Solutions & Examples for HTML5 Developers

HTML5 Cookbook

Christopher Schmitt
& Kyle Simpson

O'REILLY®

HTML5 Cookbook

With scores of practical recipes you can use in your projects right away, this cookbook helps you gain hands-on experience with HTML5's versatile collection of elements. You get clear solutions for handling issues with everything from markup semantics, web forms, and audio and video elements to related technologies such as geolocation and rich JavaScript APIs.

Each informative recipe includes sample code and a detailed discussion on why and how the solution works. Perfect for intermediate to advanced web and mobile web developers, this handy book lets you choose the HTML5 features that work for you—and helps you experiment with the rest.

- Test browsers for HTML5 support, and use techniques for applying unsupported features
- Discover how HTML5 makes web form implementation much simpler
- Overcome challenges for implementing native audio and video elements
- Learn techniques for using HTML5 with ARIA accessibility guidelines
- Explore examples that cover using geolocation data in your applications
- Draw images, use transparencies, add gradients and patterns, and more with <canvas>
- Bring HTML5 features to life with a variety of advanced JavaScript APIs

"A great next step in the search for more HTML5 knowledge. This book is as delicious as key lime pie."

—Jenn Lukas
Interactive Development Director of Happy Cog

Christopher Schmitt is principal of Heatvision.com, Inc., a new-media event, design and publishing company. He has worked with the Web since 1993, and is the author of *CSS Cookbook* (O'Reilly).

Kyle Simpson is a JavaScript systems architect from Austin, Texas. He focuses on JavaScript, web performance optimization, and "middle-end" application architecture. Kyle runs several open-source projects, including LABjs.

US $34.99 CAN $36.99
ISBN: 978-1-449-39679-4

Twitter: @oreillymedia
facebook.com/oreilly

O'REILLY®
oreilly.com

Praise for *HTML5 Cookbook*

"Written by community experts Emily Lewis, Mark Grabanski, Christina Ramey, Kimberly Blessing, Christopher Deutsch, Anitra Pavka, Kyle Simpson, and Christopher Schmitt, the *HTML5 Cookbook* provides the breadth and depth needed to use tomorrow's technology today."

— Estelle Weyl, author of *HTML5 & CSS3 for the Real World*

"There is so much for frontend designers to remember these days, it's hard to have it all memorized. The *HTML5 Cookbook* is perfect for all of us who know what we are looking for and need a quick and reliable way to find it."

— Chris Coyier, CSS-Tricks.com

"If you're ready to learn HTML5 that works today, the *HTML5 Cookbook* is the book to buy. There are some excellent books out there if you want exposition and details, but if you want to roll up your sleeves and get to work, buy this book."

— Ben Henick, Sitebuilder at-large

"The difficulty with cookbooks has always been getting the right balance between breadth and depth. I am impressed with the *HTML5 Cookbook*. Schmitt and Simpson have got this balance just right, providing enough depth on essential topics to give you what you need for implementing HTML5 features on your sites and apps, while also going further and exploring some interesting peripheral and nascent topics that you'll want to learn about soon, if not today. Covering semantics, video, audio, Canvas, progressive enhancement and backwards compatibility, forms, accessibility, geolocation and more, you're bound to get a lot out of this book whatever web disciplines you practice.

— Chris Mills, Open Standards Evangelist, Opera Software

"The *HTML5 Cookbook* is brimming with recipes to show you what you can really do with HTML5, plus a soupçon of JavaScript. Fill your studio with the aroma of drawings, animations, geolocation, audio/video, form fields, and semantic elements."

— Helen Oliver, Research Assistant,
Design Engineering Group at Imperial College London

"Great overview of the most important HTML5 technologies, with tons of code to match it! An everyday companion for your developing needs."

— Robert Nyman, Technical Evangelist, Mozilla

"The *HTML5 Cookbook* is not only a fantastic resource for creative problem solving with HTML5, it is also a great learning tool. There's no easier way to get familiar with a new (or updated) language than by combing through useful examples—something this book is just chock full of."

