THE TRUTH ABOUT HTML5

FOR WEB DESIGNERS • 2012-13 EDITION

SEMANTICS • MICRODATA • SEO • FORMS
CANVAS • AUDIO/VIDEO • SVG • WEB APPS

BY LUKE STEVENS

FOREWORD BY JOHN ALLSOPP

The Truth About HTML5 (For Web Designers)

Written and designed by
Luke Stevens

Edited by
Bill Harper
(Portions of this book have been updated post-edit, so any mistakes are entirely those of the author!)

Published by
Indie Digital Pty Ltd

http://truthabouthtml5.com
Spotted an error or typo? Let me know: luke@itsninja.com

Thanks for reading!
Luke

Foreword

HTML5 is a mess. It's also one of the most exciting technological advances perhaps ever (a big claim especially for something I just described as a mess).

There are quite a few books, most of them excellent, on HTML5. Some cover the markup exclusively Some cover markup and JavaScript APIs. Others still focus on a specific development challenge like games.

This book is a little different. Rather than simply looking at the *what* and *how* of HTML5 (though it does that as well) it endeavors to explain the *why* and *why not* of HTML5.

And it's a passionate, informed, opinionated critique of much of HTML5 to boot.

Along the way you'll learn a great deal about HTML5 markup, and additional HTML5 features such a the new audio and video elements, the Canvas element, the History API, and related features such as SVG.

But hopefully most of all you'll learn to think critically about HTML5 as a tool, and adopt the good parts, for good reasons, and ignore the less than useful parts, for the right reasons as well.

Luke Stevens has written a book all web designers and developers who care about their code should read. So go ahead and read it!

John Allsopp
Author, *Developing with Web Standards*
Co-founder of Web Directions
Web evangelist

Introduction

Hi. I'm Luke, your average, garden-variety web designer. I've been building web sites for over a decade, use ExpressionEngine as my CMS, and have enjoyed both working in-house and full-time freelancing.

I thought it would be fun to write a short book about HTML5. I thought HTML5 would be simple. I thought writing about it would be straightforward. And I thought the respected voices in the design community would be telling everyone what it is (and what it isn't) simply and clearly, particularly with the plethora of other HTML5 books out there.

I was wrong.

Fortunately this book (and hopefully your experience as a reader!) is infinitely better for it. And I hope once you've read it you'll share my concern about the strange direction basic markup has taken *and* my excitement for the new HTML5 (and related) technologies that are coming soon to a browser near you. That includes Internet Explorer 10—Microsoft finally, truly gets web standards.

What seemed impossible just a few years ago—a far-fetched, almost utopian ideal of *all* browser vendors, including Microsoft, competing tooth-and-nail to support bleeding-edge web standards—is now a reality. Innovation in web standards is happening at a break-neck speed, and my hope is this book gets you up to speed not only with the fundamentals of HTML5, but with the broader picture of where the web as a whole is heading, especially as we look towards a post-Flash future.

As you make your way through the following chapters, please keep in mind this book is as much of a *critique* as it is an explanation of HTML5. By taking a critical look at *why* things are the way they are my hope is you save hours by not having to worry about things that don't matter (particularly when it comes to basic markup), and your eyes are opened to how the HTML5 sausage gets made. It may not always be pretty, but if you spend your days in the trenches building websites, knowing *why* things are the way they are will help guide your design and development decisions in a very direct way.

That said, there's plenty of exciting technology in and around HTML5 too, so be sure not to miss the later chapters on graphics technologies like Canvas and SVG; the state of audio and video in HTML5 and the more developer-oriented HTML5 features that includes a new way of handling something as fundamental as a page request.

(Also note we will be focusing almost entirely on HTML5 as defined by the HTML5 spec, with the addition of SVG, and a few other related initiatives such as Schema.org and WebGL. "HTML5" has become a buzzword which can mean everything from the HTML5 spec itself, to CSS3 and modern JavaScript, to just "cool and new and not Flash". We'll be mostly sticking with the features in the actual HTML5 specification.)

I love the web design community because it's filled with smart, excitable, curious, opinionated folk who will call you on your BS. This is an opinionated book, not a dry explanation of the technology, and I'll be stating my views pretty strongly. I look forward to you doing the same. Passionate, considered debate makes us all smarter. So please, write it up on your blog, send me happy/sad/angry

emails (luke@itsninja.com), talk to me on Twitter (@lukestevens), or whatever you like.

I look forward to the discussion.

And now I'd like to ask a couple of favors.

First, if you enjoy my writing then please tell your friends, colleagues, Twitter followers, blog reader and pretty much anyone who will listen about this book. Like a lot of authors, I rely entirely on readers like you to spread the word (and the links). If you can help me out by spreading the word abo this book via good old fashioned word of mouth I'd *really* appreciate it. Thank you.

And second, if you use Google Analytics (and who doesn't?) and want to get more out of it, I'd love you to check out my web app Ninja for Google Analytics at http://itsninja.com. Google Analytics is a big, complex beast, but it has the best data on how your web site actually performs, it's just buried deep, deep down. Ninja for GA brings that data to the surface through a simple, elegant interface. It's web analytics for web designers, and I think you (and your clients) will like it. My hope is it will make your own design practice (and your client's sites) more productive and profitable. After all, all the HTML5 in the world wont help you if your conversion rates are lousy and your bounce rates are sky-high. (We'll return to this theme in the final chapter of this book when we look at Performance Based Design.) Check it out: http://itsninja.com.

