Building Web Pages With HTML 5

Depending on who you ask, HTML 5 is either the next important step toward creating a more semantic web or a disaster that's going to trap the web in yet another set of incomplete tags and markup soup.

The problem with both sides of the argument is that very few sites are using HTML 5 in the wild, so the theoretical solutions to its perceived problems remain largely untested.

That said, it isn't hard to see both the benefits and potential hang-ups with the next generation of web markup tools.

What's different about HTML 5?

First off, what do we mean by HTML 5? Ideally, we mean the whole thing — new semantic structural tags, API specs like canvas or offline storage and even some new inline semantic tags. However, for practical reasons (read: browser support issues) we're going to limit this intro to just the structural tags. As cool as Canvas, offline storage, native video or the geolocation APIs are, they aren't supported consistently across all the browsers yet.

"But wait," you say, "most browsers don't support the new structural elements either!" This is true, but the vast majority of them will happily accept any tag you want to make up. Even IE 6 can deal with the new elements, though if you want to apply styles using CSS, you'll need a little JavaScript help.

The one thing to keep in mind when you're applying styles to the new tags is that unknown tags have no default style in most browsers. They're also treated as inline elements. However, because most of the new HTML 5 tags are structural, we'll want them be behave like block elements. The solution is make sure that you include `display:block;` in your CSS styles.

To help make some sense of what's new in HTML 5 today, we're going to dive right in and start using some of the new structural elements.

Finally, a doctype anyone can remember

The first thing we need to do to create an HTML 5 document is use the new doctype. Now, if you've actually memorized the HTML 4 or XHTML 1.x doctypes, you're better monkeys than us. Whenever we start a new page we have to bring up an old one and cut and paste the doctype definition over.

It's a pain, which is why we love the new HTML 5 doctype. Are you ready? Here it is:

```
<!DOCTYPE html>
```

Shouldn’t be too hard to commit that to memory. Simple and obvious. Case insensitive.

The idea is to stop versioning HTML so that backwards compatibility is easier. Whether or not that pans out in the long run is a whole other story, but at least it saves you some typing in the mean time.

Semantic structure at last

OK, we have our page defined as an HTML 5 document. So far so good. Now what are these new tags we’ve been hearing about?

Before we dive into the new tags, consider the structure of your average web page, which (generally) looks something like this:

```
<html>

    <head>

    ...stuff...

    </head>

    <body>



        <div id="header">

            <h1>My Site</h1>

        </div>

        <div id="nav">

            <ul>



```

```
                <li>Home</li>

                <li>About</li>

                <li>Contact</li>

            </ul>



        </div>

        <div id=content>

            <h1>My Article</h1>

            <p>...</p>

        </div>



        <div id="footer">

            <p>...</p>

        </div>

    </body>

</html>
```

That’s fine for display purposes, but what if we want to know something about what the page elements contain?

In the above example, we’ve added IDs to all our structural divs. This is a fairly common practice among savvy designers. The purpose is two-fold — first, the IDs provide hooks which can be used to apply styles to specific sections of the page and, second, the IDs serve as a primitive, pseudo-semantic structure. Smart parsers will look at the ID attributes on a tag and try to guess what they mean, but it's hard when ID names are different on every site.

And that's where the new structural tags come in.

Recognizing that these IDs were common practice, the authors of HTML 5 have gone a step further and made some of these elements into their own tags. Here's a quick overview of the new structural tags available in HTML 5:

`<header>`

The header tag is intended as a container for introductory information about a section or an entire webpage. The `<header>` tag can include anything from your typical logo/slogan that sits atop most pages, to a headline and lede that introduces a section. If you've been using `<div id="header">` in your pages, that would be the tag to replace with `<header>`.

`<nav>`

The nav element is pretty self-explanatory — your navigation elements go here. Of course what constitutes navigation is somewhat debatable — there's primary site navigation, but in some cases there may also be page navigation elements as well. The WHATWG, creators of HTML 5, recently amended the explanation of `<nav>` to show how it could be used twice on the same page.

