THE EXPERT'S VOICE® IN WEB DEVELOPMENT

# Pro HTML5 Programming

## Powerful APIs for Richer Internet Application Development

*Use HTML5 to create cutting-edge web applications*

Peter Lubbers, Brian Albers,
and Frank Salim

*Foreword by Paul Irish, Google*

Apress®

# Pro HTML5 Programming

## Powerful APIs for Richer Internet Application Development

**PETER LUBBERS**
**BRIAN ALBERS**
**FRANK SALIM**

Apress®

**Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development**



ISBN-13 (pbk): 978-1-4302-2790-8

ISBN-13 (electronic): 978-1-4302-2791-5

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



President and Publisher: Paul Manning
Lead Editor: Clay Andres
Development Editor: Matthew Moodie
Technical Reviewer: Paul Haine
Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jonathan Hassell, Michelle Lowman, Matthew Moodie, Duncan Parkes, Jeffrey Pepper, Frank Pohlmann, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Coordinating Editor: Laurin Becker
Copy Editors: Heather Lang, Andy Rosenthal, Nancy Sixsmith
Compositor: Kimberly Burton
Indexer: Julie Grady
Artist: April Milne
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `www.springeronline.com`.

For information on translations, please e-mail `rights@apress.com`, or visit `www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at `www.apress.com/info/bulksales`.



The source code for this book is available to readers at `www.prohtml5.com` and also at `www.apress.com`.

*For my beautiful wife, Vittoria,*
*and for my sons—Sean and Rocky. I am so proud of you!*
*—Peter Lubbers*

*For John. You make it all worthwhile.*
*—Brian Albers*

*For people who still read books.*
*—Frank Salim*

# Contents at a Glance

# Contents

# Foreword

In June 2004, representatives from the semantic web community, major browser vendors, and the W3C met in San Jose, California to discuss the standards body's response to the rise of web applications. At the end of the second day, a vote was held to decide whether the W3C should augment HTML and the DOM to address the new requirements of web applications. Minutes from the event record the anonymous and curious result, "8 for, 14 against."
This schism lead to a divergence in effort: two days later, the WHATWG was formed from the major browser vendors to solve emerging issues. Meanwhile, the W3C pushed forward with the XHTML2 specification, only to drop it five years later to focus on an aligned HTML5 effort with the WHATWG.

Now, six years since, we stand to benefit greatly from the passionate minds that have designed HTML5. The features both codify de facto standards that have been in use for years and lay the groundwork for next-generation web applications. Putting them to use means a more engaging and responsive web experience for your users and, oftentimes, far less code for you.

In this book, you'll find a well-designed learning curve bringing you up to speed on the features within HTML5 and its associated specifications. You'll learn best practices of feature detection, appropriate use cases, and a lot of the whys that you won't find in the specifications. The code examples are not plain, trivial uses of each API but instead lead you through building actual web applications. I hope this book is able to serve you well, and I hope you'll be as excited about the next generation of the web as I am.

Paul Irish
Google and jQuery Dev Relations and Lead Developer for Modernizr

# About the Authors

**Peter Lubbers** is the director of documentation and training at Kaazing. An HTML5 and WebSockets enthusiast, Peter is a frequent speaker at international events and teaches HTML5 training courses worldwide. Prior to joining Kaazing, Peter worked for almost ten years as an information architect at Oracle, where he wrote award-winning books and developed patent-pending software solutions. A native of the Netherlands, Peter served as a special forces commando in the Royal Dutch Green Berets. Peter lives on the edge of the Tahoe National Forest in California, and in his spare time, he runs ultramarathons. You can follow Peter on Twitter (`@peterlubbers`).

**Brian Albers** is the vice president of research and development at Kaazing. His career in web development spans a decade and a half, including his most recent position as a senior development manager at Oracle. Brian is a regular speaker at conferences, such as Web 2.0 Expo, AJAXWorld Expo, and JavaOne, where he focuses on Web and user interface technologies. A native Texan and current California resident, Brian spends as much time as possible escaping to Hawaii. When he cannot relax on the beach, Brian can be found frequenting a variety of virtual worlds in his spare time.