How Architecture Astronauts And The W3C Tried To Kill HTM

Murder is always interesting, so let's start there.

In 1997 the W3C published a Recommendation for HTML 4.0. And two years later it was more or les completed in the form of HTML 4.01. (Don't remember? Well, you were probably too busy worrying about the dreaded Y2K "bug" wiping out civilization.)

And that was pretty much it for plain old HTML.

So what happened between HTML being "finished" in 1999 (in every sense of the word), and HTML5's emergence today?

A long, aborted march to "XMListan".

The W3C published the eXtensible Markup Language (XML) 1.0 spec in 1996 (http://www.w3.org/TR/1998 REC-xml-19980210), which they hoped would become a more flexible, machine readable, stricter and more extensible way to mark up documents and other data. And it was soon being used to do just that But the W3C believed the *web itself* would eventually move to XML.

One of the first baby steps in that direction was XHTML—an XML formulation of HTML 4.

You Probably Use XML

XML may sound foreign, but if you own or even subscribe to a blog then you're already using it. The RSS or Atom feed blogs generate to syndicate their content is just one form of XML. If you look at th source of an Atom feed, you can see tags such as `<author>`, `<published>`, `<category>` and `<summary>`. These are *specific tags that accurately describe the content they represent*. It's just one

example of the “extensible” part of XML that allows machines (parsers, RSS readers and so on) to do interesting things with the content.

Now, imagine a world where we could describe our web pages in a similar way. That was the W3C’s plan for the web—that *all* the future content on the web should be described in more accurate terms than just `<div>`s, `<span>`s, `<p>`s and `<h1>`s. And with XML, we could do it.

HTML would still exist as a legacy format. But the future was XML, baby.

XHTML Is Born, But What Does It Mean?

So if HTML was the past, and XML was the future, how would we get there? With the interim step of XHTML.

By reformulating HTML 4.0 to stick to XML’s rules, XHTML was born. And in January 2000, having barely survived the Y2K apocalypse, the XHTML 1.0 spec was adopted as a W3C Recommendation (http://www.w3.org/TR/xhtml1/). We were on the road to XMListan.

In early 2002, Jeffrey Zeldman published the landmark XHTML article “Better Living Through XHTML” on A List Apart (http://www.alistapart.com/articles/betterliving/), describing XHTML as:

> *[T]he standard markup language for web documents and the successor to HTML 4. A mixture of classic (HTML) and cutting–edge (XML), this hybrid language looks and works much like HTML but is based on XML, the web’s “super” markup language, and brings web pages many of XML’s benefits, as enumerated by the Online Style Guide of the Branch Libraries of The New York Public Library.*

Those benefits enumerated on the The New York Public Library website (http://legacy.www.nypl.org/styleguide/xhtml/benefits.html) included:

> *The web is moving to XML, a powerfully enabling technology. Writing well–formed, valid XHTML pages is the easiest way to begin this transition. All it takes is learning a few simple rules of XHTML markup.*

Web designers took heed of this call to begin the transition to XML via XHTML. In 2003 Dave Shea wrote a post called “Semantics and Bad Code” (http://www.mezzoblue.com/archives/2003/08/26/semantics_an/) where he said:

> *The move from HTML to XML requires a huge shift in developer mindset. There are a lot of obstacles to overcome yet, not the least of which being solid browser support. We’ve only started down the road, and XHTML is how we’ll get there.*

Shea’s view was a popular one at the time, and certainly reasonable given our faith in the experts in the W3C.

But we never made it to XMListan. The car ran out of gas, the wheels fell off, and the engine explode about two blocks down the road.

Draconian Error Handling, Or Why Don't I Just Punch You In The Face?

Those of you building web sites back in the early '00s may remember how important it was to have a *valid* web page. People even put dinky little "Valid XHTML" badges on their sites to show off just how forward-thinking they were. (They now put equally silly HTML5 badges on blogs—and books.) Design nerds would even run *other people's* markup through the HTML validator, and write a snarky blog post or email if it failed. (Back then there was no Twitter to bitch publicly in 140 characters.)

Yes, having valid HTML is a good thing. But as web designers adopted XHTML it became—in theor if not practice—life or death. If you had so much as single error in your XHTML, your browser woul reach out and *punch you in the face*.

Okay, Not Really. But We COULD Punch You In The Face

Well, it would if you set up your server to tell the browser to adopt XML's strict XHTML parsing rules (as Mark Pilgrim described in 2003: http://www.xml.com/pub/a/2003/03/19/dive-into-xml.html), which hardl anyone did. Internet Explorer, right up to and including version 8, didn't even support these strict XHTML parsing rules. (Ironically, IE9 now does, just as everyone stopped caring.)

Why didn't anyone do it? Because they didn't want to inflict the "draconian error handling" on their users (or themselves). And it really was draconian—one invalid character, such as "&" instead of "&", would generate a fatal error that destroyed the entire page. And as a user, all you got was a hideous error message—no content, no nothing.

