padding-top` of `2em` to give some space between the content of the FN element and the edge of the FN element. Otherwise, the top of the vCard image would be hard against the border. To see this in action, just remove the `padding-top: 2em;` declaration and preview the image in a browser.

So, with just two statements, we are well under way, and we haven't had to add any HTML so far. Let's turn to the bottom of the element and add the bottom border (well, the background image, which will serve as that border).

Now, which element are we going to use to add this background image to? OK, here I have to admit to a *teensy* bit of cheating. If you look at the HTML of the hCard, I've grouped the email and telephone properties into a div, with a class of `telecommunications`. This grouping is not strictly required for our hCard.

```
<div class="telecommunications">
  <p class="tel">Phone/Fax:
    <span class="tel"><span class="type">Work</span>:
    <span class="value">61 2 9365 5007</span>
  </p>
  <p class="email">Email:
    <a class="value" href="mailto:info@webdirections.org">
      info@webdirections.org
    </a>
  </p>
</div>
```

I chose that class name because that is what the vCard specification calls this group of properties. And typically, I do tend to group together related elements using divs when I mark up content; it makes the page structure more logical. But strictly speaking, this isn't necessary, so you may consider it cheating. My advice here is that if you are going to add markup, try to make it as meaningful as possible.

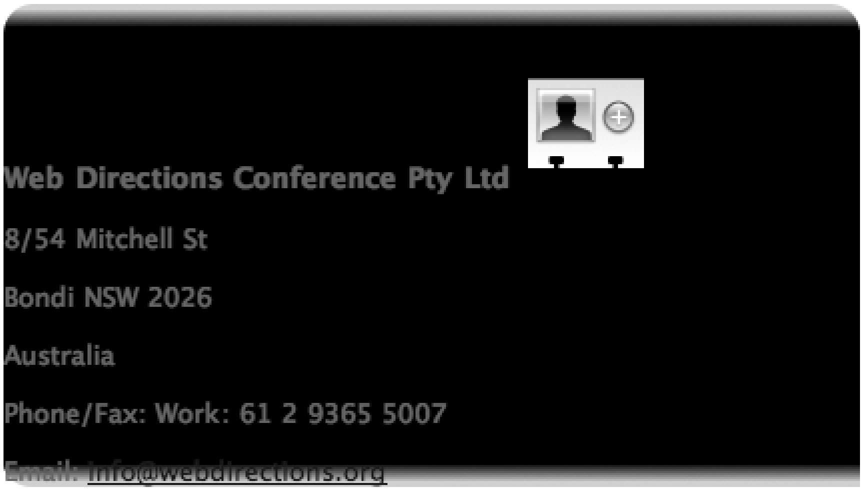
As you have probably guessed by now, we are going to add one part of the bottom border image to this element first—the main length of the bottom border—before then adding the rounded bottom-right corner as a separate image. Figure 7-17 shows this first background image.

Figure 7-17. The main bottom border image

Again, this will be a very wide image, like the top-left one, so that no matter how wide the element might get, the background image will still be wide enough. We'll need to make this image sit in the bottom left of the element we attach it to, so we use a background position of `left bottom` (we put the horizontal position before the vertical). Here's our CSS statement for this:

```
.telecommunications {
  background-image: url(images/bottom-left.png);
  background-repeat: no-repeat;
  background-position: left bottom;
  margin-bottom: 2em;
}
```

With this new CSS rule added, our hCard now looks like Figure 7-18.


Web Directions Conference Pty Ltd

8/54 Mitchell St

Bondi NSW 2026

Australia

Phone/Fax: Work: 61 2 9365 5007

Email: info@webdirections.org

Figure 7-18. The first bottom border image has been added.

We're not quite there yet, but we're well on the way. It's time for the final piece in the puzzle. OK, I admit, I might have cheated just a *little* bit more in this step. But like the previous step, it's all valid and (hopefully) quite justifiable markup. If you look at the HTML again, you'll find that our email address is marked up like this:

```
<p class="email">Email:
  <a class="value" href="mailto:info@webdirections.org">
    info@webdirections.org
  </a>
</p>
```

which is a little different from how we have previously marked up emails, which was more like this:

```
<a class="email" href="mailto:info@webdirections.org">
info@webdirections.org</a>
```

Think back to when we looked at how telephone numbers are marked up in hCard. Because telephone numbers have a type (work, home, etc.), we typically mark them up like this:

```
<div class="tel"><span class="type">Work</span>:
<span class="value">+61 2 9130 1731</a></div>
```

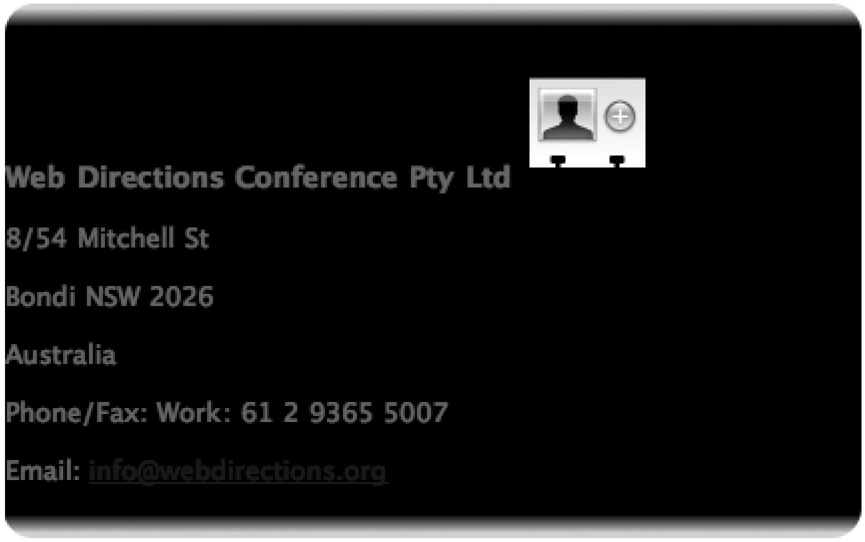
I've just used the same kind of markup here for our email address (but only because the vCard specification *allows for different types of email address*). Why have I gone to all this trouble? Well, when it came to styling the hCard, I realized I needed a block element to attach the background image for the bottom-right corner to. Typically, the last block element in the containing element is the ideal choice (and sometimes it's possible to take an inline element—for example, the link here—and use CSS to make it a block element, and attach it to that, but that really doesn't work with this design). So, if we are going to use the paragraph that contains the email link, we need a way to select it exclusively, which means that with CSS2 at least, we need a `class` or `id` as a hook for our CSS selector (in CSS3, we could use the `last-child` selector, which selects the last child element of a specified element, but again, because `last-child` is not widely supported yet, we won't rely on it here).

The least-worst thing we could do is take an existing element and add some reasonably meaningful markup to it. That's why we gave the paragraph a class of `email` and the email address a class of `value` (which reminds me a little of a moment in Hamlet: "The lady doth protest too much, methinks" . . .).

OK, let's get back to the CSS. We add the bottom-right corner image, positioning it in the bottom right of the element and making sure it doesn't repeat. We also add some padding to the bottom, to balance out the padding we added to the top of the hCard:

```
p.email {
  background-image: url(images/bottom-right.png);
  background-position: right bottom;
  background-repeat: no-repeat;
  padding-bottom: 2em;
}
```

This all goes to make our hCard look like Figure 7-19.



Web Directions Conference Pty Ltd

8/54 Mitchell St

Bondi NSW 2026

Australia

Phone/Fax: Work: 61 2 9365 5007

Email: info@webdirections.org

Figure 7-19. With the second bottom border image added, the rounded corners are finished.

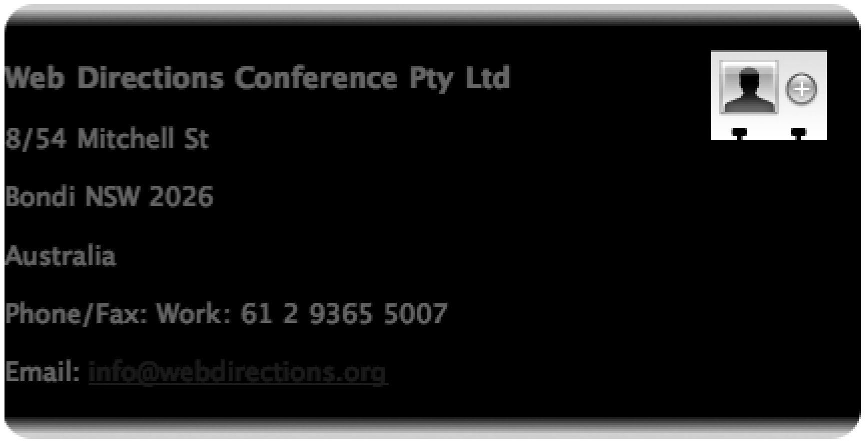
Now we need to just clean up a little. Let's start from the top. We'll float the download image to the right like this:

```
.vcard img {
  float: right;
  padding-right: 1em;
  margin-top: -1em
}
```

See how we didn't have to add a class to style the image? We used the fact that the image is a descendent of the vCard element and a descendent selector. In my experience, the very widely supported, powerful descendent selector is one of the most underused aspects of CSS. So if you don't use it frequently, look into it in more detail.

We'll add some space to the right of the image and pull it up a bit closer to the top of the hCard, giving us Figure 7-20.

We also want to add some whitespace between the edge of the hCard and the text. We would typically add padding to the left of the containing element (in this case, the vCard element), but this would break our bottom-left corner, as shown in Figure 7-21.



Web Directions Conference Pty Ltd

8/54 Mitchell St

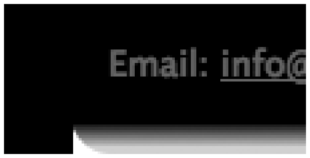
Bondi NSW 2026

Australia

Phone/Fax: Work: 61 2 9365 5007

Email: info@webdirections.org

Figure 7-20. Our design with the text vertically centered and the card flattened a bit



Email: info@

Figure 7-21. Adding padding to the left of the containing element breaks the design in this case.

That's because the div element we added to this bottom-left background image to would be moved in by the padding on its containing element. Instead, we add some left margin to all the paragraphs in the hCard:

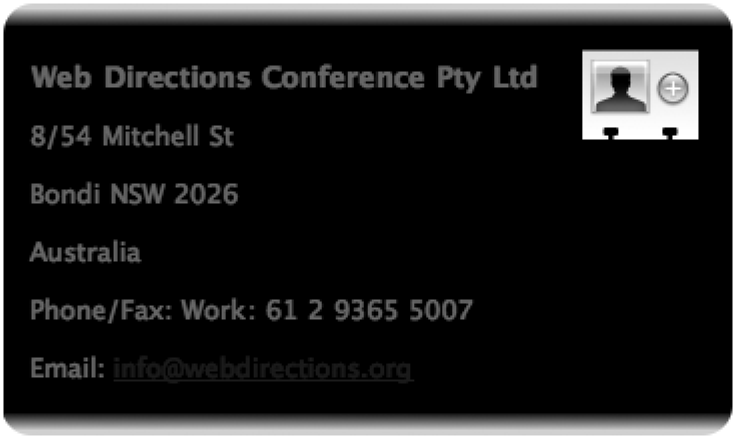
```
.vcard p {margin-left: 1em;}
```

(There's the descendent selector again—it's the Swiss Army knife of CSS.)

We haven't yet made the width of the hCard a function of the size of the text inside it (or em-driven, as discussed earlier). We do this by giving the hCard a width specified in em units. Here we set a width of 28em, which makes the hCard always roughly as wide as 28 characters (strictly speaking, 28 times the width of the capital letter "M"). The statement for our containing vCard element becomes

```
.vcard {
  background-image: url(images/vcardfill.png);
  background-repeat: no-repeat;
  color: #666;
  font-family: "Lucida Grande", Verdana, Helvetica, Arial, sans-serif;
  width: 28em
}
```

and our element looks like Figure 7-22.


Web Directions Conference Pty Ltd

8/54 Mitchell St

Bondi NSW 2026

Australia

Phone/Fax: Work: 61 2 9365 5007

Email: info@webdirections.org

Figure 7-22. Our final CSS design

We've used almost entirely the existing HTML from our original hCard (adding just a little, and trying as much as possible to keep that additional markup meaningful) and just six CSS statements.

Summary

Well, this was a chapter and then some. You learned what hCard is (in short, vCard for the Web), explored all the different properties of vCard, and discovered how to use them in hCard. You took a look at some tools for making your life as a developer easier; some of the major publishers, such as Yahoo, using hCard; and some of the services that index and aggregate hCard, such as Technorati Microformats search.

hCard is perhaps the most widely used of all microformats—certainly of the compound microformats—and it's not difficult to see why: contact information is just so common on the Web. hCard also serves as an excellent starting point for learning about other compound microformats, which you'll continue to look at in detail throughout the rest of the book.

The next chapter covers hCalendar, a microformat for events, which shares much in common with hCard, beginning with its origins as an IETF RFC. As a consequence, much of what you've learned in this chapter will be very useful in the next chapter.

8 EVENT MICROFORMAT: HCALENDAR

The rise of blogging demonstrated something that was always true about the Web, but that got somewhat lost in the feverish explosion of online sandwich delivery and multimillion-dollar sock puppets in the late 1990s web bubble: the Web is about people and their relationships. It's about what we feel and believe, and it's about conversations—online and in person.

The last chapter covered how microformats help make contacting people (and organizations) easier. In this chapter, you'll see how microformats help mark up another piece of extremely common information, events, with the hCalendar microformat. An event might be something as informal as coffee with a friend, a party you are throwing, or an internal meeting at your company. At the other end of the spectrum, an event could be the Nobel Prize conference, or even the Football (that's soccer for our American friends) World Cup, arguably the biggest sporting event in the world. This chapter provides examples of all of these types of events, using hCalendar.

hCalendar overview

The Web is full of event details—personal events, corporate events, concerts, movie timetables, television listings, conferences, and much more. Many of these are published by centralized services like Upcoming (http://upcoming.org), and many others are published on company and personal sites. Yet, as you saw with contact details, the way these events are published differs markedly from site to site.

Let's take a look at just a couple of major events and how they are marked up in HTML. Here is how the official site for the Beijing 2008 Olympics publishes event details in HTML:

```
<tr><td valign=top class=dian1 width=9>&#8226;</td><td class=f12_5959>
<A href="/28/10/article212051028.shtml">IOC Coordination Commission
plenary session held in Beijing (photos attached)(2)</A>
<span class=date1>[2006-10-24]</SPAN></td></tr>
```

While the official Academy Awards site publishes the main events associated with the Oscars on the front page like this:

```
<b>Sunday, February 25, 2007:</b> 79th Annual
Academy Awards Presentation <br>
```

(What *is it* with the `<br>` element?)

The events themselves have no particular markup whatsoever to identify them as such, which I suspect we would find to be the case for almost all events published online. These examples also show how hard it might be to extract meaningful information from an event published online—identifying and extracting dates. Here we have two significantly different date forms, and a quick look around the Web will find many others.

Now if something as seemingly simple as an address can be extremely difficult for software to recognize and extract relevant details from, as demonstrated in previous chapters, imagine how much more difficult it is to recognize an event, which may well include an address (or other location, such as Room 101 or "the Chairman's Room"), and additionally dates and times (as you learned in the date of birth discussion in the previous chapter, ambiguities can creep into even something as simple as a date) and other details.

Returning once more to the microformats principles from Chapter 1, recall that microformats are designed to "solve a specific problem." Given that so many events are published online without any real consistency, they are excellent candidates for a microformat. Another key principle is that microformats should "reuse building blocks from widely adopted standards." In the previous chapter, you saw a similar situation—contact details published online without any real consistency—and found the IETF RFC vCard fit the need very well. Rather than reinventing the wheel of creating a schema for contact details, we reused a well-established existing one.

So, is there a good candidate schema for event details, like vCard? Luckily, there is: **iCalendar**, the IETF RFC 2445. The stated purpose of iCalendar is to "provide the definition of a common format for openly exchanging calendaring and scheduling information across the Internet."[1]

You might be asking, if we have iCalendar for exchanging this information "across the Internet," why do we need a microformat? iCalendar is not designed for the Web itself; rather, it's a way of synchronizing data between different applications. In essence, it's a common file, or data interchange, instead of a publishing format. Here's an example of a typical iCalendar:

```
BEGIN:VCALENDAR
PRODID:-//xyz Corp//NONSGML PDA Calendar Version 1.0//EN
VERSION:2.0
BEGIN:VEVENT
DTSTAMP:19960704T120000Z
UID:uid1@host.com
ORGANIZER:MAILTO:jsmith@host.com
DTSTART:19960918T143000Z
DTEND:19960920T220000Z
STATUS:CONFIRMED
END:VEVENT
```

This is perhaps marginally more readable than vCard, but it's still far from human-friendly (and not at all like HTML, so it's not suitable for embedding in web pages).

In a nutshell, **hCalendar** is the schema of iCalendar, expressed in HTML. It reuses the properties of iCalendar, utilizing the class design pattern and the abbr design pattern, to meaningfully bring the rich semantics of iCalendar to the Web. Let's next take a look at how this works.

1. See www.ietf.org/internet-drafts/draft-ietf-calsify-rfc2445bis-03.txt.

The term "schema" is one you'll probably have heard used a bit, particularly in relation to XML or databases. In fact, it's been used in relation to hCard and now hCalendar in this book. In my experience, you'll be hard-pressed to get a good definition of the term from someone, outside of those who have formally studied computer science. The definition at Wikipedia of XML schema, "a way to define the structure, content and, to some extent, the semantics of XML documents,"[2] *is as good and succinct a definition as I've seen. Although the definition specifically relates to XML, it is also a good general definition in the context of microformats.*

Using hCalendar

Because hCalendar is, according to microformats.org, a "1:1 representation of the iCalendar standard, in semantic XHTML," to really get to grips with hCalendar, we need to take a close look at the iCalendar specification. This can be a little daunting, because the specification is largely written for developers working on software that uses the format, and not for publishers. Not to mention that it's 148 pages of plain, monospaced text. But you don't have to read the specification in its entirety, because I've done that for you. (Surely that's worth the price of this book alone?)

iCalendar is designed to be an interchange format for essentially all of the information that might go into a calendaring application, like Apple's iCal or Microsoft Entourage. But a great deal of the specification will hardly ever (or never) be required for the purpose for which the hCalendar microformat was designed: publishing events online. So, rather than laboriously translating all of the iCalendar specification into HTML, let's think about what we'll commonly need to publish event details online.

A straightforward event will most likely have

- A name
- A description
- A date (or, for longer events, a start date and an end date)
- A location
- A URL
- Some contact details

A more-complex event might also have

- Start and end times
- Detailed locations, with addresses and even geographic data (these might ring a bell from Chapter 6)

2. See `http://en.wikipedia.org/wiki/XML_schema`.

We'll then think about quite complex events that actually consist of a series of events. For example, a conference is an event that consists of sessions, each of which is itself an event. So we'll see how we can nest events inside calendars to group them together.

In the sections that follow, we'll start with the simplest kinds of events and work our way up to some more-complex examples.

Simple events

Typically an event will have

- A name
- A description
- A date (or for longer events a start and end date)
- A location
- A URL

Not surprisingly, iCalendar defines properties for each of these.

Recall that compound microformats require a root element, in the case of hCard, for example, an element with a class value of `vcard`, which contains the rest of the content of that microformat. With hCalendar, the root element is an element with a class value of `vcalendar`. To simplify the use of hCalendar where our calendar is simply a single event, we can use `vevent` as the class of the root element.

What? OK, it does sound a bit weird, but here is the reasoning. The iCalendar format is for whole calendars. A calendar in iCalendar is made up of one or more events, and each event can stand alone as an entity. While there is no explicit event microformat (as yet), we can think of events (inside iCalendar, an event is known as a "vevent," so we'll refer to an event in this way for now) as their own distinct microformat. In addition, in iCalendar, the calendar itself effectively has no data associated with it, other than the events it contains (and a couple of properties that are almost always used as their defaults). We can optimize a bit by leaving out the containing `vcalendar` element.

So, we have a root element with a class of `vevent`. As you saw with hCard, this element can be any kind of HTML element, provided you keep in mind the standard HTML containment rules, such as inline elements must not contain block elements. For this simple event, we'll make the containing element a div:

```
<div class="vevent">
  ...
</div>
```

The order in which details for the event—name, description, and so on—occur inside this element isn't important (this is almost always the case for microformats). But following a logical sequence, we'll add the name of the event next.

If you guessed that our next element will look something like this:

```
<h3 class="name">...</h3>
```

that's not a bad guess, but you'd be wrong. The name property in iCalendar is called summary (for which I am sure there is a good reason—it just escapes me at the moment). Then, using the class design pattern you've seen a number of times now, here's how we create the summary element:

```
<h3 class="summary">Web Directions North</h3>
```

Putting these together, we get our basic event:

```
<div class="vevent">
  <h3 class="summary">Web Directions North</h3>
</div>
```

This is not quite a valid vevent yet, because at a minimum, an event requires a summary and a start date. We'll do that in a moment.

Next, we might want to add some further information about the event, in a structured way. In iCalendar, the description property is used for this purpose. Again, using the class design pattern, we add a class value of description (i.e., a class value matching the property name in iCalendar) to an HTML element. Most likely, the description will be some kind of block of information. If it is a single paragraph, it makes sense to make this a paragraph of class description, like so:

```
<p class="description">
  A Web design and development conference in Vancouver Canada.
</p>
```

What if we want more than one paragraph? You might suggest we have two paragraphs of class description, like so:

```
<p class="description">
  A Web design and development conference in Vancouver Canada.
</p>
<p class="description">
  Featuring a who's who of international web experts....
</p>
```

This makes sense and is valid HTML (the same class value can, of course, be used on several elements), but when we look at the iCalendar specification, we find that among a number of other properties, description must not occur more than once. To comply with this aspect of iCalendar, we need to group these paragraphs together using a div element, and place the class value on the div like this:

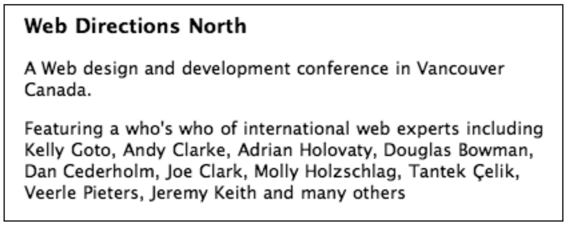
```
<div class="description">
  <p>A Web design and development conference in Vancouver Canada.</p>
  <p>Featuring a who's who of international web experts....</p>
</div>
```

Now, what would happen if we had used the previous "invalid" construct where we had two paragraphs, each with a class of `description`? Here the hCalendar specification (and this rule applies to all microformats) says, "For properties which are singular . . . the first descendant element with that class should take effect, any others being ignored." In our case, the description would only include the content from the first paragraph, and not the second, or any subsequent ones that had a class value of `description`. As I mentioned, this is true for any microformat where the property is a "singular" one. The first instance is considered to be the sole value, and all other instances are ignored.

Now we have the following:

```
<div class="vevent">
  <h3 class="summary">Web Directions North</h3>
  <div class="description">
    <p>A Web design and development conference in Vancouver Canada.</p>
    <p>Featuring a who's who of international web experts....</p>
  </div>
</div>
```

It's not yet a complete event, because it still lacks a start date, but let's take a quick look at it in a browser anyway (see Figure 8-1).



Web Directions North

A Web design and development conference in Vancouver Canada.

Featuring a who's who of international web experts including Kelly Goto, Andy Clarke, Adrian Holovaty, Douglas Bowman, Dan Cederholm, Joe Clark, Molly Holzschlag, Tantek Çelik, Veerle Pieters, Jeremy Keith and many others

Figure 8-1. Our event so far, as shown in a browser

Now, why might we be interested in this rather plain-looking page? It simply illustrates that when we use structured HTML appropriately, even when CSS is turned off or not available because it is not supported by the browsing device, our pages render meaningfully. This is usually a reasonably good basic test for whether assistive devices (screen readers, Braille devices, etc.) will be able to present the page contents in a meaningful way.

But as mentioned, we are still lacking a required element: the start date for the event. We've seen dates as part of a microformat before, with birthdays in the previous chapter. To recap, in order to deal with the potential ambiguity of a date (e.g., is 5.3.2006 May 3 or March 5?), and to make dates both human- and machine-readable, we use the date abbr design pattern. The abbr design pattern uses the abbr element of HTML, with the machine-readable data as the value of the `title` attribute, and the human-readable data as the content of the element. For instance, Buffy's birthday is as follows:

```
<abbr title="19810191" class="bday"> Jan 19 1981</abbr>
```

In the case of our conference, it began on February 6, 2007. Our machine-readable version of that date, using the ISO date-time standard (8601) is 20070206 (we go from most significant data, the year, to the least significant, the day). Or, to make it slightly more human-readable but still ISO8601-conformant, 2007-02-06. Using the abbr design pattern, we create our date element:

```
<abbr title="20070206">Feb. 6th 2007</abbr>
```

All we need to do now is to add the appropriate class value. In iCalendar, start dates (and times) are specified with the dtstart (date-time start) property (a little later you'll see how to add time information to this property):

```
<abbr title="20070206" class="dtstart">Feb. 6th 2007</abbr>
```

The conference runs for three days, so we'll need to add an end date. You would probably guess that the property in iCalendar for this is dtend, and again, we'll use the abbr design pattern to construct our end date element.

So, what's the ISO8601 form of the date February 8, 2007? You might think that it is 20070208, but in fact, that specifies midnight between February 7 and 8! So, to be exact, we either add a time component or use the date 20070209:

```
<abbr title="20070209" class="dtend">Feb. 8th 2007</abbr>
```

All we need to do now is incorporate this into our event element:

```
<div class="vevent">
  <h3 class="summary">Web Directions North</h3>

  <div class="description">
    <p>A Web design and development conference in Vancouver Canada.</p>

    <p>Featuring a who's who of international web experts....</p>
  </div>
  <p>From <abbr title="20070206" class="dtstart">Feb. 6th  ➥
    2007</abbr> to <abbr title="20070209" class="dtend">Feb. 8th ➥
    2007</abbr></p>
</div>
```

We could, of course, add our start and end dates in a number of ways. Recall from the last chapter that the actual date value is the value of the title attribute, not the human-readable content, so we could express the preceding information in a slightly more compact, human-friendly way like this:

```
<p>From <abbr title="20070206" class="dtstart">Feb. 6th</abbr> ➥
  to <abbr title="20070209" class="dtend">Feb. 8th</abbr> 2007</p>
```

We might also want to add the names of the days for the benefit of our human readers (software can work it out from the date if it is important):

```
<p>From <abbr title="20070206" class="dtstart">Tuesday Feb. 6th ➥
    </abbr> to <abbr title="20070209" class="dtend">Thursday ➥
    Feb. 8th</abbr> 2007</p>
```

We'll also want to mark up any link to the event (e.g., the main page of its website), and just as demonstrated with hCard, we can do this explicitly by adding a class value of url, which is found in iCalendar, to such links. Again, as with hCard, rather than adding a separate element for the link, we'll just turn the summary text into a link as well:

```
  <h3 class="summary"><a href="http://north.webdirections.org"
class="url">Web Directions North</a></h3>
```

This example shows that microformats, rather than imposing a particular rigid format on how we express information, give us a lot of flexibility to express content in meaningful ways for our human audience, and also for our machine audience.

Adding location information

The conference in our example took place at the Renaissance Vancouver Hotel Harbourside. I suspect our audience will have been quite interested in knowing this, so we need a way of publishing the location of the event. In this instance, iCalendar uses the obvious property name location:

```
<p class="location">Renaissance Vancouver Hotel Harbourside</p>
```

giving us the following so far:

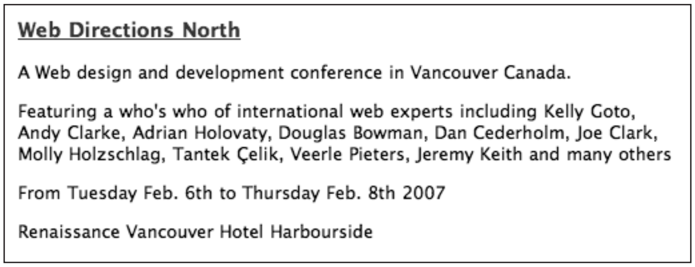
```
<div class="vevent">
  <h3 class="summary"><a href="http://north.webdirections.org"
class="url">Web Directions North</a></h3>

  <div class="description">
    <p>A Web design and development conference in Vancouver Canada.</p>
    <p>Featuring a who's who of international web experts....</p>
  </div>
  <p>From <abbr title="2007
0206" class="dtstart">Tuesday Feb. 6th ➥
   </abbr> to <abbr title="20070209" class="dtend">Thursday ➥
    Feb. 8th</abbr> 2007</p>
  <p class="location">Renaissance Vancouver Hotel Harbourside</p>
</div>
```

Finally, we'll want to add a URL for the event. Instead of adding a separate URL element, as with our hCard example, we'll add a link to the name of the event and give it a class of url (like all the other class values in this event, this one comes from iCalendar). Our final event code looks like this:

```
<div class="vevent">
  <h3 class="summary"><a href="http://north.webdirections.org"
class="url">
Web Directions North</a></h3>
```

```
      <div class="description">
        <p>A Web design and development conference in Vancouver Canada.</p>
        <p>Featuring a who's who of international web experts....</p>
      </div>
      <p>From <abbr title="20070206" class="dtstart">Tuesday Feb. 6th ➥
       </abbr> to <abbr title="20070209" class="dtend">Thursday ➥
        Feb. 8th</abbr> 2007</p>
      <p class="location">Renaissance Vancouver Hotel Harbourside</p>
    </div>
```

In a browser, the event will look as shown in Figure 8-2.



Web Directions North

A Web design and development conference in Vancouver Canada.

Featuring a who's who of international web experts including Kelly Goto, Andy Clarke, Adrian Holovaty, Douglas Bowman, Dan Cederholm, Joe Clark, Molly Holzschlag, Tantek Çelik, Veerle Pieters, Jeremy Keith and many others

From Tuesday Feb. 6th to Thursday Feb. 8th 2007

Renaissance Vancouver Hotel Harbourside

Figure 8-2. Our event is now looking more complete when rendered, with a link to the event home page and additional information.

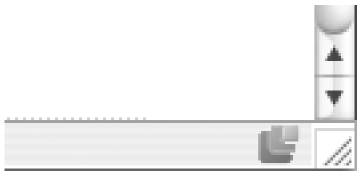
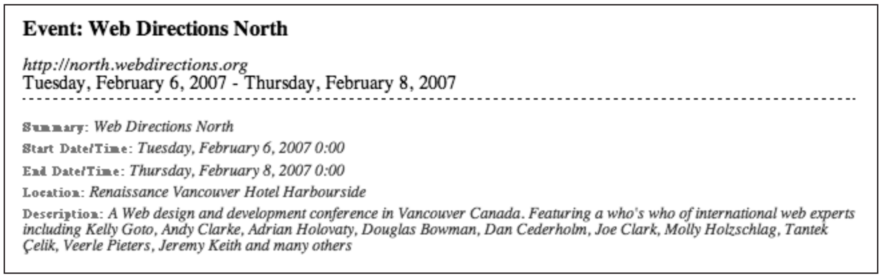


Figure 8-3. The Tails/Tails Export icon indicates that microformats are available on a web page.

It's not particularly exciting to look at, but let's see it using Firefox with the Tails extension installed. Recall from the last chapter that two popular Firefox extensions for extracting and displaying microformats are Tails Export (covered in the last chapter) and Tails (covered later in this chapter).

With Tails installed, as with Tails Export, when you load a page with microformatted content, you get a colored microformats icon (see Figure 8-3).

When you click the icon, you get a list of all the microformatted content on the page. Then you click one of the events to get its details, which are displayed as shown in Figure 8-4.



Event: Web Directions North

http://north.webdirections.org
Tuesday, February 6, 2007 - Thursday, February 8, 2007

Summary: *Web Directions North*
Start Date/Time: *Tuesday, February 6, 2007 0:00*
End Date/Time: *Thursday, February 8, 2007 0:00*
Location: *Renaissance Vancouver Hotel Harbourside*
Description: *A Web design and development conference in Vancouver Canada. Featuring a who's who of international web experts including Kelly Goto, Andy Clarke, Adrian Holovaty, Douglas Bowman, Dan Cederholm, Joe Clark, Molly Holzschlag, Tantek Çelik, Veerle Pieters, Jeremy Keith and many others*

Figure 8-4. Our hCalendar event rendered by Tails

As a basic microformats debugging tool, Tails is very useful. You can simply load your page in Firefox with Tails installed, and check whether all the data you think should be displayed actually is. Keep in mind that Tails may not be error-free (you'll see an example of when it is not, at least with the versions current in late 2006, later in this chapter), so it's not quite the same as validating your HTML with the W3C validator. But for a quick check, it's a great tool.

Tails is one of a great many extensions for the Firefox browser (there are currently over 200 developer-oriented extensions for Firefox), which many web developers find indispensable as part of their development process. You can find Tails at `http://blog.codeeg.com/tails-firefox-extension-03`.

Tails, developed by Calvin Yu, allows a user to see the microformatted content in a page. What's particularly interesting about Tails is that recently (late 2006 as this book was being written), the Firefox 3 development team and Chris Wilson, a longtime Windows Internet Explorer developer (and one of the Mosaic developers in the early 1990s), have been showing considerable interest in microformats, and inclusion of support for microformats in those browsers is being actively discussed. For example, you can see a brainstorm on Firefox microformat inclusion at `http://wiki.mozilla.org/Firefox/Feature_Brainstorming:Microformat_Handling`.

Extensions like Tails give us some idea of how native browser support for microformatted content might play out in browsers in the next couple of years.

Now that we have a straightforward vevent working, let's think about making it a little more sophisticated.

Adding contacts

For events you'll typically want to include some kind of contact details. In the last chapter, we took a detailed look at a microformat for precisely this purpose, so it would make sense to somehow use an hCard as part of an event. Remember that microformats aim to be "modular" and "embeddable," and you saw with the geo and adr microformats how these can be used inside other microformats when you need a format for geodata or addresses. In exactly the same way, you can embed an hCard in a vevent.

First, let's quickly construct our hCard for contacting the conference. We'll have an organization with a URL, an email address, and a postal address:

```
<div class="vcard contact">
  <p class="fn org">Web Directions North</p>
  <div class="adr">
    <p class="street-address">1485 Laperri&egrave;re Avenue</p>
    <p><span class="locality">Ottawa</span> <abbr class="region" ➥
      title="Ontario">ON</abbr> <span class="postal-code">K1Z ➥
      7S8</span></p>
    <p class="country-name">Canada</p>
    <p><a href="mailto:north@webdirections.org" class="email"> ➥
      north@webdirections.org</a></p>
  </div>
</div>
```

Notice that on the div of class vcard, we add a second class value, contact. The first class value, vcard, is associated with the hCard, while this second class value, contact, is actually part of the hCalendar microformat that this hCard is embedded in.

Now, where does this hCard go? Like any other part of a vevent, it will be a descendent element of the root vevent element. We'll add it to give the following vevent element:

```
<div class="vevent">
  <h3 class="summary"><a href="http://north.webdirections.org"
class="url">
Web Directions North</a></h3>

  <div class="description">
    <p>A Web design and development conference in Vancouver Canada.</p>
    <p>Featuring a who's who of international web experts....</p>
  </div>
  <p>From <abbr title="20070206" class="dtstart">Tuesday Feb. 6th ➥
    </abbr> to <abbr title="20070209" class="dtend">Thursday Â
    Feb. 8th</abbr> 2007</p>
  <p class="location">Renaissance Vancouver Hotel Harbourside</p>

  <div class="vcard contact">
    <p class="fn org">Web Directions North</p>
    <div class="adr">
      <p class="street-address">1485 Laperri&egrave;re Avenue</p>
      <p><span class="locality">Ottawa</span> <abbr class="region" ➥
        title="Ontario">ON</abbr> <span class="postal-code">K1Z ➥
        7S8</span></p>
      <p class="country-name">Canada</p>
      <p><a href="mailto:north@webdirections.org" class="email">
north@webdirections.org</a></p>
    </div>
  </div>
</div>
```

Adding start and end times

For a conference spanning several days, a start and end date will probably be sufficient information for attendees to help them make travel and accommodations plans. But for other events, like meetings or individual sessions within the conference, we are definitely going to want start and end times as well as dates.

I touched on times in the last chapter during the discussion of the ISO standard for dates. The ISO date-time standard, ISO 8601, allows us to specify both a date and a time. You know how to specify a date already—you specify the year (four digits), month (two digits), and day of the month (two digits), in that order. To specify a date and time, you use the same format but simply append the letter "T" and the time in 24-hour time, with hours, minutes, and seconds separated by a colon (:). For example, 22 minutes after 5:00 p.m. the afternoon of February 22, 2007, is 20070222T17:22 or 20070222T17:22:00 (seconds are optional).

To aid readability, the year, month, and day may be separated by hyphens (-) like this: 2007-02-22T17:22.

