» E-mail article
» Printable version

Building your first CSS site

by Michael Koch

This article is for web designers and developers who have little or no experience with cascading style sheets (CSS) but would like to move from building table-based websites to building CSS sites. However, don't expect tips on how to design your first CSS site, how to use specific CSS layouts, or how to handle content placement issues. Instead, you'll get some pointers on how to jump-start building your first CSS site.

The best ways to wrap your head around CSS design concepts are to dig through some reference materials and tutorials, to look closely at other CSS sites for inspiration and examples of what you can do with the standard CSS layout techniques, and to download some CSS sample pages or layout templates and then start changing the numbers in the CSS file to see what happens.

Good resources to help you get started with CSS include W3 Schools, DMXzone, and the Dreamweaver Developer Center. For insights into the intricacies and power of building standards-based CSS sites, you may want to check out the work of CSS evangelists Eric Meyer and Jeffrey Zeldman. And for the ultimate CSS experience and inspiration, kick back, relax, and browse CSS Zen Garden, a showcase of simply awesome CSS designs that are all based on the same HTML template.

Think inside the box

The whole point of using CSS is that it enables you to separate style (or design) from content on your web pages, which in turn helps your pages load faster and makes them more accessible to a wider variety of devices and assistive technologies, such as screen readers. CSS also makes it easier for you to implement site-wide changes and maintain consistency across your site because you can quickly alter the appearance and layout of your entire site just by editing a single CSS file. If you have ever tried changing the font or color of all the headings in all your old-fashioned HTML pages, you will appreciate the time and work you can save by using style sheets. But to make the most of these benefits, you may have to change the way you look at your HTML.

HTML tags were designed to define the content of a document. Tags such as <h1>, <p>, or <table> were supposed to instruct browsers that "this is a header," "this is a paragraph," or "this is a table" and nothing more. The actual layout of the document was supposed to be handled by the browser. However, as the major browser developers continued to inflate the original HTML specification with new HTML tags and attributes (such as the <font> tag and color attribute), it became increasingly difficult to build sites where the content of the HTML files was clearly separated from the presentation, which created all kinds of accessibility issues and other problems. (It's debatable how much blame for these problems, if any, rests with the emergence of WYSIWYG editors as well.)

To solve these problems, the World Wide Web Consortium (W3C)—the consortium responsible for standardizing HTML—created a complementary mark-up system called cascading style sheets to make it easy to change the appearance of an HTML page without affecting its HTML structure.

If you're adept at building traditional table-based HTML pages, the challenge now becomes organizing the hierarchical structure that lies beneath your beautiful designs and learning how to position content on the page using the CSS box model instead of nested tables and cells. According to this model, every element in your document is considered a rectangular box.

A box may contain any number of nested boxes, creating a hierarchy of boxes that corresponds to the nesting of page elements (for example, a list may be nested within a paragraph, which in turn may be nested within the main content area of the page, which in turn is nested within the body tags of your page, with the browser window serving as the root in this hierarchy of boxes). Although this may sound easy enough, it does take some mental aerobics to learn how to organize the content of your pages properly using the methods and rules that govern these boxes.

Identify the structure of your pages

The first step to building a CSS site, assuming you have already created a visual blueprint or comps for the site you want to build, is to strip your design of all its color and beauty and to focus instead on the actual content of the pages — that is, those elements that will be visible to users who access your site using assistive technologies, handheld devices, or alternate technologies. (After all, content is what users are after when they come to your site.) The experience users have when interacting with your site is of secondary importance (although a good user experience will encourage users to hang around or come back for more).

The goal is to organize and describe the smaller meaningful units that make up your content, such as titles, headings, paragraphs, lists, and tables, as well as the containers that hold and organize these elements. In short, you want to note the semantics of your HTML. This is easily done by marking up an Adobe PDF file or a hard copy of the site's blueprints. Simply jot down the HTML tags you need next to the individual content elements. For example, if you want to create a header, use one of the HTML heading elements (<h1>, <h2>, <h3>, and so on). Paragraphs are <p> elements. The navigation bar, like most navigation bars, is best marked up as an unordered list (<ul>). And lists may be treated as unordered lists, ordered lists (<ol>), or definition lists (<dl>), depending on the context. Remember, each element provided in the W3C's extended HTML 4 or XHTML specification has been designed to be used in a specific way, and each has a specific meaning. The idea is to structure your document in a hierarchical manner that is easily read not only by humans but by machine readers as well.

Next, you may want to draw rectangles around elements that make up a particular content area (such as the main content area, the site navigation, and the header and footer) to help you visualize the containers that hold your content and how they relate to each other. This will make it easier for you to understand the layout challenges when you start to define the HTML elements in your pages. (Remember, understanding the CSS box model is essential to creating successful layouts.)

At this point, you may also want to identify which of your elements are consistent throughout your site (such as the header, the footer, and the navigation). You can build these reusable items once, place them in separate files, and then include them globally in your pages using server-side includes (SSIs), PHP, or other scripting languages, which will save you even more time than CSS when you update your site.