In light of this, the web standards community adopted the *theory* of XHTML without its harsh realiti (or true XML nature), preferring to stick with the warm, cuddly and vastly forgiving HTML parsing from the early days.

XHTML turned out to be a baby step towards a baby step. What should have been the first move towards a strict XML formulation of the web, where we could use more descriptive (i.e. semantic) tags, was just a step towards stricter, old-style HTML. It was two steps forwards, one step back—bac to the HTML the W3C had declared finished, and was hoping to make obsolete.

XHTML Still Meant Better HTML

Nevertheless, XHTML gave the web standards community something to, well, standardize on. It allowed everyone to be a bit more serious, and dare I say *professional*, about the markup we were writing. As Jeffrey Zeldman wrote on his blog in 2009 (http://www.zeldman.com/2009/07/07/in-defense-of-web-developers/):

> *XHTML's introduction in 2000, and its emphasis on rules of construction, gave web standards evangelists like me a platform on which to hook a program of semantic markup replacing the bloated and unsustainable tag soup of the day. The web is better for this and always will be, and*

there is much still to do, as many people who create websites still have not heard the call.

For much of the ‘00s, websites built with web standards continued using XHTML. Designers got serious about separating presentation from content, and tried to write more semantic markup. Using XHTML also triggered standards mode on the major browsers of the time. All good things.

But in the W3C’s grander scheme of things, XHTML ultimately proved to be a bit of a stepping ston to nowhere.

But The Crazy Had Only Just Begun

XHTML served a useful purpose for web standards—albeit not the one originally intended. But now we step into the mad, mad, mad world of XHTML **2.0**.

While we were all happily using and advocating XHTML in web standards land (though some stuck t HTML 4.0), the W3C was working on XHTML 2.0. Sounds like a harmless update of the 1.0 spec, right?

It wasn’t.

XHTML 2.0 was day zero for the web. It wasn’t backward compatible with HTML, or even XHTML 1.0. It was a *whole new thang*.

And nothing was safe.

Among the list of sweeping changes, plain old forms would be replaced with fancy XML-style XForms. Even the `<img>` element was on the chopping block at one point, as the W3C re-envisioned the web as a more XML-ified place.

In an April 2011 blog post on software development, Joel Spolsky described what he calls “Architecture Astronauts” (http://www.joelonsoftware.com/articles/fog0000000018.html):

> *When you go too far up, abstraction-wise, you run out of oxygen. Sometimes smart thinkers just don't know when to stop, and they create these absurd, all-encompassing, high-level pictures of the universe that are all good and fine, but don't actually mean anything at all.*
>
> *These are the people I call Architecture Astronauts.*

And XHTML 2.0 was a classic case of Architecture Astronauts at work.

Here’s how Bruce Lawson, HTML5 evangelist for Opera and author of “Introducing HTML5” (New Riders, 2010) describes it (http://news.cnet.com/8301-17939_109-10281477-2.html):

> *XHTML 2 was a beautiful specification of philosophical purity that had absolutely no resemblance to the real world.*

As far as HTML was concerned, this is what the W3C—the custodians of the language that underpins much of our relationships, business, and government in the 21st century—worked on from 2002-200

over 8 drafts. Not only would it have broken backwards compatibility, it would also have sent all the talk of “forward compatibility” and “future-proofing” in the web standards community up in smoke. (You can read more about XHTML 2.0 in Wikipedia: http://en.wikipedia.org/wiki/XHTML#XHTML_2.0.)

XHTML 2.0: Unloved And Alone

While the W3C toiled away on XHTML 2.0, what did web authors, standards advocates, and browser vendors think of it?

Not much.

There was zero interest in implementing it. Even members of the working group were deeply unhapp with it. (See Jeffrey Zeldman’s thoughts on XHTML 2.0 in 2003 under “XHTML 2 and all that”: http://www.zeldman.com/daily/0103b.shtml.)

What was dopey about XHTML 2.0 wasn’t so much the spec itself (which would be fine if we could go back in time and rebuild the web from scratch). It was the idea you could do something as revolutionary as breaking backwards compatibility with millions of existing documents and create a whole new tier for the web. But that was the path the W3C set themselves on way back in 1998 (see i for yourself in "Shaping the Future of HTML"” http://www.w3.org/MarkUp/future/).

But what if the next evolution of HTML was just that—evolutionary, rather than revolutionary? One that built on the world as it was, and not some utopian world we could only hope for?

HTML5: A New Hope... We Hope

HTML5 began as a reaction to the W3C’s descent into markup madness. The problems with the W3C’s direction had not gone unnoticed.

In 2004, the so-called “Web 2.0” movement took off in a big way, and web *applications* became a bi deal. The web was no longer just a collection of text and images on pages connected through links. It was becoming a platform for applications that could run anywhere, OS be damned.

Compared to the ‘80s and ‘90s, when your OS determined what applications you could use, running applications through a browser on any OS was a revolutionary idea.

No one really predicted this (certainly not the W3C), which isn’t surprising when you think how bad we are at predicting the future in general. (Where *is* my flying car?) We’re much better at reacting ar evolving when the future arrives, which is what some people suggested we do with HTML.