This leaves us with an issue. When it is 11:00 a.m. in Sydney, Australia, where I live, it is 6:00 p.m. on the West Coast of the United States and Canada, and quite a few different times elsewhere in the world, depending on the time zone being considered. So if I tell you we'll have a conference call next Wednesday at 6:00 p.m., and you are in Vancouver while I am in Sydney, we have a problem. Clearly, we need some way of specifying a universal time. ISO 8601 does this by appending the time zone relative to Coordinated Universal Time (UTC) (which is very closely associated with Greenwich Mean Time, but based on an atomic standard time unit, rather than astronomical observation—aren't you glad you asked?) like this: +02:00, or +02, or +0200.

For example, in Sydney, which is nine hours ahead of London, the current time to the second as I write this is 20061028T17:53:17+0900, or 20061028T17:53:17+09:00, or 20061028T17:53:17+09. (If daylight saving time commences, I have to write +10 instead of +09—another thing to keep in mind.)

A good online resource for time zone differences is The World Clock – Time Zones page at `www.timeanddate.com/worldclock/difference.html?p1=769`.

Some locations are actually behind UTC, in which case, you use a minus sign instead of a plus sign. For example, Vancouver, Canada, is seven hours behind UTC when observing daylight saving time, so you would write the current time and date in Vancouver as 20062810T07:57:17-0700.

How about in London itself? Well, you might think that leaving off the time difference altogether indicates that the location is +0 UTC, but this is not the case. It actually indicates local time wherever the value is being read. You need to add the +0 (or 00, or 00:00). And, of course, to make it trickier still, when the UK observes daylight saving time, it is one hour ahead of UTC, so in that case, you'd need to add +0100.

There is a shorthand for +0000—simply append the letter "Z" after the time. When it is not daylight saving time in London (or, technically, British Summer Time), you specify a date and time in London and elsewhere in Greenwich Mean Time as 20061122T09:00:59Z.

For more detail on ISO 8601 date-time values, see www.cs.tut.fi/~jkorpela/iso8601.html *and* http://hydracen.com/dx/iso8601.htm*. If you'd like to know why the letter "Z" is associated with Greenwich Mean Time, the article "Z-Time" by Harold F. Maybeck (*www.maybeck.com/ztime*) will be of interest.*

Ironically, like many ISO standards, the 8601 standard is not available online, but only for purchase, so you'll have to rely on the descriptions of the standard in this chapter, or purchase the standard from ISO. Makes the W3C's publishing of all standards online with the aim to have them permanently available at the URL at which they are published all the better, doesn't it?

OK, so how does ISO8601 help us specify a start and/or end date and time for an event? As recommended by the W3C for dates and times on the Internet, microformats use the ISO 8601 as the date-time standard. To specify that our conference begins at 9:00 a.m. local time in Vancouver, on February 6, 2007, we have this date:

```
20070206T09:00-0800
```

(By the way, I needed to look up the time difference on that date using `http://timeanddate.com/worldclock/city.html?n=256`.)

Our start date and time for the event now becomes

```
<abbr title="20070206T09:00-0800" class="dtstart">
Tuesday Feb. 6th art 9.00am</abbr>
```

while our end date and time becomes

```
<abbr title="20070208T17:00-0800" class="dtend">
Thursday Feb. 8th at 5.00pm</abbr>
```

Note that we use 20070208, because the addition of the time component means this value no longer indicates midnight between February 7 and 8, but rather a specific time on February 8.

In the next section, we'll put all this together for one of the sessions at Web Directions North.

Marking up a complete event in hCalendar

On the first day of the conference there are two whole-day workshops. Each of these is an event, and we'll mark it up as such. In a moment, we'll add more than one event to a calendar, but let's first create these events, using what we've covered so far.

The first event has these details:

- **Title**: Accessibility 2.0 – Build applications the smart way: Make them accessible and usable.
- **Description**: Web applications are fundamentally different from websites, and their creation demands that we break out of our usual development routines and take a whole new approach. Reputations are built on web apps that excite rather than frustrate users.
- **Where**: Renaissance Vancouver Hotel Harbourside, Vancouver, Canada.
- **When**: 9:00 a.m. to 5:00 p.m., Tuesday, February 6, 2007.
- **Details**: `http://north.webdirections.org/?page_id=4/#feather`.

We wrap the whole chunk of this event's information in a `<div>` element of class vevent. Like our first example, we have the name of the event (the "summary") as a heading, and we wrap the name in a link with an `href` of the details for the event. Here's our first chunk of information:

```
<div class="vevent">
  <h3 class="summary">
    <a href="http://north.webdirections.org/?page_id=4/#feather" ➥
      class="url">Accessibility 2.0 - Build applications the smart ➥
       way: Make them accessible and usable</a>
  </h3>
</div>
```

Next we add the description. In this case, it's a single paragraph, so we can use just a paragraph with a class of `description`; there's no need to add an extra div.

```
<p class="description">Web applications are fundamentally different ➥
  from web sites, and their creation demands that we break out of ➥
  our usual development routines and take a whole new approach. ➥
  Reputations are built on web apps that excite rather than ➥
 frustrate users.</p>
```

We add the location as we did the contact for the conference as a whole—as an hCard. We could do this simply as follows:

```
<div class="vcard location">
  Location: <p class="fn org">Renaissance Vancouver Hotel ➥
    Harbourside</p>
  <p class="adr"><span class="locality">Vancouver</span>
  <span class="country-name">Canada</span></p>
</div>
```

Now, we add the start and end times:

```
<p>When: <abbr title="20070206T0900-0800" class="dtstart"> ➥
  9:00 a.m.</abbr>to <abbr title="20070206T1700-0800" ➥
 class="dtend">5:00 p.m.</abbr>, Tuesday, February 6, 2007</p>
```

And we are done, giving us the following completed HTML:

```
<div class="vevent">
  <h3 class="summary">
      <a href="http://north.webdirections.org/?page_id=4/#feather" ➥
        class="url">Accessibility 2.0 - Build applications the smart ➥
        way: Make them accessible and usable</a>
  </h3>

 <p class="description">Web applications are fundamentally ➥
    different from web sites, and their creation demands that ➥
    we break out of our usual development routines and take ➥
    a whole new approach. Reputations are built on web apps ➥
    that excite rather than frustrate users.</p>

  <div class="vcard location">
    Location: <p class="fn org">Renaissance Vancouver Hotel➥
      Harbourside</p>
    <p class="adr"><span class="locality">Vancouver</span>
    <span class="country-name">Canada</span></p>
  </div>

 <p>When: <abbr title="20070206T0900-0800" class="dtstart"> ➥
    9:00 a.m.</abbr> to <abbr title="20070206T1700-0800" ➥
    class="dtend">5:00 p.m.</abbr>, Tuesday, February 6, 2007</p>
</div>
```

Our second event will be marked up in a similar fashion:

```
<div class="vevent">
 <h3 class="summary">
  <a href="http://north.webdirections.org/schedule/tuesday/#CSS" ➥
      class="url">Creative web development with CSS and DOM ➥
      scripting</a>
  </h3>

  <p class="description">U.K. Design guru Andy Clarke and Ajax ➥
    superstar Aaron Gustafson will take your CSS-based skills and ➥
    supercharge them with a little bit of JavaScript magic. ➥
    So all you designers out there, learn how to take ➥
    your skills to the next level by combining CSS and ➥
    JavaScript.</p>

  <div class="vcard location">
    Location: <p class="fn org">Renaissance Vancouver Hotel ➥
      Harbourside</p>
    <p class="adr"><span class="locality">Vancouver</span>
      <span class="country-name">Canada</span>
    </p>
  </div>
  <p>When: <abbr title="20070206T0900-0800" class="dtstart"> ➥
    9:00 a.m.</abbr> to <abbr title="20070206T1700-0800" ➥
    class="dtend">5:00 p.m.</abbr>, Tuesday February 6, 2007
  </p>
</div>
```

Both events take place as part of the larger event—the conference—so ideally we'd like to include them in a single hCalendar.

So far, you've seen that if you have a single event, you simply create a vevent, but you also saw at the outset that the schema you are using to do this is from iCalendar. When you have multiple events, you group them together into a vCalendar microformat. To do this, you create a root element with a class value of `vcalendar`, and then each of the events is a descendent element of this root element.

Logically, you have something like this:

```
<div class="vcalendar">
  <div class="vevent">[first event]</div>
  <div class="vevent">[second event]</div>
  ...
  <div class="vevent">[last event]</div>
</div> <!-- end vcalendar -->
```

For our example, we might add the vevent for the whole conference and follow it up with individual events for the workshops, sessions, lunch breaks, parties, and so on. Note that the root `vcalendar` element is, in fact, optional—where none exists, the page itself is taken to be the root element of the hCalendar.

Finally, you might be asking, why `vcalendar`? After all, I've been referring to iCalendar throughout this whole chapter. Well, the original specification was called vCalendar (like vCard), and the root property of an iCalendar is still called a vCalendar. Version 2 of the specification, which was published in 1998, jumped on board the "i"+everything naming convention (right after Apple released the iMac, "cool" online names went from e-* to i-*, it seems).

If you want to take a look at this whole example, you can download it from `http://microformatique.com/book.chapter8/vcalendar.zip`.

Other iCalendar properties

The iCalendar specification has a great many more properties than the ones discussed in this chapter. As hCalendar is, in effect, iCalendar for HTML, strictly speaking all of these other properties are part of hCalendar. But the ones presented in this chapter are typically just about all you'll need to mark up the vast majority of events. Earlier, I mentioned that iCalendar supports, among other things, recurring events (e.g., weekly or monthly meetings).

In theory, this too is part of hCalendar, but in practice, at the time of this writing, precisely how best to implement this in HTML has yet to be fully finalized.

This incrementalist approach may sound like a drawback of the microformats process, but in a sense it demonstrates a considerable strength. If we were to wait until all the complexities and subtleties of iCalendar were fully available in HTML through the hCalendar microformat, the task would take considerably longer, and the likelihood of it being completed at all would diminish. And, ironically, we'd probably only meet the needs of a reasonably small percentage more developers this way, while making everyone's life more difficult. The incremental approach of starting with widely needed solutions (in this case, a way of marking up reasonably straightforward information about events) means that in the significant majority of cases we have a solution earlier than we otherwise would have, and with considerably less effort. This approach is sometimes referred to as the **80/20 rule**, or the **Pareto principle** (after early twentieth-century Italian economist Vilfredo Pareto). When developing a microformat, it is better to focus on the solution that meets the needs of a significant majority of users and use cases (based on observation of actual practice, not theoretical need), rather than all use cases and all possible needs.

At times, a particular microformat may seem less than complete, or a seemingly obvious (from the perspective of a sophisticated user) piece of functionality might appear to be missing. Rest assured, it is likely that this use has been considered and put to one side, possibly for later, more complete development, based in part on the consideration of this 80/20 rule. And, if you have a particular need you want fulfilled by microformats, you could always contribute your energy and expertise toward making it happen (as we'll discuss in Chapter 13).

hCalendar and tables

So far, we've been marking up our event information from scratch, and we have been able to tailor our examples to make the job seem pretty effortless. But real-world observation about how events are marked up for the Web, particularly complex events like conferences, show that often they use tabular markup. Before you get excited and start exclaiming about how tables are "bad," understand that in this situation, it makes perfect sense to use a table. After all, an event listing is literally a "timetable."

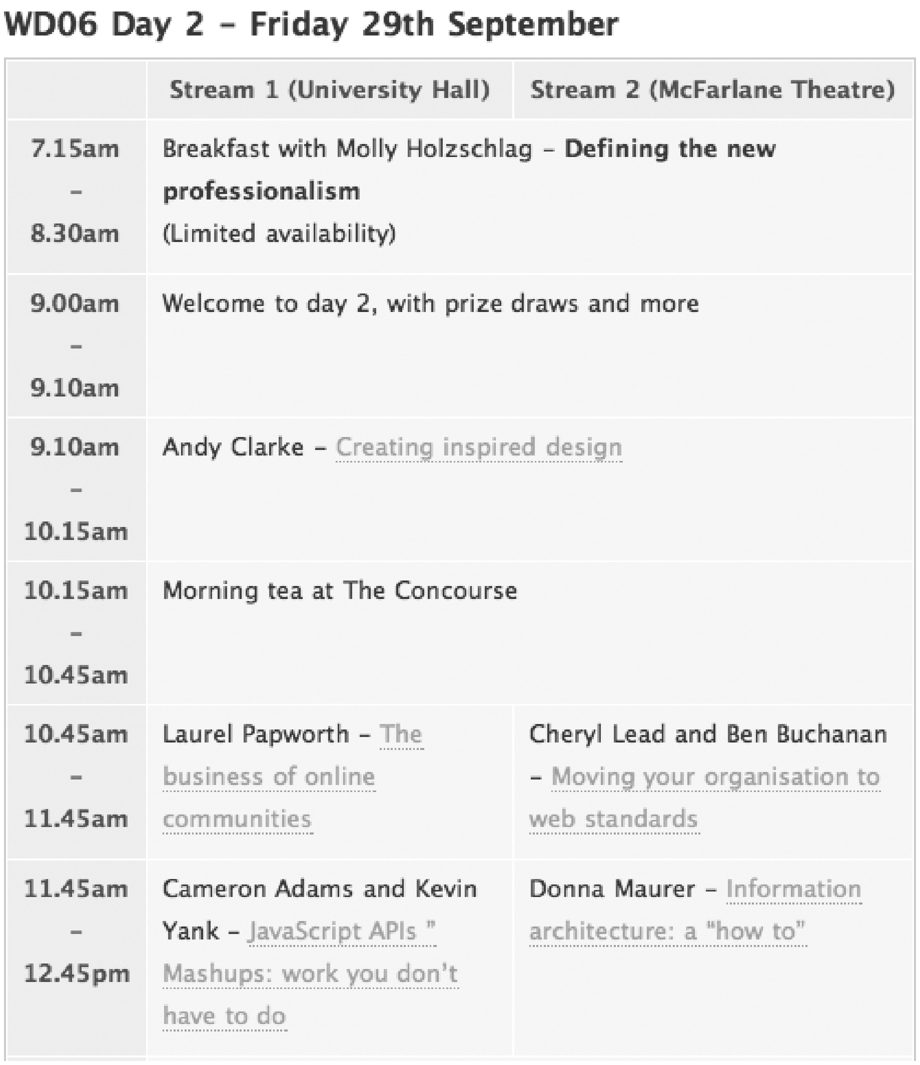
Figure 8-5 shows a typical example of how events are marked up.

Right away, you might see some problems. Because a table has been used, the time part of the event is in a different cell from the summary, location, and description, like this:

```
<tr>
  <th>10.15am - 10.45am</th>
  <td>Morning tea</td>
</tr>
```

Recall that root elements contain all the elements of a compound microformat, so what should be our "root" element be in this case? Well, the `<tr>` element is the parent of both these elements, so we could do this:

```
<tr class="vevent" >
  <th><abbr class="dtstart" title="20060928T1015+1000">10.15am</abbr> ➥
  <abbr class="dtend" title="20060928T1045+1000">
  10.45am</abbr></th>
<td colspan="2"><span class="summary">Morning tea</span></td>
</tr>
```



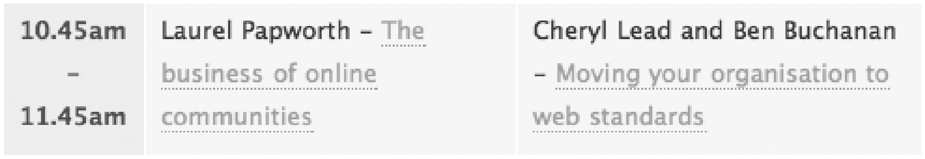
WD06 Day 2 – Friday 29th September

	Stream 1 (University Hall)	Stream 2 (McFarlane Theatre)
7.15am – 8.30am	Breakfast with Molly Holzschlag – **Defining the new professionalism** (Limited availability)	
9.00am – 9.10am	Welcome to day 2, with prize draws and more	
9.10am – 10.15am	Andy Clarke – Creating inspired design	
10.15am – 10.45am	Morning tea at The Concourse	
10.45am – 11.45am	Laurel Papworth – The business of online communities	Cheryl Lead and Ben Buchanan – Moving your organisation to web standards
11.45am – 12.45pm	Cameron Adams and Kevin Yank – JavaScript APIs " Mashups: work you don't have to do	Donna Maurer – Information architecture: a "how to"

Figure 8-5. A typical web-based calendar of events

This is fine, but we are still not able to include the location, which is in the first row of the table, not contained within the row we have just made the root of this event. So that's not the answer.

It gets even more complicated when we have simultaneous events like in Figure 8-6.



10.45am – 11.45am	Laurel Papworth – The business of online communities	Cheryl Lead and Ben Buchanan – Moving your organisation to web standards

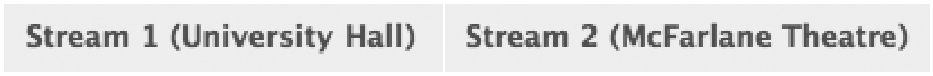
Figure 8-6. Marking up two simultaneous events in hCalendar gives us more of a challenge.

Here's our basic HTML for these two events:

```
<tr>
  <th>10.45am - 11.45am</th>
  <td>
    <p>Laurel Papworth - The business of online communities</p>
  </td>
  <td>
    <p>Cheryl Lead and Ben Buchanan -
Moving your organisation to web standards</p>
  </td>
</tr>
```

The `<tr>` is a common parent to both events, as well as the `<th>` element with the date and time details. We can't add a class of vevent to the `<tr>` element, like we did in the previous example, because we'd only have one event, when we really need two.

And in both examples, we haven't included the location, despite it being in the head of the table, like in Figure 8-7.



Stream 1 (University Hall)	Stream 2 (McFarlane Theatre)

Figure 8-7. The table head, showing the locations of the events

We could, of course, add the date, time, and location information, and then hide it using CSS with HTML like so:

```
<td class="vevent">
  <p class="summary">Laurel Papworth - The business of online ➥
    communities <abbr class="dtstart" title="20060929T1045+1000"> ➥
    10.45am</abbr> - <abbr class="dtend" title="20060929T1145+1000"> ➥
    11.45am</abbr></p>
</td>
```

and CSS like this:

```
abbr.dtstart, abbr.dtend {
  display: none
}
```

But this adds considerable complexity and redundancy to our HTML and CSS. In addition to adding information we already have elsewhere in our HTML, there will be other `<abbr>` elements of class `dtstart` and `dtend` that we *do* want displayed, so we'll probably have to make these selectors even more specific, probably something like `td.vevent abbr.dtend` to specify only `<abbr>` elements of class `dtend` and `dtstart` inside `<td>` elements of class `vevent`. And we *still* haven't even added in the location information.

Ideally, we'd want to mark up the location, date, and time information just once. This not only makes our pages smaller and our development effort simpler, but it also makes maintenance easier and reduces the chance of errors. After all, we need to make sure that the displayed information and the machine-readable information are the same, and if we change our HTML at all (e.g., if we move an event to another time on the program), we must keep these in sync. The more we need to edit and maintain, the greater the chance that we'll make mistakes.

But is there a way of using appropriate, correct HTML and achieving this goal of not repeating information? I am sure you'll have guessed that there is, but it does rely on a relatively complex and little-used aspect of HTML tables, one that has particular relevance for accessibility: table axis and headers.

Axis, scope, and headers in HTML tables

Before we see this technique in action, let's take a quick look at the `axis`, `scope`, and `headers` attributes of HTML.

One of the problems with HTML tables is that they are two-dimensional—that is, they relate information only on two axes, either in a row or a column. A header element applies only to the other cells in its row or column. For example, a common table might have items and their prices:

```
<table>
  <tr>
    <th>Item</th>
    <th>Price</th>
  </tr>
  <tr>
    <td>apple</td>
    <td>50c</td>
  </tr>
  <tr>
    <td>orange</td>
    <td>66c</td>
  </tr>
  <tr>
    <td>banana</td>
    <td>$4</td>
  </tr>
</table>
```

Even with such simple tables we have a problem. How do we associate the table data cells with the correct header? In this simple case, it seems logical that <th>Item</th> is the header for the first column, comprising the first <td> of each subsequent <tr>. But why is <th>Item</th> not a heading for the next table cell in its row, <th>Price</th>?

We can explicitly state which cells a <th> is the header for using the scope attribute. We can specify whether the header is associated with other cells in its row or column using this attribute. For example, in our simple table, we use the following:

```
<th scope="col">Item</th>
<th scope="col">Price</th>
```

to specify that each of these are headers for the other cells in their columns, not their rows.

In more complex tables, it may not be the case that the header for a particular cell is at the beginning of the column or row containing the cell itself. In this case, the scope attribute is not sufficient to specify the header for a given cell; we need to specify the header explicitly. We do this with a combination of the id attribute on a <th> element and the headers attribute on a cell. Here's how this works.

We give the <th> element an id, just as we would any element, following the id naming rules. As an example, let's give our element an id of d21045 (if you are wondering, that's short for day 2 at 10:45):

```
<th id="d21045">
```

From our example, we know that two table cells will have this as their header, so for each of them we specify a header value of d21045, like this:

```
<td headers="d21045">
```

It doesn't matter *where* in the table these cells appear—they both share the <th> element with this id as a header. But notice one more crucial thing: the attribute name is headers (note the plural). The headers attribute allows a cell to have more than one header, as well as a cell to be the header for more than one cell. Technically, we can create multidimensional tables in this way.

If you think back to the problem we had with our timetable, it essentially stemmed from not being able to contain the date and time information *and* location information for our events inside the events themselves, because of the table structure. But, logically, we can consider a header as part of the vevent construct, and so in this way, we can "share" the time and location parts among a number of vevents.

Let's see this in action. Remember our simultaneous events (see Figure 8-6)? In that case, we have two events happening at the same time, so ideally we want to have them "share" the start and end date and time. First, we create these events, which are two cells in the same row:

```
<tr>
  <td class="vevent">
    <p class="summary">Laurel Papworth - ➥
      <a href="http://www.webdirections.org/online-communities/" ➥
      class="url">The business of online communities</a></p>
  </td>
  <td class="vevent">
    <p class="summary">Cheryl Lead and Ben Buchanan - ➥
    <a href="http://www.webdirections.org/moving-to-web-standards/" ➥
     class="url">Moving your organisation to web standards</a></p> ➥
  </td>
</tr>
```

Now, we create the table header, which contains the dtstart and dtend for both events:

```
<th><abbr class="dtstart" title="20060929T1045+1000">10.45am ➥
  </abbr> - <abbr class="dtend" title="20060929T1145+1000"> ➥
  11.45am</abbr></th>
```

giving us the following:

```
<tr>
  <th><abbr class="dtstart" title="20060929T1045+1000">10.45am ➥
    </abbr> - <abbr class="dtend" title="20060929T1145+1000"> ➥
    11.45am</abbr></th>
  <td class="vevent">
    <p class="summary">Laurel Papworth - ➥
      <a href="http://www.webdirections.org/online-communities/" ➥
      class="url">The business of online communities</a></p>
  </td>
  <td class="vevent">
    <p class="summary">Cheryl Lead and Ben Buchanan - ➥
    <a href="http://www.webdirections.org/moving-to-web-standards/" ➥
     class="url">Moving your organisation to web standards</a></p> ➥
  </td>
</tr>
```

We need to associate the <th> with each of the events, using the headers attribute. Step 1 is to give our <th> an id:

```
<th id="d21045">
```

Step 2 is to use the headers attribute on the <td> elements to associate this <td> element with each of them, like so:

```
<td class="vevent" headers="td21045">
```

giving us the following:

```
<tr>
  <th id="d21045">
<abbr class="dtstart" title="20060929T1045+1000">10.45am</abbr> -
<abbr class="dtend" title="20060929T1145+1000"> ➥
    11.45am</abbr></th>
  <td class="vevent" headers="td21045">
    <p class="summary">Laurel Papworth - ➥
      <a href="http://www.webdirections.org/online-communities/" ➥
      class="url">The business of online communities</a></p>
  </td>
  <td class="vevent" headers="td21045">
    <p class="summary">Cheryl Lead and Ben Buchanan - ➥
    <a href="http://www.webdirections.org/moving-to-web-standards/" ➥
     class="url">Moving your organisation to web standards</a></p> ➥
  </td>
</tr>
```

Now we have associated the dtstart and dtend found in the <th> with these events. Next we'll want to associate the *location* information in the cells with the appropriate header.

Taking a look at the rendered table (see Figure 8-8), you can see that the location is found at the top of the table.

We'll use the same technique, giving the <th> elements, which specify a location, an id. In this case, the two events are in different locations, and so will have different headers for location. Here are our header elements, with their id values:

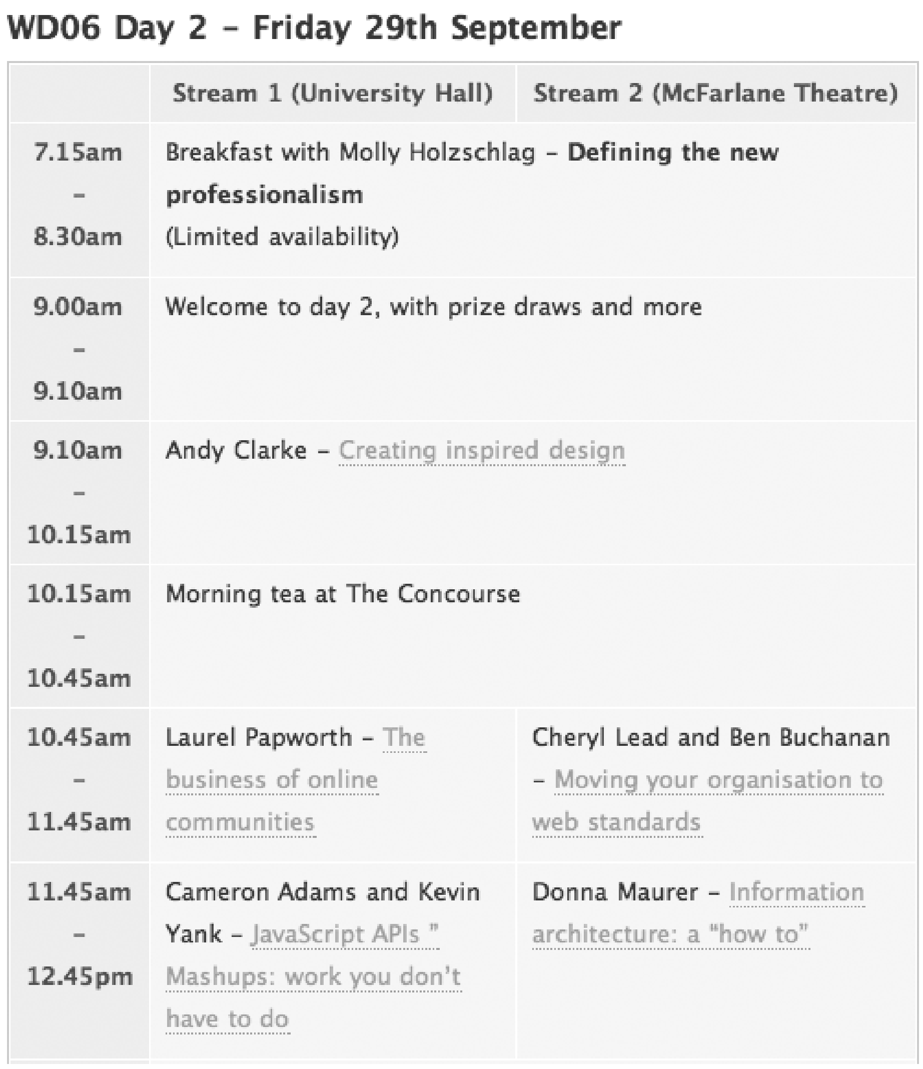
```
<th id="universityhall">Stream 1 (<span class="location"> ➥
  University Hall</span>)</th>
<th id="mcfarlane">Stream 2 (<span class="location">McFarlane ➥
  Theatre</span>)</th>
```

We associate the first event, held in University Hall, with the <th> element like this:

```
<td class="vevent" headers="d21045 universityhall">
  <p class="summary">Laurel Papworth - ➥
    <a href="http://www.webdirections.org/online-communities/" ➥
    class="url">The business of online communities</a></p>
</td>
```

The second event is in the McFarlane Theatre, which we specify similarly, like this:

```
<td class="vevent" headers="d21045 mcfarlane">
  <p class="summary">Cheryl Lead and Ben Buchanan - ➥
    <a href="http://www.webdirections.org/moving-to-web-standards/" ➥
    class="url">Moving your organisation to web standards</a></p>
</td>
```



WD06 Day 2 – Friday 29th September

	Stream 1 (University Hall)	Stream 2 (McFarlane Theatre)
7.15am – 8.30am	Breakfast with Molly Holzschlag – **Defining the new professionalism** (Limited availability)	
9.00am – 9.10am	Welcome to day 2, with prize draws and more	
9.10am – 10.15am	Andy Clarke – Creating inspired design	
10.15am – 10.45am	Morning tea at The Concourse	
10.45am – 11.45am	Laurel Papworth – The business of online communities	Cheryl Lead and Ben Buchanan – Moving your organisation to web standards
11.45am – 12.45pm	Cameron Adams and Kevin Yank – JavaScript APIs " Mashups: work you don't have to do	Donna Maurer – Information architecture: a "how to"

Figure 8-8. Our final events table. But we'd still like to attribute location details to each individual event, as well as the table header.

We have now associated the location for each event and the times for each event with the events themselves. But instead of putting all the information for a vevent inside a root element, we use headers and id to associate <th> element with more than one vevent.

A short while ago, you learned that the scope attribute lets us specify whether a header applies to a row or column. In this example, the date and time headers have a scope of row, while the location headers apply to the column they are in. Let's add a scope attribute to each <th> element to specify this:

```
<th id="universityhall" scope="col">Stream 1 (<span class="location"> ➥
  University Hall</span>)</th>
<th id="mcfarlane" scope="col">Stream 2 (<span class="location"> ➥
  McFarlane Theatre</span>)</th>
<th id="d21045" scope="row"><abbr class="dtstart" ➥
  title="20060929T1045+1000">10.45am</abbr> - ➥
  <abbr class="dtend" title="20060929T1145+1000">11.45am ➥
  </abbr></th>
```

Axis of good

We still have one more subtle problem: what is the nature of the association between a given <th> and the <td> elements it is the header for? We can specify this with another, not commonly used attribute associated with tables: the axis attribute. While it's not absolutely required for our purposes with hCalendar, it has particular benefits for accessibility with complex tables when a person is using a screen reader. It requires little extra effort, so while we are marking up our tables for the hCalendar microformat, let's spend the small amount of time to make them more accessible.

The axis property in effect allows us to create categories of cells. So, in our case, we can create two categories of cell, "time" and "location," using this attribute. To achieve this, we add axis="categoryname" to our <th> elements. To create a location category, we simply give all the <th> elements associated with location this axis value:

```
<th axis="location">
```

Similarly, to create a time category, we give the <th> elements associated with time an axis value of time:

```
<th axis="time">
```

Now, any <td> element associated with a <th> element using the headers attribute in effect belongs to the category we have created for that <th> element.

Again, let's put this into practice with our example. For our location headers at the top of the table, we create a location category like this:

```
<th id="universityhall" axis="location" scope="col">Stream 1 ➥
  (<span class="location">University Hall</span>)</th>
<th id="mcfarlane" axis="location" scope="col">Stream 2 ➥
  (<span class="location">McFarlane Theatre</span>)</th>
```

For our date/time headers, we create a time category like this:

```
<th id="d21045" axis="time" scope="row">
  <abbr class="dtstart" title="20060929T1045+1000">10.45am</abbr> ➥
   - <abbr class="dtend" title="20060929T1145+1000">11.45am</abbr>
</th>
```

So, what have we achieved with our hCalendar timetables? We started with a problem. If we use a table for our event details, then when two events share a time slot, or a location, it would seem we'd have to duplicate the time and location information if we want to use hCalendar. This makes our pages larger and more complicated than they otherwise might be, potentially impacting the maintainability of the information.

But looking more closely at the existing features of HTML, we discover that by creating an association between a cell (`<td>`) and a header (`<th>`) using the `headers` and `id` attributes, we can virtually include the headers specifying time and location into an event. By using `axis`, we can categorize the association, as time or location, making the association more accessible, too.

This feature of the hCalendar microformat is not just of theoretical value either, as you'll see when we look at X2V and how it handles hCalendars in just a moment. X2V understands the use of `headers` and `id`, and will correctly convert hCalendars marked up like this to iCalendar.

Downloading your calendar

You saw in the last chapter how to use the X2V service to add a link to a page and make an hCard on that page downloadable as a vCard, which users can use in any application that supports vCard. We can also use X2V to convert hCalendars to iCalendar, which is also widely supported by desktop applications like Apple's iCal and Microsoft Outlook. To do so, we just add a link like the following to our page:

```
<a href="http://suda.co.uk/projects/X2V/get-
vcal.php?uri=http://microformatique.com/book/tests/vCalendar.html">
Download this calendar</a>
```

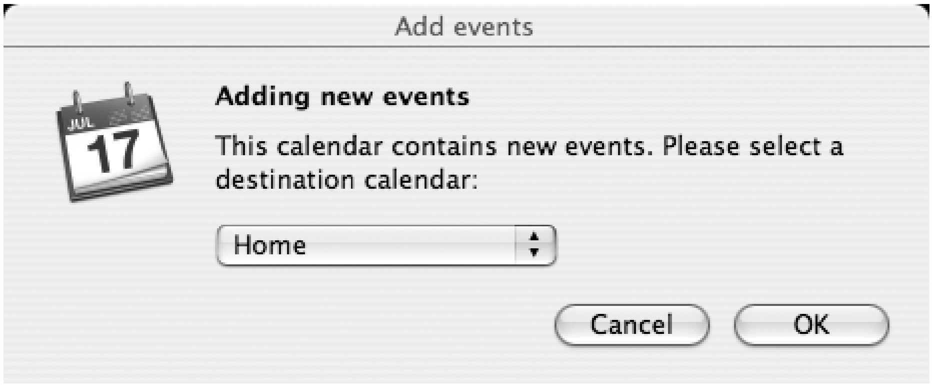
The `href` has two parts. The `href` of the X2V service

```
http://suda.co.uk/projects/X2V/get-vcal.php?
```

followed by the `href` of the page containing our hCalendar content.

```
http://microformatique.com/book/tests/vCalendar.html
```

Using Mac OS X, for example, if you click this link using Safari, the iCal application asks which calendar you want to add these events to, as shown in Figure 8-9.

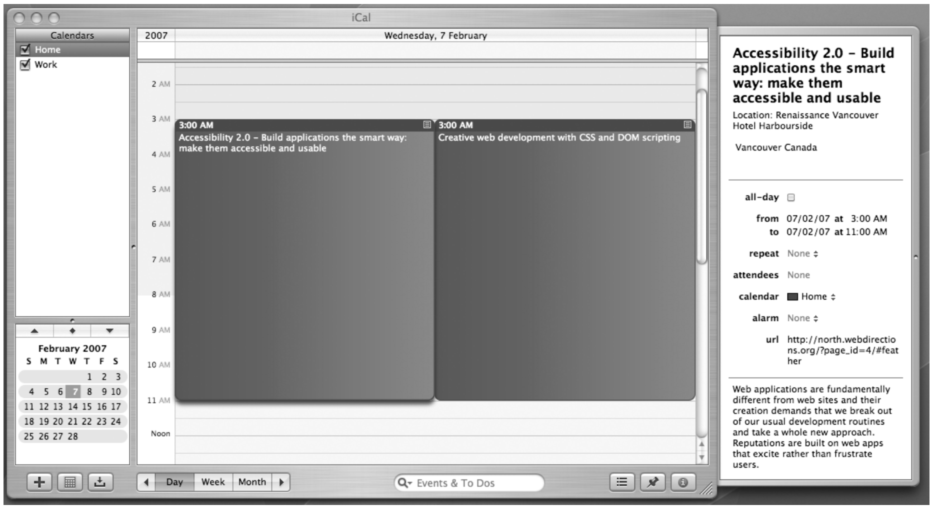

Add events

Adding new events

This calendar contains new events. Please select a destination calendar:

Figure 8-9. You can add hCalendar events to the Address Book automatically using some web browsers.

Choose a calendar, and iCal then adds the events to your calendar, as shown in Figure 8-10.



3:00 AM
Accessibility 2.0 – Build applications the smart way: make them accessible and usable

3:00 AM
Creative web development with CSS and DOM scripting

Accessibility 2.0 – Build applications the smart way: make them accessible and usable

Location: Renaissance Vancouver Hotel Harbourside

Vancouver Canada

all-day	☐
from	07/02/07 at 3:00 AM
to	07/02/07 at 11:00 AM
repeat	None
attendees	None
calendar	Home
alarm	None
url	http://north.webdirections.org/?page_id=4/#feather

Web applications are fundamentally different from web sites and their creation demands that we break out of our usual development routines and take a whole new approach. Reputations are built on web apps that excite rather than frustrate users.

Figure 8-10. Our events successfully imported into Apple's Address Book

You might be wondering why the calendar shows the event as starting at 3:00 a.m. on February 7. Didn't we specify 9:00 a.m. on February 6? We did, but we specified a time zone as well. I am actually downloading these events in Sydney, where it will be 3:00 a.m., February 7, when in Vancouver it will be 9:00 a.m. February 6. iCal knows where I am and coverts the date for me.