**Frank Salim** is one of the original engineers at Kaazing who helped craft the WebSockets gateway and client strategy. Frank is a San Diego native currently residing in Mountain View, California. He holds a degree in computer science from Pomona College. When he is not programming, Frank enjoys reading, painting, and inline skating.

# About the Technical Reviewer

■ **Paul Haine** is a client-side developer currently working in London for the Guardian newspaper. He is the author of *HTML Mastery: Semantics, Standards, and Styling* (friends of ED, 2006) and runs a personal web site at `www.joeblade.com`.

# Acknowledgments

I'd like to thank my wife Vittoria, for her love and patience, and my talented sons Sean and Rocky—reach for the stars, boys!

Thanks to my parents, Herman and Elisabeth, my sister Alice, and my brother David for always believing in me, and to my late grandmother Gebbechien whose courageous acts during the Nazi occupation of Holland were a great lesson to our family.

To my coauthors, the never-tiring Brian and the code-generating human Frank, it has been an honor to work with both of you.

Thanks also to Clay at Apress for all your support, and, finally, thanks to Jonas and John at Kaazing for pushing us to write a "real" book—an "unofficial e-book" would still be just a figment of our imagination, I am sure!

—Peter Lubbers

To my parents, Ken and Patty Albers, I offer my deepest love and appreciation for the sacrifices you made to bring me so many opportunities. Without your encouragement and the values you instilled in me, I would never have completed this or any other major life journey. You've been guiding me every step of the way.

To John, my deepest thanks for your patience every time that extra hour of work stretched to two, three, or more. You amaze and inspire me.

To Pitch, Bonnie, and Penelope, a scritch on the chin and a promise that dinner won't be so late any more. To the cats who came before, your purrs stay with me.

To my coworkers at Kaazing, much appreciation for the chance to work with the best and the brightest.

And a special thanks to the editorial staff at Apress for first believing that the time was right for an HTML5 book and then for having patience with us while we attempted to document a rapidly moving target.

—Brian Albers

I'd like to thank my parents, Mary and Sabri, who are responsible for my existence and without whom this book would literally not be possible.

—Frank Salim

# Introduction

HTML5 is brand new. Indeed, it isn't even finished yet. And if you listen to some ornery pundits, they'll tell you that HTML5 won't be ready for ten years or more!

Why, then, would anyone think that now's the time for a book called *Pro HTML5 Programming*? That's easy. Because for anyone who's looking for an extra edge to make your web application stand above the rest, the time for HTML5 is right now. The authors of this book have been working with, developing, and teaching HTML5 technologies for more than two years now and can claim with certainty that adoption of the new standards is accelerating at dizzying speeds. Even over the course of writing this book, we've been forced to continually update our browser support matrix and reevaluate our assumptions about what is ready to use.

Most users don't really understand the power that's available in the browsers they are now using. Yes, they might notice some minor interface enhancement after their favorite browser has automatically updated. But they might have no idea that this new browser version just introduced a free-form drawing canvas or real-time network communication, or any number of other potential upgrades.

With this book, we aim to help you unlock the power HTML5.

## Who This Book Is For

The content in this book is intended for the experienced web application developer who is familiar with JavaScript programming. In other words, we won't be covering the basics of web development in this text. There are many existing resources to get you up to speed in the fundamentals of web programming. That said, if you see yourself in any of the following bullets, this book will likely provide you with useful insight and information you are looking for:

- You sometimes find yourself thinking, "If only my browser could. . ."
- You find yourself using page source and developer tools to dissect a particularly impressive website.
- You enjoy reading the release notes of the latest browser updates to find out what's new.
- You are looking for ways to optimize or streamline your applications.
- You are willing to tailor your website to provide the best possible experience for users on relatively recent browsers.

If any of these apply to you, this book may be a good fit with your interests.

While we take care to point out the limitations of browser support where appropriate, our aim is not to give you elaborate workarounds to make your HTML5 application run seamlessly on a ten-year-old browser. Experience has shown that the workarounds and baseline browser support are evolving so rapidly that a book like this is not the best vehicle for such information. Instead, we will focus on the

specification of HTML5 and how to use it. Detailed workarounds can be found on the Internet and will become less necessary over time.

## An Overview of This Book

The eleven chapters of this book cover a selection of popular, useful, and powerful HTML5 APIs. In some cases, we have built on the capabilities introduced in earlier chapters to provide you with richer demonstrations.