In 2004, members representing Opera and Mozilla (with Apple “cheering [from] the sidelines”, as Ia Hickson recalls: http://www.webstandards.org/2009/05/13/interview-with-ian-hickson-editor-of-the-html-5-specification/) presented an alternative to the W3C—a spec focused on web applications. (See the original “Position Paper” here: http://www.w3.org/2004/04/webapps-cdf-ws/papers/opera.html.)

The W3C Says Go To Hell

HTML needed to adapt to the future of web *applications*, rather than a utopian world of perfectly marked-up XML-ified web *pages*. So this new group suggested an alternative direction for HTML based on backwards compatibility. No more draconian error handling (the one-error-and-you're-dead problem of XHTML as XML). New features for web applications. And an open process, which was in stark contrast to the way the W3C operates.

Essentially, their philosophy was that HTML was here to stay, and so we should concentrate on evolving it. (This may sound completely obvious now, but back then it wasn't a view shared by the W3C.)

Anyway, the group pitched their ideas to the W3C, and the W3C told them to go to hell. (Actually, they only lost by two votes—11-8 against. But this *is* the somewhat *sensationalized* history of HTML5.)

With the W3C being less than accommodating, those interested in evolving HTML and adding features for web applications, and who were backed by (and worked for) the browser vendors, decided to press on and work outside the W3C. They formed the Web Hypertext Applications Technology Working Group (WHATWG), and set up shop at whatwg.org in June 2004.

The WHATWG Is Born

And so the WHATWG was born. Here's how Hickson explains it all (http://www.thechromesource.com/interview-html5-standards-author-ian-hickson/):

> *So [after the W3C rejection] we opened a mailing list called the WHATWG to continue work on Web Forms 2.0 in public, and later that year started a new draft called Web Applications 1.0 into which we put many features aimed at writing Web apps, including a new version of HTML that we jokingly called HTML5, and a bunch of other features that later became Web Storage, Web Sockets, Server-Sent Events, and a variety of other specs. [...]*
>
> *Later, around 2006 or 2007, the W3C basically realized they had made a mistake, and they asked if they could work on HTML5 as well, so we renamed Web Applications 1.0 to HTML5, and the WHATWG and the W3C started working together. Web Forms 2.0 got merged into HTML5, and most of the bits of HTML5 that weren't really HTML got split out into separate specs.*

It's ironic, isn't it? The establishment (the W3C) was the utopian revolutionary, and the rebel outsiders (the WHATWG) were fighting for incremental conservatism. Go figure.

It's A Whole New World

It's worth noting several points here:

- The W3C failed dramatically at maintaining HTML (which is kind of scary when you think abou

it).

- Web standards are incredibly haphazard. There was—and is—no unifying vision of "HTML5". I was just a bunch of separate specifications bundled up and given the name "HTML5", and those specifications only came about as a reaction to the W3C's failures.
- Big, bold ideas like the march to XML for the web—which had many people excited a decade ag —can fade to nothing. We should learn from this, and retain some skepticism towards big, bold ideas—including some of the changes in HTML5.
- The balance of power now rests with the browser vendors.

In truth, the balance of power has *always* rested with the browser vendors. If they don't implement something, *by definition* it's a non-starter. As Hickson says (http://www.webstandards.org/2009/05/13/interview-with-ian-hickson-editor-of-the-html-5-specification/):

> *The reality is that the browser vendors have the ultimate veto on everything in the spec, since if they don't implement it, the spec is nothing but a work of fiction. So they have a lot of influence—I don't want to be writing fiction, I want to be writing a spec that documents the actual behavior of browsers.*
>
> *Whether that's too much, I don't know. Does gravity have too much influence on objects on earth? It's just the way it is.*

Nevertheless, the fact an independent standards body—*our* independent standards body—failed miserably is more than a little concerning.

To HTML5 And Beyond!

To cut a long story short, the WHATWG kept working on their own vision of evolving HTML—the *only* vision of evolving HTML. And in 2006 Tim Berners-Lee, father of the World Wide Web and Director of the W3C (read more about him here: http://en.wikipedia.org/wiki/Tim_Berners-Lee), sucked it up an announced the W3C would work with the WHATWG, saying (http://dig.csail.mit.edu/breadcrumbs/node/166):

> *The attempt to get the world to switch to XML, including quotes around attribute values and slashes in empty tags and namespaces all at once didn't work.*

Berners-Lee left the door open to switching to XML by saying "all at once". But in reality it looks very much like "The attempt to get the world to switch to XML... didn't work."

And that's fine. We need big ideas and bold directions to try and work towards, and if they don't wor out, so be it. Sometimes good ideas just don't happen.

With the WHATWG having so much momentum (and the backing of the browser vendors), the W3C had no choice *but* to work with them on HTML5. In 2007 the W3C formed a group that worked work with the WHATWG on developing HTML5. And in January 2008 the W3C released their first HTML Working Draft (http://www.w3.org/TR/2008/WD-html5-20080122/), adopting the work the WHATWG had been

doing for several years.

HTML5 Is The New Black Or Hotness Or Something

By the late ‘00s web technologies were exciting again, and after years of stagnation and dead ends we finally reached a point where the bowels of innovation were loosened. (That’s a horrible image—sorry.)