The short story is that if you've been using a `<div id="nav">` tag to hold your page navigation, you can replace it with a simple `<nav>` tag.

`<section>`

Section is probably the most nebulous of the new tags. According the HTML 5 spec, a section is a thematic grouping of content, typically preceded by a header tag, and followed by a footer tag. But sections can also be nested inside of each other, if needed.

In our example above, the div labeled "content" would be a good candidate to become a section. Then within that section, depending on the content, we might have additional sections.

`<article>`

According the WHATWG notes, the article element should wrap “a section of content that forms an independent part of a document or site; for example, a magazine or newspaper article, or a blog entry.”

Keep in mind that you can have more than one article tag on the page; for example a blog homepage might have the last ten articles, each wrapped in an article tag. Articles can also be broken into sections using the section tag, though you’ll want to be somewhat careful when planning your structure otherwise you’re liable to end up with some ugly tag soup.

`<aside>`

Another fairly nebulous tag, the aside element is for content that is “tangentially related to the content that forms the main textual flow of a document.” [1]That means a parenthetical remark, inline footnotes, pull quotes, annotations or the more typical sidebar content like you see to the right of this article.

According to the WHATWG’s notes it seems like `<aside>` would work in all those cases, despite the fact that there’s considerable difference between a pull quote and tag cloud in your sidebar.

Hey, no one said HTML 5 was perfect!

`<footer>`

Footer should also be self-explanatory, except perhaps that you can have more than one. In other words, sections can have footers in addition to the main footer generally found at the bottom of most pages.

Putting it all together

OK, let’s rewrite our original example using the new tags:

```
<!DOCTYPE html>

<html>

    <head>

    ...stuff...

    </head>

```

```html


<body>

    <header>

        <h1>My Site</h1>

    </header>

    <nav>



        <ul>

            <li>Home</li>

            <li>About</li>

            <li>Contact</li>



        </ul>

    </nav>

    <section>

        <h1>My Article</h1>

        <article>



            <p>...</p>

        </article>

        </section>

        <footer>

            <p>...</p>



        </footer>

    </body>

</html>
```

Much cleaner and easier to understand, right? A couple of notes: we could have wrapped our `<h1>My Article</h1>` headline in header tags. I opted not to since the h1 element already conveys the heading, but if you also had a pub date, byline or other data atop your post, the collective group of tags would be a good candidate for adding a header container tag.

Also note that we could add a second footer element below the article element to contain things like next/prev navigation, related posts or other content.

Styling the new tags

In most browsers, all you need to do is simply define your styles as you normally would in order to apply styles to these new elements. But make sure to add the `display:block;` rule to every element — for now anyway. In time, as browsers begin standardizing support for the new elements, that won't be necessary.

For example let's apply some styles to our header:

```
header {

    display: block;

    font-size: 36px;

    font-weight: bold;

}
```

Keep in mind that you can still add class and ID attributes to these tags. So, if you wanted to style one navigation section separately, you'd simply add a class or ID to the tag like so:

```
<nav class="main-menu">
```

Then you can apply a style:

```
nav.main-menu {

    font-size: 18px;

}
```

Compatibility with older browsers

But wait, what about IE? None of these styles are working in IE 6. If you still need to support legacy browsers like IE, there is a fix. IE 6 parses and displays these tags just fine, but it won't apply any CSS to them. The fix is to use a bit of JavaScript.

All we need to do to get IE to style our HTML 5 tags is use the `createElement` method so IE 6 becomes aware of the new tags. Add this bit to the head of your HTML 5 file. Or alternatively, you can save it in a separate file and include it that way.

```
<script>

  document.createElement('header');

  document.createElement('nav');

  document.createElement('section');

  document.createElement('article');

  document.createElement('aside');

  document.createElement('footer');

</script>
```

I know what you’re thinking: “Hey, you didn’t specify a MIME type for that script tag.”

