

# Creating Web Page Layouts Cascading Style Sheets

## What are Cascading Style Sheets?

The future of web design is in emerging standards such as Cascading Style Sheets (CSS), Extensible Markup Language (XML) and Extended Hypertext Markup Language (XHTML). While some of these are still a little ways off of becoming the norm on the internet, one already has: CSS. Cascading styles sheets are like Dreamweaver templates. They allow you to create files that define how text will look on your entire site. CSS files can even define the appearance of your page layouts. Sometimes, styles can be defined right in the <head> section of an html document. More often however, style sheets are saved as completely independent files that are uploaded as part of a website. Pages of the website, then have a line of code that tells the browser to go get that style sheet in the same way that pages refer browsers to graphics and other pages. This can be very beneficial to a web master in that it allows for the easy modification of an entire website. Is that 10 pixel type that you selected too small for some users? Simply open the CSS file and change the value to 12. Instantly, the entire website has been changed to 12 pixel type. Yes. It really can be that easy.

## How Do Cascading Style Sheets work?

With HTML, once you learned that nearly all tags had a beginning and an end tag like <b> and </b>, and that some could be modified like <font color="990000" size="3"></font> and some could stand alone like , you knew about all there was to know. After that, it was simply a matter of knowing the tags and their attributes. Cascading styles sheets are not much different. The syntax is rather easy and after you know that, you simply need to know the options that are available to you.

## Anatomy of a Style

All styles have the same basic parts. First, the selector defines the name of the style. With the selector, you can either select HTML tags to define or create your own selector names. Next, you have the declaration. This is the part between the curly brackets that tells the browser how to render parts of the page to which the selector has been applied. The declaration has two parts, the property and the value. There are many properties that can be declared in any style. These include color, border width, size, and padding. With each property, you can then define what you would like done with that property in the value. In the first example given, any text that falls between <h1> tags would be made purple.

*selector*    *declaration*  
h1 {color: purple;}  
          *property*    *value*

## Creating Your Own Styles

While you can use HTML tags as the base for your style sheet, eventually you will find yourself wanting to create your own style names. These styles are just as easy to create and give you the benefit of allowing you to use names that make sense to you or others. Here's an example:

```
.title { font-family: sans-serif, Arial, Helvetica; font-size: 18px;  
font-weight: bold; color: #039;}  
.subhead { font-family: sans-serif, Arial, Helvetica; font-size: 12px;  
font-weight: bold; color: #000;}
```

In the above case, there are two styles. They are identical except for two things. First, the *title* style has 18 pixel font while the *subhead* style has 12 pixel font. Second, the *title* style is a blue color while the *subhead* style is black. Since all web-safe colors come in the format "#aabbcc", CSS allows you to abbreviate the colors as "#abc". If you are using a color that is not web-safe, you can still list all six digits to define that color.

## Formalizing Your Stylesheet

So where do these styles go? They go in the <head> section of your HTML document. Stylesheets can either be defined directly in the <head> section or a separate file which contains the stylesheet can be referenced.

### Including the styles in your HTML document

When styles are defined directly in the <head> section, they are surrounded by the <style> tag as follows:

```
<head>...
<style type="text/css">
  .title
  { font-family: sans-serif, Arial, Helvetica;
    font-size: 18px;
    font-weight: bold;
    color: #039;
  }
  .subhead
  { font-family: sans-serif, Arial, Helvetica;
    font-size: 12px;
    font-weight: bold;
    color: #000;
  }
</style>
...</head>
```

*Note: Because, like in HTML, spaces and returns aren't recognized as part of the page, you can use often used formats, like the one above, to make your styles easier to read and manage.*

### Referencing an external CSS document

When styles are stored in a separate file, they are listed in that file without the <style> tags and that file is saved with a name ending in ".css". That file is then referenced in the <head> section of the html document like this:

```
<head>...
<link rel="stylesheet" type="text/css" href="mystyles.css">
...</head>
```

## Using Styles

Styles are typically invoked by using using one of two tags: <span> or <div>. They can also be referenced with the *class* HTML attribute. Here are some examples of the styles from the previous section being used:

```
<p class="title">This is my title</p>
<p><span class="subtitle">This is a subtitle</span><br>
this is the body of my message</p>
```

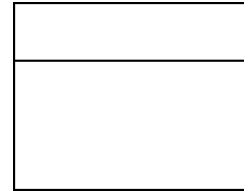
In the first line, the *title* style is applied to the entire paragraph. In the second line, the *subtitle* style is applied to only a group of words with the <span> tags. Note that while custom style names always begin with a dot as in ".title" when being defined in the style sheet, afterwards they are always referred to without the dot as in <p class="title">.

## Creating a Page Layout with CSS

While text on the big websites has been defined by CSS for years now, it is only recently that CSS has begun to pick up widespread approval for page layouts. In the past, when one has talked about separating the content from the form, it has always been in the context of discussions about server-side

scripting (like NetCloak) or developer application features (like Dreamweaver templates). Cascading style sheets allow you perform this separation of content and form right inside of a static page. Let's take a look at how to do this...

Let's create a very basic layout that will consist of a banner sitting atop the main content area. In the top area, we will place the title of our site. In the bottom section, we will place the page title, menu text and main content.



To create this layout, we will first make two basic styles. The first style will create the top banner and define the text that sits within it:

```
#banner
{
  font-family: arial, helvetica, sans-serif;
  color: #FFF;
  font-size: 20px;
  font-weight: bold;
  border-bottom: 1px dotted #000;
  border-top: 3px solid #000;
  background: #900;
  padding: 15px;
  text-transform: uppercase;
  letter-spacing: .2em;
}
```

So what does this style do? It sets the font face to sans-serif, the color to white, the size to 20 pixels and the weight to bold. It also puts a thick black border on the top, a thin dashed black border on the bottom and colors the area a dark red. There will be a 15 pixel pad between the text and the borders of the banner, the text will be all uppercase (regardless of how it is entered) and there will be a .2 em space between the letters.