Now in the early ‘10s, things are looking even better. In fact, there’s a veritable Cambrian explosion of web technology taking place. Google, Mozilla, Apple and Microsoft are competing to make the be *standards compliant* browser (with new versions coming thick and fast). There’s a whole bunch of new and interesting technology around. And web developers, designers, software companies and app developers are all interested in the new and shiny tech in and around HTML5.

To think browser makers—including Microsoft—are now trying to outcompete and even out-market each other with their *web standards support* is pretty incredible. It wasn’t that long ago (late ‘90s) th we faced the threat of them all going their own non-standard ways. Hats off to all involved.

Is HTML5 Hype, Substance, Or Both?

But back to the HTML5 specification. Two questions:

1. What exactly *is* HTML5?
2. Who’s in charge, now there’s a (decidedly uneasy) working relationship between The Establishment (the W3C) and The Rebels (the WHATWG)?

Let’s deal with what HTML5 *is* first. There’s:

- HTML5, the all-encompassing marketing buzzword
- HTML5, the bit that’s actually about HyperText Markup
- HTML5, the new functionality available through JavaScript for web applications
- HTML5, the behind the scenes stuff that’s really important and documents a whole lot of stuff browsers actually do (but you’re probably not interested in).

All this from a technical specification that runs for hundreds of pages.

For us web designers, HTML5 is currently a confusing mix of hype *and* substance, which we’ll try to sort through in the coming chapters.

In many ways HTML5 is, to put it bluntly, a mess. But it’s the most ordered mess we’ve had in a lon time. (For instance, a big part of HTML5 is written for browser vendors to ensure implementations a consistent and we can trust all browsers to do the same thing. And that’s never been done before.)

Perhaps the biggest problem is everyone thinking that if HTML5 is cool, then *all* of it (at least

according to the web design community) *must* be great, and we should adopt it post-haste without too much critical thought. And that's something I'm keen to dispel in the rest of the book.

Hixie Or Bust

As I write this, both the WHATWG and W3C versions of the HTML5 spec (the differences between the two are minor) are edited by one person: Ian Hickson.

HTML is now essentially in the hands of one man.

The W3C's working groups tried building consensus, and got absolutely nowhere with HTML. It was closed, but democratic. The WHATWG, on the other hand, has an open process, but with an editor-has-the-final-say approach.

And that editor is Ian "Hixie" Hickson.

Hickson helped start the WHATWG when working for Opera, and now works full-time for Google developing the HTML5 spec. Currently, he is the HTML5 (and now just "HTML") editor for *life*. Theoretically, the browser makers can veto him or kick him out at any time, but that seems highly unlikely. This has not gone unnoticed in the community, and is (rightly, in my opinion) a cause of some concern.

It's a classic "glass half-full/glass half-empty" situation. If Hickson flat out refuses an idea (which is known to happen), then having a single person in charge may seem like utter madness. But for those who saw the W3C's democratic processes get nowhere with XHTML 2.0, having someone who can take the reins, push things along, and actually make decisions would seem wonderful.

Of course, this invariably polarizes people.

Here's John Gruber of Daring Fireball fame (http://daringfireball.net/linked/2009/07/01/hickson-codecs):

> *Let it be said that Ian Hickson is the Solomon of web standards; his summary of the situation is mind-bogglingly even-handed and fair-minded.*

And here's Kyle Weems, creator of the CSSquirrel comic, who has been following HTML5's development for several years (http://twitter.com/#!/cssquirrel/status/58559284224589824):

> *Also... why oh why is @hixie still the editor for any world-altering spec like HTML anymore? Ego doesn't even begin to describe his antics*

As you can see, Hickson has his fans and his detractors.

I imagine editing a spec the size of HTML5 for as long as he has, with all the controversy that surrounds it, would be a pretty thankless task. But Hickson seems to go about it in a cheerful, dispassionate way.

If there's one overarching theme here, it's this: pragmatism rules.

The W3C had the "pure" spec of XHTML 2.0, and failed—it wasn't pragmatic. It also had its rules,

membership, and democratic processes, but was mired in politics and failed (with HTML at least)—i wasn't pragmatic.

The WHATWG put an editor in charge, and while this approach terrified and/or infuriated some people (including me from time to time, as you'll soon see), it *was* pragmatic (as was their approach to the spec). It got things moving (and, more importantly, shipping). And as long as it remains pragmatic it's probably how the WHATWG will stay.

XHTML 2.0 Is Dead And Everyone Is Happy

So what happened to XHTML 2.0? It was pronounced dead after being taken off life support in 2009 (http://www.w3.org/2009/06/xhtml-faq.html). I hear the death of XHTML 2.0 will soon be fictionalized in an upcoming episode of "Law & Order: Web Standards Unit".

And what about XHTML 1.0 and its various flavors? Considering it's essentially just HTML, it will keep working pretty much forever. (There's actually a continuing XML serialization of HTML5 calle XHTML5, but the chance of you actually needing to use it is practically zero.)

HTML5, err HTML, wait... HTML.next?