And if you think that's impressive, there's one more trick we can do with iCal and X2V. In iCal, you can subscribe to calendars that have been published online in the iCalendar format. The result of using X2V like we did a moment ago is an iCalendar, so our readers can simply subscribe to a URL like this: `http://suda.co.uk/projects/X2V/get-vcal.php?uri=http://microformatique.com/book/tests/vCalendar.html`, as if we had published our event information as iCalendar using iCal. In iCal, you can specify that the application check for updates, so if the hCalendar changes, it will be automatically updated in iCal. As publishers, we get the benefit of automatically updating calendars, simply by using the hCalendar microformat.

As X2V demonstrates, using the hCalendar microformat brings direct benefits today, for both publishers and users, by enabling this closer integration with desktop applications and other software that works with the iCalendar format, like Apple's iCal.

It's taken quite some effort to cover the core of hCalendar, but hopefully it's been a lot less painful than reading the 148 pages of plain, monospaced text that is the iCalendar specification. As with other microformats, help is at hand in the guise of plug-ins and applications to ease the process and make developing and publishing hCalendar content more efficient. Let's take a look at some of these now.

Tools for authoring hCalendars

As with hCard, there are quite a few tools to help make publishing hCalendar content easier than hand-coding. Let's take a look at a few options:

- **Textpattern Microformats Plug-in** (`http://placenamehere.com/TXP/pnh_mf`): As you saw in the last chapter, the pnh_mf plug-in helps publish hCard-formatted data, and it also supports publishing hCalendar content.
- **Dreamweaver Extensions Suite** (`www.webstandards.org/action/dwtf/microformats`): This extensions suite by Drew McLellan, which you saw in action in the last chapter, not only allows easy hCard creation, but also hCalendar creation in Dreamweaver.
- **hCalendar Creator** (`http://microformats.org/code/hcalendar/creator`): This tool provides an easy-to-use, form-driven interface for creating hCalendar content.
- **Conference Schedule Creator** (`http://dmitry.baranovskiy.com/work/csc`): Dmitry Baranovsky developed this tool to help easily create schedules such as the conference timetable we just developed.

In addition, two recently released desktop-based blogging tools have hCalendar support: Microsoft's **Windows Live Writer** (see `http://ideas.live.com/programpage.aspx?versionId=4372c8c2-b76f-4d44-aea1-9835b61d8dc1`; yes, they certainly do know how to construct a URL at Microsoft) and the Java-based **xfy** (`www.xfytec.com`; this product runs on Windows, Mac OS X, and Linux).

Benefits of using hCalendar

As you've seen, there can be a little work involved in publishing hCalendar content, so other than being on the bleeding edge of markup, why bother? In addition to the benefits of consistency and maintainability, there are already hCalendar aggregators as well as other direct benefits of adopting this microformat, which we'll discuss in this section.

Technorati Microformats search engine

Technorati's Microformats search engine should now be familiar to you, as it was featured in the hCard chapter. This search engine also indexes and aggregates events published using hCalendar.

If you publish your event information using hCalendar, you simply ping Pingerati by sending an HTTP GET with this format:

```
http://pingerati.net/ping/[url of update page]
```

This informs not only the Technorati Microformats search engine about your hCalendar content, but also any service that uses Pingerati to keep abreast of changes to information published using microformats. If you only infrequently publish microformatted content, you can manually inform Pingerati at `http://pingerati.net`.

X2V

You saw a little earlier how you can make your hCalendar content downloadable in iCalendar (`.ics`) format, which is compatible with most desktop calendaring software, using X2V. Users can even subscribe to your hCalendar content using Apple's iCal, by subscribing to the URL for a page containing hCalendar content.

To take the strain off X2V (itself open source, licensed under the very liberal W3C Open Source License), Technorati provides the X2V service as well. This allows end users to grab hCalendar content from any site that publishes it and download it in iCalendar format, or subscribe to it. This service is available at `http://technorati.com/events`.

The download and subscribe services also available via **favelets**, which are little bits of JavaScript that you can put in your bookmarks/favorites folder.

Life Lint parser

Similar to X2V, Life Lint (`www.lifelint.net`) converts hCalendar to iCalendar or RDF, and has optional Outlook 2002 compatibility (Outlook's support for iCalendar format is not perfect, though). Readers can use the parser to convert any hCalendar-formatted content they find, or you can add a link to your site to enable the service directly within your own pages where this content appears.

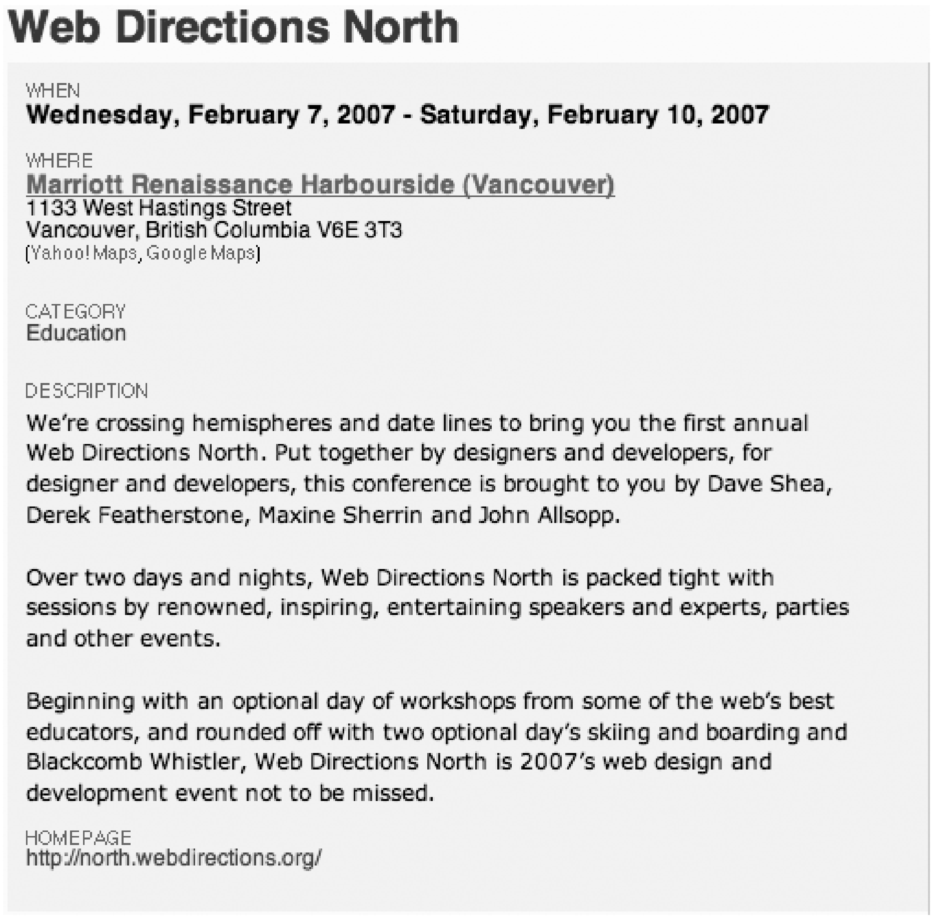
Publishers using hCalendar

hCard is increasingly being used by even large publishers, such as Yahoo, as well as innovative startup services like Cork'd. If anything, the use of hCalendar is even more widespread. In short, all the main publishers of aggregated event information are using it. The sections that follow present some examples.

Upcoming

Recent Yahoo acquisition Upcoming (`http://upcoming.org`), like a number of other services you'll see in a moment, provides a way of easily adding events you are organizing or interested in to a centralized registry of events, which others can express interest in or let you know they are going to. Upcoming, not surprisingly, publishes the event information using hCalendar, even embedding the location details using hCard.

Figure 8-11 shows how Upcoming publishes details on Web Directions North.



Web Directions North

WHEN
Wednesday, February 7, 2007 - Saturday, February 10, 2007

WHERE
Marriott Renaissance Harbourside (Vancouver)
1133 West Hastings Street
Vancouver, British Columbia V6E 3T3
(Yahoo! Maps, Google Maps)

CATEGORY
Education

DESCRIPTION
We're crossing hemispheres and date lines to bring you the first annual Web Directions North. Put together by designers and developers, for designer and developers, this conference is brought to you by Dave Shea, Derek Featherstone, Maxine Sherrin and John Allsopp.

Over two days and nights, Web Directions North is packed tight with sessions by renowned, inspiring, entertaining speakers and experts, parties and other events.

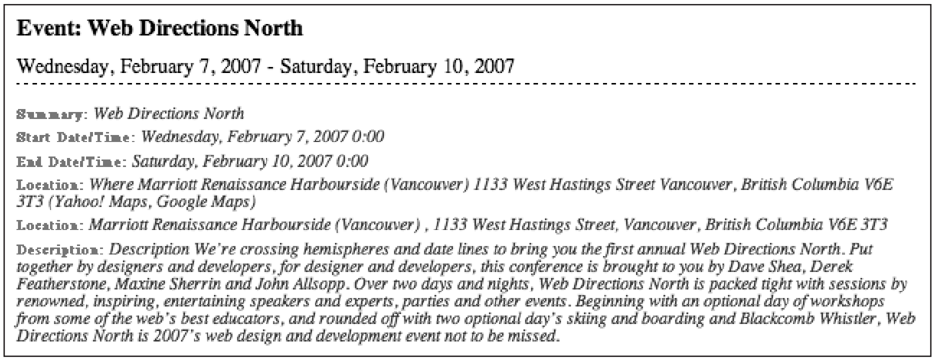
Beginning with an optional day of workshops from some of the web's best educators, and rounded off with two optional day's skiing and boarding and Blackcomb Whistler, Web Directions North is 2007's web design and development event not to be missed.

HOMEPAGE
http://north.webdirections.org/

Figure 8-11. Web Directions North, as detailed on Upcoming

Reproduced with permission of Yahoo! Inc. © 2007 by Yahoo! Inc.
YAHOO! and the YAHOO! logo are trademarks of Yahoo! Inc.

But if we look behind the scenes using the magic of the Firefox Tails extension, we see the metadata Upcoming has published using hCalendar (and hCard), shown in Figure 8-12.



Event: Web Directions North

Wednesday, February 7, 2007 - Saturday, February 10, 2007

Summary: *Web Directions North*

Start Date/Time: *Wednesday, February 7, 2007 0:00*

End Date/Time: *Saturday, February 10, 2007 0:00*

Location: *Where Marriott Renaissance Harbourside (Vancouver) 1133 West Hastings Street Vancouver, British Columbia V6E 3T3 (Yahoo! Maps, Google Maps)*

Location: *Marriott Renaissance Harbourside (Vancouver) , 1133 West Hastings Street, Vancouver, British Columbia V6E 3T3*

Description: *Description We're crossing hemispheres and date lines to bring you the first annual Web Directions North. Put together by designers and developers, for designer and developers, this conference is brought to you by Dave Shea, Derek Featherstone, Maxine Sherrin and John Allsopp. Over two days and nights, Web Directions North is packed tight with sessions by renowned, inspiring, entertaining speakers and experts, parties and other events. Beginning with an optional day of workshops from some of the web's best educators, and rounded off with two optional day's skiing and boarding and Blackcomb Whistler, Web Directions North is 2007's web design and development event not to be missed.*

Figure 8-12. Tails showing how Upcoming uses hCalendar

And here is the actual HTML Upcoming uses to publish the event (whitespace has been added for clarity, but the actual HTML is the same):

```
<div id="event" class="vevent">
  <h1 class="name summary">Web Directions North </h1>
  <div id="eventMain">
    <div id="eventMetadata">

      <div class="small">When</div>
      <div class="date">
        <abbr class="dtstart" title="2007-02-07"> ➥
        Wednesday, February 7, 2007</abbr>
        - <abbr class="dtend" title="2007-02-10"> ➥
        Saturday, February 10, 2007.</abbr>
      </div> <!-- /.date -->

      <div class="time"></div> <!-- /.time -->

      <div class="venue location vcard">
        <br />
        <div class="small">Where</div>
        <span class="fn org">
          <a href="/venue/35093">
Marriott Renaissance Harbourside (Vancouver)</a>
        </span><br />
        <div class="address adr">
          <span class="street-address">
            1133 West Hastings Street
          </span><br />
          <span class="locality">
            Vancouver</span>,
          <span class="region">
             British Columbia
          </span>
```

```
            <span class="postal-code">
              V6E 3T3
            </span>
          </div>
          <div class="directions">
            (<a href="http://maps.yahoo.com/py/maps.py?addr=1133+West+ ➥
              Hastings+Street&csz=Vancouver+British+Columbia+V6E+3T3"> ➥
              Yahoo! Maps</a>,
            <a href="http://maps.google.com/maps?q=1133+West+Hastings+ ➥
              Street,+Vancouver,+British+Columbia+V6E+3T3"> ➥
              Google Maps</a>)<br />
          </div>
        </div> <!-- /.venue -->

        <div class="category">
          <div class="small">Category</div>
          Education
        </div>

        <div class="description">
          <div class="small">Description</div>
          We're crossing hemispheres...
        </div>

        <div class="url">
          <div class="small">Homepage</div>
          <a href="http://north.webdirections.org/">
  http://north.webdirections.org/</a>
        </div>
      </div>
    </div>
  </div>
```

Upcoming mixes its own class-based semantics (e.g., venue) with hCalendar's semantic vocabulary, showing the flexibility of microformats. They work around your existing code, rather than enforcing a complete change in your development practices.

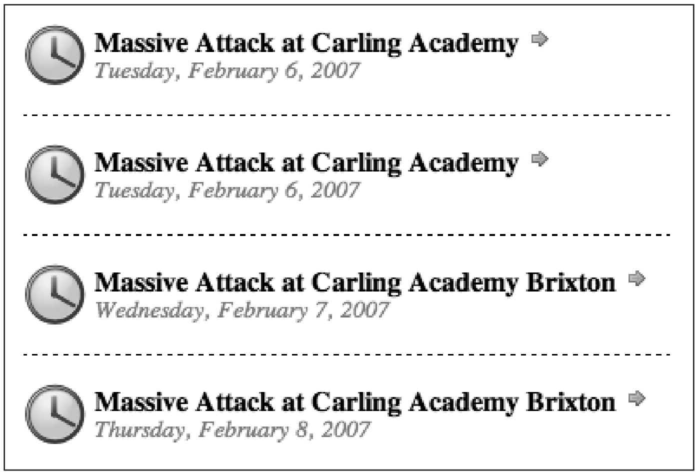
Upcoming publishes hundreds of thousands of events, from thousands of individuals, almost none of whom know anything about microformats or hCalendar. These individuals simply enter the details in a form-based interface, and out comes hCalendar when published. It's true that Upcoming and other services like it are centralized, but there's little reason that in time they can't also aggregate events published elsewhere using hCalendar. Before a standardized format like hCalendar, this would have been essentially impossible.

Meetup and Eventful

Two similar services, Meetup (`www.meetup.com`) and Eventful (`http://eventful.com`), also publish event details and contact details using hCalendar and hCard, respectively. It's clear that if you aggregate and publish event details, you should be using hCalendar. Hopefully soon these sites will also be aggregating events published elsewhere online in hCalendar, so by publishing event details that way, your content will be aggregated by all these services and more with no further effort on your part than pinging Pingerati.

Last.fm

Last.fm (`www.last.fm`) is a music community site where listeners can share their taste in music. It also provides, not surprisingly, information about upcoming concerts and other music events. And it publishes these using—you guessed it—hCalendar. For example, Figure 8-13 shows the Massive Attack artist page, as viewed with Tails, showing microformatted details of the group's upcoming concerts (which you might even miss without Tails installed).


Massive Attack at Carling Academy
Tuesday, February 6, 2007

Massive Attack at Carling Academy
Tuesday, February 6, 2007

Massive Attack at Carling Academy Brixton
Wednesday, February 7, 2007

Massive Attack at Carling Academy Brixton
Thursday, February 8, 2007

Figure 8-13. Tails showing microformatted concert details on Last.fm

World Cup KickOff

Just to show the diversity of the kind of content and uses hCalendar is being put to, here's something a little different. The biggest sporting event in the world in 2006 was the Football World Cup. The popularity of football (known in America as soccer) in most parts of the world is hard to convey. And the Football World Cup itself is very hard to keep track of, with 64 games over a month, at all kinds of times.

World Cup KickOff (`www.worldcupkickoff.com`) published event details for every one of the games of the Football World Cup using hCalendar, which enabled readers to download and import the details into their calendaring software. This kind of event shows the real benefit of the subscription approach we saw earlier that iCal allows. Because the teams contesting the later rounds are determined by earlier results, the publishers needed only to update their hCalendar content, and readers who subscribed to these pages using iCal would automatically have their calendars updated.

Summary

While hCard is probably the single most widely used compound microformat, the range of services and publishers using hCalendar makes it currently perhaps the most interesting microformat.

As this chapter demonstrated, hCalendar is certainly more complex, and to an extent more difficult to use, than the simpler microformats presented in previous chapters. But it's also certainly not rocket science. The principles underpinning hCalendar, of simple data formats based on existing widely used schemas and meeting a real current need, are the same as for all other microformats. The HTML principles and design patterns, such as the abbr design pattern, which allows us to mark up both human-friendly and machine-readable content, and the class design pattern, which provides a powerful, simple mechanism to incorporate the semantics of a schema like iCalendar into standard HTML, are also familiar and are reused by all the compound microformats. So while hCalendar in one sense appears complicated, in other ways it's simple—just a little more involved than some of the microformats you've encountered thus far.

At this point, we've actually covered all the widely adopted, finalized microformats—that is, those microformats that aren't still in draft form. However, a number of draft microformats are reasonably stable and widely used, or show considerable promise. In the following chapters, we'll look at a number of these draft microformats.

9 REVIEW AND RESUME MICROFORMATS: HREVIEW AND HRESUME

So far, all the microformats presented in this book have been at what's termed the **specification** stage. To reach this stage, according to the microformats process (which we'll discuss in more detail in Chapter 13), a format "should be stable enough so that developers can pick it up and write to it." Before a microformat reaches this stage, but is still sufficiently advanced in development, it is considered to be in **draft** stage.

In this chapter, we'll look at two draft microformats that are quite stable, advanced, and well used. These microformats, one for reviews and one for resumes, illustrate the principles and practices you've seen before and build on them nicely. They differ in one significant way from the other compound microformats you've encountered thus far: their schemas (i.e., the set of properties that make them up) are not found in other pre-existing standards (in the way that hCard, for example, implements the schema for vCard in HTML). Rather, their schemas were developed through analyzing existing practices on the Web in the related areas of reviews and resumes.

This chapter takes a slightly different approach from earlier chapters, by first looking at the properties of the two microformats under consideration, and then presenting some examples of their use. In part, this is because the formats are logically quite straightforward, and their use does not involve some of the complications we've seen with earlier formats, such as hCalendar (e.g., adapting an hCalendar to a complex timetable). It's also because we've already covered the core microformats concepts, techniques, and principles, and in this chapter we can simply watch them in action, rather than having to delve into them in detail.

hReview

At the beginning of the book we considered the issue of reviews on the Web and the challenges of aggregating related reviews published across different sites, in myriad formats. The goal of hReview is to provide a standardized format for publishing reviews and, according to microformats.org, "enable and encourage the sharing, distribution, syndication, and aggregation, of reviews."

As mentioned in the introduction to this chapter, despite the existence of a number of review formats, there was no single, widely adopted existing format, like vCard or iCalendar, for hReview to adopt. Instead, the hReview schema was developed through a brainstorming and research approach by the editors and authors of the specification and others with expertise and interest in publishing reviews online. Examples were taken from sites such as Amazon, Insider Pages, Epinions, Zagat, Yahoo Local, IMDb, and other widely used review sites. Where possible, and utilizing the 80/20 rule, which aims to provide as much possible value to adopters of a microformats with as little complication as possible, hReview has taken a minimum common subset of the properties found at this collection of sites.

It's important to note that hReview was designed from the outset to not include any "type-specific" fields. That is, it is designed to be used generally despite the type of item being reviewed—hReview can be used for reviews of films, books, restaurants, businesses, holidays, and much more. For example, it would make the hReview format much more complex to include domain-specific fields such as "Contains Spoiler," which is found at IMDb to indicate whether the review contains information that may give away important plot information.

In the sections that follow, we'll examine the schema of hReview while building an example review for the most popular film of 2006, *Pirates of the Caribbean: Dead Man's Chest*.

The hReview schema

In this section, we'll take a good look at what makes up an hReview, building a real example as we go, so you can get some firsthand experience of putting the microformat to use.

The root element

As with the other compound microformats, an hReview is contained within a root element, which may be any HTML element. Using the class design pattern, we give this root element a class value of `hreview`. Typically, to provide maximum possible flexibility, we use a block element and often a `<div>` element for the root of compound microformats, as these can contain any kind of HTML element. Using a `<span>` or another inline element would restrict us to using only inline elements within the root element.

For example, our root element will typically be as follows:

```
<div class="hreview">
</div>
```

This root element may contain one or more of the elements described in the sections that follow, as well as other elements, of course (all compound microformats allow for nonmicroformatted content within them). This approach allows for domain-specific information (such as IMDb's "Contains Spoiler" details) to be mixed in with the formal hReview content.

version

The `version` field is optional. This field does not specify the version of the review; rather, it specifies the version of hReview used to mark up the content. If no `version` is specified, then the version can be taken to mean any version from 0.2 onward. In practice, the version used has little real impact, because the hReview specification's authors are committed to maintaining backward-compatibility for hReview specifications. The current version of hReview is 0.3 (a number less than 1 typically indicates that the format has not yet reached specification status).

To indicate that the version of this hReview is 0.3, we have the following:

```
<div class="hreview">
  <p>hReview version<span class="version">0.3</span><p>
</div>
```

Again, the order in which elements appear within the root element is typically not important, so where this information appears it usually makes sense for it to appear toward the end of the review, because it's of little interest (if any) to most readers.

summary

You might remember from hCalendar that the name of a vevent (i.e., events in hCalendar) is designated by the `summary` property. Where possible, hReview takes properties from hCard and hCalendar. So, instead of inventing a new property name—for example, `title`—we reuse `summary` from hCalendar.

Here, for example, is an example title from a review of a movie I happened to watch last night:

```
<h2 class="summary">Pirates of the Caribbean: Dead Man's Chest ➥
   - Disappointing at best</h2>
```

which gives me the following hReview:

```
<div class="hreview">
  <h2 class="summary">Pirates of the Caribbean: Dead Man's Chest ➥
     - Disappointing at best</h2>
  <p>hReview version<span class="versions">0.3</span><p>
</div>
```

By the way, `summary` too is an optional property—as indeed are all properties other than the `item` property, which we'll get to shortly.

type

The Zagat website contains restaurant reviews. IMDb contains movie reviews. Amazon (largely) contains book reviews. The type of thing we are reviewing is clearly important in many instances (e.g., the indexing software for a decentralized movie review site will probably want an indication of whether a review is about a movie or something else, and then skip any review it finds that is not for a movie).

We can indicate the kind of thing a review is for with the `type` property. The field is, again, optional, but may take one of a short list of possible values:

- `product`
- `business`
- `event`
- `person`
- `place`
- `website`
- `url`

You'll note that in the preceding movie review example, hReview actually won't help our review indexing software. Remember that the hReview specification is still in draft form, and in fact explicitly notes that "the enumerated list of item types is under development and may be extended." Hopefully, some of the most commonly reviewed types, such as book, movie, song, game, and so on, will be added to hReview.

In the case of the preceding movie review, we might add type information as follows:

```
<div class="hreview">
  <h2 class="summary">Pirates of the Caribbean: Dead Man's Chest ➥
     - Disappointing at best</h2>
  <p>Review type: <span class="type">product</span></p>
  <p>hReview version<span class="versions">0.3</span><p>
</div>
```

Adding a review type of “product” is probably artificial, and it’s likely never to be used in this context, but it serves at least as an example of how this property might be used. It also presents internationalization issues. In Korean, for instance, the string `product` may make little sense. Hopefully, future versions of hReview will address internationalization concerns such as this.

item

A moment ago you saw that hReview has only one required property: `item`. It provides a name and should also provide other identifying information, particularly a URL for the item we are reviewing. As shown earlier, where it makes sense, hReview takes its property names from hCard or hCalendar. In hCard, we know there is a way of specifying names using FN. We also know that there is a `url` property. So for things like movies, books, and so on, we can reuse these.

For example, the review could have an item like so:

```
<p class="item">
  <a href="http://disney.go.com/disneypictures/pirates/"
class="fn url">Pirates of the Caribbean: Dead Man's Chest
  </a>
</p>
```

giving us the following hReview so far (and now it’s a valid hReview because it has an `item` property):

```
<div class="hreview">
  <h2 class="summary">Pirates of the Caribbean: Dead Man's Chest ➥
     - Disappointing at best</h2>
  <p class="item">
    <a href="http://disney.go.com/disneypictures/pirates/"
class="fn url">Pirates of the Caribbean: Dead Man's Chest
    </a>
  </p>
  <p>Review type: <span class="type">product</span></p>
  <p>hReview version<span class="versions">0.3</span><p>
</div>
```

If we are reviewing something and want to include a photograph of the item, we can do that using the class design pattern and a value of photo, as follows:

```
<div class="item">
  <p><a href="http://www.yourpsp.com/psp/locale.html" ➥
    class="fn url">Play Station Portable</a>
  <img src="http://www.lis186.com/works/psp_url/psp_url.swf" ➥
    class="photo" /></p>
</div>
```

This is all quite straightforward. But different values apply if we are reviewing a person, a business, or an event. Why? Well, these should all sound quite familiar from previous chapters—and we have microformats already for marking up each of them. So, again, in keeping with the microformats principles of embeddability and modularity, we can embed hCard or hCalendars inside our hReview.

In fact, not only can we do this, but in the case of businesses or people, hCard *must* be used, while in the case of events, hCalendar *should* be used for marking up the item. For example, suppose we are reviewing a concert. In that case, we should mark up the reviewed item as follows:

```
<div class="item vevent">
  <h3 class="summary fn">
    <a href="http://www.asmf.org/html/concerts/uk/30Nov.htm"
class="url">Simply Strings
</a>
  </h3>
  <p>
    <abbr title="20061130T19:30Z">
Sunday, November 30, 2006 7.30pm</abbr>
  </p>
</div>
```

Because items require names, we add the value of fn to the class of the h3.

The case of an hReview for a business or person is similar, except that here we *must* use hCard to mark up the item. For example, if we are reviewing a particular bank, we could have something like the following for our item property:

```
<div class="item vcard">
  <a href="http://www.ing.com" class="fn url org">ING Bank</a>
</div>
```

Notice how the item element also has a vcard value for its class. We saw this in the last chapter where hCards were embedded in hCalendars. The item value is in fact part of the hReview, not the hCard, which follows our general principle that the root element of a microformat only has the root class value for that microformat set on it (but where a microformat is embedded in another microformat, the root element of the embedded microformat may contain a class value for its role in the microformat in which it is embedded).

To summarize this rather complex aspect of hReview,

- The `item` property is *required*.
- The item *must* contain an element with a name, specified by a FN class value; it should provide a URL; and it *may* provide a photo.
- When the item being reviewed is a person or business, the item *must* be marked up as an hCard.
- When the item being reviewed is an event, it *should* be marked up with an hCalendar.

reviewer

While not required, the name of the reviewer is often important information for a review. When a reviewer is provided, you might guess that as with people and the `item` property, hCard is *required*. So, if the author of this book is the reviewer of the movie, we have the following:

```
<div class="hreview">
  <h2 class="summary">Pirates of the Caribbean: Dead Man's Chest ➥
     - Disappointing at best</h2>
  <p class="item">
    <a href="http://disney.go.com/disneypictures/pirates/"
class="fn url">Pirates of the Caribbean: Dead Man's Chest
</a>
  </p>
  <p class="reviewer vcard">
    <span class="fn">John Allsopp</a>
  </p>
  <p>Review type: <span class="type">product</span></p>
  <p>hReview version<span class="versions">0.3</span><p>
</div>
```

Or, if our reviewer feared the wrath of Keira Knightley, he or she may wish to be anonymous, like so:

```
<p class="reviewer vcard"><span class="fn">anonymous</a></p>
```

In short, we use an hCard, with an `fn` value of `anonymous`.

This does present some internationalization issues, because of course "anonymous" is an English word. If our review is in a language other than English, then the appearance of the word "anonymous" is likely to appear incongruous. Perhaps the current best option in a non-English language review by an anonymous reviewer is to simply leave out the reviewer altogether, as `reviewer` is an optional property.

dtreviewed

The date on which something is reviewed is often of some significance. Restaurants may change considerably over time, for instance, as do software and hardware products. Fashions and tastes change, too. For example, *Citizen Kane*, now widely considered one of the best films of all time, only started making the top ten lists of reviewers' best films in the 1970s, 30 years or so after it was made.

The dtreviewed property exists to mark up and publish review dates. As with other date values we've discussed, we can use the abbr design pattern to give both human-friendly and machine-readable values. For example, our review of the movie *Pirates of the Caribbean: Dead Man's Chest* was written on November 18, 2006. In this instance, the time is not relevant, nor indeed is the time zone, so we may leave off that information.

```
<div class="hreview">
  <h2 class="summary">Pirates of the Caribbean: Dead Man's Chest ➥
     - Disappointing at best</h2>
  <p class="item">
    <a href="http://disney.go.com/disneypictures/pirates/"
class="fn url">Pirates of the Caribbean: Dead Man's Chest
</a>
  </p>
  <p class="reviewer vcard">
    <span class="fn">John Allsopp</a>
  </p>
  <p>Date reviewed <abbr title="20061118" class="dtreviewed">
    November 18, 2006</abbr>.
  </p>
  <p>Review type: <span class="type">product</span></p>
  <p>hReview version<span class="versions">0.3</span><p>
</div>
```

ratings

In many ways, ratings are the heart and soul of a reviewing system. Three stars is the *Michelin Guide*'s ultimate restaurant accolade, and the goal of many restaurateurs. "Two thumbs up" is worth a great deal to a movie, literally. But as these two examples demonstrate, not only are ratings fundamental to reviews, but also rating systems vary dramatically. hReview allows us to easily rate the item we are reviewing and even create our own rating system.

By default, a rating is a numerical value to one fixed decimal point of precision, out of five—that is, values like 1.0, 3.5, and so on (2.59 is not a valid value).

It's clear that I didn't particularly like the movie I'm reviewing. I'd give it 2/5. To mark this up with hReview, I do something like this:

```
<p>Rating: <span class="rating">2</span>/5</p>
```

If, however, I want to use another rating system—for example, 0 on a scale of –10 to 10—I can do that as well. I specify the best (and optionally the worst, too) on my scale, as well as the actual rating value I have given the item I'm reviewing:

```
<p>
  Rating: <span class="rating"><span class="value">0</span>
  on a scale of <span class="worst">-10</span> to ➥
    <span class="best">10</span>
</p>
```

So, the lowest end of my range is marked up like this (an integer value may be used for worst):

```
<span class="worst">-10</span>
```

and the highest end of the range is marked up like this:

```
<span class="best">10</span>
```

The actual rating is as follows:

```
<span class="value">0</span>
```

The worst value is optional, and I can even have a higher value for worst than best, as occasionally is seen with rating systems. For example, while it's not quite a review, in yacht racing, the winner of a regatta is typically the boat with the lowest score.

In this case, I'll use a common score out of 10, and give the film a 4:

```
<div class="hreview">
  <h2 class="summary">Pirates of the Caribbean: Dead Man's Chest ➥
     - Disappointing at best</h2>
  <p class="item">
    <a href="http://disney.go.com/disneypictures/pirates/"
class="fn url">Pirates of the Caribbean: Dead Man's Chest
</a>
  </p>
  <p class="reviewer vcard">
    <span class="fn">John Allsopp</a>
  </p>
  <p>Date reviewed <abbr title="20061118" class="dtreviewed">
    November 18, 2006</abbr>.
  </p>
  <p>Rating: <span class="rating"><span class="value">4</span>
    out of <span class="best">10</span>
  </p>
  <p>Review type: <span class="type">product</span></p>
  <p>hReview version<span class="versions">0.3</span><p>
</div>
```

9

description

Longer reviews will usually give the reviewer's opinions, comments, and arguments, and the description property is for this purpose. The description may be any length, and it may, of course, be formatted HTML. Remember the description property of an hCalendar event? There, where the description was longer than a single paragraph, we wrapped the description content in a <div> of class description. We use the same pattern here.

With a description added, our movie review looks like this:

```
<div class="hreview">
  <h2 class="summary">Pirates of the Caribbean: Dead Man's Chest ➥
     - Disappointing at best</h2>
  <p class="item">
    <a href="http://disney.go.com/disneypictures/pirates/"
class="fn url">Pirates of the Caribbean: Dead Man's Chest
</a>
  </p>
  <p class="reviewer vcard">
    <span class="fn">John Allsopp</a>
  </p>
  <p>Date reviewed <abbr title="20061118" class="dtreviewed">
    November 18, 2006</abbr>.
  </p>
  <p>Rating: <span class="rating"><span class="value">4</span>
    out of <span class="best">10</span>
  </p>
  <div class="description">
    <p>After the success and excitement of the original Pirates of ➥
      the Caribbean...</p>
    <p>The characters lack any of the real freshness and cleverness ➥
      of the original, even Depp's Jack Sparrow has become more...</p>
    <p>Disappointing, to say the least.</p>
  </div>
  <p>Review type: <span class="type">product</span></p>
  <p>hReview version<span class="versions">0.3</span><p>
</div>
```

tags

What's a format these days without tagging? hReview doesn't disappoint, using rel-tag as a mechanism for tagging a review. We covered rel-tag extensively in Chapter 4, so there's no need to go into great detail here. One thing to note is that the tag applies to a review here, rather than the whole page, which is a more typical case. As a rule, when a rel-tag is found inside an hReview, it applies to the review it is inside.

One other specific aspect of tagging in relation to reviews is that tags may have ratings associated with them. This may sound strange, but think about restaurant reviews. Often these will give ratings for different aspects of the experience—ambience, service, and food, for instance. There are two ways of doing this: we can either put the rating inside the tag or put the tag inside a rating property. For example, we could have the following:

```
<li class="rating">
  <a href="http://technorati.com/tag/food" rel="tag">food</a>
  <span class="value">7</span> out of ><span class="best">10</span>
</li>
```

The rating contains a tag, as well as the value and scale for the rating itself. In this case, the tag is not for “food” in general, but for the *rating* of food. At the destination of the tag link, there should be a description of how the rating system for food (or whatever is being rated) works. We could also do this:

```
<a rel="tag" class="rating" href="http://technorati.com/tag/food">
  food: <span class="value">7</span> out of
<span class="best">10</span>
</a>
```

This flexibility enables the rating and tag construction to cover more-common situations than would otherwise be the case, if only one approach to markup were allowed.

permalink

Recall that permalinks are typically used in blogs to link to the permanent location of a blog post. In hReview, this is a URL for the permanent location of the review. It’s useful for when a review is aggregated, so if someone reads a review aggregated at another site, the permalink allows the reader to find the original context of the review and other reviews by the same reviewer or other related reviews.

A permalink is marked up using the rel attribute, and has the values self and bookmark added to it. The following:

```
<a href="http://blogs.westciv.com/reviews/pirates2" ➥
  rel="self bookmark">http://blogs.westciv.com/reviews/pirates2</a>
```

or this:

```
<a href="http://blogs.westciv.com/reviews/pirates2" ➥
  rel="self bookmark">Pirates of the Caribbean 2</a>
```

are both permalinks for the review found at the location specified by the href value.

license

When we looked at rel-license back in Chapter 4, we observed that a rel-license link typically refers to the license for a page, but when found inside an hReview, it specifies the license for the review, rather than the page.

To quickly recap the rel-license microformat, it’s simply a hyperlink with a rel value of license and an href value pointing to the license for the review. If we wish to license our review under a Creative Commons Attribution-ShareAlike license (that way, if others wish to use our review as part of their content, they must attribute it to the reviewer and release any work based on this under the same license conditions), we add the following to our review:

```
<a href="http://creativecommons.org/licenses/by-sa/2.5/" ➥
  rel="license">
  Licensed under a creative commons attribution share alike ➥
    2.5 license
</a>
```

Let’s add a license to the review of *Pirates of the Caribbean: Dead Man’s Chest*:

```
<div class="hreview">
  <h2 class="summary">Pirates of the Caribbean: Dead Man's Chest ➥
     - Disappointing at best</h2>
  <p class="item">
    <a href="http://disney.go.com/disneypictures/pirates/"
class="fn url"> Pirates of the Caribbean: Dead Man's Chest
</a>
  </p>
  <p class="reviewer vcard">
    <span class="fn">John Allsopp</span>
  </p>
  <p>Date reviewed <abbr title="20061118" class="dtreviewed">
    November 18, 2006</abbr>.
  </p>
  <p>Rating: <span class="rating">
    <span class="value">4</span> our of <span class="best">10</span>
  </p>
  <div class="description">
    <p>After the success and excitement of the original Pirates of ➥
      the Caribbean...</p>
    <p>The characters lack any of the real freshness and cleverness ➥
      of the original, even ➥
      <a href="http://en.wikipedia.org/wiki/Johnny_Depp" rel="tag"> ➥
      Depp's</a> Jack Sparrow has become more...
     </p>
    <p>Disappointing, to say the least.</p>
  </div>
  <p>
    <a href="http://creativecommons.org/licenses/by-sa/2.5/" ➥
      rel="license">
      Licensed under a creative commons attribution share alike ➥
      2.5 license
    </a>
  </p>
  <p>Review type: <span class="type">product</span></p>
  <p>hReview version<span class="versions">0.3</span><p>
</div>
```

Publishing tools

Although still a draft format, hReview has reasonably wide adoption from publishers, and several tools are available to help alleviate the tedium of marking up your review content.

