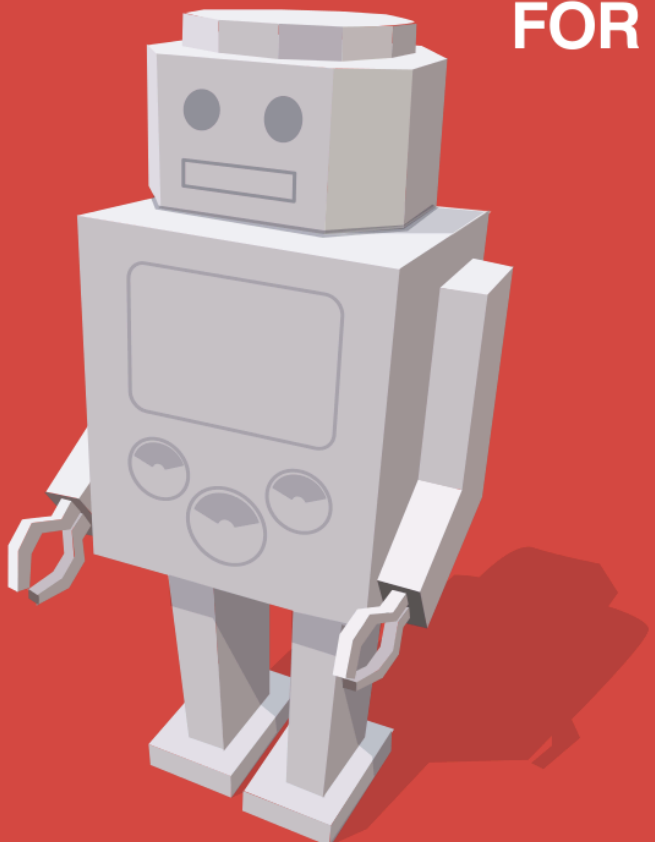
sitepoint

HTML5 & CSS3

FOR THE REAL WORLD

SECOND EDITION

BY ALEXIS GOLDSTEIN
LOUIS LAZARIS
& ESTELLE WEYL



POWERFUL HTML5 AND CSS3 TECHNIQUES YOU CAN USE TODAY!

Summary of Contents

HTML5 & CSS3 FOR THE REAL WORLD

BY ALEXIS GOLDSTEIN
LOUIS LAZARIS
ESTELLE WEYL

HTML5 & CSS3 for the Real World

by Alexis Goldstein, Louis Lazaris, and Estelle Weyl

Product Manager: Simon Mackie
English Editor: Kelly Steele
Technical Editor: Aurelio De Rosa
Cover Designer: Alex Walker
Printing History:
First Edition: May 2011
Second Edition: March 2015

Notice of Rights

Notice of Liability

Trademark Notice

Published by SitePoint Pty. Ltd.

48 Cambridge Street Collingwood
VIC Australia 3066
Web: www.sitepoint.com
Email: business@sitepoint.com

ISBN 978-0-9874674-8-5 (print)

ISBN 978-0-9874674-9-2 (ebook)
Printed and bound in the United States of America

About Alexis Goldstein

Alexis Goldstein first taught herself HTML while a high school student in the mid-1990s, and went on to get her degree in Computer Science from Columbia University. She runs her own software development and training company, aut faciam LLC. Before striking out on her own, Alexis spent seven years in technology on Wall Street, where she worked in both the cash equity and equity derivative spaces at three major firms, and learned to love daily code reviews. She taught dozens of classes to hundreds of students as a teacher and co-organizer of Girl Develop It, a group that conducts low-cost programming classes for women. You can find Alexis at her website, http://alexisgo.com/.

About Louis Lazaris

Louis Lazaris is a freelance web designer and front-end developer based in Toronto, Canada who has been involved in the web design industry since 2000, when table layouts and one-pixel GIFs dominated the industry. In recent years he has transitioned to embrace web standards while endeavoring to promote best practices that help both developers and their clients reach practical goals for their projects. Louis is Managing Editor for SitePoint's HTML/CSS content, blogs about front-end code on his website Impressive Webs (http://www.impressivewebs.com/), and curates Web Tools Weekly (http://webtoolsweekly.com/), a weekly newsletter focused on tools for front-end developers.

