CSS WEB DESIGN WITH DREAMWEAVER

By Michael L Curry

WEBSITE

http://michaelcurry.com/Default.aspx?p=73&n=Articles-on-Web-Design-Topics

DESCRIPTION

This is an advanced web design article that builds on the Dreamweaver article by teaching how to use Cascading Style Sheets (CSS) to control the visual layout of a web page. CSS is the HTML 4.0 approved method for design and offers control of every element in a Web site. Visit the link above to also download all the image files and the completed CSS and HTML files for this tutorial.

INTRO TO CSS

There are many advantages to using CSS over traditional HTML, such as:

1. The web page source code is easier to read & loads faster
2. Site wide changes can be made by updating CSS attributes in one file
3. Pages can be designed using a multidimensional layers

One disadvantage of CSS is that it is initially more difficult to design pages.

HTML Markup

```
<b>make me bold</b>
```

This is the HTML method of marking up text and there is nothing wrong with it. The disadvantage is if you had a lot of these spread throughout your site and wanted to change them, you would have to touch each one individually.

```
<font color="#FF0000" face="Verdana, Arial, Helvetica, sans-serif">
<strong>This is text</strong></font>
```

This is a more typical HTML mark-up, and you can see how messy this makes your web page source code. Again, updates need to be done manually to each instance of the font.

Replacing Font Tags with CSS Styles

CSS separates formatting from page content, as in the following example:

```
<html><head>
<style type="text/css">
<!--
. bodyTxt {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-weight: bold;
color: #FF0000;
}
-->
</head><body>
<p class="bodyTxt">My CSS styled text</p>
</body></html>
```

You can see text formatting is simplified by CSS. By defining a style if you need to make a change, it can be done it in one place. Dreamweaver will create CSS styles for you automatically as you change fonts, style and color while editing text.

External Style Documents

In the preceding example the style sheet is in the head of the web page itself. A better technique is to create an external page and have pages using this style link to the style sheet.

Page.html

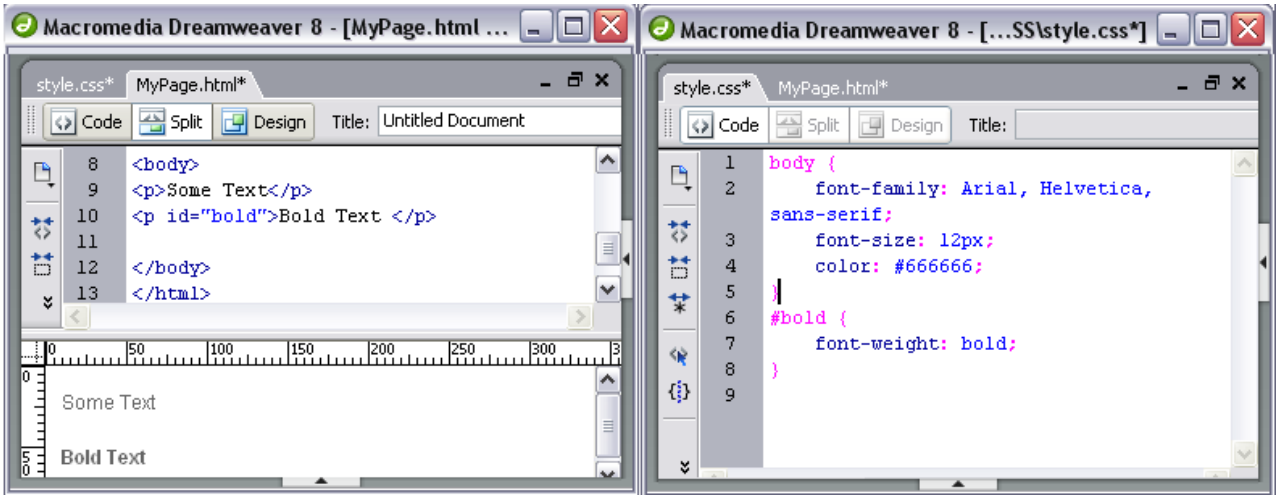
```
<html><head>
<link rel="stylesheet" type="text/css"
href="MyStyle.css">
</head>
<body>
<p class="bodyTxt">My CSS styled text</p>
</body></html>
```