Chapter 1, "Introduction to HTML5," starts off with the background of past and current versions of the HTML specification. The new high-level semantic tags are presented, along with basic changes and the rationale behind all the recent developments in HTML5. It's good to know this terrain.

Chapter 2, "Using the HTML5 Canvas API", and Chapter 3, "Working with HTML5 Audio and Video," describe the new visual and media elements. In these chapters, the focus is on finding simpler ways to spruce up your user interface without plugins or server-side interaction.

Chapter 4, "Using the HTML5 Geolocation API," introduces a truly new feature that was not easily emulated before now—the ability for an application to identify the user's current location and use that to customize the experience. Privacy is important here, so we cover some of the caveats, as well.

The next two chapters, "Using the Communication APIs" and "Using the HTML5 WebSocket API," present increasingly powerful ways that HTML5 lets you communicate with other websites and stream real-time data to an application with both simplicity and minimal overhead. The techniques in these chapters will enable you to simplify the many overly complex architectures deployed on the Web today.

Chapter 7, "Using the HTML5 Forms API," presents you with minimal tweaks you can make to your desktop or mobile web applications today to increase usability, as well as more fundamental changes you can make to detect page entry errors in very common usage scenarios.

Chapters 8, 9, and 10—"Using the HTML5 Web Workers API," "Using the HTML5 Storage API," and "Creating HTML5 Offline Web Applications"—deal with the internal plumbing of your applications. Here, you will find ways to optimize the existing functionality to obtain better performance and better data management.

Finally, Chapter 11, "The Future of HTML5," will give you a tasty preview of what's still to come.

## Example Code and Companion Web Site

The code for the examples shown in this book is available online in the Source Code section of the Apress web site. Visit `www.apress.com`, click Source Code, and look for this book's title. You can download the source code from this book's home page. In addition, we are hosting a companion site for the book at `www.prohtml5.com`, from which you can also download the sample code and some practical extras.

## Contacting the Authors

Thank you for buying this book. We hope you enjoy reading it and that you find it a valuable resource. Despite our best effort to avoid errors, we realize that things sometimes slip through the cracks, and we'd like to express, in advance, our regrets for any such slip-ups. We welcome your personal feedback, questions, and comments regarding this book's content and source code. You can contact us by sending us an e-mail at `prohtml5@gmail.com`.

CHAPTER 1



# Overview of HTML5

This book is about HTML5 Programming. Before you can understand HTML5 programming, however, you need to take a step back and understand what HTML5 is, a bit of the history behind it, and the differences between HTML4 and HTML5.

In this chapter, we get right to the practical questions to which everyone wants answers. Why HTML5, and why all the excitement just now? What are the new design principles that make HTML5 truly revolutionary—but also highly accommodating? What are the implications of a plugin-free paradigm; what's in and what's out? What's new in HTML, and how does this kick off a whole new era for web developers? Let's get to it.

## The Story So Far—The History of HTML5

HTML goes back a long way. It was first published as an Internet draft in 1993. The '90s saw an enormous amount of activity around HTML, with version 2.0, versions 3.2, and 4.0 (in the same year!), and finally, in 1999, version 4.01. In the course of its development, the World Wide Web Consortium (W3C) assumed control of the specification.

After the rapid delivery of these four versions though, HTML was widely considered a dead-end; the focus of web standards shifted to XML and XHTML, and HTML was put on the back burner. In the meantime, HTML refused to die, and the majority of content on the web continued to be served as HTML. To enable new web applications and address HTML's shortcomings, new features and specifications were needed for HTML.

Wanting to take the web platform to a new level, a small group of people started the Web Hypertext Application Working Group (WHATWG) in 2004. They created the HTML5 specification. They also began working on new features specifically geared to web applications—the area they felt was most lacking. It was around this time that the term Web 2.0 was coined. And it really *was* like a second new web, as static web sites gave way to more dynamic and social sites that required more features—a lot more features.

The W3C became involved with HTML again in 2006 and published the first working draft for HTML5 in 2008, and the XHTML 2 working group stopped in 2009. Another year passed, and that is where we stand today. Because HTML5 solves very practical problems (as you will see later), browser vendors are feverishly implementing its new features, even though the specification has not been completely locked down. Experimentation by the browsers feeds back into and improves the specification. HTML5 is rapidly evolving to address real and practical improvements to the web platform.