— Aaron Gustafson, author of *Adaptive Web Design: Crafting Rich Experiences with Progressive Enhancement*

HTML5 Cookbook

Christopher Schmitt and Kyle Simpson

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

HTML5 Cookbook
by Christopher Schmitt and Kyle Simpson

Copyright © 2012 Christopher Schmitt and Kyle Simpson. All rights reserved.
Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editors: Simon St. Laurent and Courtney Nash
Production Editor: Jasmine Perez
Copyeditor: Rachel Head
Proofreader: Linley Dolby
Indexer: Lucie Haskins
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Robert Romano

November 2011: First Edition.

Revision History for the First Edition:
2011-11-04 First release
See *http://oreilly.com/catalog/errata.csp?isbn=9781449396794* for release details.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. *HTML5 Cookbook*, the image of a common kestrel, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc., was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 978-1-449-39679-4

[LSI]

1320420716

Table of Contents

Preface

We know you want to learn about all the wonderful and exciting developments that come with HTML5, like web forms, `canvas`, and local storage. But we also know the importance of establishing a good foundation for advanced development. Let's first put HTML5 into a bit of context.

What Is HTML5?

HTML5 is a specification (see *http://dev.w3.org/html5/spec/*) under development by the World Wide Web Consortium (W3C). As we write this book, the HTML5 specification is officially a *Working Draft*, which means it may go through additional revisions before becoming a recommendation. The recommendation will then go through a formal approval process, resulting in a specific version of the markup language.

Meanwhile, independent from the W3C, the Web Hypertext Application Technology Working Group (WHATWG) also pursues development of the HTML specification (see *http://whatwg.org/html*).

Notice I didn't mention a version number. That's because the WHATWG recently decided to change tack and drop versioning entirely. A "living standard" is now how WHATWG defines HTML (see *http://blog.whatwg.org/html-is-the-new-html5*). This new development model means that HTML is defined according to how it's evolving, not as a version tied to features in a "snapshot" of time.

Feature Support, Not Browser Versions

What does this mean for us designers and developers? It could lead to a greater focus on implementing individual features, rather than a full specification: no more "this is an HTML5 site," but instead "this site features web sockets and geolocation."

Then again, some in the industry argue that designers and developers *need* stable specifications to refer to in order to validate and maintain their sites effectively. Plus, having a stable "what is true now" version makes authoring and teaching more manageable.

In the end, it could all just be fodder for yet another geek debate (my money's on Batman over Spider-Man and Peter Davison as the best Doctor), but we mention it mostly as a point of clarification that we have *two* development models and a reminder of the rather interesting politics involved in the specification processes.

Five Alive

Even though WHATWG sees "HTML" as a living document that needs no version number, in this book "HTML5" is our preferred term. Why is that? For the purposes of your daily design/development life, understanding and implementing features is what's important.

Since the whole point of the *Cookbook* series is to provide you with practical recipes you can use today, let's talk support *for* HTML5. Generally speaking, all of today's latest browsers support HTML5 to some degree or another. But, like its predecessors, HTML5 doesn't have that 100% browser support that we sadly suspect we'll always be dreaming of.

Enter JavaScript

While HTML5 markup has plenty of exciting new features, as you'll see in this book, it also involves—more than ever before—a host of related web technologies, many of which rely on rich JavaScript APIs to expose themselves to your web pages.

In an effort to give you a full taste of what HTML5 and these related technologies have to offer, we will not shy away from the JavaScript details. This book will, at times, take a very heavy JavaScript perspective, as we discuss some of those advanced functionalities that various HTML5 APIs make available to us.

If JavaScript is a scary or unfamiliar topic for you, now's the perfect time to brush up on those skills—if you're serious about using HTML5, you'll almost certainly want to get comfortable with leveraging at least some of what JavaScript has to offer.

It's also important to note that many of these APIs are still evolving, even as this book is being written, edited, revised, and published. Some are more complete than others, and thus are likely to be more stable. Other APIs are still in a state of frequent fluctuation, so you should keep that in mind as you decide how you will employ HTML5 technologies in your pages.

What's in This Book

While we may not provide a full list of all the features of HTML5 and browser support for them—depending on the recipe—we do discuss the support for each of the individual elements and implementations covered throughout this book (for tips on finding out what browsers currently support what parts of the specification, see Recipe 2.3, "Understanding Browser Support for HTML5").

We also cover workarounds to implement when browser support is patchy, and why you may or may not need them. This way, you can decide for yourself if a particular feature works for you, your client, or your employer.