MyStyle.css

```
. bodyTxt {
font-family: Verdana, Arial,
Helvetica, sans-serif;
font-weight: bold;
color: #FF0000;
}
```

Cascading Styles

Once you define a style, it will continue to cascade down into other styles unless you choose to override them. The nice part of this is we only need to redefine the attributes that important to us



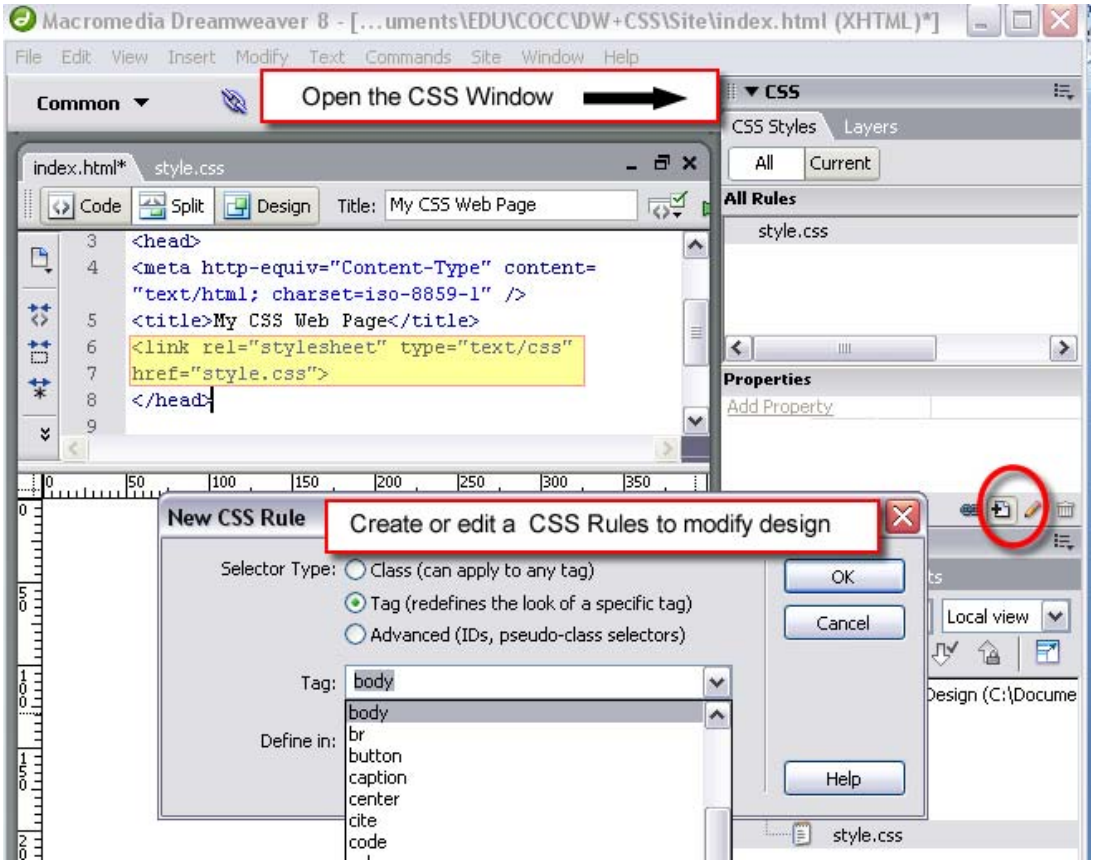
DEFINING THE BODY

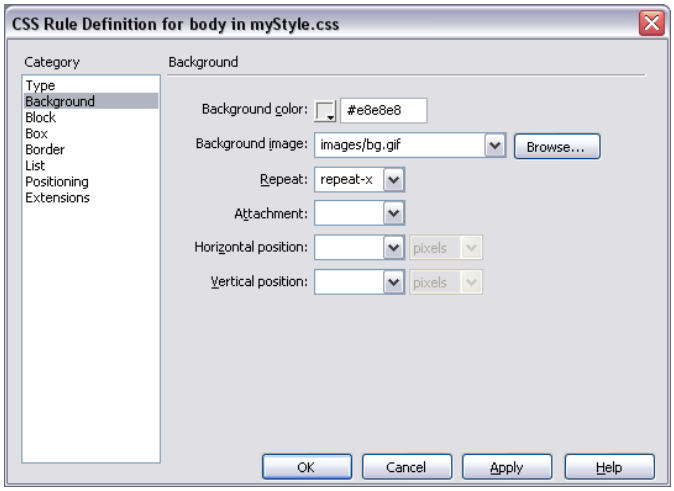
The first step we will take in our design is to define some attributes about the page.

1. Create a new blank HTML page, save it as “index.html”
2. Create a new blank CSS page, save it as style.css
3. Add a link to the CSS file inside the <head> … </head> tags of index.html

Now it is time to define the page properties, such as back ground, text font, headings and links.

Clicking new or edit CSS rule, to manage the page design.





In the Rule Definition, click different categories to see the properties.

Under **Type**.. font Arial

Background… color: #e8e8e8, Image: bg.gif, repeat-x.