### MOMENTS IN HTML

**Brian says:** "Hi, I'm Brian, and I'm an HTML curmudgeon.

I authored my first home page back in 1995. At the time, a 'home page' was something you created to talk about yourself. It usually consisted of badly scanned pictures, `<blink>` tags, information about where you lived and what you were reading, and which computer-related project you were currently working on. Myself and most of my fellow 'World Wide Web developers' were attending or employed by universities.

At the time, HTML was primitive and tools were unavailable. Web applications hardly existed, other than a few primitive text-processing scripts. Pages were coded by hand using your favorite text editor. They were updated every few weeks or months, if ever.

We've come a long way in fifteen years.

Today, it isn't uncommon for users to update their online profiles many times a day. This type of interaction wouldn't have been possible if not for the steady, lurching advances in online tools that built on each previous generation.

Keep this in mind as you read this book. The examples we show here may seem simplistic at times, but the potential is limitless. Those of us who first used `<img>` tags in the mid-1990s probably had no idea that within ten years, many people would be storing and editing their photos online, but we should have predicted it.

We hope the examples we present in this book will inspire you beyond the basics and to create the new foundation of the Web for the next decade."

## The Myth of 2022 and Why It Doesn't Matter

The HTML5 specification that we see today has been published as a working draft—it is not yet final. So when does it get cast in stone? Here are the key dates that you need to know. The first is 2012, which is the target date for the *candidate recommendation*. The second date is 2022, which is the *proposed recommendation*. Wait! Not so fast! Don't close this book to set it aside for ten years before you consider what these two dates actually mean.

The first and nearest date is arguably the most important one, because once we reach that stage, HTML5 will be complete. That's just two years away. The significance of the proposed recommendation (which we can all agree is a bit distant) is that there will then be two interoperable implementations. This means two browsers equipped with completely interoperable implementations of the entire specifications—a lofty goal that actually makes the 2022 deadline seem ambitious. After all, we haven't even achieved that in HTML4.

What *is* important, right now, is that browser vendors are actively adding support for many very cool new features. Depending on your audience, you can start using many of these features today. Sure, any number of minor changes will need to be made down the road, but that's a small price to pay for enjoying the benefits of living on the cutting edge. Of course, if your audience uses Internet Explorer 6.0, many of the new features won't work and will require emulation—but that's still not a good reason to dismiss HTML5. After all, those users, too, will eventually be jumping to a later version. Many of them will probably move to Internet Explorer 9.0 right away, and Microsoft promises to design that browser

with increased HTML5 support. In practice, the combination of new browsers and improving emulation techniques means you can use many HTML5 features today or in the very near future.

# Who Is Developing HTML5?

We all know that a certain degree of structure is needed, and somebody clearly needs to be in charge of the specification of HTML5. That challenge is the job of three important organizations:

- *Web Hypertext Application Technology Working Group (WHATWG)*: Founded in 2004 by individuals working for browser vendors Apple, Mozilla, Google, and Opera, WHATWG develops HTML and APIs for web application development and provides open collaboration of browser vendors and other interested parties.
- *World Wide Web Consortium (W3C)*: The W3C contains the HTML working group that is currently charged with delivering the HTML5 specification.
- *Internet Engineering Task Force (IETF)*: This task force contains the groups responsible for Internet protocols such as HTTP. HTML5 defines a new WebSocket API that relies on a new WebSocket protocol, which is under development in an IETF working group.

# A New Vision

HTML5 is based on various design principles, spelled out in the WHATWG specification, that truly embody a new vision of possibility and practicality.

- Compatibility
- Utility
- Interoperability
- Universal access

## Compatibility and Paving the Cow Paths

Don't worry; HTML5 is not an upsetting kind of revolution. In fact, one of its core principles is to keep everything working smoothly. If HTML5 features are not supported, the behavior must degrade gracefully. In addition, since there is about 20 years of HTML content out there, supporting all that existing content is important.

A lot of effort has been put into researching common behavior. For example, Google analyzed millions of pages to discover the common ID names for `DIV` tags and found a huge amount of repetition. For example, many people used `DIV id="header"` to mark up header content. HTML5 is all about solving real problems, right? So why not simply create a `<header>` element?