And that's our value to you, dear reader. Using HTML5 isn't an absolute proposition. You don't have to use embedded content or web sockets to use the new structural elements, and vice versa. You don't even have to use the new structural elements to have a valid HTML5 document; you just need the Document Type Definition.

If you find yourself asking "what's a Document Type Definition?," start your adventure into HTML5 by checking out Recipe 1.1, "Specifying the DOCTYPE"!

Pick what works for you and leave the rest. Or, rather, *experiment* with the rest to see if you may find a practical implementation for yourself or your projects.

Our industry is like HTML in a lot of ways, and probably always will be: it's constantly changing.

And we, as the good web designers and developers that we are, have to continue staying on top of those latest changes and developments. We have to continue to educate our clients and employers about the benefits and compromises. We have to experiment with moving targets and constantly grow our skills.

Honestly, though, that all sounds pretty good to us. So, let's start coding some HTML5, shall we?

Audience

While it would probably suffice to say that this book is for any person interested in learning about HTML5, it was particularly designed and developed for web developers who want to transition from XHTML or HTML4 into new technologies.

The chapters toward the end of the book are geared more for developers who want to utilize some of the JavaScript APIs found in HTML5.

Assumptions This Book Makes

You don't want a rundown of the HTML5 specification. Rather, you want to make things work as they relate to your job. In each of this book's recipes, the Solution gives you the quick and dirty answer to the problem presented. Check out the Discussion for greater coverage.

You could start from the beginning of the book and make your way through it in a linear fashion without feeling lost. With one or two exceptions, the chapters' recipes are set up so that they build off of each other. One of the great things about the books in the Cookbook series, though, is that they're here to help you out when you find yourself with a specific problem—in this case, an HTML5-related problem. Simply crack open the book or the ebook on your tablet and find the right recipe for a practical, usable solution.

And one more thing: we assume that geolocation will forever be linked to HTML5. Even though we know, yes, that W3C Geolocation API Working Draft is separate from the HTML5 specification. That hasn't stopped people from writing blog posts, tutorials, and even books about putting geolocation into the same conversation as HTML5. And since it's such a great, serviceable API right now, we felt we had to include it in a book about HTML5 (see Chapter 8). If it troubles you, think of Geolocation and other technologies as "HTML5 and friends." However, this book probably isn't for you if you require hyper-technical hierarchies to be adhered to in order to obtain practical knowledge.

Conventions Used in This Book

The following typographical conventions are used in this book:

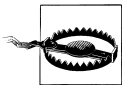
Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*HTML5 Cookbook* by Christopher Schmitt and Kyle Simpson (O'Reilly). Copyright 2012 Christopher Schmitt and Kyle Simpson, 978-1-449-39679-4."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://shop.oreilly.com/product/0636920016038.do

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

We can't see it, but time is an invisible dimension that surrounds us all.

Yet time is so very interesting in that it can be marked so well.

For instance, it can be marked by the due dates and deadlines of which a book of this sort is composed. So many, many deadlines.

When writing a book, for example, there is a deadline for the original manuscript.

Then there are the changes to the manuscript provided by many talented people like editors, technical editors, copyeditors, artists, designers, and so on.

They each give writers their deadlines, marking time.

Editorial deadline, technical edits deadline, copyedits deadline, art deadline, and so on.

Deadlines upon deadlines as time is.

Then there is another way time is marked.

The time away from loved ones.

After writing several books, I now know that writing a book is a Herculean task in that it keeps time away from other activities, like fun and family—something I wouldn't wish on anyone.

But then friends aren't just anyone, right?

Along with my co-lead author, Kyle Simpson, I congratulate and thank the contributors to the *HTML5 Cookbook*: Emily Lewis, Kimberly Blessing, Christina Huggins Ramey, Anitra Pavka, Marc Grabanski, and Christopher Deustch.

Thanks to our technical editors, Shelley Powers, Ben Henick, Dusty Jewett, Molly Holzschlag, and Helen Oliver, who diligently kept us on our toes.

Our copyeditor, Rachel Head, is a miracle worker, making our respective families proud by making them think we all are better writers than we probably are.

Many, many thanks to editors Simon St. Laurent and Courtney Nash from O'Reilly, who helped me stay sane during our time on this book.