Box… margins: 0 px.

Press OK.

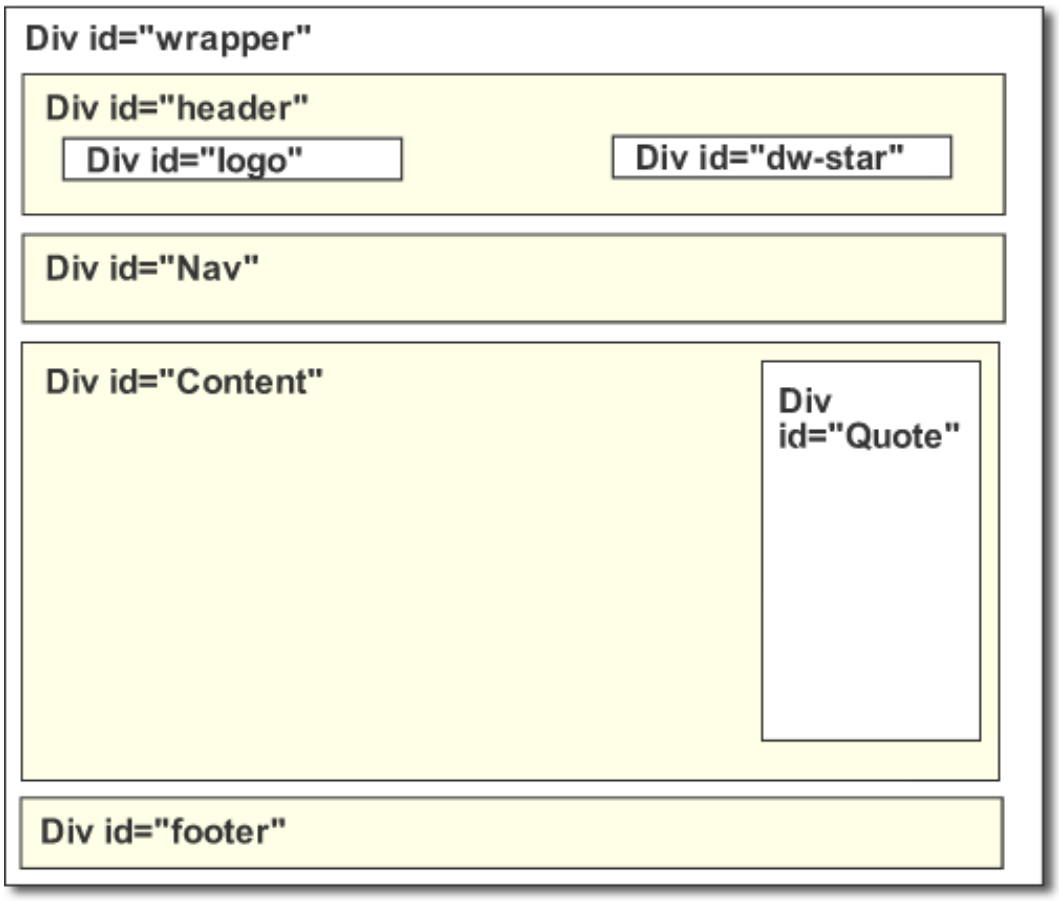
Continue to define the styles for your links, so that your CSS is as follows:

```
body {
        font-family: Arial, Helvetica, sans-serif;
        background-color: #e8e8e8;
        background-image: url(images/bg.gif);
        background-repeat: repeat-x;
        margin: 0px;
}
h1 {
        font-size: 22px;
        line-height: 1.5em;
        font-weight: bold;
        color: #636fa3;
        text-transform: uppercase;
        letter-spacing: .4em;
}
h2 {
        font-size: 18px;
        line-height: 1.2em;
        text-transform: uppercase;
        color: #333333;
        letter-spacing: .3em;
        word-spacing: 1.4em;
        font-weight: bold;
}
h3 {
        color: #333333;
        letter-spacing: 0.3em;
        word-spacing: 1em;
        font-weight: lighter;
}
a:link {
        color: #a64200;
}
a:hover {
        color: #2b3768;
}
a:visited {
        color: #996600;
}
a:active {
        color: #2e6802;
}
```