You don’t need to do that in HTML 5. In HTML 5, all scripts are assumed to be `type="text/javascript"` so there’s no need it clutter up your script tags with attributes anymore (unless your script is something other than JavaScript).

That fixes the IE problems, but we’re not out of the woods just yet. It turns out that there’s a bug in the Gecko rendering engine that causes Firefox 2 and some versions of Camino to choke on these tags as well.

There are two ways to work around this bug, neither of which are ideal.

Bear in mind though that Firefox 2 usage stats are quickly falling below 10 percent of all web traffic, so simply ignoring this bug might be a possibility depending on your site’s audience.

OK, now you can use HTML 5, but should you?

The short answer is *yes*.

The longer answer is that it depends on the site. If you’re charged with recreating the CNN homepage, well, you might want to hold off for a bit until browser support improves. But if you’re revamping your blog, we say go ahead. There are even some WordPress plugins that can help if you’re using that particular publishing system.

Also, check out the sites featured on HTML 5 Gallery and view source to see what they’re doing.

However, if IE’s shortcoming are holding you back, consider this: even Google is using the HTML 5 doctype on their main search page. Even if you don’t use all the new structural tags you can at least take advantage of things like shorter script declarations, and some of the non-structural semantic tags that we’ll cover next time around

Add Semantic Value to Your Pages With HTML 5

In the part one of our HTML 5 tutorial, we looked at some of the language's new structural markup tags that are designed to reduce the "`<div>`-soup" of HTML 4 and add semantic meaning to your page's layout.

But not every new tag in HTML is strictly structural. There are other tags that also add valuable semantic meaning to your pages in non-structural ways. Today, in part two, we'll take a look at how to use them and what they can do for your content.

The `<time>` tag

There have been a couple of cases recently where old news stories or blog posts have been recycled or republished as if they were new, breaking stories. Sometimes this happens through human error, or sometimes just through news search crawler error. In some cases, the damage was worse than just making the perpetrators look bad — it actually sent a few stocks into a tailspin.

While there's not much HTML5 can do to help human error, the `<time>` tag could have saved the search engine spiders.

Despite the timelessness of the web, most things published online carry some sort of date/time stamp, and that's exactly the data the `<time>` tag is intended to convey. The most basic usage looks something like this:

```
Published <time>12/20/2009</time>
```

However, the `<time>` tag also has an attribute called `datetime` that makes it even more useful because it allows you to display a nice date format to your human readers inside the tag. More importantly, it also provides search engine spiders with something more useful to them. Let's amend the above code slightly:

```
Published <time datetime="2009-12-20T17:22:28-05:00">

Thursday, December 20, 2009 at 10:28PM EST</time>
```

Now, you might be thinking the terribly unreadable attribute syntax (also known as the ISO 8601 date format) in our example is way more of a pain than it's worth. Indeed, if you're hand coding, it probably is. But any good content management system should be able to output this format for you. For example, to make this work in WordPress, you would simply add this bit of HTML 5 to your template:

```
Published <time datetime="

```

```
<?php the_time('c'); ?>">



<?php the_time('l, F j, Y'); ?>

</time>
```

A couple of things to note. According to the HTML 5 spec, if you omit the `datetime` attribute, then the tag must contain a valid date string. The valid part means it must be in the format: year, month, day.

Also worth mentioning: when you use the `datetime` attribute, the tag itself can be empty. While both of these variants are OK according to the spec, we suggest you use both a machine readable `datetime` for the attribute and human readable format inside the actual tag.

The `<figure>` tag

The figure tag was designed to make image embedding more descriptive and easier to recognize. The figure tag isn't limited to images though, it can also be used for diagrams, code snippets and other visual elements.

But `<figure>` also isn't intended for every image. The key defining phrase in the spec is any illustrative element "that could, without affecting the flow of the document, be moved away from that primary content, e.g. to the side of the page, to dedicated pages, or to an appendix."