About Estelle Weyl

Estelle Weyl is a front-end engineer from San Francisco who has been developing standards-based accessible websites since 1999. Estelle began playing with CSS3 when the iPhone was released in 2007, and after eight years of web application development with CSS3 she knows (almost) every CSS3 quirk, and has vast experience implementing components of HTML5. She writes tutorials and detailed grids of CSS3 and HTML5 browser support at (http://www.standardista.com/). Estelle's passion is teaching web development, where you'll find her speaking on CSS3, HTML5, JavaScript, and mobile web development at conferences around the USA and the world. You can find all her presentations at http://estelle.github.io, and find her speaking engagements at http://lanyrd.com/estellevw.

About the Technical Editor

Aurelio De Rosa is a (full-stack) web and app developer with more than 5 years' experience programming for the web using HTML, CSS, Sass, JavaScript, and PHP. He's an expert on JavaScript and HTML5 APIs, but his interests include web security, accessibility, performance,

and SEO. He's also a regular writer for several networks, speaker, and author of the books *jQuery in Action, third edition* and *Instant jQuery Selectors*.

About SitePoint

SitePoint specializes in publishing fun, practical, and easy-to-understand content for web professionals. Visit http://www.sitepoint.com/ to access our blogs, books, newsletters, articles, and community forums. You'll find a stack of information on JavaScript, PHP, Ruby, mobile development, design, and more.

To my mother, who always encourages and believes in me.

And to my father, who taught me so much about living up to my full potential. I miss you every day.

To Cakes, the most brilliant person I know. Thank you for everything you do for me. I'm so grateful for each and every day with you.

—Alexis

To Melanie, the best cook in the world.

And to my parents, for funding the original course that got me into this unique industry.

—Louis

To Amie, for putting up with me, and to Spazzo and Puppers, for snuggling with me as I worked away.

—Estelle

Table of Contents

Preface

Welcome to *HTML5 & CSS3 for the Real World*. We're glad you've decided to join us on this journey of discovering some of the latest and the greatest in front-end website building technology.

If you've picked up a copy of this book, it's likely that you've dabbled to some degree in HTML and CSS. You might even be a bit of a seasoned pro in certain areas of markup, styling, or scripting, and now want to extend those skills further by dipping into the features and technologies associated with HTML5 and CSS3.

Learning a new task can be difficult. You may have limited time to invest in poring over the official documentation and specifications for these web-based languages. You also might be turned off by some of the overly technical books that work well as references but provide little in the way of real-world, practical examples.

To that end, our goal with this book is to help you learn through hands-on, practical instruction that will assist you to tackle the real-world problems you face in building websites today—with a specific focus on HTML5 and CSS3.

But this is more than just a step-by-step tutorial. Along the way, we'll provide plenty of theory and technical information to help fill in any gaps in your understanding—the whys and hows of these new technologies—while doing our best not to overwhelm you with the sheer volume of cool new stuff. So let's get started!

Who Should Read This Book

This book is aimed at web designers and front-end developers who want to learn about the latest generation of browser-based technologies. You should already have at least intermediate knowledge of HTML and CSS, as we won't be spending any time covering the basics of markup and styles. Instead, we'll focus on teaching you what new powers are available to you in the form of HTML5 and CSS3.

The final two chapters of this book cover some of the new JavaScript APIs that have come to be associated with HTML5. These chapters, of course, require some basic familiarity with JavaScript—but they're not critical to the rest of the book. If you're

unfamiliar with JavaScript, there's no harm in skipping over them for now, returning later when you're better acquainted with it.

Conventions Used

You'll notice that we've used certain typographic and layout styles throughout the book to signify different types of information. Look out for the following items:

Code Samples

Code in this book will be displayed using a fixed-width font, like so:

```
<h1>A Perfect Summer's Day</h1>
<p>It was a lovely day for a walk in the park. The birds
were singing and the kids were all back at school.</p>
```

If the code is to be found in the book's code archive, the name of the file will appear at the top of the program listing, like this:

example.css

```
.footer {
  background-color: #CCC;
  border-top: 1px solid #333;
}
```

If only part of the file is displayed, this is indicated by the word *excerpt*:

example.css *(excerpt)*

```
  border-top: 1px solid #333;
```

If additional code is to be inserted into an existing example, the new code will be displayed in bold:

```
function animate() {
  new_variable = "Hello";
}
```

Where existing code is required for context, rather than repeat all the code, a vertical ellipsis will be displayed:

```
function animate() {
  ⋮
  return new_variable;
}
```

Some lines of code are intended to be entered on one line, but we've had to wrap them because of page constraints. A ➥ indicates a line break that exists for formatting purposes only, and should be ignored:

```
URL.open("http://www.sitepoint.com/blogs/2015/05/28/user-style-she
➥ets-come-of-age/");
```

Tips, Notes, and Warnings

Hey, You!