To show how things have come full circle with the HTML spec, the WHATWG declared in January 2011 that their HTML5 spec would be a "living standard" and renamed it to just "HTML". (See the announcement here: http://blog.whatwg.org/html-is-the-new-html5 and their rationale here: http://wiki.whatwg.org/wiki/FAQ#What_does_.22Living_Standard.22_mean.3F.)

And what of the future of HTML? The WHATWG insist they—and particularly Hickson—will maintain the HTML spec as a "living standard" indefinitely, while the W3C are sticking with the snapshot process, and have started accepting ideas for what they're unofficially calling "HTML.next" (see some of the ideas here: http://www.w3.org/wiki/HTML/next).

(A W3C member gave a personal presentation that captures the differing approaches to the future of HTML quite nicely: http://www.w3.org/2010/11/TPAC/HTMLnext-perspectives.pdf.)

Will the W3C come up with another pie-in-the-sky path to nowhere (echoing 1998's "Shaping the Future of HTML" workshop http://www.w3.org/MarkUp/future/)? Will they try to work with the WHATWG, or fork HTML5 and do their own thing? Who knows. Some have been asking if the W3C should even exist.

Should We Just Kill Off The W3C Altogether, Or Embrace It?

In September 2011, a debate broke out about the purpose of the W3C, and three broad views emerged reform, destroy, and embrace.

Before we get to those three views, let's consider why debate about the W3C is still continuing, just a it seems to have its house in order, having adopted the WHATWG's successful HTML5 specification

In short, it's because the world kept turning. The WHATWG began their work on what became HTML5 in the mid-2000s, and the details of HTML5 (and related specifications) are still being nutte out in the 2010s. Mobile is exploding, "apps" are taking us back to the platform-specific software world of the 90s, and standards development is still slow, even in this new wow-stuff-is-actually-happening environment we now enjoy.

Can the web keep up in the face of resurgent, platform-specific app development? Has the W3C outlived its usefulness, or is it now finally back on track after years in the wilderness? Here are three perspectives, all from September 2011:

Reform

In "Things the W3C Should Stop Doing" (http://infrequently.org/2011/09/things-the-w3c-should-stop-doing/), Alex Russell, who works for Google on Chrome, argues the W3C needs to drop all its XML and enterprise stuff, and refocus solely on the web. Essentially, drastic reform can save the W3C from irrelevance.

> *The time has come for the W3C to grab the mantle of the web, shake off its self-doubt, and move to a place where doing good isn't measured by numbers of specs and activities, but by impact for web developers.*

Destroy

In "Web Technologies Need an Owner" (http://joehewitt.com/2011/09/22/web-technologies-need-an-owner), Joe Hewitt, who worked on early versions of Firefox, created Firebug, and was responsible for the iPhone Facebook app, argues the web is just another platform, but without anyone taking responsibility for it (unlike Windows, Android and iOS).

> *Let's face facts: the Web will never be the dominant platform. There will forever be other important platforms competing for users' time. To thrive, HTML and company need what those other platforms have: a single source repository and a good owner to drive it. A standards body is not suited to perform this role. Browser vendors are innovating in some areas, but they are stalled by the standards process in so many areas that is impossible to create a platform with a coherent, unified vision the way Apple has with Cocoa or the way Python has with Guido.*

Therefore we should, as Hewitt tweeted (https://twitter.com/joehewitt/status/116292923288592384):

> *[D]issolve the W3C, and run the web like an open source project. No more specs, just commits. Does Linux need a standards body?*

Embrace

Finally, in "The web is a different problem" (http://www.webdirections.org/blog/the-web-is-a-different-problem/) John Allsopp, long standing web evangelist, writer, and speaker, argues that while standards development certainly stalled in the 00s, we've seen an "explosion of innovation at the browser level in the last few years, particularly with CSS3 and more modular specs, and are we really now going to throw the baby out with the bathwater?

> *So, to put it bluntly, I think the problem is overstated. We seem to have arrived at an approach that both enables the exploration and implementation of novel features in browsers, which are also widely adopted across browsers. [...]*
>
> *[But] the web is a different problem. It makes little if any sense to compare innovation of the web ecosystem with that of iOS, Android or other platforms. The web faces challenges far far greater (and has goals far more important). [...]*
>
> *So, rather than generally criticising the W3C, or going so far as calling for its dissolution, we should focus on how well in many ways it has done an almost impossible task—getting companies which are fierce commercial rivals to sit down, work together and agree on core technologies they will each, and all, implement, even while at the same time, these same competitors are involved in significant legal conflicts with one another.*

Whatever we may wish for, sheer inertia is likely to see the W3C maintain its role as the home of we standards development in the coming years (for better or worse), especially now it has brought the WHATWG and HTML5 inside the W3C tent.

How Does New Stuff Get Added To HTML5 Now?

How will HTML5 evolve from here on out? How will the WHATWG implement new HTML features in their "living standard"? They say new HTML features should first appear in browsers (experimentally at least), and *then* be codified into the spec, assuming there's a reasonable use case for them and the editor approves. (See the WHATWG FAQ for more: http://wiki.whatwg.org/wiki/FAQ#Is_there_a_process_for_adding_new_features_to_a_specification.3F.)

This means the HTML spec will capture features as they emerge, rather than dictate new features fro scratch—a somewhat odd stance given the amount of innovation the WHATWG did in the HTML5 spec before *any* browser implementation.