- **hReview Creator** (`http://microformats.org/code/hreview/creator`): Like microformat.org's other microformat creators, this is a form-driven interface that produces microformatted HTML for your review.
- **Textpattern Microformats Plug-in** (`http://placenamehere.com/TXP/pnh_mf`): For Textpattern users, Chris Casciano's pnh_mf, which has been mentioned in several previous chapters, supports publishing in the hReview format as well as a number of other microformats.

WordPress users are in luck—several plug-ins help publish hReview-formatted content:

- **hReview WordPress Plug-in** (`www.aes.id.au/?page_id=28`): Andrew Scott created this form-driven hReview plug-in, which allows for easy publishing of individual or multiple reviews on a single page or in a single post.
- **WP Movie Ratings** (`http://paulgoscicki.com/archives/2006/11/wp-movie-ratings-v14-released`): This plug-in from Paul Goscicki integrates with IMDb to allow pain-free reviewing of movies in hReview format.

Services using hReview

Between them, the following sites alone must review tens of thousands of products, of considerable diversity. Couple this with the fact that one of them, Revoo, is already aggregating hReview-formatted reviews from elsewhere on the Web, and you can see that the hReview format is already gaining significant adoption, despite still not officially being a 1.0 specification (as of the time of this book was written, at least).

- **Revoo** (`http://revoo.com`): Revoo is a review aggregation and publishing service that both aggregates reviews published on the Web in hReview format and publishes reviews in hReview format. Revoo publishes its reviews in a curious way: on the main review page for a product, where all the reviews for that product are published, Revoo does not use the hReview format. But if you follow the link at the foot of a review titled `Review Confirmed Customer`, then you can get a copy of the review formatted using hReview.
- **Cork'd** (`http://corkd.com`): As mentioned in previous chapters, this wine community site features reviews from a great many members (anyone can join up and review wine for free). Reviews are created using a form-driven interface and then published using hReview.

In Chapter 11 we take a close look at Cork'd, examining the code it uses and some of the designer's decisions in using hReview and other microformats.

- **Yahoo Tech** (`http://tech.yahoo.com`): This technology review site also uses hReview to publish technology reviews from users.

hReview is a straightforward, easy-to-use schema, so if you publish reviews, it's already pretty much an indispensable format for your efforts.

hResume

Resumes, like reviews, represent discrete, reasonably well-understood sets of information that follow a standard set of rules about what should be included. They are also published online typically—but not exclusively—by large, centralized job search sites. All these factors make resumes ideal candidates for a microformat, just as reviews were.

The hResume format, authored by Ryan King, was developed in a similar way to hReview and followed the recommended process for developing a microformat proposal (which we'll look at in detail in Chapter 13). Existing formats, like Europass (`http://europass.cedefop.europa.eu`), XML Résumé Library (`http://xmlresume.sourceforge.net`), and examples of web-based resumes "in the wild" were researched, to develop a simple schema for publishing resumes. As we saw with hReview, where possible, property names are reused from existing microformats.

I should note that the hResume format is still a reasonably early draft, but I think it is worth considering for a couple of reasons. First, as a more recent microformat, it draws heavily on lessons learned from the development of earlier microformats. It also reuses other microformats we've seen quite extensively. Finally, resumes are very common on the Web, and almost all of us have them, and will rely on them to improve our professional position in coming years.

In the sections that follow, we'll examine the schema of hReview, while I do something I didn't think I'd ever do again: build my resume, using hResume.

The hResume schema

Let's now take a closer look what makes up an hResume and build a real example as we go.

The root element

As you should know by now, all compound microformats are contained within a root element, which may be any HTML element, denoted as a root for this microformat by the use of a special class value, in this case—you guessed it—`hresume`. Within this class, hResume defines a number of properties to publish relevant resume information.

The summary property

The `summary` property of hResume is for an "overview of qualifications and objectives." The property name is reused from hCalendar, demonstrating an important, central microformats principle: "don't repeat yourself" (or DRY). With microformats, we always look to reuse existing work, schemas, patterns, class names, and so on, rather than reinventing the wheel each time we have a need.

Now, I haven't needed to have a resume for many years. And I hope to never need one in the future, for that matter—at least not for obtaining a job. I've worked for myself for over a decade, and I think everyone is better off if it stays that way. But let's try shoehorning my decade of self-employment and many different activities into an hResume:

```
<div class="hResume">
  <h1>Resume: John Allsopp</h1>
  <p class="summary">Software developer, conference organizer, ➥
    speaker, and author John Allsopp has spent the last 15 years ➥
    running and developing software for Western Civilisation Pty. ➥
    Ltd., writing and publishing training courses, speaking at ➥
    conferences, organizing conferences, and running training ➥
    seminars in standards-based web development.</p>
</div>
```

Contact details

Most people publish resumes to get work. In order to get work, you need someone to contact you. And so it makes sense that the hResume format provides for contact details. In fact, you already know a lot about this from Chapter 7—we'll reuse hCard to add contact details to the example resume.

As you've seen with other microformats, when you embed one microformat in another, then the root element of the embedded format (in this case, the hCard) has both a class value as the root of its format and another for the property it is playing in the microformat that contains it.

In this example, we have both vcard as the root of the hCard and contact for the role it plays in the hResume:

```
<div class="hResume">
  <h1 class="vcard contact">Resume:
    <a href="mailto:john@webdirections.org" class="fn email">
      John Allsopp</a>
  </h1>
  <p class="summary">Software developer, conference organizer, ➥
    speaker, and author John Allsopp has spent the last 15 years ➥
    running and developing software for Western Civilisation Pty. ➥
    Ltd., writing and publishing training courses, speaking at ➥
    conferences, organizing conferences, and running training ➥
    seminars in standards-based web development.</p>
</div>
```

The contact details *must* be marked up using hCard, and in addition, we *should* use the address element of HTML for this element. There are two complicating factors to using an address element, however. First, keep in mind that the HTML address element is not just for any addresses, but specifically (according to the HTML specification) for "contact information for a document or a major part of a document." In this case, it could be argued that the hResume is "a major part of a document." So the address element makes sense, particularly if you are publishing your own hResume. But address is an inline element, which means it may *only* contain other inline elements. This presents complications when marking up, for instance, postal addresses, where you can't simply rely on the default rendering of <div>, <p>, and <span> elements, but need to use either <br /> elements or specific CSS for formatting these addresses correctly. It also means that with CSS off, postal addresses won't format correctly, unless you use <br /> elements. Of course, this isn't a fatal problem, but it's one worth noting.

In our case, we'll avoid the issue by only publishing an email address. The <a> element is, of course, an inline element, so we can contain it in an address element:

```
<div class="hResume">
  <h1>Resume:
    <address class="vcard contact">
      <a href="mailto:john@webdirections.org" class="fn email">
        John Allsopp
      </a>
    </address>
  </h1>
  <p class="summary">Software developer, conference organizer, ➥
    speaker, and author John Allsopp has spent the last 15 years ➥
    running and developing software for Western Civilisation Pty. ➥
    Ltd., writing and publishing training courses, speaking at ➥
    conferences, organizing conferences, and running training ➥
    seminars in standards-based web development.</p>
</div>
```

Education

Typically, a resume lists the writer's educational qualifications. If you think about the microformats we've covered and the importance of reuse in microformats generally, then you can think of a person's education as a series of events, with start and end dates, which are amenable to marking up with hCalendar.

For this example's list of education achievements, we'll have a root element with a class value of hcalendar, which contains a series of vevents that also have the class education. In this case, it makes sense to explicitly use the hCalendar element (recall from Chapter 8 that the hCalendar element is optional, and where it's missing, the page itself is taken to be the calendar), because when it comes to work experience, we'll have a second hCalendar for those events. It also may be that an aggregator site publishes more than one hResume per page, in which case we'll need separate hCalendars for each resume.

In addition, if you think about the most appropriate HTML element, a list makes sense—after all, a person's education is a list of events. And lastly, it would be an ordered list, because the list is ordered in reverse chronological order.

Putting this all together, we have the following hCalendar for our example's education portion:

```
<ol class="vcalendar">
  <li class="vevent education">
    <abbr class="dtstart" title="19850101">1985</abbr>-
    <abbr class="dtend" title="19891231">1989</abbr>
    <abbr title="Bachelor of Science ">B.SC</abbr>
    <span class="vcard">
      <a class="fn org url" href="http://www.usyd.edu.au/">
University of Sydney, Australia</a>
    </span>
    <span class="summary">
      Majored in computer science and mathematics.
Also studied law, English literature, and philosophy.
    </span>
  </li>
  <li class="vevent education">
    <abbr class="dtstart" title="19790101">1979</abbr>-
    <abbr class="dtend" title="19841231">1984</abbr>
    <abbr title="Higher School Certificate">H.S.C</a>
    <span class="vcard">
      <a class="fn org url" href="http://www.riverview.nsw.edu.au">
Saint Ignatius College, Riverview</a>
    </span>
  </li>
</ol>
```

Notice that I've taken the opportunity to add informative markup, by expanding abbreviations and using hCards for educational institutions. As I mentioned in earlier chapters, once you start thinking about HTML in a semantic way, the opportunity for richer semantics in your documents presents itself all over the place. Now, you might argue it's just showing off, or a waste of time. Perhaps you are right. But the principle I keep in mind when marking up is akin to Metcalfe's Law for the value of a telecommunications network, famously proposed by Robert Metcalfe, inventor of Ethernet. He suggested that "the value of a telecommunications network is proportional to the square of the number of users of the system."

While the accuracy of Metcalfe's Law is often criticized, few people doubt the general predictive power of the observation. Similarly, I think the value of a body of information is somehow a function of the complexity of its semantic markup—or, to put it simply, the more semantic markup there is in a body of information, the more valuable it is. I'd argue that increasing the volume of information, from say 20 billion to 100 billion indexed documents, won't increase the value of the Web nearly as much as increasing the amount of structured semantic markup in the existing body information in a nontrivial way.

By taking every opportunity to meaningfully mark up our content with appropriate HTML and microformats, we are contributing to the increased value of the Web. That seems a reasonable motivation for a little extra effort to me.

Experience

Just as a person's education can be considered a series of events, so too can an individual's work experience. So again we'll use an ordered list, which is an hCalendar with a set of vevents, to represent work history in our example. Each entry in the list gets a class value of experience to denote that the event is an experience.

```
<ol class="vcalendar">
  <li class="vevent experience">
    <abbr title="19940101" class="dtstart">1994</abbr>
 - present. Chief Technology Officer,
    <span class="vcard">
      <a class="fn org url" href="http://westciv.com">
Western Civilisation Pty. Ltd.</a>
    </span>
    <span class="summary">
      Responsible for the development of software, including ➥
        Palimpsest, Style Master, and Layout Master
    </span>
  </li>
  <li class="vevent experience">
    <abbr title="20060101" class="dtstart">2006</abbr>-present,
    Director<span class="vcard">
      <a class="fn org url" href="http://webdirections.org">
Web Directions Conferences Pty. Ltd.</a>
    </span>
    <span class="summary">
      Responsible for a broad range of logistical and management ➥
      aspects of organizing the Web Directions conferences.
    </span>
  </li>
  <li class="vevent experience">
    <abbr title="20030101" class="dtstart">2003</abbr> -
    <abbr title="20051231" class="dtstart">2005</abbr> Director,
    <span class="vcard">
      <a class="fn org url" href="http://webessentials.com">
WE04 and WE05 conferences</a>
    </span>
    <span class="summary">
      Responsible for a broad range of logistical and management ➥
      aspects of organizing the WE04 and WE05 conferences.
    </span>
  </li>
</ol>
```

Here you might notice that I've used the summary property of hCalendar for the description of my role and an hCard for the company or organization I played the role at—just as I did for the school, university, or other institution at which I was educated.

If you think back to Chapter 7, the idea of roles and titles should be familiar. In this case, we have “Chief Technology Officer” and “Director,” both of which are titles. It makes sense to mark these up with hCard—but we have a problem. hCards *require* a formatted name (FN), so we would have to add an extraneous name for every hCard, perhaps hiding it with CSS, which is something you should always try to avoid. In fact, this problem has shown up more than once during the evolution of microformats, leading to the development of a new design pattern along the lines of the class and abbr design patterns: the **include pattern**.

The problem we are trying to solve with this pattern is how to include data from one microformat on a page into another. So, for example, here we have a need for several hCards on the page that all have the same FN value, but different title or role property values. Ideally we’d only include the FN after having specified it once, rather than repeating the value in every hCard with that FN value.

The include pattern enables this as follows. First, we need to uniquely identify the hCard (or other microformatted) element of the page to be included. We know the standard HTML mechanism for this: using an id value. In the first of the hCards on the page, then, we add an id value to the root element (the properties of which we’ll want to include elsewhere)—in this case, that will be the hCard.

```
<div class="hResume">
  <h1 class="summary">Resume: John Allsopp</h1>
  <div class="vcard contact" id="jafn" >
    <a href="mailto:john@webdirections.org" class="email fn"  >
      John Allsopp
    </a>
  </div>
</div>
```

Now, to include the FN into another hCard, we use a link to the hCard we want to include a value from, and additionally give the link a class value of include:

```
<span class="vcard">
 <a href="#jafn" class="include"></a>
 <span class="title">Chief Technology Officer</span>
</span>
```

This gives us experience properties like the following:

```
<li class="vevent experience">
  <abbr title="19940101" class="dtstart">1994</abbr> - present.
  <span class="vcard">
    <a href="#jafn" class="include"></a>
    <span class="title">Chief Technology Officer</span>
  </span>,
  <span class="vcard">
    <a class="fn org url" href="http://westciv.com">
Western Civilisation Pty. Ltd.</a>
  </span>
```

```
    <span class="summary">
      Responsible for the development of software, including ➥
      Palimpsest, Style Master, and Layout Master
    </span>
  </li>
```

Skills

In a resume, the skills a person has are typically presented as a list of descriptive words or phrases. For example, I might list my skill set as follows: software engineering, programming, web development, C++, REALbasic, Java, CSS, HTML, XHTML, XML, DOM, JavaScript, surf lifesaving. Not only is there a mechanism for us to list our skills in an hResume, but also we can actually *tag* our resume with these skills.

Given that our set of skills is a list, we'll use a standard HTML list to mark them up. We'll denote this as a list of skills by giving each entry in the list a `class` value of `skill`. And we'll turn each of the entries in the list into a tag by making it a link to a tag space, with a `rel` value of `tag`. For example, my set of skills might be marked up as follows:

```
<ul>
  <li>
    <a class="skill" ➥
      href="http://en.wikipedia.org/wiki/Software_engineering" ➥
      rel="tag">software engineering</a>
  </li>
  <li>
    <a class="skill" href="http://en.wikipedia.org/wiki/Programming" ➥
      rel="tag">programming</a>
  </li>
  <li>
    <a class="skill"➥
      href="http://en.wikipedia.org/wiki/Web_development" ➥
      rel="tag">web development</a>
  </li>
  <li>
    <a class="skill" href="http://en.wikipedia.org/wiki/C++" ➥
      rel="tag">C++</a>
  </li>
  <li>
    <a class="skill" href="http://en.wikipedia.org/wiki/Realbasic" ➥
      rel="tag">REALbasic</a>
  </li>
  <li>
    <a class="skill" href="http://en.wikipedia.org/wiki/Java" ➥
      rel="tag">Java</a>
  </li>
  <li>
    <a class="skill" href="http://en.wikipedia.org/wiki/CSS" ➥
      rel="tag">CSS</a>
  </li>
```

```
    <li>
      <a class="skill" href="http://en.wikipedia.org/wiki/HTML"➥
         rel="tag">HTML</a>
    </li>
    <li>
      <a class="skill" href="http://en.wikipedia.org/wiki/XHTML" ➥
        rel="tag">XHTML</a>
    </li>
    <li>
      <a class="skill" href="http://en.wikipedia.org/wiki/XML" ➥
        rel="tag">XML</a>
    </li>
    <li>
      <a class="skill"➥
        href="http://en.wikipedia.org/wiki/Document_Object_Model" ➥
        rel="tag">DOM</a>
    </li>
    <li>
      <a class="skill" href="http://en.wikipedia.org/wiki/Javascript" ➥
        rel="tag">JavaScript</a>
    </li>
    <li>
      <a class="skill" ➥
        href="http://en.wikipedia.org/wiki/Surf_lifesaving" ➥
        rel="tag">surf lifesaving</a>
    </li>
  </ul>
```

Or we could mark this up in a less highly structured, more narrative way as follows:

```
I've been a <a class="skill" href="http://en.wikipedia.org/wiki/
Software_engineering" rel="tag">software engineer</a> and
<a class="skill" href="http://en.wikipedia.org/wiki/Programming"
rel="tag">programmer</a>, with languages such as <a class="skill"
href="http://en.wikipedia.org/wiki/C++" rel="tag">C++</a>,
<a class="skill" href="http://en.wikipedia.org/wiki/Java"
rel="tag">Java</a>, and <a class="skill"
href="http://en.wikipedia.org/wiki/Realbasic" rel="tag">REALbasic</a>
for nearly 20 years, and a standards-based <a class="skill"
href="http://en.wikipedia.org/wiki/Web_development"
rel="tag">web developer</a> with a strong
understanding of <a class="skill" href="http://en.wikipedia.org/wiki/
CSS" rel="tag">CSS</a>, <a class="skill" href="http://en.wikipedia.org/
wiki/HTML" rel="tag">HTML</a>, <a class="skill"
href="http://en.wikipedia.org/wiki/XHTML" rel="tag">XHTML</a>,
<a class="skill" href="http://en.wikipedia.org/wiki/DOM"
rel="tag">the DOM</a>, and <a class="skill"
```

```
href="http://en.wikipedia.org/wiki/Javascript" rel="tag">JavaScript</a>
for over a decade. In my occasional spare moments, I am an experienced
volunteer <a class="skill" href="http://en.wikipedia.org/wiki/
Surf_lifesaving" rel="tag">surf lifesaver</a>, and regularly train
people in <a class="skill" href="http://en.wikipedia.org/wiki/
First_aid" rel="tag">first aid<a>, <a class="skill"
href="http://en.wikipedia.org/wiki/CPR" rel="tag">
<abbr title="Cardio Pulmonary Resuscitation">CPR</abbr></a>,
use of the <a class="skill" href="http://en.wikipedia.org/wiki/
defibrillator" rel="tag">defibrillator</a>, as well as surf rescue skills.
```

Again, here you can see the flexibility of microformatted markup. You aren't *required* to use a list or any specific elements—rather, you can fill in the microformat information around existing markup.

Affiliations

An important part of a professional resume is the organizations with which you are affiliated. You can use hResume to create these associations by listing or otherwise publishing the names of the organizations. In what should be becoming a familiar pattern, we'll use a list in our example. Each list item will have a class value of affiliation, and each affiliated organization will be, not surprisingly, marked up using hCard.

In my case, I've been a member of the Web Standards Project (WaSP) for quite a few years now, so I'll add that, as well as my surf lifesaving club, North Bondi SLSC.[1] To the best of my knowledge, I am the only surf lifesaving member of WaSP.

```
<ul>
  <li class="vcard affiliation">
    <a href="http://www.webstandards.org/" class="fn org url">
      Web Standards Project
    </a>
  </li>
  <li class="vcard affiliation">
    <a href="http://www.northbondisurfclub.com/" class="fn org url">
      North Bondi SLSC
    </a>
  </li>
</ul>
```

I could, of course, similarly mark up these affiliations in a more narrative manner as well:

```
Early member of the CSS Samurai, a part of
<abbr title="Web Standards Project" class= "vcard affiliation">
<a href="http://www.webstandards.org/" class="fn org url">WaSP</a>
</abbr>. Patrol captain, <span class="vcard affiliation">
<a href="http://www.northbondisurfclub.com/"
class="fn org url">North Bondi SLSC.</a></span>
```

1. For you Macintosh aficionados out there, the original Mac color, "Bondi Blue," was named after the beach at which I am a lifesaver.

Publications

In many fields, particularly academic ones, what you publish is central to your professional history, so hResume provides a way of marking up publications. At present, the citation microformat is still very much under development, so we can use the HTML compound for citations we put together in Chapter 4. In my case, I've published quite a few online articles, and of course this book. I can list these publications as follows:

```
<ol>
  <li>
    <cite href="http://www.alistapart.com/articles/dao/"> ➥
      <a href="http://www.alistapart.com/articles/dao/"> A Dao of ➥
      Web Design</a></cite> A List Apart, April 2000
  </li>
  <li>
    <cite href="http://www.westciv.com/style_master/house/good_oil/ ➥
      not_paper/"><a href="http://www.westciv.com/style_master/house/ ➥
      good_oil/not_paper/">Web Pages aren't printed on paper</a> ➥
      </cite>September 1999
  </li>
</ol>
```

You might notice that I didn't add any class values to the list of the items to denote these are publications; simply using the cite element is sufficient. This also means that any cited works like this in an hResume are assumed to be published by the person for whom this is the resume.

hResume tools and services

Although still a new draft format, a number of tools for helping publish hResume content are available:

- **hResume Creator** (`http://hresume.weblogswork.com/hresumecreator`): This form-based resume creator from Spur, Inc. is much like the other microformats creators, such as hCard Creator and hCalendar Creator.
- **hResume WordPress Plug-in** (`http://hresume.weblogswork.com/?page_id=3`): Another tool from Spur, Inc., this plug-in for WordPress helps publish hResumes on WordPress-based blogs.
- **Microformat Resume Plug-in for WordPress** (`www.ssdesigninteractive.com`): Sajid Sayid created this easy-to-use plug-in for adding hResume content to a WordPress-based site.

A couple of services are already using the format as well:

- **Emurse** (`http://emurse.com`): This service for building, hosting, and publishing resumes online uses the hResume format.
- **LinkedIn** (`www.linkedin.com`): This professional online networking service uses hResume for publishing resume details.

Styling hReview and hResume content with CSS

Hopefully by now you're starting to see how CSS can be used with microformats. What makes hReview and hResume so easy to style with CSS is that they largely work with the class design pattern, making almost all the property elements available to any browser using the class selector. In the case of publications, where we don't have an explicit class, we can easily use the descendent selector:

```
.hresume cite {}
```

which selects any `<cite>` element inside an element of class `hresume`.

Coming up in the Cork'd case study in Chapter 11, you'll see how noted designer and developer Dan Cederholm uses hReview with CSS—making anything I can tell you here redundant.

Summary

I hope that this chapter has been relatively smooth sailing for you, after all your hard work examining the intricacies of semantic HTML, the various microformats, and design patterns over the first eight chapters of this book. In this chapter, you saw how these techniques and the microformats principles combine to help you develop a logical schema to consistently mark up quite complex types of information in HTML. You also saw a new design pattern, the include design pattern, which you'll revisit in Chapter 11 during an analysis of the wine review site Cork'd.

You have one more new microformat, hAtom, to cover in the next chapter. Again, it's a draft microformat, and again you'll be putting a lot of the knowledge you've gained so far to work. Then it will be time to take a look in detail at how a number of sites are using microformats today.

10 SYNDICATED CONTENT MICROFORMAT: HATOM

Syndicated content, most commonly blog posts, has been one of the fastest growing kinds of content on the Web the last four or five years. Typically, the syndication is enabled not with HTML-based content, but with complementary formats, like RSS.

In this chapter, we'll examine hAtom, a microformat for marking up syndicated content logically in HTML, aiding archiving and content subscription. But first, we'll look at web-based subscription services in general.

Web-based subscription services

One of the significant changes in web use of the past few years has been the increased use of subscription-based services (typically blogs, and similar reasonably frequently updated information) to syndicate content, using RSS. While we typically think of **RSS** as a single, monolithic technology, it is in fact a number of somewhat incompatible versions of Rich Site Summary (RSS 0.91 and RSS 1.0), Really Simple Syndication (RSS 2.0), and RDF Site Summary (RSS 0.9 and 1.0), as well as the totally unrelated (technically at least) Atom, which is usually lumped in with RSS despite this technical difference.

For the moment, let's forget the differences and think about the similarities among these formats. They exist to allow publishing "feeds" of usually frequently updated content in a format that enables dedicated software to present new content to users, allowing users to manage a potentially large number of these feeds. While we usually associate RSS and feeds with blogs, RSS is increasingly being used by web applications to notify users of all kinds of changes to information. For example, the photo sharing site Flickr allows you to subscribe to a number of different feeds: a feed of the comments on photos you have posted, or feeds of photos by particular people or a group of people, such as your friends, family, or contacts. So the similarities are that, in effect, all these different versions of "RSS" enable more or less the same thing.

In addition, all of these RSSs are also XML-based languages, meaning that feeds must at least be well-formed documents. It is a feature of XML that any XML processor must report an error and stop processing as soon as an error occurs; compare this with the extreme leniency of HTML processors.

A significant difference between RSS (RDF Site Summary, Really Simple Syndication, and Rich Site Summary) and Atom is that Atom is in fact a draft IETF standard (RFC 4287), while the various RSSs are de facto standards, meaning that they're widely adopted and used, but nowhere are they standardized.

So what does all this have to do with microformats, which are about marking up web pages, not about creating XML-based syndication formats? Recall that microformats exist to solve problems. The feeds for blogs and other syndicated content (e.g., newspaper articles online) generally only carry relatively recently published posts or articles. This means that for archiving purposes, feeds are typically of limited value. After a reasonably short period of time, posts disappear from a feed, and finding them becomes difficult. For example, one of my blogs has several hundred posts, and I often need to search my own blog using Google to find something I posted in the last few months, because it has vanished from the feed for that blog.

Actual blog posts and other such syndicated content are almost invariably published as HTML, *as well as* in a feed format, leading both to duplication and to poorer semantics for the permanent, more widely available, and more likely search engine–indexed HTML-based content. The feed, which typically has a use-by date of a certain number of posts, is much less likely to be indexed by search engines. In fact, if you take a look at the HTML code produced by just about any widely used blogging tool such as Moveable Type or WordPress, you'll probably find some kind of idiosyncratic, pseudo-semantic markup, but of course, little if any consistency across systems.

So, having identified a problem, we turn to existing schemas to help us develop a format for marking up blogs and other syndicated content in HTML. As you've seen, there are a number of quite widely used formats, of which Atom has the benefit of being (very nearly) an IETF standard (it's in the final stages of draft status), and as such presents probably the best candidate for a syndicated content microformat. Enter hAtom.

hAtom overview

hAtom is a draft microformat, based on a "minimal" subset of the draft Atom 1.0 specification. The focus of hAtom is that "content that can be syndicated, primarily but not exclusively weblogs," and the aim is a format for "identifying semantic information in weblog posts and practically any other place Atom may be used, such as news articles" (per microformats.org). So what's left out of hAtom that's in Atom? hAtom restricts itself to that part of the Atom schema specifically associated with blog posts.

Because hAtom is focused on the kind of information typically produced by some form of tool, content management system (CMS), or application, hand-coding of Atom content will most likely be rare. But an ideal place for developing hAtom content is in blogging system and CMS templates. So if that's something you've done or are interested in doing, you can bring hAtom to a lot of people by simply modifying your templates (typically it shouldn't be all that difficult). That way, anyone who uses your templates will have hAtom marked-up content the moment he or she switches on the template.

Using hAtom

As mentioned previously, hAtom is based on a subset of Atom (we'll cover the properties of hAtom in just a moment). It's worth noting that semantically Atom and hAtom content are essentially equivalent. But in terms of markup, hAtom and Atom are often very different. This is not simply because Atom is an XML-based language, but also because while Atom feeds are typically a simple list of entries, the way real, live blogs and other forms of syndicated content are published on the Web differs markedly. hAtom also provides a set of guidelines for addressing this divergence where it may be an issue.

In many ways, hAtom is like hCalendar. The root element is an optional hfeed element, which contains one or more hentry elements. Compare this with hCalendar, which has an optional vcalendar element and one or more vevent elements.

Each hAtom entry has required title, updated, published, and author properties, and optional content, summary, and permalink (bookmark) properties. Let's take a look at these in detail, marking up a blog post as we go. Here's our unadorned HTML:

```
<h3>Introducing Web Connections</h3>

<p>Super smart, far too young and good-looking, ➥
  <a href="http://toolmantim.com/">Tim Lucas</a> and Cam (" ➥
  <a href="http://www.themaninblue.com/">the man in blue</a>") Adams ➥
  have just taken the wraps off a fantastic web app, " ➥
  <a href="http:///connections.webdirections.org">Web ➥
  Connections</a>" they've built to help networking ➥
  at our upcoming (next week!) conference, ➥
  <a href="http://www.webdirections.org/">Web Directions ➥
  </a>.</p>

<p>Tim and Cam have incorporated rel-tag, XFN, hCard, and hCalendar. ➥
  <strong>Seriously microformatted</strong>.</p>

<p>Right now you can check it out, but to keep weight off the ➥
  servers, it's only open to signup for conference attendees for ➥
  now, but I am sure that will change down the track.</p>

<p>So check it out. Fantastic work, Tim and Cam.</p>
```

First, we need a root element for the feed, of which this is an entry:

```
<div class="hfeed">
  <div class="hentry">
  </div>
</div>
```

Next, we have a title, which is one of the required properties. Illustrating how microformats aim to work with existing content, we can reuse our h3 for this, and using the class design pattern (you'll be dreaming about design patterns by the time I'm finished with you), we get this title for our entry:

```
<div class="hentry">
  <h3 class="entry-title">Introducing Web Connections</h3>
</div>
```

entry-title, like all the property names we use for hAtom, comes straight from Atom.

Next, we do the required updated date. We actually have to add this; typically, this is something that our publishing tool will do for us automatically. In this case, the post was published using WordPress, which adds the publication date. You now know a lot about dates in microformats, so it will come as no surprise that we use the date design pattern to mark up the date.

```
<p>Published:<abbr class="updated published" title="20060920"> ➥
  September 20, 2006</abbr></p>
```

Now, I did something sneaky here: as the updated date and the published date are the same, I overloaded the published element. But we could, of course, have two elements: one for the publication date and one for any updated date.

```
<p>Published:<abbr class="published" title="20060920">September 20, ➥
  2006</abbr>, last updated:<abbr class="updated" title="20060924"> ➥
  September 24, 2006</abbr></p>
```

The last of the required properties is author, so we add that:

```
<p>By: <span class="vcard author"><a href="http://blogs.westciv.com" ➥
  rel="me" class="fn url">John Allsopp</a></span></p>
```

There's a bit going on there, so let's take a look at it piece by piece. First, as with education events in hResume, we have two class values on the vcard span. The value vcard makes the element an hCard, while the value author designates that this hCard is for the author of the entry. We also add a rel="me" value, which helps identify all my blog posts wherever they might be posted (after all, in a previous chapter you saw that there are other John Allsopps out there).

Now let's take a look at the optional properties. People write and publish blog posts and other kinds of entries for people to read. Or at least I assume so (sometimes I think they post them for other people to comment on, which is not exactly the same thing). The meat of an entry, the content, is marked up in an entry property. An entry doesn't have to have content; it can have more than one content property, as well.

In our case, it makes sense to mark up all the content as a single content property, because it consists of a single chunk of information. If for some reason a post contained two discrete chunks of information, then it makes sense to mark these up as distinct content elements. Our example entry now looks like this:

```
<div class="hentry">
  <h3 class="entry-title">Introducing Web Connections</h3>
  <p>Published:<abbr class="updated published" title="20060920"> ➥
    September 20, 2006</abbr></p>
  <div class="content">
    <p>Super smart, far too young and good-looking, ➥
      <a href="http://toolmantim.com/">Tim Lucas</a> and Cam ➥
      ("<a href="http://www.themaninblue.com/">the man in blue</a>") ➥
      Adams have just taken the wraps off a fantastic web app, ➥
      "<a href="http:///connections.webdirections.org">Web ➥
      Connections </a>" they've built to help networking ➥
      at our upcoming (next week!) conference, ➥
      <a href="http://www.webdirections.org/">Web ➥
      Directions</a>.</p>

    <p>Tim and Cam have incorporated rel-tag, XFN, hCard, ➥
      and hCalendar. <strong>Seriously microformatted</strong>.</p>
```

10

```
    <p>Right now you can check it out, but to keep weight off the ➥
      servers, it's only open to signup for conference attendees ➥
      for now, but I am sure that will change down the track.</p>

    <p>So check it out. Fantastic work, Tim and Cam.</p>

  </div>
</div>
```

Now that we have the required properties of our hAtom marked-up feed done, let's take a look at some more detailed, optional, but commonly used properties.

Summaries

Blog posts often appear in a feed with a short entry and a link to the full post. The short "teaser" in Atom (which often simply turns people off—who really wants to visit a site to read a few more lines?) is called a summary. This could be a completely distinct part of the entry, for example:

```
<p class="summary">Smart young guys build cool new web app</p>
```

Or we could reuse a relevant part of the content like this:

```
<div class="content">
  <p><span, class="summary">Super smart, far too young and good- ➥
    looking, <a href="http://toolmantim.com/">Tim Lucas</a> and Cam ➥
    ("<a href="http://www.themaninblue.com/">the man in blue</a>") ➥
    Adams have just taken the wraps off a fantastic web app, ➥
    "<a href="http:///connections.webdirections.org">Web Connections ➥
    </a>"</span> they've built to help networking at our upcoming ➥
    (next week!) conference, <a href="http://www.webdirections.org/"> ➥
    Web Directions</a>.</p>

   ...

</div>
```

Notice again with the microformats approach, we don't have to bend existing content and markup out of shape. We just use the existing structure of our content intelligently, adding an element here, an attribute there, and maintaining as much of the original as possible.

Permalinks

Articles like blog posts are typically published on the main page of a blog or site section, and then after a period of time, they end up archived on secondary pages. But it's important to enable someone to link to an entry once and for that link to always be maintained as the permalink to this post. hAtom supports permalinking—all we need to do is use a draft microformat, rel-bookmark, which uses one of the HTML specification's "reserved" values for the rel attribute described in Chapter 4.

rel-bookmark is a hyperlink element, with a rel of bookmark. In effect, using this microformat, we say that the destination of the link is a bookmark, or permanent location for the part of the page where the link appears. So, to indicate that that the URL http://microformatique.com/?p=37 is the permalink for our post, we use this link:

```
<a href="http://microformatique.com/?p=37" rel="bookmark">
  Permalink
</a>
```

Categories

What's a good post without a category or three? **Categories** are the forerunners to tagging (and, in fact, Technorati tag search treats categories from blogging systems like Blogger and WordPress as tags for the purposes of indexing). It should come as no surprise that Atom, and thus hAtom, supports categories, and that it uses rel-tag to do so.

There are two kinds of categories: **feed** categories, which are labels or tags that describe what whole feeds are about, and **entry** categories, which describe what an individual entry is about. Both categories are marked up in the same way, using the rel-tag microformat. The only difference is where the tags appear. rel-tags inside an entry property are tags or categories for the *entry*, while those outside any entry, but inside a feed, apply to the *whole feed*.

For example, the blog in which the post we are marking up appears is about microformats, semantics, HTML, and patterns. To give these categories to the feed as a whole, we have this:

```
<div class="hfeed">
  <ul>
    <li><a href="http://www.technorati.com/tag/microformats"
rel="tag"> microformats</li>
    <li><a href="http://www.technorati.com/tag/semantics" rel="tag"> ➥
      semantics </li>
    <li><a href="http://www.technorati.com/tag/HTML" rel="tag"> ➥
      HTML</li>
    <li><a href="http://www.technorati.com/tag/patterns" rel="tag"> ➥
      patterns</li>
  </ul>

  ...

</div>
```

On the other hand, the particular entry is about web applications, microformats, WD06 (the tag invented for the Web Directions conference), and so on. We simply add a list like the following to our entry to add these categories to it:

```
<ul>
  <li><a href="http://www.technorati.com/tag/microformats"
 rel="tag"> microformats</li>
  <li><a href="http://www.technorati.com/tag/webapps" rel="tag"> ➥
    webapps</li>
  <li><a href="http://www.technorati.com/tag/WD06" rel="tag"> ➥
    WD06 </li>
</ul>
```

and put it anywhere inside our hentry property element.