Every once in a while I tried to count up how many total years of web development experience were brought to this project by all the talented writers and editors who contributed, but I got a bit too exhausted just thinking about it.

I'm not sure if all these talented, beautiful people knew what they signed up for when I first talked to them about the *HTML5 Cookbook*, but I'm happy with what we were able to make together. And I thank them for that.

Finally, thanks to my family and loved ones who have been there for me throughout my obsession with web design and development. Let's continue to share good times together.

Christopher Schmitt

http://christopherschmitt.com

http://twitter.com/@teleject

I would like to thank my wonderful and patient wife, Christen, for letting me be part of this project. I also want to thank my parents for their support and encouragement in all my various endeavors. Lastly, to my new son, Ethan, I hope someday you are happily coding along in HTML12 and you get to look back on this old HTML5 book with nostalgia. Just remember, I had to walk to school in the snow, uphill, both ways...

Kyle Simpson

http://blog.getify.com

http://twitter.com/@getify

Thank you to Christopher Schmitt for giving me the opportunity to contribute to this book. I'd also like to thank Rey Bango, who recommended me to Christopher and who has given me so many opportunities to write about what I love. Following the pimpage chain, I must thank Christian Heillman for recommending me to Rey in the first place and for always encouraging me to share what I know.

To my love, Jason, thank you for supporting me in everything I do and reminding me that new opportunities are never as scary as they first seem. Thank you, also, for reading everything I write and making sure I don't sound stupid. You make my life bigger and better than I ever dreamed.

To my other unofficial editors, Brian Arnold and Ian Pitts, thanks for taking the time over the months (and years) to read my work and offer critical feedback. You guys are not only brilliant developers, but you are great friends.

Thanks to my sister, Erin, for reminding me who I am and what is important in this world. And to my sista, Erin Shutes, thank you for 20 years of support through the best and worst of times.

To the readers of my blog, my social-network friends, and everyone I've met at conferences, thanks for being interested in what I have to share and for sharing back. You remind me every day why I love what I do.

Emily Lewis

http://emilylewisdesign.com

http://twitter.com/@emilylewis

I would like to thank my friend Christopher Schmitt for inviting me to contribute to this book; my students at Bryn Mawr College and my colleagues at Comcast Interactive Media for constantly challenging me; my mother and my great-aunt for their encouragement; and my cat, Punky, for her support.

Kimberly Blessing

http://www.obiwankimberly.com

http://twitter.com/@obiwankimberly

Thank you, Molly Holzschlag and Cameron Barrett, for encouraging me to share my geeky learnings.

(Of course) thank you, Christopher Schmitt, for convincing me to contribute to this book.

Thank you, Matt May, for your feedback as I considered what to write for this book.

I'd also like to thank all of the other wonderful people I've met at web conferences and networking events over the years. You are too many to name. Y'all inspire me to learn and laugh.

Thank you, Mom and Dad, for... well, everything.

I also want to thank our readers—*you*—for caring enough about your art to hone your skills. Your curiosity, knowledge, and empathy will advance the Web.

Anitra Pavka

http://anitrapavka.com

http://twitter.com/@apavka

A hearty thanks to Christopher Schmitt for continually providing me with opportunities for expanding my knowledge of web design. You've pushed me to develop my writing and coding skills far beyond my modest ambitions.

Thank you, my sweet husband, Paul Ramey, for providing support and patience while I research, code, and write. And thank you for providing welcome distractions when I've worked too long!

Christina Huggins Ramey

http://www.christinaramey.me

http://twitter.com/@fidlet

Thanks to Christopher Deutsch for taking the geolocation chapter content ideas and running with them.

Marc Grabanski

http://marcgrabanski.com

http://twitter.com/@1Marc

CHAPTER 1

Fundamental Syntax and Semantics

Emily Lewis

1.0 Introduction

This chapter is designed to get you up and running with HTML5 basics. Covering the fundamental recipes, from declaring the DOCTYPE and character set to dealing with ambiguities of the new HTML5 semantics, it helps lay the groundwork for the rest of the book.

1.1 Specifying the DOCTYPE

Problem

You want to create an HTML5 page.