Do not forget the { and } braces in your definition, and also to put a semicolon (;) after each term in the CSS definition.

PAGE LAYOUT

In order to layout a page, we break up the major components of the site using the CSS generic tag . Here is an example page layout using DIV tags.

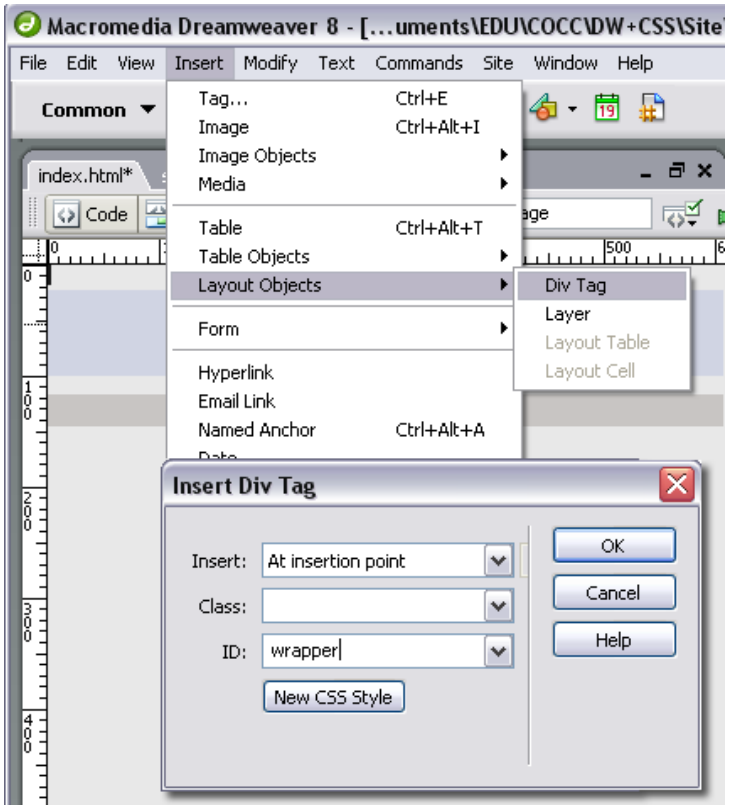


CSS has an absolute positioning model to define exact div positions. However, modern designs require more flexibility, so the "Flow" model is used so objects flow around other objects elastically.