Putting all of this together, here's the markup for our feed:

```
<div class="hfeed">
  <ul>
    <li><a href="http://www.technorati.com/tag/microformats"
rel="tag"> microformats</li>
    <li><a href="http://www.technorati.com/tag/semantics" rel="tag"> ➥
      semantics </li>
    <li><a href="http://www.technorati.com/tag/HTML" rel="tag"> ➥
      HTML</li>
    <li><a href="http://www.technorati.com/tag/patterns" rel="tag"> ➥
      patterns</li>
  </ul>

  [there may be many hentries here before we get to the entry we have ➥
   been marking up]


  <div class="hentry">
    <h3 class="entry-title">Introducing Web Connections</h3>
    <p>Published:<abbr class="updated published" title="20060920"> ➥
      September 20, 2006</abbr></p>
    <div class="content">
      <p><span, class="summary">Super smart, far too young and good- ➥
        looking, <a href="http://toolmantim.com/">Tim Lucas</a> and ➥
        Cam ("<a href="http://www.themaninblue.com/">the man in ➥
        blue</a>")Adams, have just taken the wraps off a fantastic CCC
        web app, "~<a href="http:///connections.webdirections.org"> ➥
        Web Connections </a>"</span> they've built to help networking ➥
        at our upcoming (next week!) conference, ➥
        <a href="http://www.webdirections.org/">Web Directions</a>.</p>

      <p>Tim and Cam have incorporated rel-tag, XFN, hCard, and ➥
        hCalendar. <strong>Seriously microformatted</strong>.</p>

      <p>Right now you can check it out, but to keep weight off the ➥
        servers, it's only open to signup for conference attendees ➥
        for now, but I am sure that will change down the track.</p>
      <p>So check it out. Fantastic work, Tim and Cam.</p>
    </div>

    <p>Categories</p>

    <ul>
      <li><a href="http://www.technorati.com/tag/microformats"
rel="tag"> microformats</li>
      <li><a href="http://www.technorati.com/tag/webapps" rel="tag"> ➥
        webapps</li>
      <li><a href="http://www.technorati.com/tag/WD06" rel="tag"> ➥
        WD06 </li>
    </ul>

  </div>

  [there may be many more entries after this one in the feed]

</div>
```

Services using hAtom

Because hAtom is still young, it has not seen (as yet) the level of adoption of some of the other microformats. However, there are already some interesting hAtom developments, both on the publishing and processing side, as outlined in the sections that follow.

Publishing

The open source WordPress blogging system (which is increasingly used as a content management system) is where the most hAtom-related action seems to be taking place.

The Sandbox theme (`www.plaintxt.org/themes/sandbox`), hinted to likely be the basis for the upcoming default WordPress theme, by Scott Allan Wallick and Andy Skelton, is a skeleton theme (i.e., you can use it to develop your own themes on top of) that supports hAtom. There's also a version of the venerable Kubrick theme (the default 2.0 theme for WordPress) by Bill Humphries that uses hAtom.

If you are interested in adding hAtom to your existing WordPress theme or install, Frances Berriman has a WordPress loop to add hAtom to your blog (see `www.fberriman.com/?p=86`).

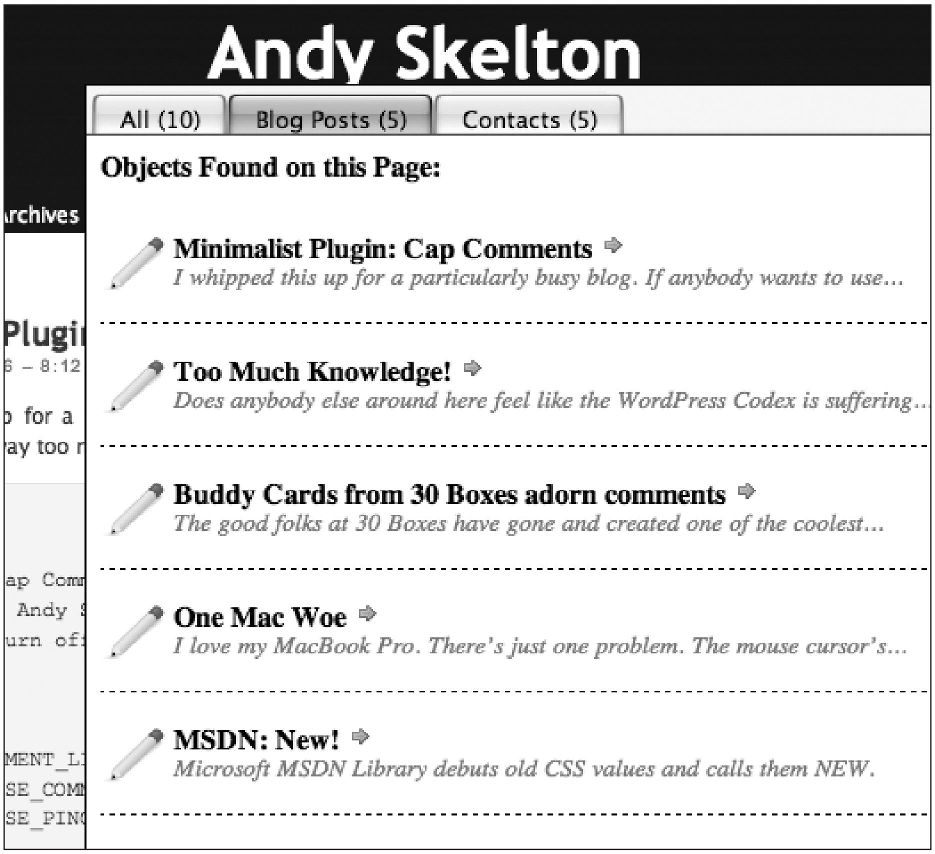
There's also an hAtom Creator for creating individual posts, at `http://dichotomize.com/uf/hatom/creator.html`. Along with the other microformats creators covered in this book, it's a great way of experimenting with a new format.

Processing

On the processing side, a number of projects and toolkits are available that can help make processing hAtom content pain-free.

First, there's hAtom2Atom, an XSLT processor for transforming hAtom marked-up content to Atom (see `http://rbach.priv.at/hAtom2Atom/`). It's used by the online hatom2atom service by Luke Arno (find it at `http://lukearno.com/projects/hatom2atom`) and by Chris Casciano's extension for the Mac OS news reader NetNewsWire (see `http://placenamehere.com/mf/netnewswire`), which allows users to subscribe to any page marked up using hAtom just like they would subscribe to an RSS or Atom feed.

The venerable Tails extension for Firefox (`http://blog.codeeg.com/tails-firefox-extension-03`) supports the hAtom microformat. Figure 10-1 shows one of the developers of Sandboxes blog, as seen through the eyes of Tails.



Objects Found on this Page:

Minimalist Plugin: Cap Comments
I whipped this up for a particularly busy blog. If anybody wants to use…

Too Much Knowledge!
Does anybody else around here feel like the WordPress Codex is suffering…

Buddy Cards from 30 Boxes adorn comments
The good folks at 30 Boxes have gone and created one of the coolest…

One Mac Woe
I love my MacBook Pro. There's just one problem. The mouse cursor's…

MSDN: New!
Microsoft MSDN Library debuts old CSS values and calls them NEW.

Figure 10-1. hAtom in action, as seen through the eyes of Tails

This demonstrates how with hAtom, every page becomes its own feed, without the need for RSS or Atom.

Summary

It's early days for hAtom, but by leveraging the hard work of those who came before (modular open source software like WordPress and hAtom2Atom, as well as a community-based approach to development), hAtom has come a long way very quickly and shows how agile the evolution of a microformat can be.

I'll leave the last word on hAtom to one of the rising stars of the Web, Matt Mullenweg, developer of WordPress and founder of Automattic:

I'm not excited about hAtom because I think it will replace RSS or Atom. (Not in a million years.) Rather, it offers a normalized semantic way to class elements of a page, which is something we've been trying to get folks to agree on for years. I think consistency in templates here will be a big boon to people just starting to learn HTML and CSS.[1]

With hAtom, we've finished with our in-depth coverage of specific microformats. In this part of the book we took a look at each of the current "specification" level formats and some of the fast-emerging draft specifications.

In the next section of the book, we'll look at two of the current innovators with microformats, Cork'd and Yahoo, to see how and why nimble startups as well as the biggest publishers on the Web use microformats. Then we'll finish off the book by detailing the process by which microformats come into being.

1. See the Comments section of `http://factoryjoe.com/blog/2006/08/06/wordpress-makes-a-move-towards-hatom-gets-upgrades`.

PART THREE **CASE STUDIES**

In this part of the book, we look in detail at two publishers using microformats extensively today, from two ends of the publishing spectrum. In Chapter 11, we study Cork'd, a startup wine review site. Then in Chapter 12, we look at how Yahoo—arguably the biggest publisher on the Web—uses several different now-familiar microformats on a number of its sites.

11 CASE STUDY: CORK'D

In this chapter, we take a look at what a fresh new startup, Cork'd (`http://corkd.com`), is doing. In the process, one of the founders of Cork'd, renowned developer, designer, blogger, and author Dan Cederholm, talks about microformats, why they interest him, what he's been doing with them, and what he thinks he'll be doing with them in the near future.

Introducing Dan Cederholm

The name Dan Cederholm will be familiar to many web designers and developers. His site SimpleBits (`http://simplebits.com`) is widely read in the web design community. His SimpleQuiz publication introduced a great many developers to the ideas associated with semantic HTML markup, his clients include such popular sites as MTV and ESPN.com, and his two books, *Bulletproof Web Design* (New Riders, 2005) and *Web Standards Solutions* (friends of ED, 2004; ISBN: 1-59059-381-2), are pretty much required reading for web designers and developers.

Photo by Jeremy Keith

Dan's also been heavily involved with microformats for some time, designing both the microformats community website and logo, as well as extensively using microformats with his new project, Cork'd. The fact that innovative, widely respected designers and developers like Dan, who were early adopters of web standards, CSS, and the more semantic use of HTML, have embraced microformats is, to me, a clear indication of the future importance of microformats. People like Dan have been leading the way in web development for some time.

When I interviewed Dan recently, I started by asking him how he first came to hear about microformats.

I'm sure I first heard about them from either Matt Mullenweg or Tantek Çelik, but more specifically XFN (the XHTML Friends Network). I'm not sure if the term "microformats" was even being used at this point, but in any case, I can remember lots of buzz around using XFN to mark up your "blogroll," noting relationships to the various people you were linking to. This was right around the time of the annual SxSWi festival, and so updating your XFN lists was a favorite pastime for newly acquainted geeks.

At this point, it was microformats at their birth—a simple tweak of markup using the `rel` ***attribute—but one that sparked the rest.***

You learned earlier that microformats were legendarily born in the halls of the South by Southwest (SxSW) conference (an annual conference in Austin, Texas). If you get the chance you should try attending, but make your plans months in advance, as it's very popular.

Dan focuses on the simplicity of microformats—and simplicity of design is one of Dan's hallmarks, as you'll see. It's noteworthy, too, that he zooms in on one of the features of microformats I've remarked on a number of times in this book: microformats work with your existing content and markup rather than forcing changes on you.

Dan continues:

I was drawn to the "indie" spirit of microformats right away. I liked how microformats aimed to work right away, rather than waiting for the Semantic Web to take hold (will it ever?). One of my favorite bands is Guided by Voices. Many of their early recordings were done crudely on cassette boom boxes and 4-track recorders. They used what was lying around, and in the end it didn't matter—the songs rang true. Microformats, to me, are about "using what's lying around," namely XHTML, XML, etc., and using what already works and is understandable in a really new and powerful way.

Back in 2004 and 2005, microformats were very much an early adopter technology. Just why did Dan invest the energy in something that may have shown a lot of promise, but that didn't really hold any immediate payoffs?

Early on, choosing to use microformats to describe a relationship (XFN) or a contact (hCard) was more about "planting seeds"—there were few if any tools/apps/developers taking advantage of this rich, bubbled-to-the-surface data, but the promise was there. By planting the seeds, I knew that eventually people would come along and start to do cool stuff with the microformats I'd implemented. And we're starting to see more and more of that now.

We should be very thankful that early adopters like Dan made this investment, just as we owe a great deal of thanks to early web pioneers and early advocates of CSS and web standards. Their names are often not well known, but the seeds they sowed have made the Web a more fruitful place for us all.

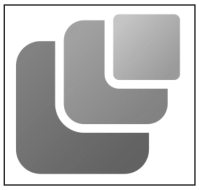


Figure 11-1. The microformats logo, designed by Dan Cederholm

Dan, as I mentioned, designed both the microformats.org site and logo (see Figure 11-1). Of that experience, he says,

That was a fun project—helping to build microformats into a "brand" by designing the logo and website. We went through quite a few concepts for the logo itself, settling on the three-tiered layered box mark that you'll see today. I like talking about the logo, because as simple as it is, it actually conveys a lot about what microformats are. The three layers represent the "building blocks" of semantic markup, and the idea of smaller blocks being built on top of larger, more established ones. The large base block could be XML, while the middle block could represent XHTML jutting out from XML, and then the third, smallest layer could be microformats, which build from the others that precede it. That third small block is also a brighter green, signifying "new growth." As microformats are growing from the current technologies that available, it's the third "branch" that the community has taken on itself to create off the W3C's previous specs. A renegade, if you will.

Introducing Cork'd

So Dan's a fan of microformats, but the proof of the pudding is in the eating—let's see how he's been putting his code where his mouth is. Recently, Cederholm started Cork'd with Hivelogic's Dan Benjamin. Cork'd uses microformats extensively, as you'll see in a moment, but first, what is Cork'd? According to Cederholm,

Cork'd is a community-based, free service for wine aficionados. Cork'd members can keep track of wines they've tasted, wines they own, or wines they'd like to buy. Many wine fans use a "wine journal"—a notebook of sorts to keep tasting notes, remembering what they thought of a particular bottle. Dan Benjamin (cofounder and developer of Cork'd) and I thought it'd be fun to place this information on the Web, and then allow people to share those notes and reviews with their friends. Community plays a large role in Cork'd, in that friends can add each other as "Drinking Buddies," keeping tabs on what wines they've tasted as well as recommendations and messaging. The thought was, let's bring wine to people who would rather trust a friend's opinion than that of a highbrow, snobby wine publication. So, we're trying to take the intimidation out of wine, while providing a fun, free service. It was launched in May 2006.

Just from Dan's description, you can probably start imagining how Cork'd uses microformats (I promised you you'd start seeing them everywhere way back in Chapter 1). "Cork'd member" sounds like the kind of thing we could mark up using hCard, while a "wine journal" would logically contain "reviews" of the wine people have tasted. Because community is important, you can imagine that establishing the relationships between people is an important part of Cork'd as well, and we know XFN suits that task well.

You might think that as an early promoter and adopter, Dan built microformats right into the application from the beginning, but he says,

(Embarrassingly) it [our adoption of microformats] didn't happen right away. The way Cork'd was constructed was very organic, with Dan B. and I working in tandem via SVN. This means we could both chip away at the app independently, without getting in each other's way. This also gives the interface designer a great deal of flexibility—to swap markup/CSS/images in and out to experiment with different combinations. So, once the basic framework of the application was in place, I then went back and modified the markup to support microformats.

This was largely a nonissue, however, as we were already being pretty semantic about everything, and what was needed was the classes that describe those pieces correctly. I think that's a good point about implementing microformats.

I've commented on this quite a number of times: microformats can generally be retrofitted into a site or application without too much effort, particularly where the site uses good development practices, such as the use of valid, semantic HTML or XHTML. That's why we devoted quite a bit of time to the issue of valid and semantic HTML in Chapter 3.

And where else has Dan been using microformats?

I'd also been using microformats on my business/blog, SimpleBits. Again, it's a case where I have total control over the markup, so it's a nice playground. Just recently, I coded some templates for a well-known design organization and used hCalendar to mark up the listing of events. It'll be interesting to see if the CMS developers take care in preserving that structure, what obstacles they encounter, or if they ignore them completely. Unfortunately, it takes at least a basic understanding of what microformats are in order to make sure they'll work. That could change, though. People are creating new tools every day, and as we get CMS and software developers on board, much of the implementation I hope will be trans-

parent, second-nature, and omnipresent. Thankfully, since microformats are equally human- and machine-readable, it doesn't take long to grasp the concepts.

So, let's get into the details of how Dan uses microformats in Cork'd. He had this to say:

On Cork'd we use three microformats: hCard, hReview, and rel-tag. We're also "sort of" using XFN, currently marking all of your contacts as "friend" in order for people to take advantage of that info (more on that in a minute). Ideally, we'll have the ability to mark relationships in the future.

Using hReview was a no-brainer, as member-created wine reviews are what drive the site. And since Cork'd is a community-based site, it also makes good sense to use hCard to mark up your Drinking Buddies (aka contacts). The contact information of each member is somewhat limited on the site, but we've seen benefits from using hCard here, even if to describe sparse data.

In terms of motivation, I was excited to use microformats beyond tinkering with a personal blog, and this was a prime opportunity. I was particularly excited to play with hReview, as in a perfect world, every wine blogger could write their tasting notes on their own respective sites, and then a site like Cork'd could merely be an aggregator of that information if hReview is involved. That's the future I suppose.

So, has it been worth their while? What benefits have they seen?

I think it's the unexpected benefits that are the most exciting. For example, a developer contacting me saying that he'd scraped the XFN'd contact list, then the hReviews, ran everything though a SPARQL engine that could generate a list of reviews **only** ***from his trusted friends. Now this would be a great feature to have as part of Cork'd—but the beautiful thing is that this guy was able to build it himself using microformatted data that was revealed on Cork'd. I have to admit I had no idea what he was talking about—and that's fantastic! I'm calling this*** **oblivious development**—***that I as an interface designer can choose to use microformats, and then people who actually know what they're doing can come along and do really cool stuff with the data.***

Drew McLellan gave a great presentation recently titled "Can Your Website be Your API?" and this is a great example of that. Cork'd has no official public API, yet because of simple microformats, we do. There's another site called Scrugy that's scraping our hReviews and adding them to their wine search engine. No work on our part, and more Cork'd content distributed elsewhere. This, to me, is a beautiful thing. And those seeds that we've been planting are now legitimately being harvested. Microformats are no longer just an ideal, but rather real benefits are popping up every day.

I've reiterated throughout the book that enabling "decentralized development, content, services" is a key principle of microformats, and Dan's experience at Cork'd definitely emphasizes this.

I asked Dan whether they'd run into any issues or "gotchas" while working with microformats, and here was his response:

I did run into an issue with the (then) recommended way of including a wine's title for each hReview on the page. There was a method that utilized the `<object>` element that caused issues in Safari. I ended up abandoning that in favor of repeating the wine name for each hReview, and then hiding that using CSS. Later, a new and improved method for include patterns (`http://microformats.org/wiki/include-pattern`) emerged. I need to give this a second look.

In Chapter 9, we looked at the now recommended way of including details from one microformat in another, the include pattern, which was redesigned precisely because of this issue with the `<object>` element in Safari.

Other gotchas may have gone unnoticed. In other words, without a "microformats validator" it can be difficult to be confident in the way you've implemented them. This can be a stumbling block for those just getting started. I recommend using the code creators to visualize how the structure is easily set up (more on that in a bit).

Dan is very much known as a designer as well as a developer. I wondered whether using microformats had any impact on his designs. Did they help in implementing designs? Did they hinder his work as a designer in any way? One common and, I think, completely unfounded criticism of web standards, which I've also heard in relation to microformats, is that they "inhibit creativity." Did Dan find that?

Since I designed first, and then retrofitted the microformats after the fact, they didn't impact the design all that much. I think that speaks volumes about the simplicity of what microformats are—there's nothing out of the ordinary in terms of required markup. One thing I need to keep in mind when dealing with microformats is that I can attach the recommended classes to elements other than `<div>` and `<span>`. If a list or heading makes more semantic sense, then so be it. I hope that designers realize that flexibility is there. Another thing to keep in mind is that CSS will be able to make your microformats **appear** ***however you'd like. Utilize the display property in CSS to change inline elements to "block," to force things on another line. Or conversely, give block elements the "inline" value to string things together. Just because the markup behaves a certain way by default doesn't mean we can position the elements however we'd like later on. This is a core principle of CSS-based design—that separation of content and design—but it rings true with microformats as well, and I hope designers understand that while certain markup patterns are recommended via their specifications, using CSS to alter the presentation is always available.***

I'll also mention that the code creators on microformats.org are incredibly helpful for designers implementing microformats. They'll take the guesswork out of using the correct markup and create the structure for you automatically. It's the first place I'd send a curious designer new to microformats. Oftentimes seeing is believing, and seeing the way in which the creator plugs in data is extremely useful in grasping how a microformat works. I used both the hCard and hReview creators in setting up the microformats on Cork'd.

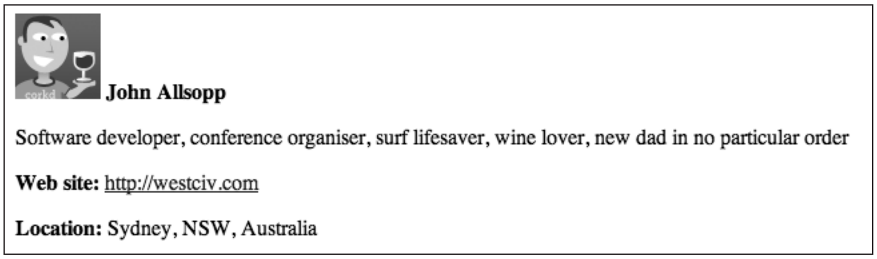
Next, let's take a look at how Cork'd uses microformats, both the HTML markup and the CSS used to style the resulting content. We'll begin with our old favorite, hCard, and then move on to hReview.

hCard on Cork'd

Here's my membership hCard at Cork'd. (I've used mine as an example because you never know who might have an account and is reviewing very nice wine, but their husband or wife [or creditors] might not know about this purchase.)

```
<div class="vcard">
  <img src="/img/icon-user-64.gif" height="64" width="64" alt="icon" ➥
    class="photo" />
  <span class="fn"><strong>John Allsopp</strong></span>

  <p>Software developer, conference organiser, surf lifesaver, wine ➥
    lover, new dad in no particular order</p>
  <p><strong>Web site:</strong> <a href="http://westciv.com" ➥
    class="url">http://westciv.com</a></p>
  <p><strong>Location:</strong> <span class="region">Sydney, ➥
    NSW, Australia</span></p>
</div>
```

In many ways, it's a very straightforward hCard. It has an FN (which as you know is a required property for an hCard), a URL, a photo, and a location, which Cork'd simply adds as a region, rather than a fully elaborated adr element. If you look at this hCard in a browser, with no styling (see Figure 11-2), you can see one of the key principles and benefits of microformats: they are human-friendly.


John Allsopp

Software developer, conference organiser, surf lifesaver, wine lover, new dad in no particular order

Web site: http://westciv.com

Location: Sydney, NSW, Australia

Figure 11-2. The hCard is perfectly human-readable, without any styling whatsoever.

So what techniques does Dan use to style this nice, simple piece of markup? In keeping with Dan's philosophy of simplicity, reflected in the HTML, the CSS is also unfussy. Much of the style relies on inheritance—of font family and size—and on background color "shining through" from other elements.

For example, here is how the member's name is styled:

```
#content div.vcard span.fn strong {
  font-size: 140%;
  font-weight: bold;
  color: #630;
}
```

By itself, this looks like Figure 11-3.

John Allsopp

Figure 11-3. In an isolated scenario, the FN looks like this.

With the full page, the member's name looks as shown in Figure 11-4.

John Allsopp

Figure 11-4. In the full page, the FN has inherited styles from the rest of the page.

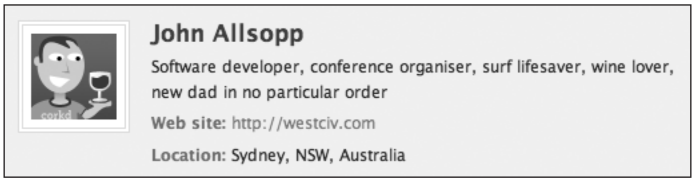
Rather than redefining the font family for every element, Dan relies on the inheritance from the body. While it seems to be a simple point, developers often needlessly add styling to elements instead of relying on inheritance. The simpler a style sheet is, the easier it is to debug, maintain, and modify.

If you look at the CSS in total, you'll see that Dan uses descendent selectors extensively. Microformats are ideal in many ways for descendent selectors, as compound microformats create a context via their root elements that is tailor-made for this approach.

The last little trick we might pick up from Dan is the simple way he uses `float` to position the user's icon to the left of the user's details. In fact, we used this technique in Chapter 7 to add an hCard icon to the right of our hCard, but such simple powerful techniques are always worth reinforcing. Here's the CSS:

```
#content div.vcard img.photo {
  float: left;
  margin: 0 10px 0 0;
  text-align: center;
  padding: 6px;
  border: 4px double #E7DAC0;
  background: #fff;
}
```

The margin of `0 10px 0 0` means that only the right edge will have a 10-pixel margin, creating the separation from the text of the hCard. The fully styled hCard looks like Figure 11-5.


John Allsopp

Software developer, conference organiser, surf lifesaver, wine lover, new dad in no particular order

Web site: http://westciv.com

Location: Sydney, NSW, Australia

Figure 11-5. The fully styled hCard

Note, too, how Dan has used a plain image but a white background color, padding, and border to create the stylized effect of the image. Why do this and not add the effects straight onto the image? Although it's not available right now, Cork'd plans uploadable images, so members can upload their own photo or avatar. Styled this way, all these images will have a similar appearance, and yet there will be no need to process the images at all on the server.

Dan has again used descendent selectors and the inherent semantics in hCard to select this image element (that way, if a wine was marked up using a future hItem microformat, for example, and it included the photo class, too, this selector would not select those elements, only photos inside hCards).

hReview on Cork'd

Although hReview is still a draft format, we've seen that it is widely used. Cork'd makes use of hReview, which makes sense because at the heart of Cork'd are wine reviews. Let's take a look at how hReviews get used at Cork'd.

I am quite a fan of Pinot Noir; its lightness and subtlety make it a perfect red wine for the Australian climate. Ironically, it grows best in cool climates, and while we do make some good Pinot here, the best Pinots definitely come from elsewhere.

Here's the review for an inexpensive, highly rated Pinot I found at Cork'd:

```
<div id="review_7713" class="hreview">
  <h5 class="item"><span class="fn">2000 Flowers Camp Meeting Ridge ➥
    Pinot Noir</span></h5>
  <abbr class="dtreviewed" title="20061009">(49 days ago)</abbr>

  <span class="reviewer vcard">
    <img class="photo" src="/img/icon-user-64.gif" height="48" ➥
      width="48" alt="buddy icon" />
    <a class="url fn" href="/people/mikewillison">mikewillison</a>
  </span>
  <abbr class="rating" title="5"><img src="/img/icon-rate5.gif" ➥
    alt="*****" /></abbr>
  <blockquote class="description">Well, quite frankly it has ➥
    exceeded every expectation... It was only a glass of wine, ➥
    wasn't it?</blockquote>
  <p class="tags">Tasting Tags:
    <a href="/tags/elegant" rel="tag" class="rel-tag" title="view ➥
      all wines with this tag">elegant</a>
    <a href="/tags/moue" rel="tag" class="rel-tag" title="view ➥
      all wines with this tag">moue</a>
    <a href="/tags/sexy" rel="tag" class="rel-tag" title="view ➥
      all wines with this tag">sexy</a>
  </p>
</div>
```

As I said, it's an inexpensive wine, so I hope Mike doesn't mind me spilling the beans.

Let's take a look at some of the specific aspects that Cork'd is using. We can see how the existing logic of the application works well with the hReview format. Here the container element for the review is given an internal unique identifier using id, but also serves as the root element for the review.

```
<div id="review_7713" class="hreview">
```

I like the way the review date is really human-friendly (49 days ago), giving us an immediate sense of how "fresh" the review is, rather than using a specific date, like October 14, 2006. The abbr design pattern is used to give the machine-readable value for this date.

The reviewer is marked up as a simple hCard, with a link to his profile page. Interestingly, Dan had marked up the reviewer's comments (the description field) as a blockquote, which makes sense, as the review in effect quotes the reviewer. The other thing to note is how wine is "categorized" using tags that capture the reviewer's impression of the wine's taste, using rel-tag. This way, we can actually find all wines given a particular tasting tag by any reviewers, so if we like "berry" flavors, we can find all the wines other members think have this flavor. I am not sure that "sexy" is a strictly technical tasting term, but I think I know what Mike means.

One thing to note is the use of the abbr design pattern for ratings. In the hReview chapter, you saw that ratings typically look like this:

```
<p>Rating: <span class="rating">2</span>/5</p>
```

But a very common way of presenting ratings is with stars, chefs' hats, and so on. Figure 11-6 shows an example of how Cork'd does it.

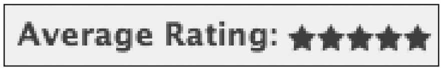


Figure 11-6. Cork'd uses a star rating system for the wines reviewed on the site.

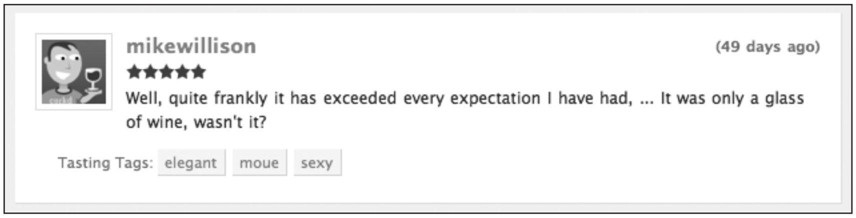
How can we mark up such a common form of rating with hReview? Enter the abbr design pattern once more.

```
<abbr class="rating" title="5"><img src="/img/icon-rate5.gif" ➥
  alt="*****" /></abbr>
```

Here, the title value is the rating, while the image inside is the human-readable version (of course, with alt text to help those using assistive technologies).

The hReview CSS

Figure 11-7 shows how hReview is styled on Cork'd using CSS.


mikewillison

(49 days ago)

★★★★★

Well, quite frankly it has exceeded every expectation I have had, ... It was only a glass of wine, wasn't it?

Tasting Tags: elegant moue sexy

Figure 11-7. Styling an hReview, Cork'd style

Some of the familiar tricks you've seen are in evidence again. This time the reviewer icon is floated to the left, and the review date, (49 days ago), is floated to the right.

The tasting tags are styled to look like buttons, with raised edges, using the CSS `border` property:

```
a.rel-tag {
  margin: 0 2px 0 0;
  padding: 3px 5px;
  background: #FFF6D3 url(../img/search-bg.gif) repeat-x top left;
  border-right: 1px solid #DFD5AF;
  border-bottom: 1px solid #DFD5AF;
}
```

You'll note that because the attr selector isn't widely supported, Dan's added a class of `rel-tag` to the tasting tags. When the attr selector is better supported, this class will no longer need to be added to these links.

The overall review effect of a cutout box is achieved very simply as well:

```
div.hreview {
  margin: 15px 0 8px 0;
  padding: 15px;
  border-bottom: 1px solid #E5DFC7;
  border-right: 1px solid #E5DFC7;
  background: #fff;
}
```

with margin, padding, borders, and a simple colored background. No explicit width is set, because the hReview element is 100% the width of its parent element, which is where the width is actually set.

While these are simple, well-understood, long-used, and widely supported features of CSS, they achieve a contemporary look, and again use the structure and markup of our microformat—working with the existing code, rather than against it, or ignoring it, as many of us all too frequently do.

I think any aspiring designer or developer can glean a lot from Dan's code. There is a simplicity, a straightforwardness, and an intelligence to his coding that never becomes overly clever or fussy. Yes, Dan, like most of us, uses the odd hack to work around Internet Explorer's box model problems, but on the whole, his coding style and very nicely commented CSS mean that the code is sensible and readable.

Summary

So what's the future for Cork'd and microformats? What other uses of microformats can Dan see adding value to Cork'd? Here are his answers to these questions:

We could definitely add more XFN functionality, giving the members the ability to specify relationships (rather than mark them all as friends automatically). That'd be one area to improve. Ditto for making hCard consistent for all instances of contacts on the site. There is actually a lot more we can do. Currently it's up to us to manually add it in. My hope is that over time, more software developers begin to build microformat functionality into their CMSs, aggregators, browser extensions, etc. And we're starting to see that already.

Whether you are beginning a new project or reworking an existing one, taking a page from the book of someone as successful and innovative as Dan Cederholm and looking at how you can use microformats in these projects surely can't be a bad idea.

In the next chapter, we'll go to the other end of the spectrum and look at how one of the Web's biggest publishers is using microformats across a wide variety of sites.

12 CASE STUDY: YAHOO

In the last chapter, we examined Cork'd, a small, innovative, recent startup. In this chapter, we move on to take a look at a company that represents the other extreme on the Web: Yahoo, arguably the Web's most visited site. Unlike many other highly visited sites, such as Google, Yahoo's traffic comes largely from the content it publishes across a broad spectrum of sites. Yahoo is almost certainly the largest publisher on the Web in this respect.

As you've seen throughout the book, Yahoo uses a range of microformats extensively. In this chapter, we'll examine some of these uses in more detail and talk to a Yahoo developer about the company's interests in microformats, the benefits seen from adopting them, and the lessons learned along the way.

Introducing Nate Koechley

Nate Koechley, Senior Engineer and Technical Evangelist on the Yahoo User Interface (YUI) Library team, generously agreed to talk with me about how Yahoo is using microformats. This is just one of many examples of an openness at Yahoo that I really admire—the company opens up not only APIs and data for any web developer to use, but also a lot of underlying code.

Take a look at the YUI Library for articles, documentation, and open source user interface components you can use in your development: `http://developer.yahoo.com/yui`.

I asked Nate why such an established, huge publisher like Yahoo might be interested in a new technology like microformats.

Why adopt microformats? First and foremost, at Yahoo! we're just plain passionate about the Web. We want to see it thrive. Microformats seem good for the vitality of the Web, and ultimately good for users. It's true there are sometimes constraints on bigger websites and companies, but at the same time our size can be an advantage. In cases like microformats, our scale can help technology reach its tipping point. We're proud to play that role, and look forward to microformats making the Web even more functional for users.

Did Yahoo consider that it was taking risks by doing so?

In terms of risks, people sometimes wonder if microformats void copyright. After speaking with Tantek Çelik, I believe the answer is no, and that the format of data doesn't impact the ownership of data. That said, we're still examining these questions on a case-by-case basis, and we're being cautions in a few cases. (Disclaimer: I am not a lawyer. This statement has not been evaluated by any lawyer. This is NOT legal advice or legal counsel.)

On balance, however, I wish more equations took the "easy to implement + low risk + upside for users + good for the Web's health" format.

How does Yahoo use microformats?

We use microformats all over the place, and more every day. Flickr and Upcoming have been fortified with microformats for longer than I can remember. Yahoo! Tech launched in May with hReview, and in June Yahoo! Local added hCards, hCalendars, and hReviews. You can find microformats on Yahoo! Food, Finance, Movies, Games, and probably many more. I believe our sites in Europe are rolling them out even faster than here in the States.

In addition to the formats I listed, we're also using the hAtom, vCard, and adr formats, and the home link and tag patterns. We have some pretty cool new implementations baking in the oven, but unfortunately I can't talk about them quite yet.

What benefits have you seen from using these microformats? What unexpected benefits?

I think the benefits of microformats haven't been fully realized yet, and won't really until the network effect hits. I believe that day draws near. That said, a developer needs to mark up content in some fashion no matter what, so it's often pretty convenient that we can turn to microformats for guidance.

I wondered, too, whether Yahoo had encountered any difficulties using microformats. The company has adopted several microformats across a broad range of sites, so if anyone was going to run into issues caused by using microformats, it was going to be Yahoo. When I asked about that, Nate had the following to say:

There haven't been too many gotchas. In one case, one of our site's modular template systems made it difficult to associate the product name with the review, but these are solvable problems. Also, some of the formats aren't fully cooked yet, so in some cases there are gaps that I expect will get closed over time. In general, it's been pretty painless.

In the sections that follow, we'll take a look at some of Yahoo's microformat implementations in action, and learn from the experience gained and efforts made by Yahoo's developers. But first, I'd like to emphasize that Yahoo is an early adopter (and in part developer) of microformats. There is a real danger in early adoption of changing specifications. (I know this firsthand—years ago I implemented a draft of CSS2 in Style Master, a CSS editor I develop, and a good deal changed between that draft and the final specification, necessitating a lot of additional work.) In this chapter's case study, you'll see some uses of microformats at Yahoo that aren't 100% "correct," but much if not all of that is a consequence of being an early adopter of changing specifications, or being a true pioneer going into uncharted territory. More than anything, we should be thankful that Yahoo has taken the risks, and appreciative that the company is willing to share its experiences—I know I certainly am.

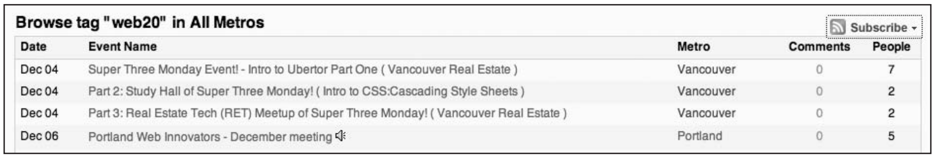
Upcoming

Acquired by Yahoo in late 2005, Upcoming (`http://upcoming.org`) hosts details about tens of thousands of events, and lets users find and track events, and build communities around events (by letting others know they'll attend, or by adding comments about an event). It features all the expected Web 2.0 goodness—an API for creating mashups, event tagging (by organizers and by others using the rel-tag microformat, no less) and, of course, RSS for subscribing to an event or a collection of events (e.g., all events tagged "web2.0", all events in Vancouver or some other area, etc.). It should come as no surprise that Upcoming events are marked up using the hCalendar microformat. In fact, Upcoming doesn't use hCalendar just on an event's page, as you would expect, but also when presenting lists of events, like all those with a given tag or in a given place.