Wow! What control, eh? This is not your old font tag.

Let's create the bottom section now:

```
#mainarea
{
  font-family: arial, helvetica, sans-serif;
  color: #000;
  font-size: 12px;
  font-weight: normal;
  background: #fff;
  padding: 15px;
}
```

This tag is considerably simpler. We define the text as sans-serif, black, 12 pixel and of a normal weight. We set the background of the area to white and gave a padding of 15 to maintain consistency with our previous area. Because we are going to defining an area of our page, we use the “#” key instead of the “.” before the style name.

If we want the banner of our page to run right up to the edges of the browser screen, we need to eliminate the natural padding that all browsers automatically impose on web pages. Let's do that by redefining the <body> tag:

```
body
{
  font-family: arial, helvetica, sans-serif;
  margin: 0px 0px 10px 0px;
  background: #fff;
}
```

This tag is pretty self-explanatory except for the margin property. You can define the margin of a body tag using just a series of values. If there is one value, it sets the margin for all four sides. If there are two value, they set the margins for the top/bottom and left/right sides, respectively. If there are three values, they set the margin for the top, the right/left, and the bottom. If you use four values, as in the style above, it sets the margins for the top, right, bottom and left, respectively. One can also set the margin for just one side without worrying about the others by simply specifying that side with a margin-right, margin-left, etc.

Finally, let's create a couple of text styles that we can use in the main area of our page. We'll use ones similar to the styles we examined earlier:

```
.title
{ font-family: sans-serif, Arial, Helvetica;
  font-size: 18px;
  font-weight: bold;
  color: #900;
}
.subhead
{ font-family: sans-serif, Arial, Helvetica;
  font-size: 12px;
  font-weight: bold;
  color: #000;
}
```

## Saving the Style

Now that we have our styles all figured out. They can be put together in a text file and saved as something like, "styles.css". Put it in your web site directory and that's it!

## Let's Create a Page Based On This Style

Once you have your style sheet created, pages become pretty easy to put together. Here we go:

```
<html>
<head>
<title>My CSS Page</title>
<link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<div id="banner">My First CSS Based WebSite</div>
<div id="mainarea">
<p class="title">Page One</p>
<p><span class="subhead">This is my subhead</span><br>
la de da de da</p>
</div>
</body>
</html>
```

With our style sheet in place, the above HTML is able to correctly assemble our page. Let's take a look at some of the tags we used and why we used them.

### The <div> tag

When we are defining a whole section of our web page, as in the banner or main area, we use the <div> tag. This tag is able to span large amounts of your page and set them apart visually. Typically, <div> tags use styles that begin with "#" symbols. Note that while other tags use *class* to indicate the appropriate style, this usage of <div> requires *id*.

### The <span> tag

The span tag is nice when you simply want to apply a style to a few words or a sentence here and there.

`<p class="title">`

You can also apply a style to an entire paragraph, table cell or other natural area of your page by applying a class attribute to an html tag.

## Modifications Made Easy

When the time comes that you need to make some changes, CSS makes that very easy to do. Simply open your style sheet and begin making the appropriate modifications. Go ahead and make some changes of your own!

### Let's create a fixed-width column

First we need to create a style that will hold this column. Let's do this in our style sheet file that we saved earlier. Add the following style:

```
#rightcolumn
{
  font-family: arial, Helvetica, sans-serif;
  color: #000;
  font-size: 12px;
  position: absolute;
  right: 0px;
  top: 75px;
  width: 150px;
  padding-right: 15px;
  background: #fff;
}
```

This creates a space that sits on the right side of the page (right: 0px) and below the banner (60 px). The width of this area will be fixed (width: 150px). If we don't want our *mainarea* style to intrude on the area of this style, we need to now make a change to it:

```
#mainarea
{
  font-family: arial, helvetica, sans-serif;
  color: #000;
  font-size: 12px;
  font-weight: normal;
  background: #fff;
  padding: 15px;
  margin-right: 165px;
  border-right: 1px dotted #666;
}
```

This will keep the *mainarea* style far enough off of the right side of the page to leave room for our new column. It will also place a gray dotted border between the columns to give some visible separation.

Save and close the style sheet and open your HTML document. Place the following line right before your `<div id="mainarea">` section:

```
<div id="rightcolumn">
<p>Menu Item 1<br><Menu Item 2<br>Menu Item 3</p>
</div>
```

Save and close the html document and check it out in the web browser. Did it work?

## **Good Work!**

You are well on your way to creating great sites with Cascading Style Sheets. I hope that you have found, through this lesson, that creating sites based on css is not only easy, but can give you layout and design powers that you never had before, while making maintaining sites easier as well. As it becomes more and more important to create pages that are visible to numerous devices, css will only become more vital as it allows page designers to create different layouts that can quickly be applied to content.

For more information about Cascading Style Sheets, check out Brian's web log. Go to <http://radio.weblogs.com/0109265> and click on the css link in the menu. You can find links to great sites and information about helpful books there.

**Brian Fitzgerald** · email: [bfitz@lps.org](mailto:bfitz@lps.org)

Brian Fitzgerald is the Web and User Interface Developer for Lincoln Public Schools in Lincoln, Nebraska. He maintains the district web presence and aids district schools in establishing and maintaining theirs. His most commonly used applications are Dreamweaver, Fireworks, BBEdit and OS X's Mail.app. He does most of his work on Mac OS X, but uses Windows often. His design philosophy is to "Keep it simple, but pay never-ending attention to the details." You can find out what Brian is learning at his web log: <http://radio.weblogs.com/0109265/>