How long will the WHATWG/W3C relationship last? Your guess is as good as mine. Hickson has been openly hostile to the W3C's process at times (http://lists.w3.org/Archives/Public/www-archive/2012Jan/0032.html), and his decisions and refusals continue to be a source of considerable friction on the W3C mailing lists.

At the end of the day, either party can dream up all the specs they like. What really matters is what th browser vendors choose to implement. As far as HTML is concerned, the WHATWG's extremely close relationship with the browser vendors means they'll probably be calling the shots for the foreseeable future.

So, after all that, we're back to HTML. And that wraps up our somewhat sensationalized (and highly condensed) history of HTML5. Or HTML. Or... you get the idea.

TL;DR

In summary, the W3C tried to kill HTML and took us on a decade-long journey to nowhere; some

people from browser vendors formed a group interested in web apps and evolving HTML's forms; they worked outside the W3C on what became HTML5; the W3C realized they were screwed and agreed to use their work; browser vendors are implementing it (or their existing implementations of certain features have been standardized); web standards have become a *Microsoft* marketing buzzword; hell has not frozen over.

What We'll Be Focusing On

HTML5 is a massive specification, filled with mind-numbing detail for browser vendors.

But that detail is actually the best thing about it. Removing the implementation ambiguities has led t more predictable behavior, which is good news for designers and developers alike. (Before that, browser vendors were looking over each other's shoulders to see how parts of the spec were interpreted.)

It's not sexy work, but rather years of careful documentation and clarification by the WHATWG that we can all be grateful for.

The other parts of HTML5 very much reflect its origins as Web Applications 1.0 and Web Forms 2.0 We'll touch on the web app stuff in chapter twelve, and look at the web forms in chapter eight.

As designers, the biggest point of interest are the changes and additions to the actual markup side of HTML. And that's what we'll focus on: semantics, forms, graphics, and audio/video.

We'll also touch on the new features for web apps in HTML5, which we'll hopefully see in our Content Management Systems sooner rather than later.

Most importantly though, we'll be looking at the *ideas* in markup and the practical—sometimes critical—dos and don'ts of HTML5 along the way.

Let's jump in, and look at how we start a document in HTML5.

A Doctype For Every Occasion (And The Other Bits)

Let's start with the first line of a web page. It's now just:

```
<!doctype html>
```

That's it. Short, memorable, and triggers standards mode in all major browsers (including IE6). It's also case insensitive.

In HTML5 the opening `<html>` tag has also been simplified to:

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

Browsers will cope without the lang attribute, but it's good practice to specify the page's primary language—especially for non-English pages. (See this helpful article on declaring languages in HTML5: http://nimbupani.com/declaring-languages-in-html-5.html.)

Next comes the `<head>` tag, which will contain our `<title>`, `<meta>`, CSS and JavaScript tags as per usual. You don't actually need to specify `<head>` tags if you want to be ultra minimal (see Bruce Lawson's minimal HTML5 document discussion here: http://www.brucelawson.co.uk/2010/a-minimal-html5-document/), but we will.

Inside our `<head>` tags we have:

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

This specifies the character encoding for the page. Again, it's been reduced to the simplest form possible in HTML5. You should *always* specify this for security reasons (there's a technical discussion here: http://code.google.com/p/doctype-mirror/wiki/ArticleUtf7), and it should come before the `<title>` tag.

```
<meta name="description" content="My HTML5 Website">
```

This hasn't changed. Google and other search engines sometimes use this tag in their search results pages, but not for rankings. (You can forget all about `<meta content="keywords">` though. Search engines have been ignoring it for years. We'll look at markup and SEO in chapter six.)

For more on meta tags Google does understand, see: http://www.google.com/support/webmasters/bin/answer.py?answer=79812.

The `<title>` tag hasn't changed.

To link CSS and JavaScript files, we can just use:

```
<link rel="stylesheet" href="styles.css">
```

And:

```
<script src="myscript.js"></script>
```

There's no need to specify `type="text/css"` or `type="text/javascript"` anymore—the browsers assume it anyway.

We can start using these techniques now. There's no harm in them, they just make it simple enough t start writing our documents from memory. (The old techniques will continue to work though—probably forever.)

So, a basic HTML5 page (with basic body content) looks like this:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="description" content="My HTML5 Website">
  <title>My HTML5 page</title>
  <link rel="stylesheet" href="mystyles.css">
  <script src="myscript.js"></script>
</head>