In the sections that follow, we'll look at an example of Upcoming's use of hCalendar and explore definition lists in more detail.

hCalendar on Upcoming

In Figure 12-1, I've subscribed to all the events tagged "web2.0" (everything about the Web is now tagged "2.0" it seems—even events for realtors).



Browse tag "web20" in All Metros

Date	Event Name	Metro	Comments	People
Dec 04	Super Three Monday Event! - Intro to Ubertor Part One (Vancouver Real Estate)	Vancouver	0	7
Dec 04	Part 2: Study Hall of Super Three Monday! (Intro to CSS:Cascading Style Sheets)	Vancouver	0	2
Dec 04	Part 3: Real Estate Tech (RET) Meetup of Super Three Monday! (Vancouver Real Estate)	Vancouver	0	2
Dec 06	Portland Web Innovators - December meeting	Portland	0	5

Figure 12-1. Search results for Web 2.0 events in Upcoming

Reproduced with permission of Yahoo! Inc. © 2007 by Yahoo! Inc. YAHOO! and the YAHOO! logo are trademarks of Yahoo! Inc.

Let's take a look at the code underneath:

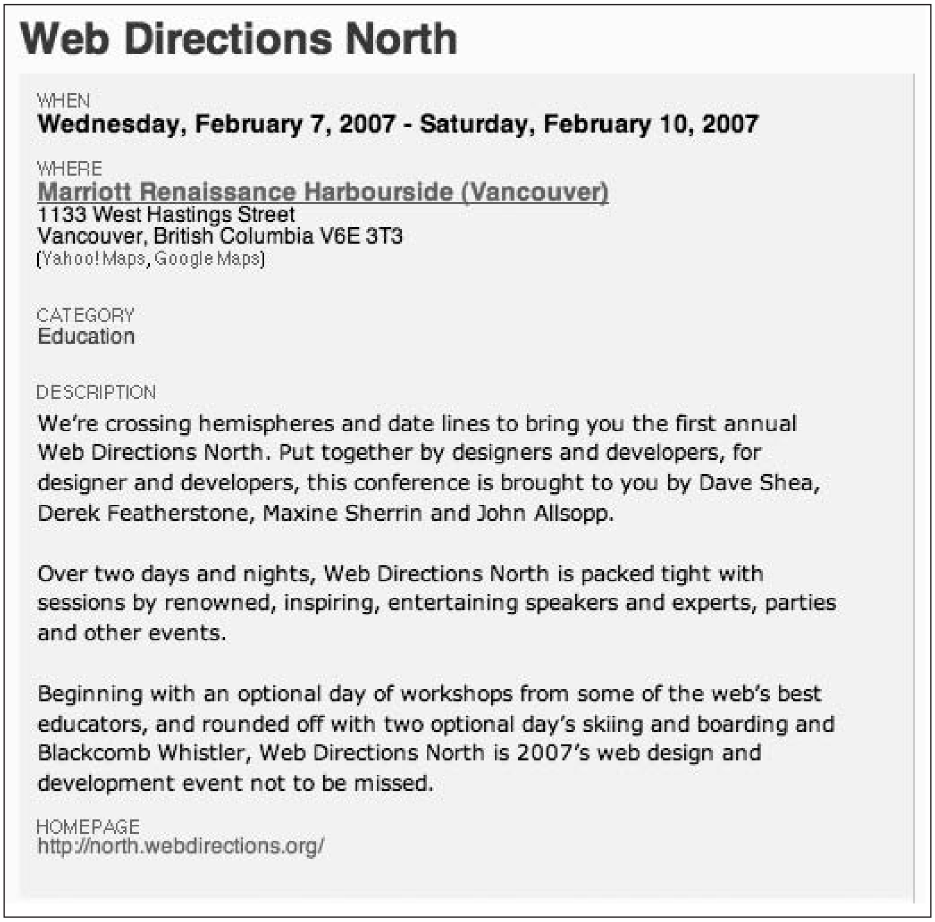
```
<tr class="vevent">
  <td class="even" nowrap="nowrap"><abbr class="dtstart" title=""> ➥
    Dec 04</abbr></td>
  <td class="even"><a href="http://upcoming.org/event/118987" ➥
    class="url summary" >Super Three Monday Event! - Intro to Ubertor ➥
    Part One ( Vancouver Real Estate )</a></td>
  <td class="even"><a href="/metro/ca/bc/van/"><abbr class="location" ➥
    title="Ubertor @ Milan&#039;s Bistro, 1223 Hamilton Street, ➥
    Vancouver, V6B6A8">Vancouver</abbr></a></td>
  <td class="even" align="center"><span class="inactive">0</span></td>
  <td class="even" align="center">7</td>
</tr>
```

As you can see, it's not quite perfect yet—for example, the start date has a human-friendly component, but not a machine-readable part, which makes it far less valuable than if that machine-readable date was there. Upcoming has also interestingly used an abbr design pattern to give the location details:

```
<abbr class="location" title="Ubertor @ Milan&#039;s Bistro, 1223 ➥
  Hamilton Street, Vancouver, V6B6A8">Vancouver</abbr>
```

Keep in mind that the page is a summary, and as such space is at a premium. As a result, an additional usability benefit in most browsers is that by hovering over the element, the reader gets the full details of the location. It's a feature users will need to learn, but if they use the site frequently, it might start coming in very handy, because if they just want a brief overview of the event they no longer need to follow the link to the full details page, saving a click.

If users do want more details, they can follow the link and get a more detailed hCalendar, as shown in Figure 12-2.



Web Directions North

WHEN
Wednesday, February 7, 2007 - Saturday, February 10, 2007

WHERE
Marriott Renaissance Harbourside (Vancouver)
1133 West Hastings Street
Vancouver, British Columbia V6E 3T3
(Yahoo! Maps, Google Maps)

CATEGORY
Education

DESCRIPTION
We're crossing hemispheres and date lines to bring you the first annual Web Directions North. Put together by designers and developers, for designer and developers, this conference is brought to you by Dave Shea, Derek Featherstone, Maxine Sherrin and John Allsopp.

Over two days and nights, Web Directions North is packed tight with sessions by renowned, inspiring, entertaining speakers and experts, parties and other events.

Beginning with an optional day of workshops from some of the web's best educators, and rounded off with two optional day's skiing and boarding and Blackcomb Whistler, Web Directions North is 2007's web design and development event not to be missed.

HOMEPAGE
http://north.webdirections.org/

Figure 12-2. Further details on each event are obtained by following the relevant links.

The hCalendar has this HTML (I've elided it a little to make it more readable):

```
<div id="event" class="vevent">
  <h1 class="name summary">Web Directions North</h1>
  <div id="eventMain">
    <div id="eventMetadata">
      <div class="small">When</div>
      <div class="date">
        <abbr class="dtstart" title="2007-02-07">Wednesday, ➥
          February 7, 2007</abbr> - <abbr class="dtend" title="2007-02
          10">Saturday, February 10, 2007</abbr>
      </div> <!-- /.date -->
      [...]
      <div class="venue location vcard">
        <br/><div class="small">Where</div>
        <span class="fn org"><a href="/venue/35093">Marriott ➥
          Renaissance Harbourside (Vancouver)</a></span><br />
        <div class="address adr">
          <span class="street-address">1133 West Hastings ➥
            Street</span><br />
          <span class="locality">Vancouver</span>, <span ➥
            class="region">British Columbia</span> <span ➥
            class="postal-code">V6E 3T3</span>
        </div>
        <div class="directions">
          (<a href="http://maps.yahoo.com/...6E+3T3">Yahoo! Maps</a>,
          <a href="http://maps.google.com/maps?q=1133+West+...3"> ➥
            Google Maps</a>)<br />
        </div>
      </div> <!-- /.venue -->

      <div class="category">
        <div class="small">Category</div>
        Education
      </div>

      <div class="description">
        <div class="small">Description</div>
        We're crossing hemispheres ...
      </div>

      <div class="url">
        <div class="small">Homepage</div> <a ➥
          href="http://north.webdirections.org/"> ➥
          http://north.webdirections.org/</a>
      </div>

      [...]

</div> <!-- /#eventMain -->
```

One "gotcha" you've already seen with dates in hCalendar is evidenced here: the dtend property has a value of title="2007-02-10", which as you might recall actually means in effect midnight between February 9 and 10. To get around this nonintuitive aspect of hCalendar, a finish time with the date such as 2007-02-10T1700-0800 or the date value of 2007-02-11 would correctly specify the machine-readable date.

We can see how the microformat markup has most likely been added to the existing markup, for example:

```
<div class="venue location vcard">
```

and

```
<div class="address adr">
```

Commenting is something that many web developers don't pay enough attention to. Particularly with many nested elements like the divs here, knowing where one ends can be difficult to work out, so adding comments like this makes the markup much more readable and maintainable:

```
</div> <!-- /#eventMain -->
```

There are some little things that might need some fixing up, for example:

```
<div class="url">
  <div class="small">Homepage</div> <a ➥
    href="http://north.webdirections.org/"> ➥
    http://north.webdirections.org/</a>
</div>
```

Because the div with the class of url contains the element <div class="small">Homepage </div> as well as the link, the actual value of the URL includes the text Homepage. Perhaps better markup might be as follows:

```
<dl>
  <dt>Homepage </dt>
  <dd><a href="http://north.webdirections.org/" class="url"> ➥
    http://north.webdirections.org/</a></dd>
</dl>
```

In fact, while not strictly associated with microformats, this pattern of a definition list could replace a number of multiple div constructs. For example, the following:

```
<div class="category">
  <div class="small">Category</div>
  Education
</div>
```

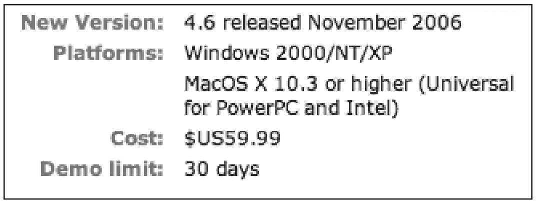
could be more cleanly marked up as follows:

```
<dl>
  <dt class="small">Category</dt>
  <dd class="category">Education</dd>
</dl>
```

In the original code, the value of the `category` property is in fact `Category Education`. But what we really want is for the value of the `category` property to simply be `Education`, which it is in the reformulation.

Definition lists

If you haven't used **definition lists** before (most developers use them rarely if at all), they are actually more broadly useful than is assumed or their name would imply. According to the HTML specification, "Definition lists vary only slightly from other types of lists in that list items consist of two parts: a term and a description." So definition lists can be used in not uncommon situations like the example shown in Figure 12-3 (from my website), where I have labeled text entries.



New Version:	4.6 released November 2006
Platforms:	Windows 2000/NT/XP
	MacOS X 10.3 or higher (Universal for PowerPC and Intel)
Cost:	$US59.99
Demo limit:	30 days

Figure 12-3. A good use for the noble definition list

This list is marked up as follows in HTML:

```
<dl class="prod-summary">
  <dt>New Version:</dt>
  <dd>4.6 released November 2006</dd>
  <dt>Platforms:</dt>
  <dd>Windows 2000/NT/XP</dd>
  <dd>MacOS X 10.3 or higher (Universal for PowerPC and Intel)</dd>
  <dt>Cost:</dt>
  <dd>$US59.99</dd>
  <dt>Demo limit:</dt>
  <dd>30 days</dd>
</dl>
```

In the preceding code, the `<dt>` elements are floated to the left to put the `<dt>` and its associated `<dd>` element on the same line.

As for CSS, Upcoming doesn't really use the hCalendar markup for styling with CSS. That would make sense where the HTML has had the microformat markup retrofitted to it.

Nate more or less confirmed my speculation in an email when he commented, "Your hunches about legacy impact were basically spot on."

Reviews at Yahoo Local

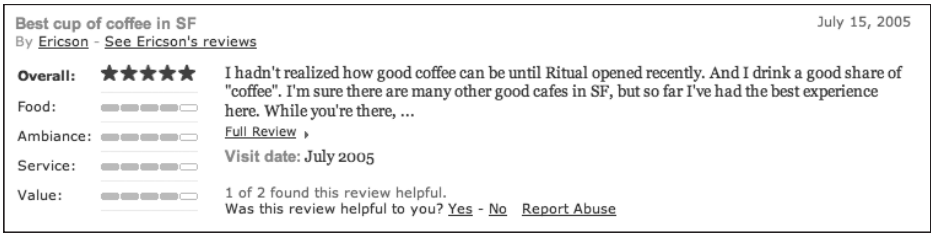
Review sites seem to be the flavor of the month, and as you learned earlier, Yahoo uses hReview at a couple of its very popular sites.

Yahoo Local embodies the "user-generated content" nature of many recent sites. The idea is to let users add their opinions and expertise to a site, in order to tap into the "wisdom of the crowd." At present, much of this content is still centralized at sites like Cork'd and Yahoo Local, but as you've seen on several occasions, one of the goals of microformats is to enable decentralized services. By adopting hReview, Yahoo makes it possible for others to aggregate the reviews they publish much more easily than would otherwise be possible.

In the following sections, we'll explore Yahoo Local's use of hReview and some lessons we can learn from that.

Review example

So what does a review look like at Yahoo Local? Figure 12-4 shows the review page for a well-liked coffee shop in San Francisco.


Best cup of coffee in SF
July 15, 2005
By Ericson - See Ericson's reviews

Overall: ★★★★★
Food:
Ambiance:
Service:
Value:

I hadn't realized how good coffee can be until Ritual opened recently. And I drink a good share of "coffee". I'm sure there are many other good cafes in SF, but so far I've had the best experience here. While you're there, ...
Full Review
Visit date: July 2005
1 of 2 found this review helpful.
Was this review helpful to you? Yes - No Report Abuse

Figure 12-4. A review from Yahoo Local
Reproduced with permission of Yahoo! Inc. © 2007 by Yahoo! Inc. YAHOO! and the YAHOO! logo are trademarks of Yahoo! Inc.

Stop for a moment and think of how you would mark this up. What hReview properties are there? Now, how does Yahoo Local do it? Let's have a look (I've elided the code that is not directly relevant to us):

```
<tbody class="hreview">

  <tr valign="top"><th colspan="2"><h3><u style="float:right;"><abbr ➥
    style="border-bottom-style: none;" class="dtreviewed" ➥
    title="2005-07-15">July 15, 2005</abbr></u><span class="summary"> ➥
    Best cup of coffee in SF</span></h3><h6 class="reviewer vcard">By ➥
    <a ...>Ericson</a> - <a href="..." class="url">See Ericson's ➥
    reviews </a></h6></th></tr>
  <tr valign="top"><td width="170" nowrap><table class="ylsrevrating">

  <tr valign="middle"><th><b>Overall: </b></th><td><abbr ➥
    style="border-bottom-style: none;" class="rating" title="5"><img ➥
    src="http://us.i1.yimg.com/us.yimg.com/i/us/ls/gr/ ➥
    read_star_5.gif"alt="5" width="78" height="13"></abbr></td></tr>

  <tr valign="middle"><th>Food:</th><td><abbr ➥
    style="border-bottom-style: none;" class="rating" title="5"><img ➥
    src="http://us.i1.yimg.com/us.yimg.com/i/us/sh/karma/ ➥
    ur_bar_4.gif"alt="4" width="79" height="6"></abbr></td></tr>

  <tr valign="middle"><th>Ambiance:</th><td><abbr ➥
    style="border-bottom-style: none;" class="rating" title="5"><img ➥
    src="http://us.i1.yimg.com/us.yimg.com/i/us/sh/karma/ ➥
    ur_bar_4.gif"alt="4" width="79" height="6"></abbr></td></tr>

  <tr valign="middle"><th>Service:</th><td><abbr ➥
    style="border-bottom-style: none;" class="rating" title="5"><img ➥
    src="http://us.i1.yimg.com/us.yimg.com/i/us/sh/karma/ ➥
    ur_bar_4.gif"alt="4" width="79" height="6"></abbr></td></tr>

  <tr valign="middle"><th>Value:</th><td><abbr ➥
    style="border-bottom-style: none;" class="rating" title="5"><img ➥
    src="http://us.i1.yimg.com/us.yimg.com/i/us/sh/karma/ ➥
    ur_bar_4.gif" alt="4" width="79" height="6"></abbr></td></tr> ➥
    </table></td><td? ➥
    width="100%"><p class="ylsratrevfull" id="full_4"><span ➥
    class="description">I hadn&#39;t realized how good coffee can be ➥
    until Ritual opened recently. ...<a ...>Full Review</a><script> ➥
    revMin("4");</script></p>...</td></tr>

</tbody>
```

The first thing to note is that the review is a table. While the knee-jerk response might be to say, “Hey, `that’s` not tabular data!”, I think it is fair to argue that it is. Whether I’d recommend it be marked up that way is a separate matter—some data is clearly of a tabular nature, and tables are ideal for marking them up, but other data is less clearly so. But if we delve more deeply, we’ll find that there are in fact two tables at play.

Inside each review is a table of the ratings. To me, this is an excellent use of tabular markup. We have five rows of related data—`Overall`, Food, `Ambience`, `Service`, and `Value`—and for each of these, a rating.

What about the other table? Well, from the example I’ve pulled out here, you might think that the review itself is in a table. But in fact it’s in a table body element (`<tbody>`).

If you’ve used `<tbody>` elements, you might assume that every table has at most one `<tbody>`, but a table may have any number of `<tbody>` elements. If we zoom out just a fraction, we find the following structure for the collection of reviews:

```
<table ... class="ylsreviewrow">
  <tbody class="hreview">
    ...
  </tbody>

  <tbody class="hreview">
    ...
  </tbody>

  <tbody class="hreview">
    ...
  </tbody>

</table>
```

Each review is not in a table to itself; rather, it's in a `<tbody>` for the set of reviews, which is in a table. We can think of the set of reviews as a one-dimensional table of reviews. There are certainly other ways this set of reviews could be marked up, but of course that's one of the beauties of HTML, and indeed microformats.

It could be a list—after all, a one-dimensional table is in many ways a list. Or, each review could be contained within a div. I suspect—though it's purely speculation on my part—that the choice of markup is a complex one involving legacy solutions. For example, the reviews page allows sorting of the reviews by newest, oldest, most helpful, and so on, and there may be constraints imposed by the server-side application providing this functionality.

None of my commentary here should be construed as criticism, by the way. Rather, I want to demonstrate that while sometimes we have the luxury of the "ideal" or (arguably) "perfect" solution, in reality, we are very often constrained by factors such as the following, which make solving a problem much harder:

- Time
- Resources
- Legacy systems

I think Yahoo Local has done an admirable job of incorporating microformats into a complex existing service and using HTML appropriately despite considerable constraints.

Sometimes as developers we are fortunate that we can build a brand-new system from the ground up. But frequently we'll be maintaining and upgrading existing systems. That's when our job often becomes much more difficult.

Something specific worth noting about Yahoo Local's use of hReview is that multiple ratings are used—not just an overall rating, but individual ratings for specific aspects of the experience, like ambience and service. And Yahoo has marked these up using the abbr design pattern, just as Cork'd marks up ratings. In both cases, rather than a textual, numerical value, the reviews use a graphical indicator—stars in the case of Cork'd and bars in the case of Yahoo Local. The use of a table in this case actually improves accessibility, as screen readers can read out the `<th>` and `<td>` elements.

Styling Yahoo Local reviews

One indicator that the hReview microformatting on Yahoo Local has been retrofitted is that little if any of the page's styling is done using the hReview microformat. Again, this demonstrates the flexibility of microformats—they can often be added to even very complex, longstanding applications, without any significant impact on the existing code. Here's something you might have noticed in the code from the previous section:

```
<abbr style="border-bottom-style: none;" class="rating" title="5">
```

Why this inline style? Some browsers add dotted underlining via the CSS `border` property to the `<abbr>` element as part of their default style sheets. This inline CSS overrides that styling. Ideally, rather than add the inline style to each `<abbr>` element, we'd simply have a CSS rule:

```
abbr {border-bottom-style: none}
```

In fact, when I raised this with Nate Koechley, he mentioned that this statement is part of the YUI Reset CSS file (see `http://developer.yahoo.com/yui/reset`), which "normalizes the default rendering of all HTML elements, for example it sets margin, padding, and border to 0, font sizes to [a] default, italic and bold styles to normal, and list-style to none."

However, at present, Yahoo Local doesn't use Reset CSS, and I imagine there are legacy or perhaps internal procedural implications that make simply adding the inline style a much easier and possibly shorter-term solution.

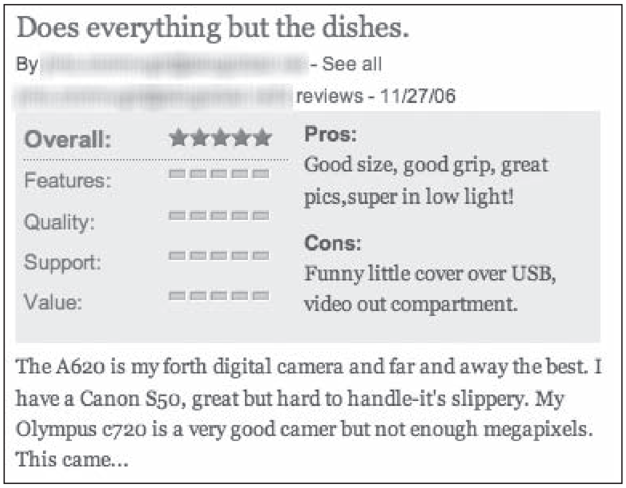
hReviews at Yahoo Tech

Let's compare Yahoo Local's reviews with those over at Yahoo Tech. Interestingly, Yahoo Tech uses hReview for the summary page of reviews of a particular product, but not for the full review page by an individual reviewer.

In the following sections, we'll see how Yahoo Tech uses hReview, and how this differs from Yahoo Local.

Review example

Figure 12-5 shows what a Yahoo Tech review looks like in a browser.



Does everything but the dishes.

By [redacted] - See all [redacted] reviews - 11/27/06

Overall: ★★★★★

Features: ▭▭▭▭▭

Quality: ▭▭▭▭▭

Support: ▭▭▭▭▭

Value: ▭▭▭▭▭

Pros:
Good size, good grip, great pics,super in low light!

Cons:
Funny little cover over USB, video out compartment.

The A620 is my forth digital camera and far and away the best. I have a Canon S50, great but hard to handle-it's slippery. My Olympus c720 is a very good camer but not enough megapixels. This came...

Figure 12-5. A review from Yahoo Tech

Reproduced with permission of Yahoo! Inc. © 2007 by Yahoo! Inc. YAHOO! and the YAHOO! logo are trademarks of Yahoo! Inc.

In many ways it's the same as (or similar to) a review at Yahoo Local. But let's take a look at the HTML (again, I've elided some code to get to the most relevant parts for our purposes):

```
<div class="arating hreview">
  <div class="hd">
    <h3 class="summary"><a ...</a></h3>
    <p class="metadetails ">By <a href="..."  ... class="reviewer ➥
      fn"> reviewer@obfuscated.com</a> ... - <span class="dtreviewed"> ➥
      11/27/06</span><a href="#prodname" class="include ➥
      microformatdetail"></a></p>
  </div>

  <div class="bd">
    <div class="ratreviewsummary">
      <ul class="ratingslist">
        <li class="overall stars10 ">Overall: <span ➥
          title="reviewer@obfuscated.com gave this product 5 out ➥
          of 5 stars for Overall quality" class="rating"> ➥
          <em class ="value">5</em>/<em class="best">5</em></span>
        </li>
        <li class="bars10">Features: <span ➥
          title="reviewer@obfuscated.com gave this product 5 out ➥
          of 5 stars for Features" class="rating">5/5 </span>
        </li>
        <li class="bars10">Quality: <span ➥
          title="reviewer@obfuscated.com gave this product 5 out ➥
          of 5 stars for Quality" class="rating">5/5 </span>
        </li>
```

12

```
        <li class="bars10">Support: <span ➥
          title="reviewer@obfuscated.com gave this product 5 out ➥
          of 5 stars for Support" class="rating">5/5 </span>
        </li>
        <li class="bars10">Value: <span ➥
          title="reviewer@obfuscated.com gave this product 5 out ➥
          of 5 stars for Value" class="rating">5/5 </span>
        </li>
      </ul>

      <dl class="procons description">
        <dt>Pros:</dt>
        <dd>Good size, good grip, great pics, super in low light!</dd>
        </dl>
        <dl class="procons description ">
        <dt>Cons:</dt>
        <dd>Funny little cover over USB, video out compartment.</dd>
      </dl>
    </div>
    <div class="ytuserreviewtext">
      <div class="ytReviewContainer">
        <p class="description item fn">The A620 is my forth digital ➥
          camera and far and away the best. I have a Canon S50, ➥
          great but hard to handle-it's </p>
      </div>
    </div>
  </div>
</div>
```

This is much closer to the div- and class-based markup we might usually expect. Unordered lists for ratings and definition lists for marking up pros and cons are used as well.

But there are some little issues that need fixing. Again, I don't mean to be critical—many of these things are still evolving, and indeed it's most likely that this code was implemented before hReview got to its current state.

First, while the overall rating is correctly marked up, as follows:

```
<li class="overall stars10 ">Overall: <span ➥
  title="reviewer@obfuscated.com gave this product 5 out of 5 ➥
  stars for Overall quality" class="rating"><em class="value">5 ➥
  </em>/<em class="best">5</em></span>
```

The individual ratings are not quite right. Instead of this:

```
<li class="bars10">Features: <span ➥
  title="reviewer@obfuscated.com gave this product 5 out of 5 ➥
  stars for Features" class="rating">5/5 </span>
```

where the value is 5/5, but should be a numerical value to a single decimal place, we could use the same pattern as for the overall ratings like so:

```
<li class="bars10">Features: <span ➥
  title="reviewer@obfuscated.com gave this product 5 out of 5 ➥
  stars for Features" class="rating"><em class="value">5 ➥
  </em>/<em class="best">5</em></span>
```

Or, because the default "best" is 5, we could simply use the following:

```
<li class="bars10">Features: <span ➥
  title="reviewer@obfuscated.com gave this product 5 out of 5 ➥
  stars for Features" class="rating">5</span>/5
```

Another important thing to note is that the FN for an item *must* appear inside the element for the item, not as a class value on the same element that has `item` as a class value.

Here, we'll need to make a small change, because for now, the `fn` for the item is on the item element, not on a descendent element:

```
<p class="description item fn">The A620 is my forth digital camera ➥
  and far and away the best. I have a Canon S50, great but hard to ➥
  handle-it's ...</p>
```

Also, while it is not an invalid hReview to have both a description and item for the same element (i.e., "The A620 is my fourth digital camera . . ."), this is not really the item we are reviewing. So, here it would be best to remove the `item` and `fn` class values from the description paragraph:

```
<p class="description">The A620 is my forth digital camera and ➥
  far and away the best. I have a Canon S50, great but hard to ➥
  handle-it's ...</p>
```

But we still need an item for the review. Remember, an `item` property is *required* in hReview. There is more than one review on this page of the same item, so that might bring to mind the include design pattern discussed in Chapter 9. We can use this design pattern to include details from one microformat to another, essentially when the microformats represent the same thing.

The first step is to uniquely identify the microformatted element we want to include a property or properties from. In this case, we'll have to look outside the hReview element itself for that element. If we look at the page itself, we find the following HTML:

```
<strong class="item fn" id="prodname">Canon PowerShot A620</strong>
```

We still have the problem that the `fn` for the item appears as a class on the element for the item, rather than on a descendent element for the item element. So, we change it as follows:

```
<strong class="item" id="prodname"><span class="fn">Canon PowerShot ➥
  A620</span></strong>
```

We give the item a unique identifier, using the id. Now we can return to our individual review, and include this in there:

```
<p class="description">The <span class="item"><a href="#prodname" ➥
  class="include"></a>A620</span> is my forth digital camera and ➥
  far and away the best. I have a Canon S50, great but hard to ➥
  handle-it's ...</p>
```

If we look back to the top of the review, we find something quite similar to this:

```
<a href="#prodname" class="include microformatdetail"></a>
```

A couple of other little things need attention. For example, a reviewer, if included, *must* be an hCard:

```
By <a href="..."  ... class="reviewer fn">reviewer@obfuscated.com</a>
```

Here, we need to add a class value of vcard and then make the fn on a descendent element of that element (remember the properties of a compound microformat are descendents of the root element). We end up with something like this:

```
By <span class="reviewer vcard"><a href=".."class="fn"> ➥
  reviewer@obfuscated.com</a></span>
```

Styling Yahoo Tech reviews

As with Yahoo Local, the Yahoo Tech site doesn't use the microformat markup for styling with CSS. Once more, I suspect it's because the markup has had microformats retrofitted onto it, so styling was already well and truly in place with the existing markup.

Is it worth it?

Having seen how Yahoo is adopting microformats in some pretty significant ways, a question comes to mind: is it worth it? I next asked Nate whether he felt adopting microformats has been worth the effort.

Anything that makes the Web a better place feels like the right thing to do. When it's as technically low-impact as microformats, why not? Each decision needs to be considered for each new context, of course, but implementing microformats seems like an easy win. That our efforts might positively impact the overall evolution of the Web is a bonus.

Finally, I asked Nate if he could tell us where Yahoo might go next with microformats.

I can't talk about future plans, but I will say that many people here are pretty excited about microformats, and some are actively involved with microformats.org. I think we've already demonstrated our willingness to deploy them, and I think it's safe to expect more of the same from Yahoo!.

Summary

This case study on Yahoo and the previous one on Cork'd demonstrate the versatility of microformats. New projects like Cork'd can take advantage of microformats from the ground up, not just for markup, but for styling with CSS as well. Existing projects—even large, sophisticated applications like those presented in this chapter from Yahoo—can be adapted to use microformats with little impact on the existing code base, HTML, and CSS.

I hope that seeing how such a broad range of industry leaders have adopted microformats and benefited from doing so will reassure you that microformats aren't just great bleeding-edge concepts, but a practical, useful, and increasingly commonly used technology. I hope, too, that these case studies inspire you to start using microformats in your existing and new projects.

In the next and final chapter, you'll see how new microformats come into being. Not only is this of interest if you want to be involved in the development of new microformats, but also the process that has developed is applicable if you want to standardize internal markup conventions for your company or develop specific microformat-like solutions to common development problems you face.

PART FOUR **DEVELOPING MICROFORMATS**

Throughout this book, you’ve learned about the major microformats, how they work, and what they enable. You've seen who is publishing microformats, who is building services around published microformat content, and the kinds of tools available to help make developing microformatted content and services easier. But you've haven’t yet examined how they come into being, which is what you’ll turn to in this part of the book.

13 THE PROCESS OF DEVELOPING MICROFORMATS

Surprisingly, one of the most common questions I get when speaking about microformats (which for my sins I do quite a bit) is, "How can I develop a new microformat?" or at the very least "How are microformats developed?"

I've stated a number of times that microformats are more than just a technology; rather, they are an approach to solving the problem of bringing richer, more consistent semantic markup to commonly found kinds of data on the Web. An important part of this approach is the principles underpinning microformats, which you've seen in action throughout the book, but so too is the process of developing a new format, a process that has developed over the period that microformats have matured.

The process is quite different, much more open, and more bottom-up when compared with the top-down, "closed-shop" approach by which "standards" are typically developed. Yet it strives to avoid the anarchy of the free-for-all browser-driven "development" of HTML in the early part of the 1990s.

In this chapter, we'll take a look at the process of developing microformats and the benefits it brings. While it's unlikely that a great many of the readers of this book will develop their own microformats soon, the principles and processes are useful generally for the internal standardization of development practices, patterns, coding conventions, and so on. And if you do ever feel the interest or need to get involved with the process of developing a new microformat, this chapter will give you a good starting point and inform you of the dos and don'ts.

We'll start by reviewing the microformat principles in the next section, and then move on to cover determining the problem a microformat might solve, researching the problem thoroughly, documenting the process and presenting examples, and developing a draft schema.

Microformat principles revisited

Let's very quickly revisit the principles underpinning microformats, which we first examined in Chapter 1. Microformats

- Solve a specific problem
- Start as simply as possible
- Are designed for humans first, machines second
- Reuse building blocks from widely adopted standards
- Are modular and embeddable
- Enable and encourage decentralized development, content, and services

Our process should enable and encourage these principles.

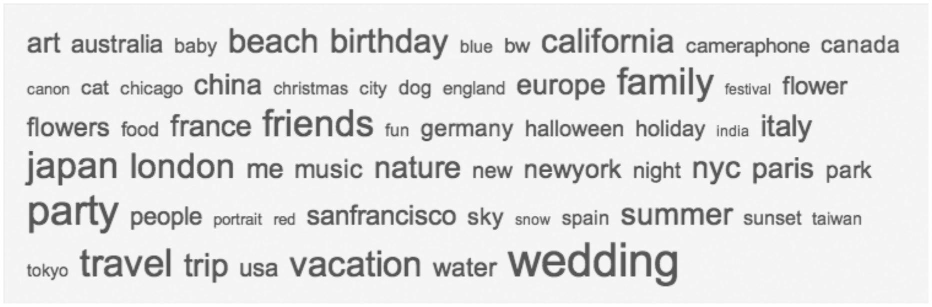
Determining the problem

The first step in developing a microformat is to establish whether there really is a problem to be solved. Microformats deal in real-world problems, not hypothetical or theoretical "what ifs" or "Wouldn't it be great if . . .?" kinds of problems.

If you find yourself thinking, "We tend to mark up this kind of information over and over again at our site," then standardizing your markup is probably an excellent idea. But the problem or need you are facing may not extend widely beyond your particular circumstances—it's only then that a particular problem might be amenable to a microformat.

The question is how to determine whether that problem is more generalizable. Because the microformats approach is community oriented and open, it's best to go directly to the microformats community, in particular the uf-discuss mailing list (`http://microformats.org/mailman/listinfo/microformats-discuss`), to get some feedback from people with considerable experience in developing microformats on whether you have a suitable candidate for a microformat. It also makes sense to search the Web for discussions elsewhere about similar problems.

For example, an area I recently did some research on, after discussions with a number of people who showed interest in the problem, is tag clouds. **Tag clouds**, first made popular by web applications like Flickr, use the relative text size of tags displayed together to indicate how popular these tags are with respect to one another. For example, Figure 13-1 shows Flickr's tag cloud.



art australia baby beach birthday blue bw california cameraphone canada canon cat chicago china christmas city dog england europe family festival flower flowers food france friends fun germany halloween holiday india italy japan london me music nature new newyork night nyc paris park party people portrait red sanfrancisco sky snow spain summer sunset taiwan tokyo travel trip usa vacation water wedding

Figure 13-1. An example tag cloud from Flickr

It's clear from a brief inspection that tag clouds are very widely used by sites and web applications, so this indicates that it is a problem for which the microformats approach *might* be valuable. A subsequent search of the Web finds that there doesn't appear to be any discussion of how to best standardize the markup of tag clouds, which despite their very common appearance across many sites are marked up differently at almost every site.

So here I have identified a problem and determined it is one for which there is as yet no reasonable solution (if there is a solution, there's no need for a microformat). And after further investigation, it appears that the problem is a candidate for a microformat.

Researching (or "paving the cowpaths")

The goal of microformats has been described as "paving the cowpaths"—that is, microformats aim to codify current behavior, rather than stipulate new behavior. You've no doubt seen a park where the natural path is across the grass, rather than along the paved area. Consequently, no one uses the "proper" path, instead deepening the natural path, and typically killing the grass in the process. There might even be a "Keep Off the Grass" sign, which of course is pointless, because people have made it clear precisely where they want to walk. Like me, you've probably wondered why the path wasn't actually designed where people wanted to go in the first place.

Microformats aim to avoid precisely this problem of putting paths where they aren't needed, by watching what people actually do. And we do that, as part of the process of developing a microformat, by researching how developers currently solve the problem under consideration.

We've seen this in action with the hReview microformat, where the final schema of hReview was a minimal subset of the common properties of a number of different widely used web-based review formats. It's recommended that this research be conducted in public—by creating a page on the microformats wiki (but only after the earlier steps are followed; one of the key aspects of this process is that we don't just start dozens or hundreds of ultimately unfinished microformats).

The microformats wiki is located at `http://microformats.org/wiki`, and is, like most wikis, open. To edit or add pages requires only an account, which anyone may create.

What sort of information should the research try to document? According to the process outlined on the "So you wanna develop a new microformat?" page on microformats.org (see `http://microformats.org/wiki/process`), it should be "real-world sites and pages which are publishing the kind of data you wish to structure with a microformat." For example, for the tag cloud research, the following were documented for several sites and web applications:

- What tag clouds model or represent (e.g., most popular tags over time, most popular tags in the last day or week)
- Screenshots of what tag clouds look like
- The coding conventions and implied schemas these tag clouds adopted, with code examples
- Analysis of these conventions and schemas (e.g., strengths and weaknesses of the particular design decisions of each of the various implementations)

This tag cloud–related information can be found on the microformats wiki: `http://microformats.org/wiki/tagcloud-examples`.