Tips will give you helpful little pointers.

Ahem, Excuse Me ...

Notes are useful asides that are related—but not critical—to the topic at hand. Think of them as extra tidbits of information.

Make Sure You Always ...

... pay attention to these important points.

Watch Out!

Warnings will highlight any gotchas that are likely to trip you up along the way.

Supplementary Materials

http://www.learnable.com/books/htmlcss2/

The book's website, which contains links, updates, resources, and more.

https://github.com/spbooks/htmlcss2/
: The downloadable code archive for this book.

http://community.sitepoint.com/
: SitePoint's forums, for help on any tricky web problems.

books@sitepoint.com
: Our email address, should you need to contact us for support, to report a problem, or for any other reason.

Acknowledgments

We'd like to offer special thanks to the following members of the SitePoint and Learnable community who made valuable contributions to this edition of the book:

Martin Ansdell-Smith, Ilya Bodrov, Jacob Christiansen, Ethan Glass, Gerard Konars, Dityo Nurasto, Thom Parkin, Guilherme Pereira, Jason Rogers, Bernard Savonet, and Julian Tancredi.

Alexis Goldstein

Thank you to Simon Mackie and Aurelio DeRosa. Simon, you always kept us on track and helped to successfully wrangle three co-authors, no small feat. And Aurelio, your incredible attention to detail, impressive technical expertise and catching of errors has made this book so much better than it would have been without your immense contributions. Thank you to my co-authors, Louis and Estelle, who never failed to impress me with their deep knowledge, vast experience, and uncanny ability to find bugs in the latest browsers. A special thank you to Estelle for the encouragement, for which I am deeply grateful.

Louis Lazaris

Thank you to my wife for putting up with my odd work hours while I took part in this great project. Thanks to my talented co-authors, Estelle and Alexis, for gracing me with the privilege of having my name alongside theirs, and, of course, to our expert reviewer Aurelio De Rosa for always challenging me with his great technical insight. And special thanks to the talented staff at SitePoint for their super-professional handling of this project and everything that goes along with such an endeavor.

Estelle Weyl

Thank you to the entire open source community. With the option to "view source," I have learned from every developer who opted for markup rather than plugins. I would especially like to thank Jen Mei Wu and Sandi Watkins, who helped point me in the right direction when I began my career. And thank you to my developer friends at Opera, Mozilla, and Google for creating awesome browsers and even better developer tools, providing us with the opportunity to not just play with HTML5 and CSS3, but also to write this book.

Want to Take Your Learning Further?

Thanks for buying this book—we appreciate your support. Do you want to continue learning? You can now gain unlimited access to courses and ALL SitePoint books at Learnable for one low price. Enroll now and start learning today! Join Learnable and you'll stay ahead of the newest technology trends: http://www.learnable.com.

Chapter 1

Introducing HTML5 and CSS3

This chapter gives a basic overview of how the web development industry has evolved and why HTML5 and CSS3 are so important to modern websites and web apps. It will show how using these technologies will be invaluable to your career as a web professional.

Of course, if you'd prefer to just get into the meat of the project that we'll be building, and start learning how to use all the new bells and whistles that HTML5 and CSS3 bring to the table, you can always skip ahead to Chapter 2 and come back later.

What is HTML5?

What we understand today as HTML5 has had a relatively turbulent history. You probably already know that HTML is the predominant markup language used to describe content, or data, on the World Wide Web (another lesser-used markup language is XML). HTML5 is the latest iteration of the HTML5 language and includes new features, improvements to existing features, and JavaScript APIs.

That said, HTML5 is not a reformulation of previous versions of the language—it includes all valid elements from both HTML4 and XHTML 1.0. Furthermore, it's

been designed with some principles in mind to ensure it works on just about every platform, is compatible with older browsers, and handles errors gracefully. A summary of the design principles that guided the creation of HTML5 can be found on the W3C's HTML Design Principles page.[1]

First and foremost, HTML5 includes redefinitions of existing markup elements in addition to new elements that allow web designers to be more expressive in describing the content of their pages. Why litter your page with `div` elements when you can use `article`, `section`, `header`, `footer`, and so on?