The most common use case is an image that you refer to in the body of an article. To use `<figure>`, you simply wrap your image tag and add a legend like this:

```
<figure>



 <img src="/images/tpsreport.jpg"

      alt="TPS Report, July 2009">

 <legend>July 2009 TCP Reports</legend>

</figure>
```

The `<figure>` tag can also be used for code snippets. For example, to markup the code snippet in the previous example, you'd have something like this:

```
<figure>

 <legend>Using the HTML 5 Figure tag</legend>

 <pre><code>

 <figure>

 <img src="/images/tpsreport.jpg"

      alt="TPS Report, July 2009">



 <legend>July 2009 TPS Reports</legend>

</figure>

 </code></pre>

</figure>
```

The most important thing to remember about the `<figure>` tag is that you don't need to use it for every image you want to embed in a page — just those cases where you refer to the image in question to illustrate a point or use as an example in your article.

The `<dialog>` tag

The dialog element in HTML 5 is for marking up conversations — a chat transcript, an interview, a bit dialog from a screenplay, and so on. The `<dialog>` tag works pretty much like a definition list, but specifically denotes dialog.

For example, here's the famous Abbot and Costello "Who's on First" routine represented in HTML 5:

```
<dialog>

 <dt> Costello </dt>

 <dd> Look, you gotta first baseman? </dd>

 <dt> Abbott </dt>



 <dd> Certainly. </dd>

 <dt> Costello </dt>

 <dd> Who's playing first? </dd>



 <dt> Abbott </dt>

 <dd> That's right. </dd>

</dialog>
```

You could also use `<dialog>` for something like a blog's comment section, since comments are, generally speaking, a conversation between a writer and an audience. Something like this:

```
<dialog>

 <dt id="comment_1">Commenter Name</dt>

 <dd>HEllo. I really like your site. want to buy some WOW gold?</dd>

</dialog>
```

Revisiting the `<aside>` tag

In part one of this walk-through of HTML5, we looked at the `<aside>` tag as way to mark up a sidebar or other structural content wells. But `<aside>` can also be used in less structural ways. For example, it can also be used to highlight a quote within an article.

Here's one of the official examples (also note the use of the `<q>` tag, a little-used HTML 4 element):

```
<p>He later joined a large company, continuing on the same work.



<q>I love my job. People ask me what I do for fun when I'm not

at work. But I'm paid to do my hobby, so I never know what to

answer. Some people wonder what they would do if they didn't

have to work... but I know what I would do, because I was

unemployed for a year, and I filled that time doing exactly what

I do now.</q></p>



<aside>

 <q> People ask me what I do for fun when I'm not at work. But I'm

 paid to do my hobby, so I never know what to answer. </q>

</aside>



<p>Of course his work - or should that be hobby? - isn't his

only passion. He also enjoys other pleasures.</p>
```

This is essentially what newspapers and magazines refer to as a “pull-quote,” an excerpt from an article that’s been “pulled out” and typeset separately (usually a larger font). Other examples include the traditional, Hamlet-style aside, but not, according to the spec, parenthetical asides that fit the normal flow of the document.

Who knew HTML 5 would require a linguistics degree? Like we said in the first tutorial, the spec isn’t perfect.

The `<mark>` tag

The mark tag is intended to denote text that is particularly relevant. It’s a bit like `<strong>` but instead of denoting the importance of its contents, it denotes relevance.

The `<mark>` tag isn’t one that you’ll probably use very often, but it does have one particularly handy use case: highlighting search terms that brought a visitor to your site. You’ve probably seen sites that do this, somewhat like the way Google highlights your search terms when you access a cached page rather than the live site. That’s the perfect time to use the `<mark>` tag.

Conclusion

That about does it for the rest of the semantic tags in HTML 5, next time around we’ll start playing with the API related tags like `<audio>`, `<video>` and `<canvas>`