<body>
  <h1>My HTML5 Page</h1>
</body>
</html>
```

As you can see it's pretty much what we're used to, just simpler.

Formatting Changes In HTML5

A few things to note about how we write HTML in HTML5:

- **Quotes are optional.** You no longer need to quote attribute values, so you can write `<meta charset=utf-8>` or `<div class=myclass>` if you like. Personally I prefer quoting values, but HTML5 leaves it up to you.
- **It's case-insensitive.** You can write your markup in upper or lowercase, or even a mix like `<DiV ClAsS=VaLuE>` if you really hate your coworkers and/or feel nostalgic for YoUr WaCkY MySpAcE days.
- **Closing slashes are optional.** You no longer need to close standalone tags with a closing slash (e.g. `<meta charset=utf-8 />`). As you probably guessed, this was a relic of the move to XML Likewise, `<br>` and `<br />` are both perfectly valid—it's up to you.

If you're a stickler for XHTML's stricter syntax (always writing in lowercase, quoting attribute valu and closing standalone tags), you can keep doing it—it will always be happily supported.

What About A HTML5 Shim And CSS For The New Elements?

HTML5 introduces new elements such as `<nav>`, `<header>`, `<article>`, `<section>`, and so on. These sound fine in theory, but are terrible in practice.

To support these elements in IE6-8, others suggest you include a small script that tells IE6-8 these elements exist and to use whatever styles you specify for them (it will leave them unstyled otherwise I don't recommend using these new elements, so we don't need the HTML5 shim. (If you really want to use them, here's the code to do it: http://code.google.com/p/html5shiv/. But seriously, don't use the new elements. You'll thank me later.)

You also need to set the new elements to `display: block;`, as shown in this HTML5doctor.com boilerplate: http://html5doctor.com/html-5-boilerplates/. Again, don't use these elements. (You'll see why in th next two chapters.)

What About The HTML5 Boilerplate And Modernizr?

If you want an everything-and-the-kitchen-sink boilerplate for new HTML5 pages, check out http://html5boilerplate.com/ and the markup documentation https://github.com/paulirish/html5-boilerplate/wiki/The-marku (There's more documentation in the wiki.)

While I appreciate the effort they've put into the HTML5 Boilerplate, if you're just finding your way with HTML5 it's pretty intense. I prefer to start simple and work with my own bare-minimum approach. But if you prefer the start-with-everything-and-delete-what-you-don't-want approach, the HTML5 Boilerplate may be right up your alley.

Modernizr (http://www.modernizr.com/) is a handy script for detecting support for HTML5 and CSS3 features. (It doesn't *add* support, it only *detects* it.) It's become a staple for designers who live on the bleeding edge and experiment with new features, so if that's what you're interested in check it out. (We'll talk more about Modernizer, and the merits of *feature* detection rather than *browser* detection

when we look at HTML5's web application features in chapter twelve.)

Well, that was easy. Almost *too* easy. Now let's take a big left turn into the proverbial ditch that is th
new structural tags.

New Structural Elements - This Isn't Going To End Well (Plus, Controversy!)

One of the most common tasks web designers do is mark up page structure, which usually consists of a header, footer, navigation, sidebar and content area. It's the sort of thing you can probably do blindfolded and handcuffed to your chair after being spun around for five minutes.

HTML5 introduces a handful of new elements to help us define the structure of a given web page, such as `<section>`, `<article>`, `<nav>`, `<aside>`, `<header>`, and `<footer>`.

We shouldn't use them. They were made up on a whim by (probably) *one guy* in 2004 and even *he* seems to have forgotten what their purpose is.

If that's all you needed to know, great. Keep using `<div>`s with meaningful class and ID names, and appropriate `<h1>`-`<h6>` headings. They'll be valid forever (more or less), and you're not missing out on anything.

However, I suggest using some non-HTML5 features when marking up documents, such as ARIA attributes for blind and sight-impaired users and microdata schemas (when appropriate) for search engine results. (We'll talk more about these in later chapters.)

Nevertheless, we'll tackle these new elements in depth because everyone gets them wrong. And we'll set the record straight on how they found their way into the spec and their real intended purpose, which involves a radically different way of structuring your pages.

A Little Taste Of Pain

Here are just some of the problems these new structural elements introduce:

- They give terms web designers already use (such as header and footer) new uses, while claiming to be just doing what web designers are already doing.
- They introduce a new method of structuring documents that's vague, complicated, and unnecessary.
- They seriously hurt accessibility for some users (specifically those using IE6, IE7, and even IE8 with JavaScript switched off).
- They introduce broad, unclear, poorly-defined use cases that will make web standards harder to learn (and harder to teach).

These are serious problems that hurt, rather than help, web standards. Markup should be lightweight, easy to learn, and easy to apply. It should *not* require mental gymnastics to try and work out what to use where.

But these new structural tags have created a strange, quasi-religious experience where you have to consult the high priests (the HTML5 gurus) for their interpretation of vague religious texts (the HTML5 spec) just to mark up a darn web page.

"But, but... these elements are in the official HTML5 spec! Surely there *must* be a good reason for them?"

Read on...

Where Did These Elements Come From?

Quiz question: How were these elements added to the HTML5 spec?

a. Experts considered various use cases, weighed up various options and alternatives, and after extensive consultation and careful deliberation included the most important ones.

b. The community of web designers and HTML authors (such as you and me) cried out for certain elements to enable particular functionality, and after much discussion the community came up with a shortlist of necessary elements.

c. A scientific, research-based approach was taken, where markup patterns were studied "in the wild" and codified into a bunch of new elements.

d. Some markup wonks thought they'd be a good idea and threw them in the spec 7+ years ago.

And the answer is… (d).

"But I read in [insert HTML5 book of your choice here] that it was more like answer (c). The WHATWG studied real-world usage of ID and class names, and that's how they came about!"

We'll get to that.

I was intrigued about who added these elements, when they added them, and why. So I put those