The term "HTML5" has also been used to refer to a number of other new technologies and APIs. Some of these include drawing with the `canvas` element, offline storage, the new `video` and `audio` elements, drag-and-drop functionality, Microdata, and embedded fonts. In this book, we'll be covering a number of those technologies, and more.

Application Programming Interface

API stands for Application Programming Interface. Think of an API in the same way you think of a graphical user interface or GUI—except that instead of being an interface for humans, it's an interface for your code. An API provides your code with a set of "buttons" (predefined methods) that it can press to elicit the desired behavior from the system, software library, or browser.

API-based commands are a way of abstracting the more complex workings that are done in the background (or sometimes by third-party software). Some of the HTML5-related APIs will be introduced and discussed in later sections of this book.

Overall, you shouldn't be intimidated if you've had little experience with JavaScript or other APIs. While it would certainly be beneficial to have some experience with JavaScript or other languages, it isn't mandatory. Whatever the case, we'll walk you through the scripting parts of our book gradually, ensuring that you're not left scratching your head!

At the time of writing, it's been a good 5-plus years since HTML5 has had wide use in terms of the semantic elements and the various APIs. So it's no longer correct to categorize HTML5 as a "new" set of technologies—but it is still maturing and there

[1] http://www.w3.org/TR/html-design-principles/

are ongoing issues that continue to be addressed (such as bugs in browsers, and inconsistent support across browsers and platforms).

It should also be noted that some technologies were never part of HTML5 (such as CSS3 and WOFF), yet have at times been lumped in under the same label. This has instigated the use of broad, all-encompassing expressions such as "HTML5 and related technologies." In the interest of brevity—and also at the risk of inciting heated arguments—we'll generally refer to these technologies collectively as "HTML5."

How did we get here?

The web development industry has evolved significantly in a relatively short time period. In the late 1990s, a website that included images and an eye-catching design was considered top of the line in terms of web content and presentation.

Today, the landscape is quite different. Simple performance-driven, Ajax-based websites (usually differentiated as "web apps") that rely on client-side scripting for critical functionality are becoming more and more common. Websites today often resemble standalone software applications, and an increasing number of developers are viewing them as such.

Along the way, web markup has evolved. HTML4 eventually gave way to XHTML, which is really just HTML4 with strict XML-style syntax. HTML5 has taken over as the most-used version of markup, and we now rarely, if ever, see new projects built with HTML4 or XHTML.

HTML5 originally began as two different specifications: Web Forms 2.0[2] and Web Apps 1.0.[3] Both were a result of the changed web landscape and the need for faster and more efficient maintainable web applications. Forms and app-like functionality are at the heart of web apps, so this was the natural direction for the HTML5 spec to take. Eventually, the two specs were merged to form what we now call HTML5.

For a short time, there was discussion about the production of XHTML 2.0,[4] but that project has long since been abandoned to allow focus on the much more practical HTML5.

[2] http://www.w3.org/TR/web-forms-2/
[3] https://whatwg.org/specs/web-apps/2005-09-01/
[4] http://www.w3.org/TR/xhtml2/

Would the real HTML5 please stand up?

Because the HTML5 specification is being developed by two different bodies (the WHATWG and the W3C), there are two versions of the spec. The W3C (or World Wide Web Consortium) you're probably familiar with: it's the organization that maintains the original HTML and CSS specifications, as well as a host of other web-related standards such as SVG (Scalable Vector Graphics) and WCAG (Web Content Accessibility Guidelines).

The WHATWG (aka the Web Hypertext Application Technology Working Group), on the other hand, was formed by a group of people from Apple, Mozilla, and Opera after a 2004 W3C meeting left them disheartened. They felt that the W3C was ignoring the needs of browser makers and users by focusing on XHTML 2.0, instead of working on a backwards-compatible HTML standard. So they went off on their own and developed the Web Apps and Web Forms specifications that we've discussed, which were then merged into a spec they called HTML5. On seeing this, the W3C eventually gave in and created its own HTML5 specification based on the WHATWG's spec.

This can seem a little confusing. Yes, there are some politics behind the scenes that we, as designers and developers, have no control over. But should it worry us that there are two versions of the spec? In short, no.