Although some features of the HTML5 standard are quite revolutionary, the name of the game is evolution not revolution. After all, why reinvent the wheel? (Or, if you must, then at least make a better one!)

## Utility and the Priority of Constituencies

The HTML5 specification is written based upon a definite *Priority of Constituencies*. And as priorities go, "the user is king." This means, when in doubt, the specification values users over authors, over implementers (browsers), over specifiers (W3C/WHATWG), and over theoretical purity. As a result, HTML5 is overwhelmingly practical, though in some cases, less than perfect.

Consider this example. The following code snippets are all equally valid in HTML5:

```
id="prohtml5"
id=prohtml5
ID="prohtml5"
```

Sure, some will object to this relaxed syntax, but the bottom line is that the end user doesn't really care. We're not suggesting that you start writing sloppy code, but ultimately, it's the end user who suffers when any of the preceding examples generates errors and doesn't render the rest of the page.

HTML5 has also spawned the creation of XHTML5 to enable XML tool chains to generate valid HTML5 code. The serializations of the HTML or the XHTML version should produce the same DOM trees with minimal differences. Obviously, the XHTML syntax is a lot stricter, and the code in the last two examples would not be valid.

## Secure by Design

A lot of emphasis has been given to making HTML5 secure right out of the starting gate. Each part of the specification has sections on security considerations, and security has been considered up front. HTML5 introduces a new origin-based security model that is not only easy to use but is also used consistently by different APIs. This security model allows us to do things in ways that used to be impossible. It allows us to communicate securely across domains without having to revert to all kinds of clever, creative, but ultimately insecure hacks. In that respect, we definitely will not be looking back to the good old days.

## Separation of Presentation and Content

HTML5 takes a giant step toward the clean separation of presentation and content. HTML5 strives to create this separation wherever possible, and it does so using CSS. In fact, most of the presentational features of earlier versions of HTML are no longer supported (but will still work!), thanks to the compatibility design principle mentioned earlier. This idea is not entirely new, though; it was already in the works in HTML4 Transitional and XHTML1.1. Web designers have been using this as a best practice for a long time, but now, it is even more important to cleanly separate the two. The problems with presentational markup are:

- Poor accessibility
- Unnecessary complexity (it's harder to read your code with all the inline styling)
- Larger document size (due to repetition of style content), which translates into slower-loading pages

## Interoperability Simplification

HTML5 is all about simplification and avoiding needless complexity. The HTML5 mantra? "Simple is better. Simplify wherever possible." Here are some examples of this:

- Native browser ability instead of complex JavaScript code
- A new, simplified `DOCTYPE`
- A new, simplified character set declaration
- Powerful yet simple HTML5 APIs

We'll say more about some of these later.

To achieve all this simplicity, the specification has become much bigger, because it needs to be much more precise—far more precise, in fact, than any previous version of the HTML specification. It specifies a legion of well-defined behaviors in an effort to achieve true browser interoperability by 2022. Vagueness simply will not make that happen.

The HTML5 specification is also more detailed than previous ones to prevent misinterpretation. It aims to define things thoroughly, especially web applications. Small wonder, then, that the specification is over 900 pages long!

HTML5 is also designed to handle errors well, with a variety of improved and ambitious error-handling plans. Quite practically, it prefers graceful error recovery to hard failure, again giving A-1 top priority to the interest of the end user. For example, errors in documents will not result in catastrophic failures in which pages do not display. Instead, error recovery is precisely defined so browsers can display "broken" markup in a standard way.

## Universal Access

This principle is divided into three concepts:

- *Accessibility*: To support users with disabilities, HTML5 works closely with a related standard called Web Accessibility Initiative (WAI) Accessible Rich Internet Applications (ARIA). WAI-ARIA roles, which are supported by screen readers, can be already be added to your HTML elements.
- *Media Independence*: HTML5 functionality should work across all different devices and platforms if at all possible.
- *Support for all world languages*: For example, the new `<ruby>` element supports the Ruby annotations that are used in East Asian typography.