1. Insert a new layout object… DIV tag onto the page

2. Give it an ID of "**wrapper**"

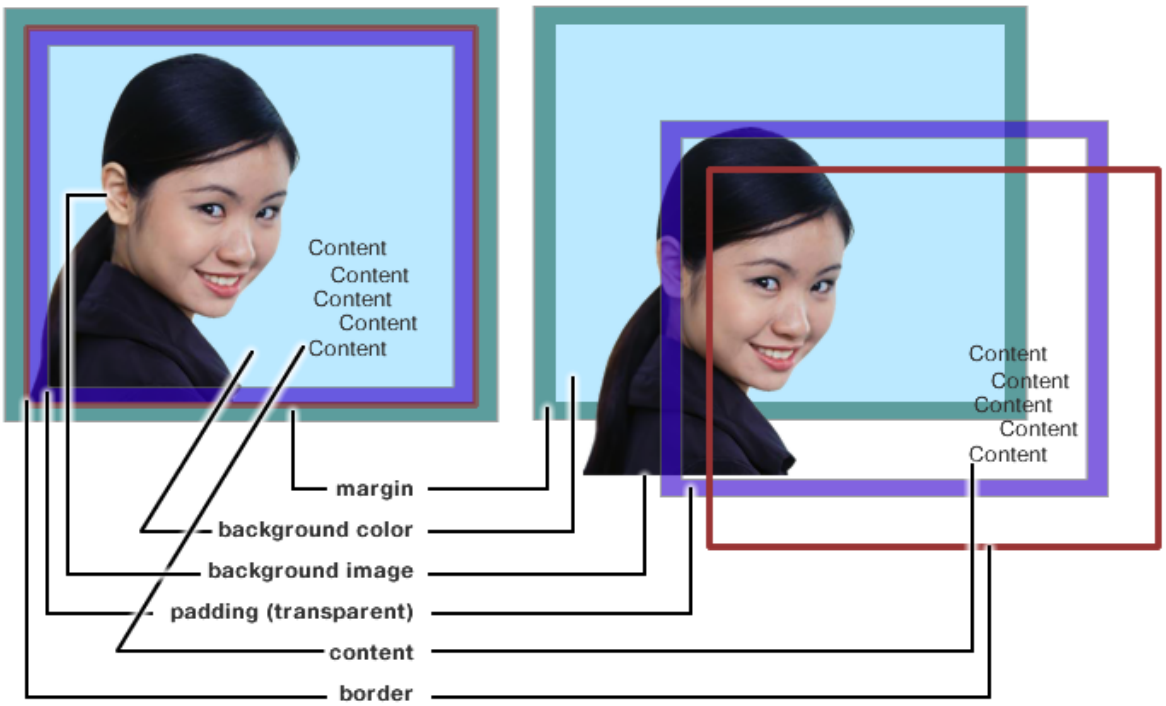
3. Define the following properties:
   a. background-color: #FFFFFF;
   b. height: 100%;
   c. width: 800px;
   d. margin-left: auto;
   e. margin-right: auto;



The wrapper will be used to encapsulate all the content of our page design.

THE CSS BOX MODEL

A tag uses the CSS Box model to manage layout. The box can define attributes about those divisions (such as height, width, background, etc.). While it will take some experience to appreciate, the following image may be helpful to understand how these properties work.



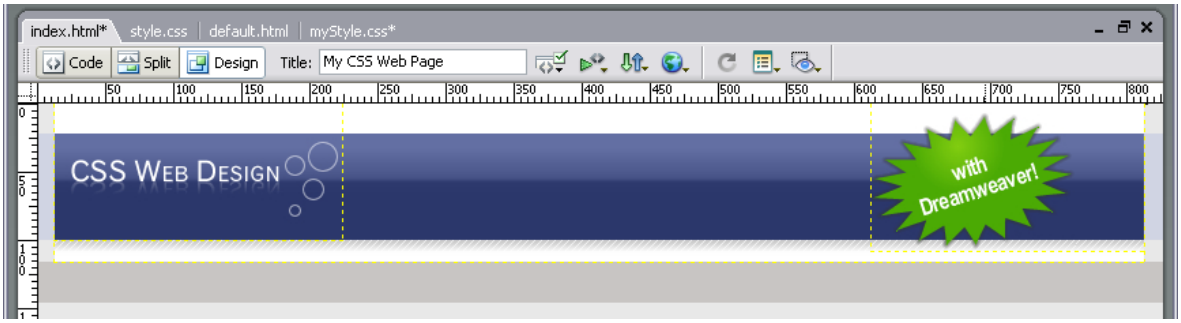
Add a Header to the Design

Adding a header will help illustrate the Box model better.