The WHATWG's version of the specification can be found at http://www.whatwg.org/html/, and in January 2011 was renamed "HTML" (dropping the "5"). It's now called a "living standard,"[5] meaning that it will be in constant development and will no longer be referred to using incrementing version numbers.

The WHATWG version contains information covering HTML-only features, including what's new in HTML5. Additionally, there are separate specifications being developed by WHATWG that cover the related technologies. These specifications include Microdata, Canvas 2D Context, Web Workers, Web Storage, and others.

[5] http://blog.whatwg.org/html-is-the-new-html5

The W3C's version of the spec can be found at http://www.w3.org/html/wg/drafts/html/master/, and the separate specifications for the other technologies can be accessed through http://dev.w3.org/html5/.[6]

So what's the difference between the W3C spec and that of WHATWG? Besides the name ("Living Standard" versus "HTML5.1"), the WHATWG version is a little more informal and experimental (and, some might argue, more forward-thinking). But in most places they're identical, so either one can be used as a basis for studying new HTML5 elements and related technologies.[7]

Why should I care about HTML5?

As mentioned, at the core of HTML5 are a number of new semantic elements, as well as several related technologies and APIs. These additions and changes to the language have been introduced with the goal of allowing developers to build web pages that are easier to code, use, and access.

These new semantic elements, along with other standards such as WAI-ARIA and Microdata (which we cover in Appendix B and Appendix C respectively), help to make our documents more accessible to both humans and machines—resulting in benefits for both accessibility and search engine optimization.

The semantic elements, in particular, have been designed with the dynamic Web in mind, with a particular focus on making pages more accessible and modular. We'll go into more detail on this in later chapters.

Finally, the APIs associated with HTML5 help improve on a number of techniques that web developers have been using for years. Many common tasks are now simplified, putting more power in developers' hands. Furthermore, the introduction of HTML5 audio and video means that there will be less dependence on third-party software and plugins when publishing rich media content on the Web.

---

[6] Technically, the W3C's version has now been upgraded to a new version: "HTML5.1" [http://www.w3.org/TR/html51/]. For simplicity we'll continue to refer to both versions as "HTML5". In addition, the W3C's website has a wiki page dedicated to something called "HTML.next [http://www.w3.org/wiki/HTML/next]", which discusses some far-future features of HTML that we won't cover in this book.

[7] There's a document published by the W3C [http://www.w3.org/wiki/HTML/W3C-WHATWG-Differences] that details many of the differences between the two specs, but most of the differences aren't very relevant or useful.

Overall, there are good reasons to start looking into HTML5's new features and APIs, and we'll discuss more of those reasons as we go through this book.

What is CSS3?

Another separate—but no less important—part of creating web pages is Cascading Style Sheets (CSS). As you probably know, CSS is a style language that describes how HTML markup is presented to the user. CSS3 is the latest version of the CSS specification.

CSS3 contains just about everything that's included in CSS2.1, the previous version of the spec. It also adds new features to help developers solve a number of presentation-related problems without resorting to scripting plugins or extra images.

New features in CSS3 include support for additional selectors, drop shadows, rounded corners, updated layout features, animation, transparency, and much more.

CSS3 is distinct from HTML5. In this publication, we'll be using the term CSS3 to refer to the current level of the CSS specification, with a particular focus on what's been added since CSS2.1. Thus, CSS3 is separate from HTML5 and its related APIs.

One final point should be made here regarding CSS and the current "version 3" label. Although this does seem to imply that there will one day be a "CSS4," Tab Atkins, a member of the CSS Working Group, has noted that there are no plans for it.[8] Instead, as he explains, the specification has been divided into separate modules, each with its own version number. So you might see something like "CSS Color Module Level 4"[9]—but that does not refer to "CSS4." No matter what level an individual module is at, it will still technically be under the umbrella of "CSS3," or better yet, simply "CSS." For the purposes of this book, we'll still refer to it as "CSS3," but just understand that this is likely to be the last version number for the language as a whole.

[8] http://www.xanthir.com/b4Ko0
[9] http://dev.w3.org/csswg/css-color/

Why should I care about CSS3?

Later in this book, we'll look in greater detail at many of the new features in CSS. In the meantime, we'll give you a taste of why CSS3's new techniques are so exciting to web designers.