Set up your HTML pages

Once you have an idea of the structure of your pages and are finished with the preliminary markup of the pages, you are ready to set up your development site (which is rather easy to do with a tool like Macromedia Dreamweaver from Adobe because it includes robust site management capabilities). Start with a barebones home page that includes the correct doctype declaration, which is a critical piece of information that browsers and other tools processing the page require. If you are using Dreamweaver 8, starting a new page is as easy as selecting File > New > Basic Page > HTML and then selecting the document type to fill in your blank canvas with the necessary HTML elements:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
 <html xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
 <title>Untitled Document</title>
 </head>
 <body>
 </body>
 </html>
```

Next, populate the content area (between the <body> tags) of your page with the elements and HTML tags you have identified on your blueprint. For example:

```
<body>
<h1>Banner</h1>
<!-- Navigation -- >
  <ul>
  <li>
  <a href="http://domain.com/index.php">Home</a>
  <a href="http://domain.com/about/">About</a>
  <a href="http://domain.com/services/">Services</a>
  <a href="http://domain.com/portfolio/">Portfolio</a>
  <a href="http://domain.com/contact/">Contact</a>
  </li>
  </ul>

<!-- Main Body -->
<h2>Content title</h2>
  <p>content</p>

<h2>Content title</h2>
  <p>content</p>
  <p>content</p>
<!--an ordered list -->
  <ol>
  <li>step 1</li>
  <li>step 2</li>
  <li>step 3</li>
  <li>step 4</li>
  </ol>
<!-- Footer -->
Copyright
</body>
 </html>
```

Define your HTML elements

Next, you have to identify the main sections or containers that hold your HTML elements. The main sections of a page typically consist of a header (or banner), the main content, the navigation, and a footer. If you are using a multicolumn layout, you would also include secondary content areas or menus in addition to your primary content and navigation:

```
<body>
<!-- the "root" container that holds all the elements on your page-->
<div id="container">
<div id="banner">
  <h1>Banner</h1>
  </div> <!--banner -->
<!-- Navigation -- >
<div id="navigation">
  <ul>
  <li>
  <a href="http://domain.com/index.php">Home</a>
  <a href="http://domain.com/about/">About</a>
  <a href="http://domain.com/services/">Services</a>
  <a href="http://domain.com/portfolio/">Portfolio</a>
  <a href="http://domain.com/contact/">Contact</a>
  </li>
  </ul>
  </div><!-- navigation -->
<!-- Main Body -->
<div id="mainContent">
  <h2>Content title</h2>
  <p>content</p>

<h2>Content title</h2>
  <p>content</p>
<!--an ordered list -->
  <ol>
  <li>step 1</li>
  <li>step 2</li>
  <li>step 3</li>
  <li>step 4</li>
  </ol>
  </div><!-- mainContent -->
<!-- Footer -->
<div id="footer">
  Copyright
  </div><!-- footer -->
</div><!-- container -->
</body>
 </html>
```

After you've defined the content blocks within your HTML page, it's time to define where the content blocks should appear on the page and how they should look. That's where style sheets come in.

Create the style sheet

Style sheets consist of groups of style rules, which in turn are groups of properties that define how an HTML element appears in a web browser. When applying a style sheet to your pages, you have three options: you can use an external style sheet, an embedded style sheet, or inline styles. External style sheets are best because they enable you to control the appearance and layout of all the pages in your site from within one CSS file. An embedded style sheet works fine if you're dealing only with a single page. Inline styles may come in handy on those rare occasions when you want to override the appearance of a general element on your page as defined in an external style sheet.

CSS resolves conflicting styles by giving preference to the style that is closest to the tag in question. For example, a tag's style attribute has priority over a surrounding element's style, which takes precedence over a style defined in the head of the document, which in turn wins over an attached style sheet. Likewise, you can merge as many external style sheets as you want by using multiple <link> elements or @import statements. Where they conflict, succeeding links or imports will override preceding ones, and will themselves be overridden by rules within the HTML document itself.

Setting up your style sheet is pretty straightforward—you can start a new document in Dreamweaver or use a text editor and save the file with a CSS extension (for example, layout.css). In a nutshell, you want to include the container layouts (dive with an ID) you specified earlier, heading styles, paragraph styles, and any other styles you defined in your HTML, as well as a rule for the entire body of your document:

```
body {
}
#banner {
}
#mainContent {
}
#footer {
}
#navigation {
}
h1 {
}
h2 {
}
h3 {
}
p {
}
a:link {
}
a:visited {
}
a:hover {
}
a:active {
}
ul li {
}
ul li {
 }
```

Although there are no rules for the order in which to define style sheets, you may want to organize your style sheets consistently and in a meaningful way. For example, you can lump all container layouts (divs with an ID of banner, footer, and mainContent) at the top, followed by the heading styles, paragraph and links styles, and list styles. (For more tips on how to get organized with style sheets, see 5 CSS Tips by Mike Rundles.)

When you know how to style each element, fill in the gaps between the curly brackets with properties and attributes. Here's an example of what this might look like for the banner rule:

```
#banner {
....padding: 0px 0px 0px 0px;
....margin-bottom: 0px;
....height: 80px;
....border-top: 0px solid #003366;
....border-bottom: 1px solid #003366;
....background-image: url(../graphics/logo.gif);
....background-position: left;
....background-repeat: no-repeat;
....background-color: #003366;
....}
```

When building CSS sites, you'll soon realize that one of the most time-consuming tasks is getting started on the HTML markup and remembering all the configurations of styles that may be necessary to control the appearance of your pages. To help you jump-start future projects and streamline your workflow, you may want to create a set of custom starter templates that contain all the basic information you need to make a site's header structure, body, and CSS file compliant and complete. (For more information and samples, check out Kevin Hale's Quick Start Your Design with XHTML Templates.)

‹ Back

Validate and test your pages

As a rule, you should always validate your style sheet to ensure conformance with W3C recommendations, using the W3C CSS Validation Service. And, as with traditional HTML designs, you always want to test your CSS pages in different browsers as you progress, especially since some browsers may not support a given feature or may interpret a certain rule incorrectly. This is especially true with Microsoft Internet Explorer. However, for every snag you run into, you will be able to find a workaround. In a pinch, remember that Google can be your best friend.

Michael Koch is a technical communicator with a penchant for tools and gadgets that make his life easier and more enjoyable.