Our header will be defined as an 800 x 116 pixel box. Rather than using a large graphic to be our header, we will use a very small image that is repeated along the X axis to create the header bar.

Then we position the logo and a star at both ends of our header, using additional DIV tags.

So the end result looks like this:



Here is the new code that you should have added to this point. I have used a highlighter so you can see how the CSS relates to the HTML:

Index.html

```
<div id="wrapper">
        <div id="header">
                <div id="logo">

                <a href="default.html"><img
                src="images/logo.jpg" width="211"
                height="100" border="0" /></a>

                </div>

                <div id="dw-star">

                <img src="images/dreamweaver-
                star.jpg" width="150" height="108"
                border="0" />

                </div>
        </div>
</div>
```

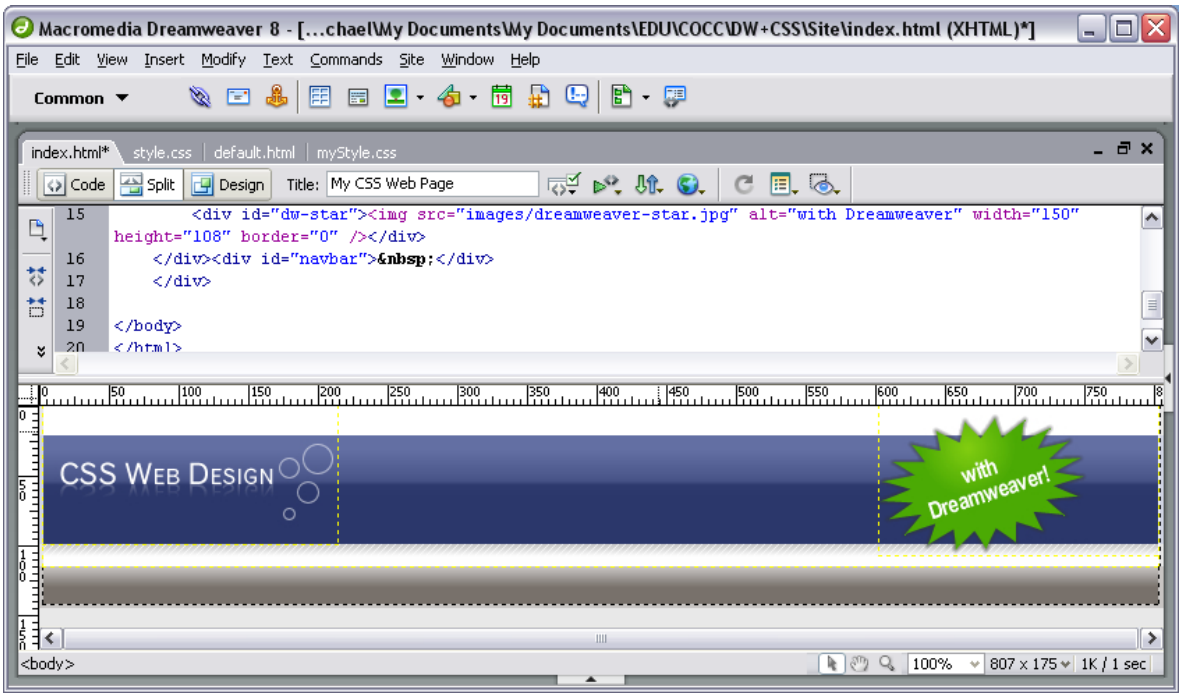
Style.css

```
#wrapper {
        background-color: #FFFFFF;
        height: 100%;
        width: 800px;
        margin-left: auto;
        margin-right: auto;
}
/*
Header will be used to create the top header image.
It also contains two sub DIVs,  logo & the star
*/
#wrapper #header {
        background-image: url(images/header_bg.jpg);
        background-repeat: repeat-x;
        background-color: #2B376B;
        float: left;
        height: 116px;
        width: 800px;
}
/*
Float Left means additional content will flow around
the right side of this item
*/
#wrapper #header #logo {
        float: left;
}
#wrapper #header #dw-star {
        margin-left: 600px;
}
```

SITE NAVIGATION

One misconception of CSS design is we no longer use tables to design. There are many instances where using a table is the ideal way to layout information even in CSS. To illustrate that concept, we will create the site navigation using a table.