Some design techniques find their way into almost every project. Drop shadows, gradients, and rounded corners are three good examples. We see them everywhere. When used appropriately, and in harmony with a site's overall theme and purpose, these enhancements can make a design flourish. Perhaps you're thinking: we've been creating these design elements using CSS for years now. But have we?

In the past, in order to create gradients, shadows, and rounded corners, web designers have had to resort to a number of tricky techniques. Sometimes extra HTML elements were required. In cases where the HTML is kept fairly clean, scripting hacks were required. In the case of gradients, the use of extra images was inevitable. We put up with these workarounds, because there was no other way of accomplishing those designs. CSS3 allows you to include these and other design elements in a forward-thinking manner that leads to so many benefits: cleaner markup, maintainable code, fewer extraneous images, and faster-loading pages.

A Short History on Vendor Prefixes

Ever since experimental features in CSS3 have begun to be introduced, developers have had to use prefixes in their CSS to target those features in various browsers. Browsers add vendor prefixes to features that might still be experimental in the specification (that is, they're not very far along in the standards process).[10] For example, at one time it was common to see something like this for a simple CSS transition:

```
a {
    color: #3381d6;
    -webkit-transition: color 0.4s ease;
    -moz-transition: color 0.4s ease;
```

[10] For more info, see: http://www.sitepoint.com/web-foundations/vendor-specific-properties/

```
    -o-transition: color 0.4s ease;
    transition: color 0.4s ease;
}
```

This would seem counterproductive to what was just mentioned, namely that CSS3 makes the code cleaner and easier to maintain. Fortunately, many prefixes are no longer needed. Additionally, we highly recommend that developers use a tool that will add prefixing automatically to your CSS.

One such tool is called Autoprefixer.[11] Autoprefixer can be included as part of your Grunt[12] workflow to post-process your CSS. With this, you need to include only the standard version of any CSS feature, and Autoprefixer will look through the Can I use... database[13] to determine if any vendor prefixes are needed. It will then build your CSS automatically, with all necessary prefixes. You also have the option to manually process your CSS using an online tool such as pleeease.[14] Whatever the case, in many places in this book we will include vendor prefixes, however be sure to use an online resource for up-to-date information on which features still require prefixes.

What do we mean by "the Real World"?

In the real world, we create web applications and we update them, fine-tune them, test them for potential performance problems, and continually tweak their design, layout, and content.

In other words, in the real world, we don't write code that we have no intention of revisiting. We write code using the most reliable, maintainable, and effective methods available, with every intention of returning to work on that code again to make any necessary improvements or alterations. This is evident not only in websites and web apps that we build and maintain as personal projects, but also in those we create and maintain for our clients.

We need to continually search out new and better ways to write our code. HTML5 and CSS3 are a big step in that direction.

[11] https://github.com/postcss/autoprefixer
[12] http://gruntjs.com/
[13] http://caniuse.com/
[14] http://pleeease.io/play/

The Current Browser Market

Although HTML5 is still in development, presenting significant changes in the way content is marked up, it's worth noting that those changes won't cause older browsers to choke, nor result in layout problems or page errors.

What this means is that you could take any old project containing valid HTML4 or XHTML markup, change the doctype to HTML5 (which we'll cover in Chapter 2), and the page will appear in the browser the same as it did before. The changes and additions in HTML5 have been implemented into the language in such a way as to ensure backwards-compatibility with older browsers—even older versions of Internet Explorer! Of course, this is no guarantee that the new features will work, it simply means they won't break your pages or cause any visible problems.

Even with regards to the more complex new features (for example, the APIs), developers have come up with various solutions to provide the equivalent experience to non-supporting browsers, all while embracing the exciting new possibilities offered by HTML5 and CSS3. Sometimes this is as simple as providing fallback content, such as a Flash video player to browsers without native video support. At other times, though, it's been necessary to use scripting to mimic support for new features.

These "gap-filling" techniques are referred to as **polyfills**. Relying on scripts to emulate native features isn't always the best approach when building high-performance web apps, but it's a necessary growing pain as we evolve to include new enhancements and features, such as the ones we'll be discussing in this book. Fortunately, as of writing, older browsers such as Internet Explorer 6 through 9 that fail to support many of the new features in HTML5 and CSS3, are used by less than 10% of web visitors today.[15] More and more people are using what has been termed evergreen browsers;[16] that is, browsers that automatically update. This means that new features will be functional to a larger audience, and eventually to all, as older browser shares wane.