# A Plugin–Free Paradigm

HTML5 provides native support for many features that used to be possible only with plugins or complex hacks (a native drawing API, native sockets, and so on). Plugins, of course, present problems:

- Plugins cannot always be installed.
- Plugins can be disabled or blocked (for example, the Apple iPad does not ship with a Flash plugin).

- Plugins are a separate attack vector.
- Plugins are difficult to integrate with the rest of an HTML document (because of plugin boundaries, clipping, and transparency issues).

Although some plugins have high install rates, they are often blocked in controlled corporate environments. In addition, some users choose to disable these plugins due to the unwelcome advertising displays that they empower. However, if users disable your plugin, they also disable the very program you're relying on to display your content.

Plugins also often have difficulty integrating their displays with the rest of the browser content, which causes clipping or transparency issues with certain site designs. Because plugins use a self-contained rendering model that is different from that of the base web page, developers face difficulties if pop-up menus or other visual elements need to cross the plugin boundaries on a page. This is where HTML5 comes on the scene, smiles, and waves its magic wand of *native* functionality. You can style elements with CSS and script with JavaScript. In fact, this is where HTML5 flexes its biggest muscle, showing us a power that just didn't exist in previous versions of HTML. It's not just that the new elements provide new functionality. It's also the added native interaction with scripting and styling that enables us to do much more than we could ever do before.

Take the new canvas element, for example. It enables us to do some pretty fundamental things that were not possible before (try drawing a diagonal line in a web page in HTML4). However, what's most interesting is the power that we can unlock with the APIs and the styling we can apply with just a few lines of CSS code. Like well-behaved children, the HTML5 elements also play nicely together. For example, you can grab a frame from a video element and display it on a canvas, and the user can just click the canvas to play back the video from the frame you just grabbed. This is just one example of what a native code has to offer over a plugin. In fact, virtually *everything* becomes easier when you're not working with a black box. What this all adds up to is a truly powerful new medium, which is why we decided to write a book about HTML5 *programming*, and not just about the new elements!

## What's In and What's Out?

So, what really *is* part of HTML5? If you read the specification carefully, you might not find all of the features we describe in this book. For example, you will not find HTML5 Geolocation and Web Workers in there. So are we just making this stuff up? Is it all hype? No, not at all!

Many pieces of the HTML5 effort (for example, Web Storage and Canvas 2D) were originally part of the HTML5 specification and were then moved to separate standards documents to keep the specification focused. It was considered smarter to discuss and edit some of these features on a separate track before making them into official specifications. This way, one small contentious markup issue wouldn't hold up the show of the entire specification.

Experts in specific areas can come together on mailing lists to discuss a given feature without the crossfire of too much chatter. The industry still refers to the original set of features, including Geolocation, as HTML5. Think of HTML5, then, as an umbrella term that covers the core markup, as well as many cool new APIs. At the time of this writing, these features are part of HTML5:

- Canvas (2D and 3D)
- Channel messaging
- Cross-document messaging
- Geolocation

- MathML
- Microdata
- Server-Sent events
- Scalable Vector Graphics (SVG)
- WebSocket API and protocol
- Web origin concept
- Web storage
- Web SQL database
- Web Workers
- XMLHttpRequest Level 2

As you can see, a lot of the APIs we cover in this book are on this list. How did we choose which APIs to cover? We chose to cover features that were at least somewhat baked. Translation? They're available in some form in more than one browser. Other (less-baked) features may only work in one special beta version of a browser, while others are still just ideas at this point.

In this book, we will give you an up-to-date overview (at the time of this writing) of the browser support available for each of the HTML5 features we cover. However, whatever we tell you will soon be out of date, because this is very much a moving target. Don't worry though; there are some excellent online resources that you can use to check current (and future) browser support. The site `www.caniuse.com` provides an exhaustive list of features and browser support broken down by browser version and the site `www.html5test.com` checks the support for HTML5 features in the browser you use to access it.

Furthermore, this book does not focus on providing you with the emulation workarounds to make your HTML5 applications run seamlessly on antique browsers. Instead, we will focus primarily on the specification of HTML5 and how to use it. That said, for each of the APIs we do provide some example code that you can use to detect its availability. Rather than using *user agent* detection, which is often unreliable, we use *feature* detection. For that, you can also use *Modernizr*—a JavaScript library that provides very advanced HTML5 and CSS3 feature detection. We highly recommend you use Modernizr in your applications, because it is hands down the best tool for this.