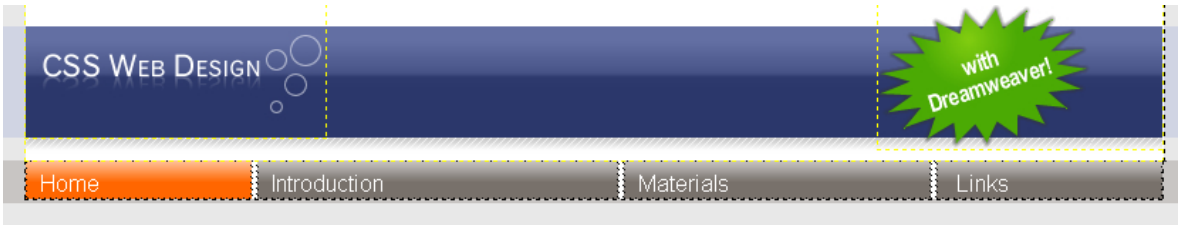
First, we create a DIV id="navbar" that is 800 x 26 pixels. We will use the same technique we used for our header bar and set the background of the navbar to be the nav_off.gif, which we will also repeat along the X axis.



Next we insert a 1 row and 13 column table which we assign the id= NavTable. The trick here is to give each cell of the table an appropriate class, such as:

```
<td class="navOn"><a href="index.html">Home</a></td>
 <td class="navSpacer"> </td>
 <td class="navOff"><a href="aboutus.html">About Us</a></td>
```

Then in the CSS we will assign different properties to each of these classes, so that in the end we have this



You can refer to the complete CSS for the details of the navOn, navSpacer & navOff formatting commands.

THE MAIN CONTENT

In the main content we will have two additional DIV containers. One will be used for the main text of the page, and the other will have a small area for text off to the side of the page.

The Content DIV

The content DIV is unremarkable:

```
<div id="main"> Content goes here </div>
```

But in the CSS page, we define an orange border that will tie in nicely with our menu bar, as well as padding to the left and right of the page.

```
#wrapper #main {
        float: left;
        width: 786px;
        height: 100%;
        border: 2px solid #FF6600;
        background-color: #FFFFFF;
        padding-top: 15px;
        padding-bottom: 10px;
        padding-left: 10px;
}
```

We also define that any content in this DIV should flow to right of our text. This float will be helpful for our next step.

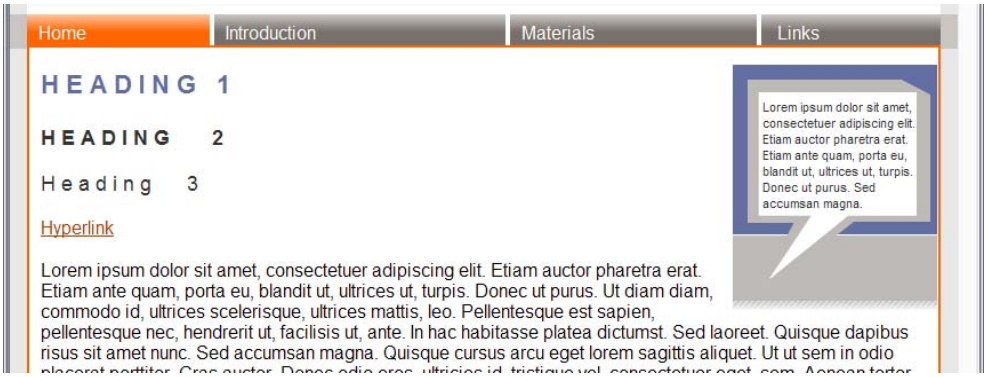
The Quote DIV

To place a small quote inside of a dialog balloon on the right of the page. To do this, we will use the same technique we have now used on the header and the navigation of using a small image to repeat over the DIV. To accomplish this, we will need 3 DIV tags. Two of the tags will hold an image, while one has the quote.

```
<div id="blockquote">
        <div id="quoteHead"><img src="images/blockquote-head.gif" /></div>

        <div id="quote">  Quote goes here </div>

        <div id="quoteFoot"><img src="images/blockquote-foot.gif" /></div>
</div>
```



