The Power of HTML 5 and CSS 3

By Jeff Starr on July 19, 2009 • 46 Comments »

Web designers can do some pretty cool stuff with HTML 4 and CSS 2.1. We can structure our documents logically and create information-rich sites without relying on archaic, table-based layouts. We can style our web pages with beauty and detail without resorting to inline `<font>` and `<br>` tags. Indeed, our current design methods have taken us far beyond the hellish era of browser wars, proprietary protocols, and those hideous flashing, scrolling, and blinking web pages.

As far as we've come using HTML 4 and CSS 2.1, however, we can do better. We can refine the structure of our documents and increase their semantic precision. We can sharpen the presentation of our stylesheets and advance their stylistic flexibility. As we continue to push the boundaries of*existing* languages, HTML 5 and CSS 3 are quickly gaining popularity, revealing their collective power with some exciting new design possibilities.

Goodbye `<div>` soup, hello semantic markup

In the past, designers wrestled with semantically incorrect table-based layouts. Eventually, thanks to revolutionary thinking from the likes of Jeffrey Zeldman and Eric Meyer, savvy designers embraced more semantically correct layout methods, structuring their pages with `<div>` elements instead of tables, and using external stylesheets for presentation. Unfortunately, complex designs require significant differentiation of underlying structural elements, which commonly results in the "`<div>`-soup" syndrome. Perhaps this looks familiar:

```
<div id="news">

   <div class="section">

      <div class="article">

         <div class="header">

            <h1>Div Soup Demonstration</h1>
            <p>Posted on July 11th, 2009</p>

         </div>

         <div class="content">

            <p>Lorem ipsum text blah blah blah.</p>

            <p>Lorem ipsum text blah blah blah.</p>

            <p>Lorem ipsum text blah blah blah.</p>

         </div>

         <div class="footer">

            <p>Tags: HMTL, code, demo</p>

         </div>

      </div>

      <div class="aside">

         <div class="header">

            <h1>Tangential Information</h1>

         </div>

         <div class="content">

            <p>Lorem ipsum text blah blah blah.</p>

            <p>Lorem ipsum text blah blah blah.</p>

            <p>Lorem ipsum text blah blah blah.</p>

         </div>

         <div class="footer">

            <p>Tags: HMTL, code, demo</p>

         </div>

      </div>
```

```
   </div>

</div>
```

While slightly contrived, this example serves to illustrate the structural redundancy of designing complex layouts with HTML 4 (as well as XHTML 1.1 et al).
Fortunately, HTML 5 alleviates `<div>`-soup syndrome by giving us a new set of structural elements. These new HTML 5 elements replace meaningless `<div>`s with more semantically accurate definitions, and in doing so provide more “natural” CSS hooks with which to style the document. With HTML 5, our example evolves:

```
<section>

   <section>

      <article>

         <header>

            <h1>Div Soup Demonstration</h1>

            <p>Posted on July 11th, 2009</p>

         </header>

         <section>

            <p>Lorem ipsum text blah blah blah.</p>

            <p>Lorem ipsum text blah blah blah.</p>

            <p>Lorem ipsum text blah blah blah.</p>

         </section>

         <footer>

            <p>Tags: HMTL, code, demo</p>

         </footer>

      </article>

      <aside>

         <header>
```

```
            <h1>Tangential Information</h1>

         </header>

         <section>

            <p>Lorem ipsum text blah blah blah.</p>

            <p>Lorem ipsum text blah blah blah.</p>

            <p>Lorem ipsum text blah blah blah.</p>

         </section>

         <footer>

            <p>Tags: HMTL, code, demo</p>

         </footer>

      </aside>

   </section>

</section>
```

As you can see, HTML 5 enables us to replace our multitude of `<div>`s with semantically meaningful structural elements. This semantic specificity not only improves the underlying quality and meaningfulness of our web pages, but also enables us to remove many of the `class` and `id` attributes that were previously required for targeting our CSS. In fact, CSS 3 makes it possible to eliminate virtually all `class` and `id` attributes.

Goodbye `class` attributes, hello clean markup

When combined with the new semantic elements of HTML 5, CSS 3 provides web designers with god-like powers over their web pages. With the power of HTML 5, we obtain significantly more control over the *structure* of our documents, and with the power of CSS 3, our control over the *presentation*of our documents tends toward infinity.

Even without some of the advanced CSS selectors available to us, the new variety of specific HTML 5 elements enable us to apply styles across similar sections *without* the need for defining `class` and `id`attributes. To style our previous `<div>`-soup example, we would target the multitude of attributes via the following CSS:

```
div#news    {}

div.section {}
div.article {}

div.header  {}

div.content {}

div.footer  {}

div.aside   {}
```

On the other hand, to style our HTML 5 example, we may target the various documents regions directly with this CSS:

```
section {}

article {}

header  {}

footer  {}

aside   {}
```

This is an improvement, but there are several issues that need addressed. With the `<div>` example, we target each area specifically through use of `class` and `id` attributes. Using this logic allows us to apply styles to each region of the document, either collectively or individually. For example, in the`<div>` case, `.section` and `.content` divisions are easily distinguished; however, in the HTML 5 case, the `section` element is used for both of these areas and others as well. This is easily resolved by adding specific attribute selectors to the different `section` elements, but thankfully, we instead may use a few advanced CSS selectors to target the different `section` elements in obtrusive fashion.