## MORE MOMENTS IN HTML

**Frank says:** "Hi, I'm Frank, and I sometimes paint.

One of the first HTML canvas demonstrations I saw was a basic painting application that mimicked the user interface of Microsoft Paint. Although it was decades behind the state of the art in digital painting and, at the time, ran in only a fraction of existing browsers, it got me thinking about the possibilities it represented.

When I paint digitally, I typically use locally installed desktop software. While some of these programs are excellent, they lack the characteristics that make web applications so great. In short, they are disconnected. Sharing digital paintings has, to date, involved exporting an image from a painting

> application and uploading it to the Web. Collaboration or critiques on a live canvas are out of the question. HTML5 applications can short-circuit the export cycle and make the creative process fit into the online world along with finished images.
>
> The number of applications that cannot be implemented with HTML5 is dwindling. For text, the Web is already the ultimate two-way communication medium. Text-based applications are available in entirely web-based forms. Their graphical counterparts, like painting, video editing, and 3D modeling software, are just arriving now.
>
> We can now build great software to create and enjoy images, music, movies, and more. Even better, the software we make will be on and off the Web: a platform that is ubiquitous, empowering, and online."

---

# What's New in HTML5?

Before we start programming HTML5, let's take a quick look at what's new in HTML5.

## New DOCTYPE and Character Set

First of all, true to design principle 3—simplification—the `DOCTYPE` for web pages has been greatly simplified. Compare, for example, the following HTML4 `DOCTYPE`s:

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

Who could ever remember any of these? We certainly couldn't. We would always just copy and paste some lengthy `DOCTYPE` into the page, always with a worry in the back of our minds, "Are you absolutely sure you pasted the right one?" HTML5 neatly solves this problem as follows:

```
<!DOCTYPE html>
```

Now *that's* a `DOCTYPE` you might just remember. Like the new `DOCTYPE`, the character set declaration has also been abbreviated. It used to be

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

Now, it is:

```
<meta charset="utf-8">
```

Using the new `DOCTYPE` triggers the browser to display pages in standards mode. For example, Figure 1-1 shows the information you will see if you open an HTML5 page in Firefox, and you click Tools ➤ Page Info. In this example, the page is rendered in standards mode.