Are we ready to propose a microformat now? In fact, we aren't sure we need one yet. We've seen that HTML contains elements and attributes that are still not fully used as designed, so the next questions we ask are as follows:

- Is there an HTML element or attribute that could solve our problem, without the need for the complexities of a microformat?
- If not, how about an HTML compound? (You saw in Chapter 3 how using two or more HTML elements together can give you a much richer semantic vocabulary.)

Only if the answer to both of these questions is no should you really start thinking about proposing a new microformat.

Reusing others' work

Isaac Newton, one of the great minds and civilization-changing forces in history, remarked of his predecessor Descartes, "If I have seen a little further it is by standing on the shoulders of Giants."

Sometimes we overlook the work that lays the foundations for a breakthrough. In the case of microformats, we look expressly for those giants' shoulders to stand on. After all, it's arrogant to assume we are the first to encounter a problem. As we saw with hCard and hCalendar, often there exists "well established, interoperably implemented standards we can look at which address [our] problem"[1]—and at this stage, that's precisely what we want to see if we can find. Very smart, hardworking people may have spent a lot of effort solving the problem we face. Reusing that work has a number of benefits:

- It saves a lot of time and effort on our part.
- It will almost certainly be a better solution than one we can develop in a reasonable time frame.
- By adopting an existing schema, we get interoperability with the services and applications that have already adopted that schema. For example, with hCalendar, we get integration with iCal subscriptions for no extra effort simply by using the hCalendar markup, which is based on the iCalendar schema.

Starting from scratch

Sometimes, as we saw with hReview, there is no existing schema for us to rely on. Or, as in the case of the long-standing work by the microformats community on citations, there may be a number of well-established, effectively competing standards. Both situations present more complex challenges than taking an existing widely adopted schema and mapping it onto HTML.

There are no hard-and-fast rules as to how this aspect of the process is undertaken, but now, whether or not there is an established format, it's time to develop a draft proposal and iteratively improve that proposal through discussion and feedback from the microformats community.

1. See `http://microformats.org/wiki/process`.

Documenting the process

As part of the process of developing a microformat, creating several pages at the wiki is recommended, based on the experience of developing microformats so far.

Examples page

First is an examples page, which brings together real-world examples from the Web of content that could be marked up using the proposed microformat. The examples page should also document the implied schemas in the researched pages' markup. Here's an example from the tag cloud examples page at the wiki—the hot tags code at Flickr looks like this:

```
<table id="Recently">
  <tr>
    <td>
      <p><b>In the last 24 hours</b><br />
        <b><a href="/photos/tags/pics2006/">pics2006</a>, </b>
        <b><a href="/photos/tags/ubicomp2006/">ubicomp2006</a>, </b>
          ...
      </p>
    </td>
    <td>
      <p><b>Over the last week</b><br />
      <b><a href="/photos/tags/itunes7/">itunes7</a>, </b>
      <b><a href="/photos/tags/futureofwebappssf06/"> Â
        futureofwebappssf06</a>, </b>
Flickrs main tagcloud looks like this in HTML
      <p id="TagCloud">
      <a href="/photos/tags/06/" style="font-size: 12px;">06</a>
      <a href="/photos/tags/amsterdam/" style="font-size: 15px;"> ➥
        amsterdam</a>

      ...
```

Notes page

My notes on the Flickr tag cloud implementation are as follows:[2]

1. The hot tags are simply cells in a table with the id "recently".
2. The tag cloud proper is a p with the id of "tagcloud"—which would allow only a single tag cloud per page.

2. See http://microformats.org/wiki/tagcloud-examples.

3. The links are simply links—despite this being a list of words, ordered alphabetically.
4. The rank or weight or popularity of the tag is visually created using inline style and font-size.
5. There are at least a dozen different levels of popularity—difficult to determine without tedious work, as these are reflected in font-sizes in pixels.

This much simpler example comes from the hReview examples page, which documents the schema of a Yahoo Local review:

- Yahoo Local
- `http://local.yahoo.com`
- author (review author)
- publication date
- title
- description
- rating
- overall and by category [1–5]
- positive summary
- negative summary

Documenting these examples helps ensure any microformat we propose is "designed for humans first, machines second," in keeping with the third microformats principle.

Page for documenting existing formats and schemas

Next, where the example and notes pages exist, we need a page that documents the current, widely used data formats and standards "that attempt to or have attempted to solve the problem."

In some cases, like that of the citation format, there may be several; in other cases, like tag clouds, there may be none at all. This type of page helps ensure we adhere to the principle "reuse building blocks from widely adopted standards."

Brainstorming page

Now, it's time to start developing a proposal, referred to as the "brainstorming" phase and typically documented on a brainstorming page—for example, hResume-brainstorming (`http://microformats.org/wiki/resume-brainstorming`). More on developing a proposal in the next section.

Developing a draft schema

Based on the schema of real-world cases or existing standards, a draft schema is then developed. As we saw with hReview, this will typically be a subset of the common set of all properties of the different schema uncovered by the research phase. The goal is not to solve all possible problems, but rather to solve the minimal set of commonly occurring problems and needs developers have. I referred to this earlier as the 80/20 rule. Typically, a small part (20%) of a schema or other standard will meet the significant majority of most users' needs. In developing a microformat, we are aiming to hit this sweet spot.

An example from the real world touched on previously in this book can be found with iCalendar. The iCalendar format includes all kinds of complex properties to support non-Gregorian calendars, events recurring according to all kinds of complex and subtle rules, invitations to events, and so on. Not only does hCalendar not really concern itself with most of these aspects of iCalendar, but neither do most applications that actually support iCalendar! In fact, the IETF is developing a subset of iCalendar, iCalendar Basic, that covers the small part of iCalendar that is widely used. It's such lessons as these that the microformats community wishes to learn, in order to avoid a lot of effort developing features that will never be used widely, if at all.

There are several important aspects to this phase:

- What other microformats might play a role within this microformat? We've seen rel-tag, hCard, and even hCalendar reused in other microformats, for instance.
- What names will the properties or "fields" of this microformat take? We've seen that formats like vCard, as well as existing microformats (e.g., `summary` from hCalendar is reused a number of times), provide a source for these names.
- What name will the microformat itself take?

As this process continues, a **straw man proposal** (i.e., a proposal that is put up largely to discuss its shortcomings and iteratively develop more robust proposals) will hopefully evolve to a state where the consensus is that it is ready to be considered as a draft proposal. At this stage, this proposal is documented on a separate draft proposal page.

So how does a draft become a specification? As with other steps in the process, it is essentially by consensus. One criterion is that a specification "should be stable so that developers can pick it up and write to it."[3] Among other things, actual implementations of the specification are considered an important aspect in terms of determining whether it is stable. The IETF, which we have seen in the source of standards such as vCard and iCalendar, refers to this approach as "rough consensus and running code."

3. See `http://microformats.org/wiki/process`.

Summary

It may seem that the process of developing microformats is somewhat chaotic, or at least unstructured. No matter how well documented or structured a theoretical process is, the reality is typically less so. As Otto von Bismarck, the Minister-President of Prussia and a great practitioner of *realpolitik*, famously said, "The less people know about how sausages and laws are made, the better they'll sleep at night." As with a number of aspects of microformats, there is a pragmatism—indeed, an element of *realpolitik*—in their development. The solution to complex problems often involves compromise, at the very least.

The process of developing microformats has evolved and is still evolving. But the very fact that this evolution is documented by its practitioners means that hopefully it is also improving, and we continue to learn the lessons of others who have developed microformats, just as we hope to stand on the shoulders of giants when it comes to other aspects of microformats.

Microformats are young—at the time of this writing, they're not much more than a couple of years old. But while it's a decade or more older, the Web too is young, in the context of hypertext and hypermedia systems and networks, let alone publishing media. But despite this youth, one of the signs of an emerging maturity among developers for the Web, which is exemplified by the microformats community, is that the brash "everything is new, and you just don't get it" mentality of earlier years (which applied to design and development practices, as well as business models—and we saw how well that turned out a few years back) is being replaced with the recognition that there is much to learn from those who came before us—both on and off the Web—again, in terms of design, development practices, business development, and more.

Where the Web takes us in 5 or 50 years is impossible to say, as I think the last 5 or so years have demonstrated. But I have little doubt that the principles microformats embody and help enable, of openness, interoperability, iterative development, and distributed and decentralized data and services, will help shape that future.

PART FIVE **APPENDIXES**

APPENDIX A **MICROFORMAT SPECIFICATION REFERENCE**

This appendix details all of the microformats covered in this book. For each format, the authors or editors, a description, how the microformat is best used, the schema, and any restrictions or important things to keep in mind when using the microformat are noted. Each section also features one or more detailed examples for each microformat and points to publishing tools available to help developers.

rel-license

rel-license uses the `rel` attribute on a hyperlink element (`<a href=...>`) to indicate the license or licenses for the current document. The document at the destination of a rel-license link is the license or one of the licenses for this document.

Where a rel-license is found inside an hReview, the license is for the review, rather than the page itself. This enables aggregating reviews from multiple sources that have different license conditions.

- **More information**: `http://microformats.org/wiki/rel-license`
- **Status**: Specification
- **Author**: Tantek Çelik (`http://tantek.com`)

Usage

rel-license uses standard hyperlink elements (`<a href=...>`). The addition of a `rel="license"` attribute indicates that the destination document is a license for this document.

If a rel-license is embedded in an hReview, the license is the license for the review, not the document.

Examples

Here are some examples of using rel-license:

```
<a rel="license" href="http://creativecommons.org/licenses/by/2.5/"> ➥
  Creative Commons Attribution 2.5 License<a>

<a rel="license" href="http://www.gnu.org/licenses/gpl.txt"> ➥
  GNU General Public License<a>
```

Tools

Some tools that can help you create rel-license content are as follows:

- **Textpattern Microformats Plug-in** (`http://placenamehere.com/TXP/pnh_mf`): Created by Chris Casciano, the pnh_mf plug-in provides a simple way for adding various microformats, including rel-license, to Textpattern-based sites.
- **Dreamweaver Microformats Extension** (`www.webstandards.org/action/dwtf/microformats`): This suite, created by Drew McLellan and available from the Web Standards Project, provides tools for adding various microformats, including rel-license, to your HTML.

Services

A couple of services using rel-license are as follows:

- **Yahoo Creative Commons Search** (`http://search.yahoo.com/cc`): This service enables searching for content based on its use license. The search uses rel-license to find content matching certain license criteria, such as content that may be used for commercial purposes.
- **Google Usage Rights Search** (`www.google.com/advanced_search?`): This feature of Google's Advanced Search allows you to specify usage rights as a filter for searching. It uses rel-license to facilitate this searching.

rel-tag

rel-tag uses a `rel` value of `tag` on a hyperlink element to "tag" or "label" the page, blog post, or other major part of the page it appears in. When a rel-tag link appears inside another microformat (e.g., in hAtom) it applies to that microformat and not the page as a whole.

By adding a rel-tag link to a page (or other microformat), you make an assertion about what this page or part of a page is about.

- **More information**: `http://microformats.org/wiki/rel-tag`
- **Status**: Specification
- **Authors**: Tantek Çelik (`http://tantek.com`) and Kevin Marks (`http://epeus.blogspot.com`)

Usage

A rel-tag is a standard hyperlink element, with a `rel` value of (or including) `tag`. The `href` value points to an entry in a tag space. A **tag space** is a set of URLs of the form `http://tagspaceURL/tagvalue`. For example, `http://wikipedia.org/wiki/RSS` uses Wikipedia as a tag space, and the value of this tag is `RSS`. Also, `http://technorati.com/tag/RSS` uses Technorati as a tag space, but the value of this tag is identical to the previous one—`RSS`.

The actual value of the tag is the final segment of the `url` value, after the last `/`, meaning that the value of a tag is not dependent on the tag space being pointed to. Should the last character be a slash (`/`), then this final slash is ignored when determining the tag value. A resource should exist at the URL.

Technorati (`http://technorati.com/tag`) and Wikipedia (`http://wikipedia.org/wiki`) are common tag spaces, but anyone may implement his or her own tag space.

Examples

`<a href="http://technorati.com/tag/microformats"` **`rel="tag"`**`>microformats</a>` tags the page or major part of the page it is in as being about microformats.

`<a href="http://technorati.com/tag/cooking" lang="it"` **`rel="tag"`**`>cucina</a>` tags the page or major part of the page it is in as being about cooking. This demonstrates how tagging works with internationalization—the human-friendly language here is Italian, the language of the page in which it appears, but the tag value is `cooking`, meaning that we can aggregate pages based on their tags despite being in different languages.

Of course, there is no reason why we could not use markup like this: `<a href="http://technorati.com/tag/cucina" lang="it" rel="tag">cucina</a>`. But in this case, the value of the rel-tag is `cucina`. There is no requirement that tag values be English-language words, but by using the English translation of a tag value, pages can be aggregated on the basis of their content, despite their human language.

Tools

Some tools that can help you create rel-tag content are as follows.

For WordPress:

- **Simpletags**: `www.broobles.com/scripts/simpletags/`
- **Bunny's Technorati Tags**: `http://dev.wp-plugins.org/wiki/BunnysTechnoratiTags`

For Textpattern:

- **pnh_mf**: `http://placenamehere.com/TXP/pnh_mf/`
- **tru_tags**: `www.truist.com/blog/493/trutags-a-tagging-plugin-for-textpattern`

For Bloxsom:

- **Tagging plug-in from Axel Beckert**: `http://noone.org/blog/tags/Tagging`

For Dreamweaver:

- **Dreamweaver Microformats Extension**: `www.webstandards.org/action/dwtf/microformats`

Services

A number of services using rel-tag including the following.

Tag search:

- **Technorati Tag search** (`www.technorati.com/ping`): Technorati Tag search indexes tagged blog posts and other content. Use Ping to ensure Technorati indexes your rel-tagged content.
- **IceRocket Tags** (`http://blogs.icerocket.com/tag`): Blog search engine IceRocket also allows for tag-based searching.

Debugging:

- **rel-lint tool** (`http://tools.microformatic.com/help/xhtml/rel-lint`): You can "validate" your rel-tags using this service from Drew McLellan.
- **Operator** (`http://labs.mozilla.com/2006/12/introducing-operator/`): Created by Michael Kaply, Operator is an extension for Firefox 2 that extracts tag (and other microformatted) data from a page and enables various actions to be performed on it. In debug mode it shows errors to help debug problems.

Publishers

The following publishers use rel-tag for user tags:

- **ODEO**: `http://odeo.com`
- **Eventful**: `http://evdb.com`
- **LiveJournal**: `http://news.livejournal.com/86492.html?thread=24881884`

rel-nofollow

rel-nofollow uses a `rel` value of `no-follow` for links to external resources to indicate that search engines should not use the link to give additional PageRank to the linked page. Search engines such as Google typically use the number and authority of inbound links to a page to rank its place in search results.

rel-nofollow, introduced by Google in an attempt to address the issue of link spam, particularly in blog comments, is not without controversy. A thorough criticism can be found at `www.nonofollow.net`.

- **More information**: `http://microformats.org/wiki/rel-nofollow`
- **Status**: Specification
- **Editors**: Tantek Çelik (`http://tantek.com`) and Kevin Marks (`http://epeus.blogspot.com`)

Usage

To use rel-nofollow, add a `rel` value of `nofollow` to links.

Example

To link to a page while indicating to search engines that the link should not add to the linked page's PageRank, use the following form:

```
<a href="http://westciv.com" rel="nofollow">Style Master</a>
```

Tools

Many blogging systems automatically add `rel="nofollow"` to links in comments or make this feature available through plug-ins or other extension mechanisms.

- WordPress 1.5 and higher automatically add `nofollow` to user-submitted comment links. There are plug-ins to disable this feature, including dofollow (`http://wp-plugins.net/plugin/sem-dofollow`) and Follow URL (`http://wp-plugins.net/?filter=follow+url`).
- Moveable Type and Textpattern both require a plug-in to enable rel-nofollow on user-submitted comments.

Services

Recent research indicates that

- Google follows a link with the `rel` value of nofollow but does not index the linked page.
- MSN does not follow a link with rel-nofollow, nor does it index the page linked.
- Yahoo both follows a link with rel-nofollow and indexes the page linked.

VoteLinks

Whereas rel-nofollow allows a publisher to withhold PageRank from a linked page, VoteLinks enables a publisher to explicitly endorse, vote against, or abstain from voting for the resource linked to. The specification leaves open the issue of whether the vote pertains to the page linked to, or what the page linked to represents.

- **More information**: `http://microformats.org/wiki/votelinks`
- **Status**: Specification
- **Authors**: Tantek Çelik (`http://tantek.com`) and Kevin Marks (`http://epeus.blogspot.com`)

Usage

VoteLinks uses the `rev` attribute of a link, rather than the `rel` attribute. The `rev` attribute describes the relationship between the document containing the link and the linked document. Earlier versions of VoteLinks used the `rel` attribute (which describes the relationship between the linked document and the document containing the link), but this is logically incorrect and is now deprecated.

The `rev` attribute can take one of three values with VoteLinks:

- `vote-for`
- `vote-abstain`
- `vote-against`

Examples

To vote for a linked resource, use a link like this:

```
<a href="http://westciv.com" rev="vote-for">Style Master</a>
```

To express indifference or abstain from voting, use a link like this:

```
<a href="http://westciv.com" rev="vote-abstain">Style Master</a>
```

To vote against a linked resource, use a link like this:

```
<a href="http://westciv.com" rev="vote-against">Style Master</a>
```

Tools and services

At present there is not a significant amount of adoption of VoteLinks in tools or services.

XHTML Friends Network (XFN)

XFN enables publishers to encode in HTML, XHTML, or XML markup for common professional and personal relationships, by means of the `link` element and the `rel` attribute.

- **More information**: `www.gmpg.org/xfn`
- **Status**: Specification version 1.1
- **Authors**: Tantek Çelik (`http://tantek.com`), Matthew Mullenweg (`http://photomatt.net`), and Eric Meyer (`http://meyerweb.com`)

Usage

XFN uses a set of possible values for the `rel` attribute. These are grouped into seven categories. Some of these values are mutually exclusive. The designations from the XFN specification are as follows:

- **Friendship**
  - **Friend**: Someone you are a friend to. (I have a feeling the developers of XFN kept this one vague, as what constitutes a "friend" will vary perhaps significantly from person to person.)
  - **Acquaintance**: Someone you have exchanged greetings with and not much (if anything) more—maybe a short conversation or two.
  - **Contact**: Someone you know how to get in touch with.
- **Physical**
  - **Met**: Someone you have actually met in person.
- **Professional**
  - **Co-worker**: Someone you work with or who works at the same organization as you.
  - **Colleague**: Someone in the your same field of study/activity.
- **Geographical**
  - **Co-resident**: Someone you share a street address with.
  - **Neighbor**: Someone who lives nearby, perhaps only at an adjacent street address or doorway. (Like "friend," "neighbor" is a more fluid concept, varying from context to context.)
- **Family**
  - **Child**: Your genetic offspring, or someone you have adopted and take care of.
  - **Parent**: Your biological (or adopted) mother or father.
  - **Sibling**: Someone you share a parent with.
  - **Spouse**: Someone you are married to.
  - **Kin**: A relative, or someone you consider part of your extended family who doesn't fit into any of the preceding criteria.

- **Romantic**
  - **Muse**: Someone who brings you inspiration.
  - **Crush**: Someone you have a crush on.
  - **Date**: Someone you are dating.
  - **Sweetheart**: Someone with whom you are intimate and at least somewhat committed.
- **Identity**
  - **Me**: A link to yourself at a different URL.

Examples

The following are some examples of XFN in action:

```
<a href="http://marxandmarzipan.com" ➥
  rel="friend met colleague neighbor">Maxine, the co-founder ➥
  of westciv</a>

<a href="http://blogs.westciv.com/dog_or_higher" rel="me"> ➥
 Also by me</a>
```

Tools

Some tools that can help you create XFN content are as follows:

- **WordPress 2** (`http://wordpress.com`): WordPress has direct support for XFN in its standard “add link” feature.
- **XFN Creator** (`http://gmpg.org/xfn/creator`): This online-based tool creates XFN links.
- **Autoxfn** (`http://weblog.scifihifi.com/2004/02/08/autoxfn-10`): This is a plug-in for Bloxsom.
- **Dreamweaver Microformats Extension** (`www.webstandards.org/action/dwtf/microformats`): The Dreamweaver microformats toolbar supports XFN.

Services

The following services use XFN content:

- **Rubhub** (`http://rubhub.com`): This search engine uses XFN to present information about the relationships between sites.
- **Flickr** (`http://flickr.com`): Flickr uses XFN to publish relationships on its profile pages.
- **Upcoming** (`http://upcoming.org`): This site uses XFN to publish user-defined relationships.
- **Cork’d** (`http://corkd.com`): This wine community site uses XFN to publish the relationships between members.

geo

The geo microformat enables publishers to add geodata about the location of a person, place, event, or thing to a page. The geo microformat is a one-to-one representation of the geo property in vCard, and thus a subset of hCard.

geo microformatted content has a root element with a `class` value of geo and two properties, `longitude` and `latitude`.

- **More information**: `http://microformats.org/wiki/geo`
- **Status**: Draft (see `http://microformats.org/wiki/geo#Draft_Specification` for the latest draft specification)
- **Author**: Tantek Çelik (`http://tantek.com`)

Usage

Use the geo format to indicate a geographic location. The geo specification notes the following:

If the publisher knows and is publishing the address of the location, OR if the address of the location was what was actually entered by a human, and the publisher simply turned that into lat/long using some sort of a service, then the publisher SHOULD use adr to publish the actual human entered address information since that communicates far more semantic information than a simple geo lat/long coordinate.

Examples

There are two forms of the geo microformat: a standard "compound" form and an abbreviated form. Here's an example of the standard compound microformat:

```
<span class="geo">
 <span class="latitude">27.976628</span>, <span class="longitude"> ➥
   86.933302</span>
</span>
```

Here's an abbreviated form when the longitude and latitude shouldn't actually be visible, but should still be machine-readable:

```
Apollo 11 splashed down <abbr title="13.150000;169.150000" ➥
class="geo"> 400 miles (640 km) South West of Wake Island, in the ➥
North Pacific Ocean</abbr>
```

Tools

Some tools that can help you create geo content are as follows:

- **GEO Microformats to XML** (`http://suda.co.uk/projects/microformats/geo`): Brian Suda's XSLT service extracts geodata, and it can mash up this data with Google Maps.
- **Operator** (`http://labs.mozilla.com/2006/12/introducing-operator/`): Created by Michael Kaply, Operator is an extension for Firefox 2 that extracts geodata from a page and enables various actions on it.

Services

The following services use geo content:

- **Flickr** (`http://flickr.com`): Flickr uses geo for all geotagged photos and provides a drag-and-drop interface for geotagging.
- **Multimap** (`www.multimap.com`): Multimap uses geo to publish the location of a location search result.
- **Open Guides** (`http://openguides.org`) and **Wikitravel** (`http://wikitravel.org`): These open source projects use the geo microformat for publishing location geodata.

adr

The adr microformat enables publishers to mark up address information in a standard, interoperable way. The adr microformat is a one-to-one representation of the `adr` property in vCard, and thus a subset of hCard.

- **More information**: `http://microformats.org/wiki/adr`
- **Status**: Draft (see `http://microformats.org/wiki/adr#Draft_Specification` for the latest draft specification)
- **Author**: Tantek Çelik (`http://tantek.com`)

Usage

adr elements have a root element with a class of `adr` and one or more of the following subproperties (i.e., child elements with classes equal to the following values):

- `type`: One of `work`, `home`, `pref`, `postal`, `dom`, or `intl` (these values are taken directly from the vCard specification)
- `post-office-box`
- `extended-address`
- `street-address`
- `locality`
- `region`

- `postal-code`
- `country-name`

Example

Here's an example of adr in use:

```
<div class="adr">
  <div class="street-address">1600 Pennsylvania Ave</div>
  <span class="locality">Washington</span>,
  <span class="region">DC</span>,
  <span class="postal-code">20520</span>
  <span class="country-name">USA</span>
</div>
```

Tools

Because adr is a subset of hCard, the tools that help publish hCard content also help publish adr content. See the "hCard" section of this appendix for more details.

Services

Because adr is a subset of hCard, the services that use hCard content also use adr content. See the "hCard" section of this appendix for more details.

hCard

hCard is a one-to-one representation of the properties and values of vCard (IETF RFC 2426) in HTML. hCard extends the limited scope of vCard—essentially people—to include companies, organizations, and places.

- **More information**: `http://microformats.org/wiki/hcard`
- **Status**: Specification (see `http://microformats.org/wiki/hcard#Specification` for the latest specification)
- **Authors**: Tantek Çelik (`http://tantek.com`) and Brian Suda (`http://suda.co.uk`)

Usage

hCard allows publishers to mark up details (particularly, but not exclusively, contact details) for people, organizations, companies, and places.

hCards have a root element with a class value of `vcard` and a required subproperty of `fn` (formatted name), for the name of the entity represented by the hCard. Where an hCard is for an organization, the organization `fn` also includes a class value of `org` to specify this.

Optional subproperties include the following:

- Name and personal details
  - N (name): This is for the structured name of the entity, and it must take one or more subproperties from the following list: honorific-prefix, given-name, additional-name, family-name, honorific-suffix.
  - nickname: A nickname.
  - bday: Birthday. This subproperty typically uses the abbr design pattern to have both human-friendly and machine-readable values (see the "The abbr design pattern" section in Appendix B for more information).
  - photo: An image element (<img>) with a class of photo.
  - url: A URL for the person or organization.
- Contact information
  - adr (address): See the "adr" section of this appendix. The adr microformat is a subproperty of hCard.
  - geo: A longitude and latitude for the person, place, or organization represented by the hCard. See the "geo" section of this appendix for details. geo is a subproperty of hCard.
  - email: A contact email address. It can take a type and value, or just a value. The type may be one of the following: INTERNET, x400, pref, or "other IANA registered address types."
  - tel: A contact telephone number. This subproperty may have multiple numbers, and it can take a type and value, or just a value. The type may be one of the following: voice, home, msg, work, pref, fax, cell, video, pager, bbs, modem, car, isdn, or pcs.
- Organizational details
  - org: Where the hCard is for a person, this subproperty is for an organization the person works at, or has some other role with. Where the hCard is for an organization, this is the name of the organization. In this case, the element with a class value of fn *must* also have a class value of org.
  - role: The role a person plays within the organization specified by the org property.
  - title: The title a person has at the organization specified by the org property.
  - logo: An image element (<img>) with a class value of logo specifies a graphic image for the entity represented by the hCard. For a company, it will typically be the company logo. For an individual, it may be an avatar. See also the photo property, for photographic images of people.
- Other properties

Because hCard is basically vCard represented in HTML, it acquires all the vCard properties. Some of these are not commonly used (even by applications that support vCard), and so are probably going to be rarely used in hCard as well. They are included here for completeness. The descriptions are largely taken from the vCard specification.

- agent: Specifies information about another person who will act on behalf of the entity represented by the hCard.
- categories: In effect, these are tags for the hCard, implemented using the rel-tag microformat.
- key: Specifies the public key or authentication certificate associated with the entity the hCard represents.
- class: Specifies the access classification for the hCard (e.g., public, private, or confidential).
- label: Specifies the formatted text corresponding to delivery address of the entity represented by the hCard.
- mailer: Specifies the type of electronic mail software that is used by the entity represented by the hCard.
- note: Specifies supplemental information or a comment that is associated with the hCard.
- rev: A date/time value specifying revision information about the hCard.
- sound: Specifies digital sound content information that annotates some aspect of the hCard. By default, this type is used to specify the proper pronunciation of the name type value of the hCard.
- tz: Specifies information related to the time zone of the entity represented by the hCard.

Examples

First up, here's an hCard for a person:

```
<div class="vcard">
  <span class="fn n"><span class="given-name">William</span> <span ➥
    class="additional-name">Henry</span> <span class="family-name"> ➥
    Gates</span> <span class="honorific-suffix">III</span></span> ➥
    (born <abbr class="bday" title="1955-10-28">October 28, 1955
    </abbr>)
  <div class="adr">
    <p><span class="street-address">1835 73rd Ave NE</span>,</p>
    <p><span class="locality">Medina</span>,</p>
    <p><span class="region">WA</span> <span class="postal-code">98039 ➥
      </span></p>
    <p class="country-name">USA</p>
  </div>
  <p><span class="title">Chairman</span>, <span class="org">Microsoft ➥
    </span></p>
  <p><a href="http://www.microsoft.com/billgates/"
class="url">http://www.microsoft.com/billgates/</a></p>
  <p><a href="mailto:bill@microsoft.com" class="email">
bill@microsoft.com</a></p>
</div>
```

Here's an hCard for an organization:

```
<div class="vcard">
  <p class="fn org">Web Directions Conference Pty Ltd
  <a href="http://suda.co.uk/projects/X2V/get-vcard.php?uri= ➥
    http://microformatique.com/book/chapter7/index.html">
  <img src="images/vcard-add.png" alt="download vcard icon"></a></p>

  <div class="adr">
    <p class="street-address">8/54 Mitchell St</p>
    <p><span class="locality">Bondi</span> <span class="region">NSW ➥
      </span> <span class="postal-code">2026</span></p>
    <p class="country-name">Australia</p>
  </div>
  <p>Phone/Fax: <span class="tel"><span class="type">Work</span>: ➥
  <span class="value">61 2 9365 5007</span></p>
  <p>Email: <a class="email" href="mailto:info@webdirections.org"> ➥
    info@webdirections.org</a></p>
</div>
```

Tools

A number of tools can help you create hCard content:

- **Dreamweaver Microformats Extension** (`www.webstandards.org/action/dwtf/microformats`): The Dreamweaver microformats toolbar from the Web Standards Project, by Drew McLellan, supports creating hCard.
- **hCard Creator** (`http://microformats.org/code/hcard/creator`): From microformats.org, this tool generates the HTML for you, based on your form input.
- **Textpattern Microformats Plug-in** (`http://placenamehere.com/TXP/pnh_mf`): Created by Chris Casciano, the pnh_mf plug-in provides a simple way to mark up microformatted content, including hCard.

Services

The following services use hCard content:

- **Technorati Microformats search**: `http://kitchen.technorati.com`
- **Pingerati**: `http://pingerati.net`
- **X2V**: `http://suda.co.uk/projects/X2V`

hCalendar

hCalendar is a one-to-one representation of the properties and values of iCalendar (IETF RFC 2445) in HTML. iCalendar is an interoperable format for calendaring and scheduling applications.

- **More information**: `http://microformats.org/wiki/hcalendar`
- **Status**: Specification (see `http://microformats.org/wiki/hcalendar#Specification` for the latest specification)
- **Authors**: Tantek Çelik (`http://tantek.com`) and Brian Suda (`http://suda.co.uk`)

Usage

iCalendar is a reasonably complex standard that is often not entirely implemented by supporting applications. As such, there is a current effort to create a simplified subset of iCalendar, called iCalendar Basic. Aspects of iCalendar, while theoretically part of hCalendar, have yet to be implemented in HTML. As such, here we'll only consider the fully implemented aspects of hCalendar.

According to the hCalendar specification, the basic model of hCalendar is to model "iCalendar object/property names in lower-case for class names, and to map the nesting of iCalendar objects directly into nested XHTML." hCalendar allows publishers to mark up events and other time/date based episodes (such as educational experience and work experience).

hCalendars have an optional `vcalendar` root element. Where only a single calendar of events occurs on a page, where no root element is found, the page itself is assumed to be the root element. Where more than one calendar of events is found on a single page, the root element is any HTML element with a class value of `vcalendar`.

An hCalendar consists of one or more events. These are marked up in hCalendar as descendent elements of the root element, with a class value of `vevent`.

Required properties of events

At a minimum, a `vevent` requires a `summary` and a `dtstart` to appear as children of it:

- `summary`: This is typically the name of the event.
- `dtstart`: This is the event's start date, an ISO 8601–formatted date/time for the commencement of the event. It is recommended that the abbr design pattern be used to give both a human-readable version and machine-readable version of the date, for example:

  ```
  <abbr title="20070207" class="dtstart">Feb. 6th 2007</abbr>
  ```

Common optional properties of events

The following are optional properties of events:

- `description`: This property provides a "more complete summary" (according to the iCalendar specification) of the event. An event may only have one `description` subproperty, so where, for instance, the description spans multiple paragraphs, a single div with class `description` should contain these paragraphs.
- `dtend`: This is the event's end date (i.e., the date/time at which the event ends), in ISO 8601 format. A date without a time is in effect midnight of the day before the date, so for timeless dates you should use the date *following* the event date to be correct. It is recommended that the abbr design pattern be used to give both a human-readable version and machine-readable version of the date, for example:

  ```
  <abbr title="20070206" class="dtstart">Feb. 6th 2007</abbr>
  ```

Due to a quirk of hCalendar, a DTEND date without a time value is midnight between the day before and the date specified—in effect, the day before—hence the mismatch between the `title` *value and the human-readable version.*

  If a date and time value were used, the date part would be 20070206.

- `location`: The location of the event. The adr, geo, or hCard microformat can be used to mark these up in a more information-rich manner.
- `url`: A URL for the event (e.g., for more information).

Example

Note that the following is an example of an hCalendar for a single event. The containing vCalendar element would typically be optional (unless there were more than one distinct calendar on this page).

```
<div class="vevent">
  <h3 class="summary"><a href="http://north.webdirections.org" ➥
    class="url">Web Directions North</a></h3>

  <div class="description">
    <p>A Web design and development conference in Vancouver ➥
      Canada.</p>
    <p>Featuring a who's who of international web experts....</p>
  </div>
  <p>From <abbr title="20070206" class="dtstart">Tuesday, ➥
    Feb. 6th </abbr> to <abbr title="20070209" class="dtend"> ➥
    Thursday, Feb. 8th </abbr> 2007</p>
  <p class="location">Renaissance Vancouver Hotel Harbourside</p>
</div>
```

Tools

Some tools you can use to create hCalendar content are as follows:

- **Textpattern Microformats Plug-in** (`http://placenamehere.com/TXP/pnh_mf`): The pnh_mf plug-in for Textpattern helps publish hCard-formatted data and also supports publishing hCalendar content.
- **Dreamweaver Microformats Extension** (`www.webstandards.org/action/dwtf/microformats`): This Dreamweaver microformats extension by Drew McLellan enables easy, form-driven hCalendar creation in Dreamweaver.
- **hCalendar Creator** (`http://microformats.org/code/hcalendar/creator`): This tool provides an easy-to-use, form-driven interface for creating hCalendar content.
- **xfy** (`https://www.xfytec.com`) **and LiveWriter** (`http://ideas.live.com/programpage.aspx?versionId=4372c8c2-b76f-4d44-aea1-9835b61d8dc1`): These recently released desktop-based blogging tools both have hCalendar support. Microsoft LiveWriter is a Windows-only product, while xfy is written in Java and runs on Windows, Mac OS X, and Linux.

Services

These services use hCalendar:

- **Technorati Microformats search** (`http://kitchen.technorati.com`): This search engine aggregates events marked up using hCalendar. You should also ping Pingerati (`http://pingerati.net`) so that Technorati and other hCalendar aggregators know about your new or updated hCalendar content.
- **X2V** (`http://suda.co.uk/projects/X2V`): This service by Brian Suda converts hCalendar content to the iCalendar format, enabling applications that support iCalendar to import hCalendar content. It also enables applications such as Thunderbird and iCal, which support subscribing to iCalendar content to subscribe to hCalendar marked-up calendars on web pages.
- **Life Lint parser** (`www.lifelint.net`): Life Lint is similar to X2V in that it converts hCalendar to iCalendar (and RDF), and it has optional Outlook 2002 compatibility (although bear in mind that Outlook's support for iCalendar format is not perfect).

hReview

hReview is a simple format for marking up reviews on the Web. The schema of hReview was developed as a minimal subset of a broad range of review formats currently being published by the likes of Amazon, Epinions, Zagat, iTunes, and many more, as well as earlier attempts at common review formats, such as RVW, RDF Review Vocabulary, and Simple-Review XML.

- **More information**: `http://microformats.org/wiki/hreview`
- **Status**: Draft (see `http://microformats.org/wiki/hreview#Microformats_Draft_Specification_2006-02-22` for the latest draft specification)
- **Authors**: Tantek Çelik (`http://tantek.com`), Ali Diab (`http://360.yahoo.com/alidiabali`), Ian McAllister (`http://spaces.msn.com/members/ianmcallister`), John Panzer (`http://journals.aol.com/panzerjohn/abstractioneer`), Adam Rifkin (`http://ifindkarma.com/blog`), and Michael Sippey (`http://sippey.typepad.com`)

Usage

hReview is designed for publishing structured reviews of people, businesses, films, products, websites—indeed anything that can be reviewed online. It is used as follows.

- `root`: An hReview has a root element, which can be any HTML element, with a class value of `hreview`.