Targeting HTML-5 elements without classes or ids

Rounding out the article, let's look at some practical examples of targeting HTML-5 elements without`class`es or `id`s. There are three types of CSS selectors that will enable us to target and differentiate the elements in our example. They are as follows:

- The descendant selector [CSS 2.1]: **E F**
- The adjacent selector [CSS 2.1]: **E + F**
- The child selector [CSS 2.1]: **E > F**

Let's have a look at how these selectors enable us to target each of our document sections without the need for `class`es or `id`s.

Targeting the outermost `<section>` element
Due to the incompleteness of our example, we will assume that the outermost `<section>` element is*adjacent* to a `<nav>` element which itself is a direct descendant of the `<body>` element. In this case, we may target the outermost `<section>` as follows:

```
body nav+section {}
```

Targeting the next `<section>` element
As the only direct descendant of the outer `<section>`, the next `<section>` element may be specifically targeted with the following selector:

```
section>section {}
```

Targeting the `<article>` element
There are several ways to target the `<article>` element specifically, but the easiest is to use a simple descendant selector:

```
section section article {}
```

Targeting the `header`, `section`, and `footer` elements
In our example, each of these three elements exists in two locations: once inside the `<article>`element and once inside the `<aside>` element. This distinction makes it easy to target each element individually:

```
article header {}

article section {}

article footer {}
```

..or collectively:

```
section section header {}

section section section {}

section section footer {}
```

So far, we have managed to eliminate all `class`es and `id`s using only CSS 2.1. So why do we even need anything from CSS 3? I’m glad you asked..

Advanced targeting of HTML 5 with CSS 3

While we have managed to target every element in our example using only valid CSS 2.1, there are obviously more complicated situations where the more advanced selective power of CSS 3 is required. Let's wrap things up with a few specific examples showing how CSS 3 enables us to style any element*without* extraneous `class` or `id` attributes.

Targeting all posts with a unique post ID
WordPress provides us a way of including the ID of each post in the source-code output. This information is generally used for navigational and/or informational purposes, but with CSS 3 we can use these existing yet unique ID attributes as a way to select the posts for styling. Sure, you could always just add a `class="post"` attribute to every post, but that would defeat the point of this exercise (plus it's no fun). By using the "substring matching attribute selector," we can target all posts and their various elements like this:

```
article[id*=post-] {}           /* target all posts */

article[id*=post-] header h1 {} /* target all post h1 tags */

article[id*=post-] section p {} /* target all post p tags */
```

Now that's just sick, and we can do the same thing for numerically identified comments, enabling us to apply targeted styles to associated constructs:

```
article[id*=comment-] {}           /* target all comments */

article[id*=comment-] header h1 {} /* target all comment h1 tags */

article[id*=comment-] section p {} /* target all comment p tags */
```

Target any specific section or article
Many sites display numerous of posts and comments. With HTML 5, the markup for these items consists of repetitive series of `<section>` or `<article>` elements. To target *specific* `<section>` or`<article>` elements, we turn to the incredible power of the "`:nth-child`" selector:

```
section:nth-child(1) {} /* select the first <section> */

article:nth-child(1) {} /* select the first <article> */



section:nth-child(2) {} /* select the second <section> */

article:nth-child(2) {} /* select the second <article> */
```

In a similar manner, we may also target specific elements in *reverse* order via the “`:nth-last-child`” selector:

```
section:nth-last-child(1) {} /* select the last <section> */

article:nth-last-child(1) {} /* select the last <article> */



section:nth-last-child(2) {} /* select the penultimate <section> */

article:nth-last-child(2) {} /* select the penultimate <article> */
```

More ways to select specific elements
Another way to select specific instances of HTML-5 elements such as `<header>`, `<section>`, and`<footer>`, is to take advantage of the “`:only-of-type`” selector. With these HTML-5 elements appearing in multiple locations in the web document, it may be useful to target elements that appear only once within a particular parent element. For example, to select only `<section>` elements that are the only `<section>` elements within another `<section>` element (insane, I know), as in the following markup:

```
<section>

   <section></section>

   <section>

      <section>Target this section</section>

   </section>

   <section>

      <section>Target this section</section>

   </section>

   <section>

      <section>But not this section</section>

      <section>And not this section</section>

   </section>

   <section></section>
```

```
</section>
```

..we could simply use the following selector:

```
section>section:only-of-type {}
```

Again, you could always add an `id` to the target element, but you would lose the increased scalability, maintainablity, and clarity made possible with an absolute separation of structure and presentation.

The take-home message for these examples is that CSS 3 makes it possible to target virtually anyHTML-5 element *without* littering the document with superfluous presentational attributes.

Much more to come

With the inevitable, exponential rise in popularity of both HTML 5 and CSS 3, designers can look forward to many new and exciting possibilities for their web pages, applications, and scripts. Combined, these two emerging languages provide designers with immense power over the structure and presentation of their web documents. In my next article on this topic, we will explore some of the controversial aspects of HTML 5 and also examine some of the finer nuances of CSS 3. Stay tuned!

Note to WordPress users

You can start using HTML 5 *right now*. To see a live, working example of a WordPress theme built entirely with HTML 5, CSS, and of course PHP, drop by the Digging into WordPress site and visit our newly revamped Theme Playground. There you will find my recently released H5 WordPress Theme Template available for immediate download. And while you're there, be sure to secure your copy of*Digging into WordPress*, coming this Fall.

Related articles

- Rethinking Structural Design with New Elements in HTML 5
- CSS/(X)HTML Tutorial: Hovering Accessibility Jump Menu
- HTML Frames Notes Plus
- Exploring the (X)HTML Link Element
- biz koanz / art haiku
- The Friendliest Link Targets in the Neighborhood
- Bare-Bones HTML/XHTML Document Templates