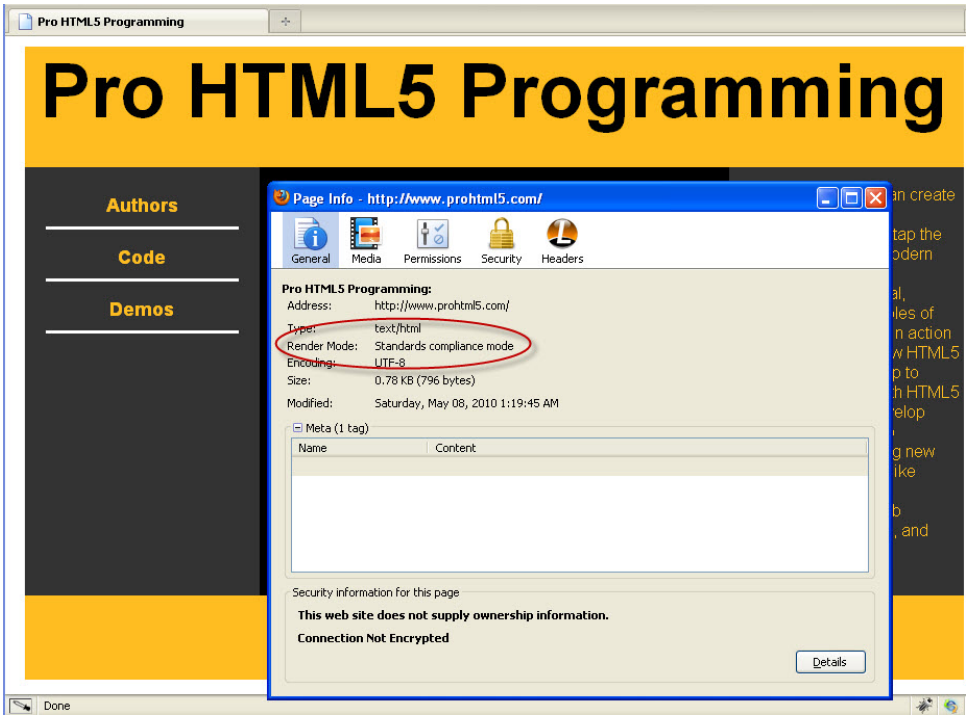


***Figure 1-1.*** *A page rendered in standards-compliant mode*

When you use the new HTML5 `DOCTYPE`, it triggers browsers to render the page in standards-compliant mode. As you may know, Web pages can have different rendering modes, such as Quirks, Almost Standards, and Standards (or no-quirks) mode. The `DOCTYPE` indicates to the browser which mode to use and what rules are used to validate your pages. In Quirks mode, browsers try to avoid breaking pages and render them even if they are not entirely valid. HTML5 introduces new elements and others (more on this in the next section) so that if you use deprecated elements, your page will not be valid.

## New and Deprecated Elements

HTML5 introduces many new markup elements, which it groups into seven different content types. These are shown below in Table 1-1.

***Table 1-1.*** *HTML5 Content Types*

| Content Type | Description |
|---|---|
| Embedded | Content that imports other resources into the document, for example `audio`, `video`, `canvas`, and `iframe` |
| Flow | Elements used in the body of documents and applications, for example `form`, `h1`, and `small` |
| Heading | Section headers, for example `h1`, `h2`, and `hgroup` |
| Interactive | Content that users interact with, for example `audio` or `video controls`, `button`, and `textarea` |
| Metadata | Elements—commonly found in the `head` section— that set up the presentation or behavior of the rest of the document, for example `script`, `style`, and `title` |
| Phrasing | Text and text markup elements, for example `mark`, `kbd`, `sub`, and `sup` |
| Sectioning | Elements that define sections in the document, for example `article`, `aside`, and `title` |

All of these elements can be styled with CSS. In addition, some of them, such as `canvas`, `audio`, and `video`, can be used by themselves, though they are accompanied by APIs that allow for fine-grained native programmatic control. These APIs will be discussed in much more detail later in this book.

It is beyond the scope of this book to discuss all these new elements, but the sectioning elements (discussed in the next section) are new. The `canvas`, `audio`, and `video` elements are also new in HTML5.

Likewise, we're not going to provide an exhaustive list of all the deprecated tags (there are many good online resources online for this), but many of the elements that performed inline styling have been removed in favor of using CSS, such as `big`, `center`, `font`, and `basefont`.

## Semantic Markup

One content type that contains many new HTML5 elements is the *sectioning* content type. HTML5 defines a new *semantic* markup to describe an element's content. Using semantic markup doesn't provide any immediate benefits, but it does simplify the design of your HTML pages, and in the future, search engines will definitely be taking advantage of these elements as they crawl and index pages.

As we said before, HTML5 is all about paving the cow paths. Google analyzed millions of pages to discover the common ID names for `DIV` tags and found a huge amount of repetition. For example, since many people used `DIV id="footer"` to mark up footer content, HTML5 provides a set of new sectioning elements that you can use in modern browsers right now. Table 1-2 shows the different semantic markup elements.

***Table 1-2.*** *New sectioning HTML5 elements*

| Sectioning Element | Description |
|---|---|
| `header` | Header content (for a page or a section of the page) |
| `footer` | Footer content (for a page or a section of the page) |
| `section` | A section in a web page |
| `article` | Independent article content |
| `aside` | Related content or pull quotes |
| `nav` | Navigational aids |

All of these elements can be styled with CSS. In fact, as we described in the utility design principle earlier, HTML5 pushes the separation of content and presentation, so you have to style your page using CSS styles in HTML5. Listing 1-1 shows what an HTML5 page might look like. It uses the new `DOCTYPE`, character set, and semantic markup elements—in short, the new sectioning content. The code file is also available in the `code/intro` folder.

***Listing 1-1.*** *An Example HTML5 Page*

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" >
  <title>HTML5</title>
  <link rel="stylesheet" href="html5.css">

</head>

<body>

   <header>
     <h1>Header</h1>
     <h2>Subtitle</h2>
     <h4>HTML5 Rocks!</h4>
   </header>

    <div id="container">
```