Required properties

hReview has one required property, `item`, with the following further requirements:

- When the review is for a person, the `item` property must be an hCard.
- When the review is of an event, the `item` property should be an hCalendar.
- Any item must have at least a nested `fn` subproperty, for the name of the item. Where the item is an hCard or hCalendar, this `fn` property will be automatically part of the nested microformat.
- Where an item has a photo or URL, subproperties of `item` with class values of `url` and `photo` should be used to mark these up.

Optional properties

An hReview may also have the following optional properties:

- `summary`: The title of the review itself.
- `type`: The type of item being reviewed, which is one of the following list: `product`, `business`, `event`, `person`, `place`, `website`, or `url`.
- `reviewer`: The author of the review. This is must be an hCard. Where the reviewer is anonymous, the value of the hCard `fn` is `anonymous`. For example, the following is the markup for a review by Francine Smith:

  ```
  <span class="reviewer vcard"><span class="fn">Francine Smith ➥
  </span></span>
  ```

  An anonymous review would be marked up as follows:

  ```
  <span class="reviewer vcard"><span class="fn">anonymous ➥
  </span></span>
  ```

- dtreviewed: The date on which the review was written or published, in ISO 8601 date/time format. Using the abbr design pattern for providing both human- and machine-readable dates and times is recommended. For example, a review written on May 17, 2007, in Sydney, Australia, might be marked up as follows:

  ```
  Written <abbr class="dtreviewed" title="20070517"> ➥
  May 17, 2007</abbr>
  ```

- description: The full text of the review.
- rating: The reviewer's rating for the item reviewed. By default, this is a fixed-point integer to one decimal point from 1.0 to 5.0 (e.g., 2.5 is a valid rating value, but 4.99 is not). Optionally, integer values from a specified "best" to "worst" may be used. For example, a standard score of 4.5 out of 5 may be marked up like this:

  ```
  I gave the film <span class="rating">4.5</span> out of 5
  ```

  To give a nondefault rating, you could use something like this:

  ```
  full marks, <span class="rating"> <span class="value">10</span> ➥
    out of <span class="best">10</span></span>
  ```

- version: The version of hReview used for marking up the review. The current version is 0.3, while this and all future versions will be backward compatible with version 0.2.
- tags: A list of rel-tag tags describing the review. No particular class value is required for the tag links to indicate they are tags; the rel="tag" value is sufficient for this.
- license: To indicate a license for the review, a rel-license link may be added to the review. In this instance, the license applies to the review, not the page containing the review.
- permalink: A link with rel values of bookmark and self indicates the permanent URL for the review.

Example

This example demonstrates the core features of hReview:

```
<div class="hreview">
  <h2 class="summary">Pirates of the Caribbean: Dead Man's Chest ➥
    - Disappointing at best</h2>
  <p class="item"> ➥
    <a href="http://disney.go.com/disneypictures/pirates/" ➥
    class="fn url">Pirates of the Caribbean 2 - Dead Man's Chest</a>
  </p>
  <p class="reviewer vcard"><span class="fn">John Allsopp</span></p>
  <p>Date reviewed <abbr title="20061118" class="dtreviewed"> ➥
    November 18 2006</abbr>.</p>
  <p>Rating: <span class="rating"><span class="value">4</span> ➥
    out of <span class="best">10</span></p>
```

```
    <div class="description">
      <p>After the success and excitement of the original Pirates ➥
        of the Caribbean...</p>
      <p>The characters lack any of the real freshness and ➥
        cleverness of the original, even <a ➥
        href="http://en.wikipedia.org/wiki/Johnny_Depp" rel="tag"> ➥
        Depp's </a> Jack Sparrow has become more...</p>
      <p>Disappointing, to say the least.</p>
    </div>
    <p><a href="http://creativecommons.org/licenses/by-sa/2.5/" ➥
      rel="license">Licensed under a creative commons attribution ➥
      share alike 2.5 license</a></p>
    <p>Review type: <span class="type">product</span></p>
    <p>hReview version<span class="version">0.3</span><p>
  </div>
```

Tools

Some tools you can use to create hReview content are as follows:

- **hReview Creator** (`http://microformats.org/code/hreview/creator`): Like microformat.org's other microformat creators, this is a form-driven interface that produces microformatted HTML for your review.
- **Textpattern Microformats Plug-in** (`http://placenamehere.com/TXP/pnh_mf`): For Textpattern users, Chris Casciano's pnh_mf supports publishing in the hReview format as well as a number of other microformats.
- **hReview WordPress Plug-in** (`www.aes.id.au/?page_id=28`): Andrew Scott has created this form-driven hReview plug-in, which allows for easy publishing of individual or multiple reviews on a single page or in a single post.
- **WP Movie Ratings** (`http://paulgoscicki.com/archives/2006/11/wp-movie-ratings-v14-released`): This plug-in from Paul Goscicki integrates with IMDb to allow pain-free reviewing of movies in hReview format.

Services

Services using hReview content include the following:

- **Revoo** (`http://revoo.com`): Revoo is a review aggregation and publishing service, which both aggregates reviews published on the Web in hReview format and publishes reviews in hReview format.
- **Cork'd** (`http://corkd.com`): This wine community site features reviews created using a form-driven interface and published using hReview.
- **Yahoo Tech** (`http://tech.yahoo.com`): This technology review site also uses hReview to publish technology reviews from users.

hResume

hResume is a format for semantically marking up resumes or curriculum vitae (CVs) for publishing on the Web, typically in HTML. The set of properties of hResume (its schema) was developed through research of existing publishing schemas for online resumes and resume formats such as Europass and HR-XML.

- **More information**: `http://microformats.org/wiki/hresume`
- **Status**: Draft (see `http://microformats.org/wiki/hresume#Status` for the latest draft specification)
- **Author**: Ryan King (`http://theryanking.com`)

Usage

hResume may be used by individuals to publish their resumes or by aggregators to publish resumes or CVs aggregated from the Web or otherwise obtained.

The root of an hResume is any HTML element with a class value of `hresume`. Its required property and optional properties are as follows.

Required property

hResume has a single required property, `contact`, which provides the name and contact details for the person whose resume this is. `contact` must be an hCard.

Optional properties

An hResume may have the following properties (the heading for each property is the class value of that property when marking up in HTML):

- `summary`: This property is for an "overview of qualifications and objectives" (per the hResume specification).
- `education`: This property is an hCalendar, the individual events of which are each an education event (e.g., secondary schooling, undergraduate degree, etc.). Each event has a `class` value of `education`.
- `experience`: This property is an hCalendar, the individual events of which are each a work event (e.g., jobs). Each event has a class value of `experience`. hCard should be used for job titles or roles. In this case, to prevent tedious repetition, the include design pattern can be used to include the `fn` from the contact hCard into all such hCards.
- `skill`: The person's skills can be listed as a rel-tag link for each skill, with the addition of the class value of `skill` on each such link.

- affiliation: Organizations with which the person has an affiliation can be marked up with hCards, with the additional class value of affiliation on the root element for each organization's hCard.
- publications: Articles, books, and other publications by the person can be added to an hResume simply by using the <cite> element. Where a <cite> element appears inside an hResume, the cited work is taken to have been authored by the person whose resume this is.

Example

This is my (somewhat truncated) resume, shown here as an example, using all the preceding features from hResume. The class values specific to hResume are emphasized.

```
<div class="hresume">
  <h1>Resume: <address class="vcard contact"><a ➥
    href="mailto:john@webdirections.org" class="fn email"> ➥
    John Allsopp</a></address></h1>
  <p class="summary">Software developer, conference organizer, ➥
    speaker, and author John Allsopp has spent the last 15 years ➥
    running and developing software for Western Civilisation ➥
    Pty. Ltd., writing and publishing training courses, speaking ➥
    at conferences, organizing conferences, and running training ➥
    seminars in standards-based web development.</p>
  <h3>Education</h3>

  <ol class="vcalendar">
    <li class="vevent education"><abbr class="dtstart" ➥
      title="19850101">1985</abbr>-<abbr class="dtend" ➥
      title="19891231">1989</abbr> <abbr title="Bachelor of ➥
      Science "> B.SC</abbr> <span class="vcard"><a class="fn ➥
      org url"href="http://www.usyd.edu.au/">University of ➥
      Sydney, Australia </a></span> <span class="summary">Majoring ➥
      in Computer Science and Mathematics. Also studied Law, ➥
      English Literature, and Philosophy.</span></li>
    <li class="vevent education"><abbr class="dtstart" ➥
      title="19790101">1979</abbr>-<abbr class="dtend" ➥
      title="19841231"> 1984</abbr> <abbr title="Higher School ➥
      Certificate">H.S.C</a><span class="vcard"><a class= ➥
      "fn org url" href="http://www.riverview.nsw.edu.au">Saint ➥
      Ignatius College, Riverview</a></span></li>
  </ol>
```

```
<h3>Experience</h3>
<ol class="vcalendar">
  <li class="vevent experience"><abbr title="19940101" ➥
    class="dtstart">1994</abbr> - present. Chief Technology ➥
    Officer, <span class="vcard"><a class="fn org url" ➥
    href="http://westciv.com">Western Civilisation Pty. Ltd.</a> ➥
    </span> <span class="summary">Responsible for the development ➥
    of Software, including Palimpsest, Style Master, and Layout ➥
    Master</span></li>
  <li class="vevent experience"><abbr title="20060101" ➥
    class="dtstart">2006</abbr>- present Director, <span ➥
    class="vcard"><a class="fn org url" ➥
    href="http://webdirections.org">Web Directions Conferences ➥
    Pty. Ltd.</a></span> <span class="summary">Responsible ➥
    for a broad range of logistical and management aspects of ➥
    organizing the Web Directions conferences.</span></li>
  <li class="vevent experience"><abbr title="20030101" ➥
    class="dtstart">2003</abbr> - <abbr title="20051231" ➥
    class="dtstart">2005</abbr> Director, <span class="vcard"> ➥
    <a class="fn org url" href="http://webessentials.com">WE04 and ➥
    WE05 conferences</a></span> <span class="summary">Responsible ➥
    for a broad range of logistical and management aspects of ➥
    organizing the WE04 and WE05 conferences.</span></li>
</ol>
<h3>Skills</h3>

<ul>
  <li><a class="skill" ➥
    href="http://en.wikipedia.org/wiki/Software_engineering" ➥
    rel="tag">Software Engineering</a></li>
  <li><a class="skill" ➥
    href="http://en.wikipedia.org/wiki/Programming" ➥
    rel="tag">programming</a></li>
  <li><a class="skill" ➥
    href="http://en.wikipedia.org/wiki/Web_development" ➥
    rel="tag">web development</a></li>
    ...
</ul>

<h3>Affiliations</h3>
```

```
  <ul>
    <li class="vcard affiliation"> ➥
      <a href="http://www.webstandards.org/" class="fn org url"> ➥
      Web Standards Project</a></li>
    <li class="vcard affiliation"> ➥
      <a href="http://www.northbondisurfclub.com/" class= ➥
      "fn org url"> North Bondi SLSC</a></li>
  </ul>

  <h3>Publications</h3>

  <ol>
    <li><cite href="http://www.alistapart.com/articles/dao/"> ➥
      <a href="http://www.alistapart.com/articles/dao/">A Dao of Web ➥
      Design</a></cite> A List Apart, April 2000</li>
    <li><cite ➥
      href="http://www.westciv.com/style_master/ ➥
      house/good_oil/not_paper/"> ➥
      <a href="http://www.westciv.com/style_master/ ➥
      house/good_oil/not_paper/"> ➥
      Web Pages aren't printed on paper</a></cite>September 1999</li>
  </ol>

</div>
```

Tools

Some tools you can use to create hResume content are as follows:

- **hResume Creator** (http://hresume.weblogswork.com/hresumecreator): This form-based resume creator from Spur, Inc. is much like the other microformats creators, such as hCard Creator and hCalendar Creator.
- **hResume WordPress Plug-in** (http://hresume.weblogswork.com/?page_id=3): Another tool from Spur, Inc., this plug-in for WordPress helps publish hResumes on WordPress-based blogs.
- **Microformat Resume Plug-in for WordPress** (www.ssdesigninteractive.com): Sajid Sayid created this easy-to-use plug-in for adding hResume content to a WordPress-based site.

Services

Here are some services using hResume:

- **Emurse** (`http://emurse.com`): This service for building, hosting, and publishing resumes online uses the hResume format.
- **LinkedIn** (`www.linkedin.com`): This professional online networking service uses hResume for publishing 9 million resume details.

hAtom

hAtom is a microformat based on a subset of Atom, the XML syndication format for web feeds. hAtom specifically focuses on the aspect of Atom concerned with blog posts. Despite being a draft specification, because it is based on the nearly 1.0 standard Atom, hAtom's schema is quite stable.

- **More information**: `http://microformats.org/wiki/hatom`
- **Status**: Draft (0.1; see `http://microformats.org/wiki/hatom#Draft_Specification` for the latest draft specification)
- **Authors**: David Janes (`http://blogmatrix.blogmatrix.com`), with contributions by Tantek Çelik (`http://tantek.com`) and Benjamin Carlyle (`http://members.optusnet.com.au/benjamincarlyle/benjamin/blog`)

Usage

hAtom is used to mark up feeds in HTML, rather than in separate files, such as RSS or Atom. Rather than replacing RSS or Atom, hAtom's aim is to use the schema of Atom to create a standardized format for publishing blogs and other feed-based content in HTML. Because Atom and RSS feeds typically list only recent posts, hAtom is particularly beneficial for archiving posts.

Schema

hAtom feeds have an optional root element of `hfeed`. It can be any HTML element, as long as it has a class value of `hfeed`. Where there is no feed element, the page itself is assumed to be the feed (in the same way that the hCalendar element is optional).

A feed consists of individual posts. A post element is any HTML element with a class value of `hentry`.

Required properties

An entry must have the following properties (the heading for each property is the class value of that property when marking up in HTML):

- `entry-title`: The title of a particular entry (usually a heading of some level).
- `author`: The author of the entry. This must be marked up using an hCard, and it should, where possible, use an `<address>` element. Because the `<address>` element is an inline element, this may cause problems for complex contact details (e.g., postal addresses), but with hAtom, typically contact details are the URL of a form-based contact page or email address, so this is less of a potential problem than the suggested use of `<address>` in hResume.
- `updated`: The date the entry was last updated. Use the datetime design pattern to publish both human- and machine-readable versions.

Optional properties

An entry may have the following properties (the heading for each property is the class value of that property when marking up in HTML):

- `entry-summary`: A short summary or extract from the entry.
- `entry-content`: The full content of the post (an entry may have more than one `entry-content`).
- `published`: The date the entry was published. Use the datetime design pattern to publish both human- and machine-readable versions. A single element can be both the published and updated dates like so:

  ```
  <abbr title="20070101" class="updated published"> ➥
  1st January 2007</abbr>
  ```

- `permalink`: A link with a `rel` value of `bookmark` is the permalink for an entry. No specific class value is required for the link.
- `tag`: Both a feed and individual entries can have one or more “categories.” Both feed and entry categories are defined by the use of a link with a `rel` value of `tag`. Feed categories appear inside a feed root element, but outside any particular hEntry elements. Where a link with a `rel` value of `tag` appears inside an hEntry element, it defines a category for the entry. No specific class value is required for the link.

Example

This hFeed example demonstrates all the features of hAtom. The hAtom class values are emphasized.

```
<div class="hfeed">
<!-- these are the categories for the feed because they appear ➥
  inside the hfeed element but outside any specific hEntry element -->
<ul>
  <li><a href="http://www.technorati.com/tag/microformats" rel="tag"> ➥
    microformats</li>
  <li><a href="http://www.technorati.com/tag/semantics" rel="tag"> ➥
    semantics </li>
  <li><a href="http://www.technorati.com/tag/HTML" rel="tag">HTML</li>
  <li><a href="http://www.technorati.com/tag/patterns" rel="tag"> ➥
    patterns</li>
</ul>
<!-- there may be many hentries here before we get to the entry we ➥
  have been marking up -->
<div class="hentry">
  <h3 class="entry-title">Introducing Web Connections</h3>
  <p>Published:<abbr class="updated published" title="20060920"> ➥
    September 20, 2006</abbr></p>
  <div class="content">
    <p><span, class="summary">Super smart, far too young and good ➥
      looking, <a href="http://toolmantim.com/">Tim Lucas</a> and ➥
      Cam ("<a href="http://www.themaninblue.com/">the man in ➥
      blue</a>") Adams, have just taken the wraps off a fantastic ➥
      web app, "~<a href="http:///connections.webdirections.org">Web ➥
      Connections </a>"</span> they've built to help networking at ➥
      our upcoming (next week!) conference, ➥
     <a href="http://www.webdirections.org/"> Web Directions</a>.</p>
    <p>Tim and Cam have incorporated reltag, XFN, hCard, and ➥
      hCalendar. <strong>Seriously microformatted</strong>.</p>
      ...
    <p>Categories</p>
    <!-- these are the categories for the entry because they appear ➥
      inside the hEntry element -->
    <ul>
      <li><a href="http://www.technorati.com/tag/microformats" ➥
        rel="tag"> microformats</li>
      <li><a href="http://www.technorati.com/tag/webapps" ➥
        rel="tag">webapps</li>
      <li><a href="http://www.technorati.com/tag/WD06" ➥
        rel="tag"> WD06 </li>
    </ul>
  </div> <!-- end hEntry -->
<!--there may be many more entries after this one in the feed -->
</div> <!-- end hFeed -->
```

Tools

Some tools you can use to create hAtom content are as follows:

- **hAtom Creator** (`http://dichotomize.com/uf/hatom/creator.html`): This is a form-based creator for individual hAtom entries.
- **WordPress Sandbox theme** (`http://www.plaintxt.org/themes/sandbox`): This "skeleton" theme for WordPress (which you can use to develop your own themes on top of), by Scott Allan Wallick and Andy Skelton, supports hAtom and is rumored to be the basis for the next default WordPress theme.
- **WordPress loop with hAtom support** (`www.fberriman.com/?p=86`): If you are interested in adding hAtom to your existing WordPress theme or install, use this WordPress loop by Frances Berriman to add hAtom to your blog.

Services

The following services use hAtom:

- **hAtom2Atom** (`http://rbach.priv.at/hAtom2Atom`): This XSLT processor is for transforming hAtom marked-up content to Atom.
- **Tails** (`http://blog.codeeg.com/tails-firefox-extension-03`): The venerable Firefox extension supports hAtom.

APPENDIX B MICROFORMAT DESIGN PATTERNS

Microformat design patterns are not themselves microformats; rather, they formalize commonly reused pieces of code that are generally useful in developing new microformats (and may also be useful in developing semantic HTML other than just in relation to microformats).

This appendix covers the design patterns you'll encounter most frequently in your work with microformats.

The abbr design pattern

The abbr design pattern uses the `<abbr>` (abbreviation) element of HTML to provide a mechanism for marking up both human-friendly and machine-readable values for the same element. This is important to meet the microformats principles of being "designed for humans first, machines second" and to "enable and encourage decentralized development, content, services."

Usage

The human-readable value is the content of the `<abbr>` element, while the machine-readable part is the value of the `title` attribute of the `<abbr>` element. The pattern is commonly used with dates and times (see also the "The datetime design pattern" section), but also with a specific form of the geo microformat. It can also be used where both a human- and machine-readable version of the marked-up information is required.

Some common uses are as follows:

- **Date/time**: The pattern is used to specify a human-readable date/time and an unambiguous machine-readable date/time value (using ISO 8601, as recommended by the W3C for dates on the Web).
- **Geodata**: The pattern is used to specify a machine-readable decimal longitude and latitude value, and a human-readable location.

Examples

Here's an example for date/time:

```
<abbr title="20061028T17:53:17+09:00">5:53pm, 28th October 2006, in ➥
  Sydney Australia</abbr>
```

Here's an example for geodata (latitude precedes longitude):

```
<abbr title="13.150000;169.150000" class="geo">400 miles (640 km) ➥
  South West of Wake Island, in the North Pacific Ocean</abbr>
```

The class design pattern

The class design pattern is a fundamental microformat design pattern. It is used to include the semantics of a schema, either existing, like vCard, or novel, like hReview, into HTML, by the use of specific `class` attribute values on HTML elements.

Usage

For more complex microformats, there is a vocabulary of possible `class` values. These `class` values are added to HTML elements to give the semantics of the particular microformat to that element. Microformats never mandate the use of particular HTML element with this design pattern (e.g., requiring a div or heading with a specific `class` value), but they do at times recommend elements—such as the `<address>` element for the contact property of an hResume.

To use the class design pattern, simply add a specific `class` value from the schema for the microformat to an HTML element. Multiple `class` values are permitted and are not uncommon.

There are some restrictions on the use of `class` values. `class` values for the subproperties of a microformat must be on descendent elements of the root element, not on the root element itself. However, a root element may take multiple classes, where the microformat is nested inside another microformat. In this case, it takes both the `class` value for the root element and the `class` value for the property it represents in the microformat it is nested inside.

Examples

A simple use of the class design pattern is to add the semantic value `country-name` to an element:

```
<span class="country-name">Australia</span>
```

In the next example, the value `vcard` is the root for the hCard element, while `reviewer` is the `class` value for the role this element plays in the hReview it is part of. hCard requires an `fn` property, but this must be on a descendent element of the root element, not on the root element itself.

```
<span class="vcard reviewer"><span class="fn">Francine ➥
  Smith</span></span>
```

In some cases, multiple class values are required. In hCard, where the details are for an organization, the `fn` (formatted name) property must also have an `org` value for its `class`:

```
<span class="fn org">IBM</span>
```

The datetime design pattern

The datetime design pattern is a specific use of the abbr design pattern. It enables marking updates in both human-friendly and unambiguous machine-readable formats, using the ISO 8601 date/time format. See the section "The abbr design pattern" for details and examples.

The include pattern

The same microformatted information, with small changes, may appear a number of times on the same page. For example, in hResume, hCard is recommended for each `experience` property where a title or role is involved. Because `fn` is required in an hCard, this necessitates repeating the `fn` property each time the hCard for a specific person is used. Not only is this more verbose than necessary, but it also requires some mechanism for hiding the formatted names, as their appearance numerous times would appear incongruous.

The include pattern enables a property from one microformat (e.g., the `fn` from an hCard) to be included in other microformats on the same page (the include pattern does not allow including parts of a microformat on another page into a microformat—it is strictly intrapage).

Usage

Because the initial include pattern proposal used the `<object>` element, which caused difficulties with Safari, the include pattern is, at the time of this writing, not entirely finalized.

The basic concept of the include pattern is to uniquely identify one microformat using an `id` attribute, and then include this microformat in other microformats by using a link to this uniquely identified element from within subsequent microformats. This has the effect of including or "importing" the properties of the included microformat into the other microformats.

Examples

The first part of the include pattern is to add a unique identifier to a microformat using the `id` attribute:

```
<div class="hResume">
  <h1 class="summary">Resume: John Allsopp</h1>
  <div class="vcard contact" id="jafn">
    <a href="mailto:john@webdirections.org" ➥
      class="email fn">John Allsopp</a>
  </div>
</div>
```

Now you include this in a subsequent microformat on the same page, using a link of class include pointing to this element, as part of your microformat:

```
<li class="vevent experience"><abbr title="19940101" class="dtstart"> ➥
  1994</abbr> - present.
  <span class="vcard">
    <a href="#jafn" class="include"></a>
  <span class="title">Chief Technology Officer</span></span>
</li>
```

The rel design pattern

Links often define the relationship between documents. Where that is the case, the rel attribute of HTML can be used to encode this relationship. This aspect of HTML is used extensively by microformats.

Usage

The rel design pattern uses a keyword value for the rel attribute and a link to indicate the relationship between the document specified by the link href and the document containing the link. A number of microformats use a single rel value (rel-license, rel-tag, rel-nofollow), while XFN uses a small set of rel values.

Example

rel-tag uses a rel value of tag to identify a link as a microformatted tag link:

```
<a href="http://wikipedia.org/wiki/Pareto_principle" rel="tag"> ➥
Pareto Principle</a>
```

Attribute conventions

While these are not strictly design patterns in themselves, I'll touch on a number of conventions regarding the use and priority of certain HTML attributes. Because attributes typically carry their own semantics (or, more relevantly, their own particular purpose), it's a convention in microformats that when a given attribute is most relevant, it should be used.

You can find examples of this throughout the book, but no particular mention has been made of the convention. For example, where a microformat (such as hCard or hCalendar) has a url property, rather than marking this up as follows:

```
<span class="url"><a href="http://north.webdirections.org">Company ➥
  Web Site</a></span>
```

where the url value would actually be Company Web Site, we make use of the semantics and purpose of the href attribute and mark it up like this:

```
<a href="http://north.webdirections.org" class="url">Company ➥
  Web Site</a>
```

Now, why don't microformat parsers take Company Web Site as the value of the url property in this instance? Because parsers understand the convention, which says that the href attribute makes the most sense for the kind of data we expect to find for a url property.

Similarly, where a property is applied to an image or other replaced element (such as an object element), because these kinds of elements have no content, we can use the alt attribute where we'd typically use the content of the element. So, for example, instead of the following:

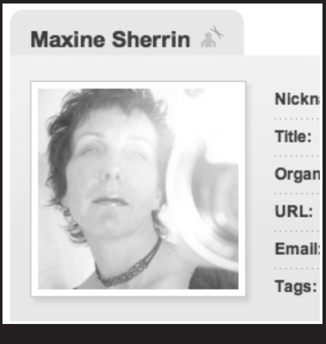
```
<span class="fn">Brian Suda</span>
```

if we wanted to have both a photo and fn combined, we could use the alt attribute on the <img> element, like so:

```
<img class="fn photo" ➥
  src="http://flickr.com/photos/paulhammond/149463680/" ➥
  alt="Brian Suda"/>
```

Because we know the value of fn is text, and because the alt attribute is alternate text for the image (in effect, the text equivalent), the convention is that we use the value of the alt attribute for the value of the fn property. Similarly, as the standard way of including image data on the Web is via the src attribute of an <img> element, the value of the photo property is the image at the src location.

APPENDIX C PEOPLE, TOOLS, SERVICES, AND PUBLISHERS



This appendix brings together many of the people, services, tools, and publishers covered in the book. By no means is it definitive; it aims to give a broad overview of the more important developments, people, and organizations involved with microformats. For more information, see the microformats wiki `http://microformats.org/wiki`.

People

- **Wolfgang Bartelme** (`www.bartelme.at`): Developer of the XFN icon set
- **Chris Casciano** (`http://placenamehere.com/TXP/pnh_mf`): Developer of the Textpattern microformats plug-in pnh_mf
- **Dan Cederholm** (`www.simplebits.com`): Microformats icon designer, and early adopter of microformats at Cork'd and elsewhere
- **Tantek Çelik** (`http://tantek.com`): Microformats pioneer; CTO of Technorati; author/editor of hCard, hCalendar, hReview, XFN, rel-nofollow, rel-tag, rel-license, and VoteLinks; and developer of the hCard Creator
- **Michael Kaply** (`www.kaply.com/weblog`): Developer of the Operator extension for Firefox
- **Ryan King** (`http://theryanking.com`): Author of hResume, and developer of hCalendar Creator and hReview Creator
- **Nate Koechly** (`http://developer.yahoo.com/yui`): Senior Engineer and Technical Evangelist on the Yahoo User Interface (YUI) Library team, and microformats early adopter
- **Hans Lube**: Inventor of Smell-O-Vision
- **Kevin Marks** (`http://epeus.blogspot.com`): Editor of rel-nofollow, VoteLinks, and rel-tag
- **Drew McLellan** (`http://allinthehead.com`): Developer of the WaSP Dreamweaver microformats toolbar
- **Chris Messina** (`http://factoryjoe.com`): Developer of the XFN icon set, and early microformats adopter and evangelist
- **Eric Meyer** (`http://meyerweb.com`): Microformats pioneer and author of XFN
- **Matt Mullenweg** (`http://photomatt.net`): WordPress founder, microformats pioneer, author of XFN, and developer of the hCard Creator and Exefen
- **Brian Suda** (`http://suda.co.uk`): Developer of X2V and other innovative microformats and proof of concept services and applications, and author of hCard and hCalendar
- **Thomas Vander Wal** (`www.vanderwal.net/random`): Coined the term "folksonomy" for bottom-up, tag-based taxonomies
- **Calvin Yu** (`http://blog.codeeg.com`): Developer of the Tails extension for Firefox

Tools

- **Autoxfn** (`http://weblog.scifihifi.com/2004/02/08/autoxfn-10`): A Blosxom plug-in for creating XFN.
- **Bunny's Technorati Tags** (`http://dev.wp-plugins.org/wiki/BunnysTechnoratiTags`): A rel-tag plug-in for WordPress.
- **Conference Schedule Creator** (`http://dmitry.baranovskiy.com/work/csc`): Authored by Dmitry Baranovskiy, this tool helps to easily create complex, table-based schedules such as conference timetables.
- **Exefen** (`http://photomatt.net/tools/exefen/`): Authored by Matt Mullenweg, this tool creates XFN links for existing links in online pages.
- **hAtom Creator** (`http://dichotomize.com/uf/hatom/creator.html`): A form-based tool for creating individual posts, by Ben West.
- **hAtom2Atom** (`http://rbach.priv.at/hAtom2Atom`): An XSLT processor for transforming hAtom marked-up content to Atom. Authored by Robert Bachmann.
- **hCalendar Creator** (`http://microformats.org/code/hcalendar/creator`): An easy-to-use, form-driven interface for creating hCalendar content, by Ryan King.
- **hCard Creator** (`http://microformats.org/code/hcard/creator`): An easy-to-use, form-driven creator for hCards, by Tantek Çelik.
- **hKit Microformats Toolkit for PHP5** (`http://allinthehead.com/hkit`): Created by Drew McLellan, this PHP5 toolkit extracts common microformats, including hCard.
- **hResume Creator** (`http://hresume.weblogswork.com/hresumecreator`): This form-based resume creator from Spur, Inc. is much like the other microformats creators.
- **hResume WordPress Plug-in** (`http://hresume.weblogswork.com/?page_id=3`): Another tool from Spur, Inc., this plug-in for WordPress helps publish hResumes on WordPress-based blogs.
- **hReview Creator** (`http://microformats.org/code/hreview/creator`): This form-driven interface, created by Ryan King, produces microformatted HTML for your review.
- **hReview WordPress Plug-in** (`http://aes.id.au/?page_id=28`): Authored by Andrew Scott, this tool allows for easy publishing of individual or multiple reviews on a single page or in a single post in WordPress.
- **Life Lint** (`www.lifelint.net`): This tool converts hCalendar to iCalendar or RDF, and it has optional Outlook 2002 compatibility.
- **Microformat Parser for Ruby** (`http://blog.labnotes.org/2005/11/20/microformat-parser-for-ruby`): A microformat parser for Ruby and Rails developers, authored by Assaf Arkin.
- **Microformat Resume Plug-in for WordPress** (`www.ssdesigninteractive.com`): Authored by Sajid Sayid, this is an easy-to-use plug-in for adding hResume content to a WordPress-based site.
- **Microformatic** (`http://microformatic.com`): This site is home to a number of microformats tools, such as hKit and "hCard n best-guess."

- **Microformats Bookmarklets** (`http://microformats.org/wiki/bookmarklets`): This page presents a collection of several **bookmarklets**, which can be added to your browser bookmarks or favorites to perform simple microformats actions, such as extracting iCalendar or hCard data, or converting iCalendar to RDF.
- **Operator extension for Firefox** (`https://addons.mozilla.org/firefox/4106`): This extension extracts microformatted content from pages and presents these in a toolbar enabling related actions relevant to the particular content. Authored by Michael Kaply.
- **pnh_mf** (`http://placenamehere.com/TXP/pnh_mf`): A Textpattern plug-in with support for rel-tag, rel-nofollow, rel-license, hCard, and hReview.
- **Sandbox theme for WordPress** (`www.plaintxt.org/themes/sandbox`): Authored by Scott Allan Wallick and Andy Skelton, this theme is hinted to likely be the basis for the next default WordPress theme. It's a skeleton template you use to base your templates on, and it utilizes hAtom for the structure of the document.
- **SimpleTags** (`www.broobles.com/scripts/simpletags/`): A rel-tag plug-in for WordPress.
- **Structured Blogging** (`http://structuredblogging.org`): This plug-in for WordPress and Moveable Type has support for creating structured content, including hCard and hCalendar.
- **Tails extension for Firefox** (`http://blog.codeeg.com/tails-firefox-extension-03`): Authored by Calvin Yu, this extension presents microformatted content in pages in a separate floating "window" within the browser window. It recognizes hCard, hCalendar, hReview, xFolk, geo, hAtom, and hResume.
- **WaSP Dreamweaver Microformats toolbar** (`www.webstandards.org/action/dwtf/microformats`): Authored by Drew McLellan, this form-based toolbar can be used for developing microformatted content for Dreamweaver. It supports hCalendar, hCard, rel-license, rel-tag, and XFN.
- **WebCards** (`www.whymicroformats.com/webcards/index.html`): This Firefox extension displays microformatted content contextually. It also features an in-page hCard creator. Authored by Andy Mitchell.
- **WordPress** (`www.wordpress.com`): A widely used, open source blogging and CMS platform, with built-in XFN support. It features templates, such as Sandbox, that use hAtom, and numerous plug-ins exist for most if not all microformats.
- **WP Movie Ratings** (`http://paulgoscicki.com/archives/2006/11/wp-movie-ratings-v14-released`): Authored by Paul Goscicki, this tool integrates with IMDb to allow pain-free reviewing of movies in hReview format.
- **X2V** (`http://suda.co.uk/projects/X2V`): A service (with open source XSLT code available) for converting hCard to vCard, and hCalendar to iCalendar. X2V enables hCard content to be downloaded for use in any desktop application that supports vCard, and hCalendar content to be downloaded for use in any application that supports iCalendar.
- **XFN Creator** (`http://gmpg.org/xfn/creator`): An easy-to-use, form-driven creator for XFN content, by Matt Mullenweg and Tantek Çelik.

Services

- **Alexa hCard Search** (`www.alexa.com/site/devcorner/samples?page=hcard`): A proof-of-concept search engine for hCard-formatted content in Alexa's web index.
- **Apple .Mac** (`www.apple.com/dotmac`): Publishes a user's address book information using the hCard format.
- **Google Advanced Search** (`www.google.com/advanced_search?hl=en`): This search form allows you to specify a license type when searching for pages.
- **Pingerati** (`http://pingerati.net`): A ping service for microformatted content. Be sure to notify Pingerati when publishing or modifying microformatted content.
- **Rubhub** (`http://rubhub.com`): A lookup engine for determining the relationships between people who author personal websites, based on their XFN markup.
- **Technorati Microformats Search** (`http://kitchen.technorati.com/search`): A ping-based search service for microformatted content, which indexes hCard, hCalendar, and hReview. Notify it of new or changed microformats based content by "pinging" Pingerati.
- **Technorati Tag Search** (`http://technorati.com/tag`): A search service from Technorati that indexes content marked up with the rel-tag microformat. Ping Technorati at `http://technorati.com/ping` to notify it of new tagged content.
- **Yahoo Creative Commons Search** (`http://search.yahoo.com/cc`): Searches for web-based content based on its licensing information, using rel-license to determine the use license of the content.

Publishers

- **Cambodian Yellow Pages** (`www.yellowpages-cambodia.com`): Publishes contact details for Cambodian businesses using hCard.
- **Cork'd** (`http://corkd.com`): A wine community site that uses hReview for marking up members' wine reviews, XFN for marking up relationships between members, and hCard for member profiles.
- **Emurse** (`http://emurse.com`): A service for building, hosting, and publishing resumes online using the hResume format.
- **Eventful** (`http://eventful.com`): Publishes details of user-submitted event details using hCalendar and contact details using hCard.
- **Flickr** (`http://flickr.com`): Supports geotagging of photos with the geo microformat and marks up users' profiles using hCard.
- **Last.fm** (`http://last.fm`): Publishes details of upcoming concerts using hCalendar.
- **LinkedIn** (`http://linkedin.com`): A professional online networking service that uses hResume for publishing resume details.

- **Meetup** (`http://meetup.com`): Publishes details of user-submitted event details using hCalendar and contact details using hCard.
- **Multimap** (`http://multimap.com`): Publishes the longitude and latitude of location-based search results using the geo microformat.
- **Revoo** (`http://revoo.com`): A review aggregation and publishing service that both aggregates reviews published on the Web in hReview format and publishes reviews in that format.
- **Upcoming** (`http://upcoming.org`): Publishes details of user-submitted upcoming events using hCalendar and contact details using hCard.
- **Yahoo Local** (`http://local.yahoo.com`): Publishes reviews of businesses in the United States, using hCard for contact details and hReview for the review itself.
- **Yahoo Tech** (`http://tech.yahoo.com`): Publishes "professional" reviews of technology, along with user-generated reviews, all published using hReview. Reviewer details are published using hCard.

Related organizations

- **Creative Commons** (`http://creativecommons.org`): This organization is devoted to encouraging more liberal licensing of copyrighted material.
- **Global Multimedia Protocols Group** (`http://gmpg.org`): This group is the publisher of XFN and began the early work that later formed the basis of microformats.
- **IETF** (`www.ietf.org`): The Internet Engineering Task Force (IETF) developed vCard (RFC 2426), iCalendar (RFC 2445), and hAtom (RFC 4287).

INDEX