Solution

Specify the HTML5 DOCTYPE at the very beginning of your page:

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML5, for Fun &amp; Profit</title>
    </head>
    <body>
    </body>
</html>
```

Note that the DOCTYPE is not case sensitive. Feel free to go CaMeL cAsE with the characters.

Discussion

The Document Type Definition, or DOCTYPE, tells browsers and validators what version of HTML the page is written in. Previous versions of HTML specified the version number, such as the DOCTYPE for XHTML 1.0 Strict:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

With HTML5, the version is dropped from the DOCTYPE. This allows HTML5 to be backward compatible in terms of syntax and hopefully makes the transition to HTML5 easier.

Let's say you have a site that is valid HTML 4.0, but you want to transition it to HTML5. All you have to do to make this a valid HTML5 site is make this DOCTYPE change.

Additionally, all browsers recognize the shortened DOCTYPE and render in strict standards mode.

There are some elements that have changed between HTML4 and HTML5, so you will need to watch for elements that have been removed or deprecated. For example, `center` might not *technically* validate as HTML5.

See Also

The W3C Working Draft discussion on differences between HTML4 and HTML5 includes DOCTYPE at *http://www.w3.org/TR/html5-diff/#doctype*.

1.2 Specifying the Character Set

Problem

You need to define the character encoding of your web page.

Solution

In your document `head`, add a `meta` declaration for the character set:

```
<meta charset="UTF-8" />
```

Discussion

The *character encoding* instructs browsers and validators what set of characters to use when rendering web pages. If you do not declare the character set in your HTML, browsers first try to determine the character set from your server's HTTP response headers (specifically, the `Content-Type` header).

The character set declared in the response headers is generally taken in preference over the character set specified in your document, so the headers are the preferred method of communicating this information. However, if you cannot control what headers your server sends, declaring the character set in your HTML document is the next best option.

If a character set is declared neither in the document nor in the response headers, the browser might choose one for you, and it *may* be the wrong one for your site's needs. This not only can cause issues with rendering, but also poses a security risk.

Several years ago, a cross-site scripting vulnerability was discovered at Google that demonstrated the importance of character encoding: *http://shiflett.org/blog/2005/dec/googles-xss-vulnerability*.

In previous versions of HTML, the character set needed to be declared with additional attributes and values:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

But, as with the DOCTYPE, HTML5 only needs the minimum information required by browsers. Again, this helps with backward compatibility and makes it easier for authors to implement.

Special characters

Unicode (UTF-8) is a versatile encoding that covers most web builders' needs. Sometimes, though, you need to include a character that is outside the UTF-8 encoding.

A great resource for character entities is at *http://www.digitalmediaminute.com/reference/entity/*. It includes the numeric, named, and Unicode references for many of the more common characters allowed in HTML.

You can specify such characters with Numeric Character References (NCRs) or as named entities in order to help browsers render them correctly. If you wanted a copyright symbol, for example, you could include it in your HTML as an NCR:

```
&#169;
```

or you could include it as a named entity:

```
&copy;
```

See Also

Mark Pilgrim's "Dive Into HTML5" discussion about character encoding at *http://diveintohtml5.info/semantics.html#encoding*.

1.3 Specifying the Language

Problem

You want to specify the primary language of your web page.

Solution

Add the `lang` attribute and appropriate value to your opening `html` element:

```
<html lang="en">
```

Discussion

Browsers, screen readers, and other user agents use the `lang` attribute to determine the language in which the content should be interpreted. The example above specifies English via the `en` value.

Declaring a document's primary language isn't a requirement for HTML5 (or any of the previous versions, for that matter). It is, however, a good practice for both usability and accessibility.

See Also

Mark Pilgrim's "Dive Into Accessibility" discussion about identifying your document language at *http://diveintoaccessibility.info/day_7_identifying_your_language.html*.

1.4 Optimizing <script>s and <link>s

Problem

You want to reference JavaScripts and include links to external CSS files in your web page as simply as possible.

Solution

Include `script` and `link` declarations, but *without* the `type` attribute:

```
<link rel="stylesheet" href="styles.css" />
<script src="scripts.js"></script>
```

Discussion

HTML5 requires only the minimum amount of information needed for user agents. In previous versions of HTML, both CSS `link`s and `script`s required the `type` attribute to indicate the language. If you forgot to include `type`, though, most browsers assumed the correct value.

HTML5 makes `type` *officially* optional, but still validates older documents that do include the attribute. This makes sense, as there is really only one standard scripting language and only one styling language for the Web in use today.

See Also

The W3C Working Draft discussion on differences between HTML4 and HTML5 includes changed attributes at *http://www.w3.org/TR/html5-diff/#changed-attributes*.

1.5 Adding Document Structure with HTML5's New Elements

Problem

You want to define your document structure with the new `header`, `footer`, `nav`, `aside`, `section`, and `article` elements.

Solution

Examine your content and document structure to determine which of the new elements work with your page:

`header`
: Is used to contain the headline(s) for a page and/or `section`. It can also contain supplemental information such as logos and navigational aids.

`footer`
: Contains information about a page and/or `section`, such as who wrote it, links to related information, and copyright statements.

`nav`
: Contains the *major* navigation links for a page and, while not a requirement, is often contained by `header`.

`aside`
: Contains information that is *related* to the surrounding content but also exists *independently*, such as a sidebar or pull-quotes.

`section`
: Is the most generic of the new structural elements, containing content that can be grouped thematically or is related.

`article`
: Is used for self-contained content that could be consumed independently of the page as a whole, such as a blog entry.

A simple blog structure, with a headline, navigation, a sidebar, blog posts, and a footer, could be marked up in HTML5 as:

```
<header>
    <h1><abbr title="Hypertext Markup Language">HTML</abbr>5, for Fun &amp;
```

```
        Profit</h1>
    <nav>
        <ul>
            <li><a href="/Archive/">Archive</a></li>
            <li><a href="/About/">About</a></li>
        </ul>
    </nav>
</header>
<article>
    <h2><code>nav</code> Isn't for <em>All</em> Links</h2>
    <p>Though the <code>nav</code> element often contains links, that doesn't mean
       that <em>all</em> links on a site need <code>nav</code>.</p>
</article>
<article>
    <h2>You've Got the <code>DOCTYPE</code>. Now What?</h2>
    <p>HTML5 isn't an all or nothing proposition. You can pick and choose what
       works best for you. So once you have the <code>DOCTYPE</code> in place, you
       should explore.</p>
</article>
<aside>
    <h2>HTML5 Elsewhere</h2>
    <p>Feed your HTML5 fix with resources from our partners:</p>
    <ul>
        <li><a href="http://lovinghtml5.com">Loving HTML5</a></li>
        <li><a href="http://semanticsally.com">Semantic Sally</a></li>
    </ul>
</aside>
<footer>
    <p>Copyright &copy; 2011 <a href="http://html5funprofit.com">HTM5, for Fun
       &amp; Profit</a>. All rights reserved.</p>
</footer>
```

And, with the right CSS and supporting HTML, this markup could render on the browser as shown in Figure 1-1.

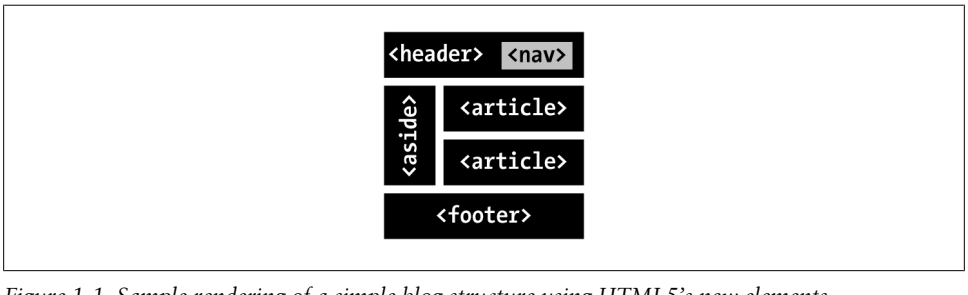


Figure 1-1. Sample rendering of a simple blog structure using HTML5's new elements

Discussion

These new structural elements were developed based on actual practices. A review of over a billion web pages (see *http://code.google.com/webstats/*) revealed the naming conventions markup authors were already using to structure and describe their content via `class` and `id`, including:

- `header`
- `footer`
- `nav`

The new elements in HTML5 simply attempt to reflect what authors are already doing.

Structural elements

Using these structural elements helps you make your markup more semantic, but they also help define the main landmarks in the document.

Consider how important it is for screen reader users and folks who navigate with the keyboard to be able to skip to different areas of the page, like the navigation. Previously, we've tried to provide such functionality via "skip links" and shortcuts (see *http://www.vdebolt.com/nmmug/flow.html*), but HTML5 establishes formal landmark elements that can be used instead. In fact, the Web Accessibility Initiative's Accessible Rich Internet Applications (WAI-ARIA) specification already addresses how to use HTML5 with ARIA landmark roles (see Chapter 2) for this very purpose.

That said, this is still a hypothetical. As of this writing, no browsers or assistive technologies reference the structural elements for any navigational purposes.

When to use structural elements

How do you know when to use these new elements? Again, focus on your content and consider the semantics of each element. For example, most web pages have an area considered a "header," composed of a logo, maybe the company name, and a tagline. And that certainly sounds like a good case for `header`. You may also have a `section` or `aside` in your document with its own set of headlines and navigation, which may also be contained in a `header`.

The same holds true for `footer`. While most pages have content appropriate for the new `footer` element, perhaps regarding the author, copyright, and related information, `section`s, `article`s, and `aside`s can feature the same information—and, in turn, can also include a `footer`.

Finally, consider `nav`. You may have many groups of links on your site, some of which are for navigation, while others are external.

`nav` is only appropriate for *major* site navigation, not search results links or a blogroll.

When to use elements

As you can see from the blog markup example, the new structural elements can replace many of the non-semantic container `div`s that you may have grown accustomed to using. But `div` still has a place at the HTML5 party.

If you ever need a containing element *strictly for style purposes*, `div` is the element to use. You don't want to use one of the new structural elements just to serve as a hook for your CSS. That is not what semantic markup is about.

Just remember to focus on your content and avoid *unnecessary* use of `div`, such as when another element is more semantic. For example, if you have a paragraph of text, use the `p` element, not `div`. Both give you block-level formatting, but `p` is more semantic for that particular purpose.

Styling structural elements

All of today's browsers render the content contained by these new elements. However, some browsers don't recognize them and, as such, treat them like inline elements. This default rendering can cause some serious problems with styling.

Fortunately, it doesn't take much to fix. The current cast of browsers simply needs to be told to treat the elements as block-level elements:

```
header, footer, nav, article, aside, section {
    display: block;
}
```

With that single CSS declaration, you can happily style the structural elements—well, almost. In versions of Internet Explorer (IE) prior to IE9 you have to add a bit of JavaScript to your `document` so IE recognizes the elements and allows you to style them:

```
<script>
    document.createElement('header');
    document.createElement('footer');
    document.createElement('nav');
    document.createElement('article');
    document.createElement('aside');
    document.createElement('section');
</script>
```

If you want cooperation from an earlier version of IE, any time you add a new HTML5 element to your document you'll need to add the corresponding `document.createElement`. See Chapter 2 for a more detailed discussion of using JavaScript with IE.

See Also

Script Junkie's article "Using HTML5's New Semantic Tags Today" at *http://msdn.microsoft.com/en-us/scriptjunkie/gg454786*.

1.6 Choosing Between <article> and <section>

Problem

You don't know whether `article` or `section` is the most appropriate element to use on your web page.

Solution

Focus on your content and the semantic definitions of `article` and `section` (refer back to Recipe 1.5).

<article>

`article` can be considered a specialized form of `section`. It is intended for content that could stand on its own, outside of all surrounding content, such as "syndicatable" content like blog posts.

`article` is suitable for other types of content, including:

- Video and accompanying transcript
- News articles
- Blog comments

Often the name of an article or a blog post is in the title of the URI. If that's the case with a page you are working on, that content should be wrapped in an `article` element.

In the code example in Recipe 1.5, we used `article` to contain the individual blog posts:

```
<article>
    <h2><code>nav</code> Isn't for <em>All</em> Links</h2>
    <p>Though the <code>nav</code> element often contains links, that doesn't mean
       that <em>all</em> links on a site need <code>nav</code>.</p>
</article>
<article>
    <h2>You've Got the <code>DOCTYPE</code>. Now What?</h2>
    <p>HTML5 isn't an all or nothing proposition. You can pick and choose what
       works best for you. So once you have the <code>DOCTYPE</code> in place, you
       should explore.</p>
</article>
```