[15] http://gs.statcounter.com/#browser_version-ww-monthly-201502-201502-bar
[16] http://tomdale.net/2013/05/evergreen-browsers/

In this book we may occasionally recommend fallback options or polyfills to plug the gaps in browser incompatibilities; we'll also try to do our best in warning you of potential drawbacks and pitfalls associated with using these options.

Of course, it's worth noting that sometimes no fallbacks or polyfills are required at all; for example, when using CSS3 to create rounded corners on boxes in your design, there's often no harm in users of really old browsers seeing square boxes instead. The functionality of the site has no degradation, and those users will be none the wiser about what they're missing.

As we progress through the lessons and introduce new subjects, if you plan on using one of these in a project we strongly recommend that you consult a browser-support reference such as the aforementioned Can I use...[17] That way, you'll know how and whether to provide fallbacks or polyfills. Where necessary, we'll occasionally discuss ways you can ensure that non-supporting browsers have an acceptable experience, but the good news is that it's becoming less and less of an issue as time goes on.

The Growing Mobile Market

Another compelling reason to start learning and using HTML5 and CSS3 today is the exploding mobile market. According to one source, in 2009 less than 1% of all web usage was on mobile devices and tablets.[18] By the middle of 2014, that number had risen to more than 35%![19] That's an astounding growth rate in a little more than five years. So what does this mean for those learning HTML5 and CSS3?

HTML5, CSS3, and related cutting-edge technologies are very well supported in many mobile web browsers. For example, mobile Safari on iOS devices such as the iPhone and iPad, Opera Mobile, Android Browser, and UC Browser all provide strong levels of HTML5 and CSS3 support. New features and technologies supported by some of those browsers include CSS3 animations, CSS flexbox, the Canvas API, Web Storage, SVG, Offline Web Apps, and more.

In fact, some of the new technologies we'll be introducing in this book have been specifically designed with mobile devices in mind. Technologies such as Offline Web Apps and Web Storage have been designed, in part, because of the growing

[17] http://caniuse.com/

[18] http://gs.statcounter.com/#desktop+mobile+tablet-comparison-ww-monthly-200901-200901-bar

[19] http://gs.statcounter.com/#desktop+mobile+tablet-comparison-ww-monthly-201408-201408-bar

number of people accessing web pages with mobile devices. Such devices can often have limitations with online data usage, and thus benefit greatly from the ability to access web applications offline.

We'll be touching on those subjects in Chapter 11, as well as others throughout the course of the book, providing the tools you'll need to create web pages for a variety of devices and platforms.

On to the Real Stuff

It's unrealistic to push ahead into new technologies and expect to author pages and apps for only one level of browser. In the real world, and in a world where we desire HTML5 and CSS3 to make further inroads, we need to be prepared to develop pages that work across a varied landscape. That landscape includes modern browsers, any remaining older versions of Internet Explorer, and an exploding market of mobile devices.

Yes, in some ways, supplying a different set of instructions for different user agents resembles the early days of the Web with its messy browser sniffing and code forking. But this time around, the new code is much more future-proof: when older browsers fall out of general use, all you need to do is remove any fallbacks and polyfills, leaving only the code base that's aimed at modern browsers.

HTML5 and CSS3 are the leading technologies that have ushered in a much more exciting world of web page authoring. Because all modern browsers provide excellent levels of support for a number of HTML5 and CSS3 features, creating powerful and simple-to-maintain future-proof web pages is easier than ever before.

So, enough about the "why," let's start digging into the "how"!

Chapter 2

Markup, HTML5 Style

Now that we've given you a bit of a history primer, along with some compelling reasons to learn HTML5 and start using it in your projects today, it's time to introduce you to the sample site that we'll be progressively building in this book.

After we briefly cover what we'll be building, we'll discuss some HTML5 syntax basics, along with some suggestions for best-practice coding. We'll follow that with some important info on cross-browser compatibility, and the basics of page structure in HTML5. Lastly, we'll introduce some specific HTML5 elements and see how they'll fit into our layout.

So let's get into it!

Introducing *The HTML5 Herald*

For the purpose of this book, we've put together a sample website project that we'll be building from scratch. The website is already built—you can check it out now at thehtml5herald.com.[1] It's an old-time newspaper-style website called *The HTML5 Herald*. The home page of the site contains some media in the form of video, images,

[1] http://thehtml5herald.com/