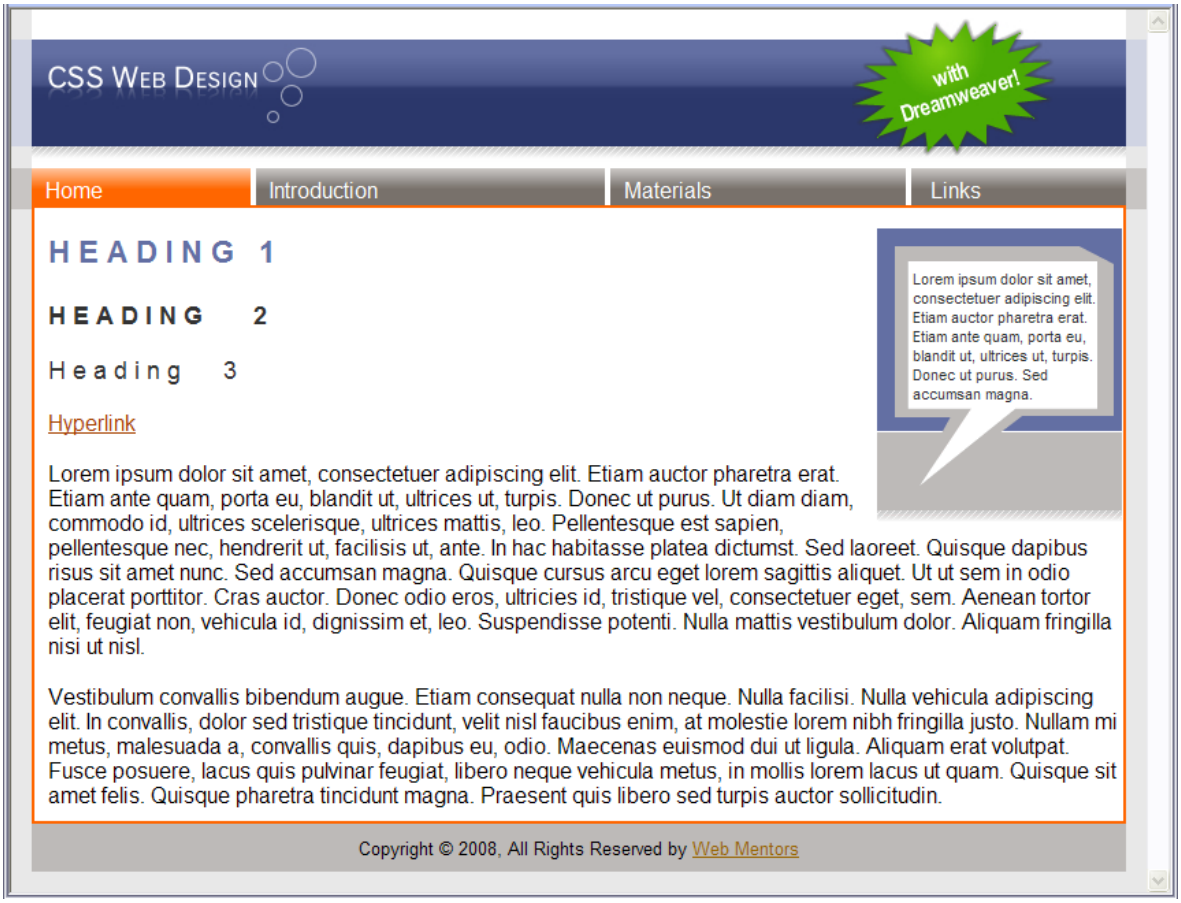
THE FOOTER

The final piece is to add a footer to the page, this is simply accomplished by adding another DIV tag

 Footer goes here

And formatting it so the background color is gray and the text is center aligned.

```
#wrapper #footer {
        float: left;
        height: 25px;
        width: 800px;
        font-size: smaller;
        text-align: center;
        background-color: #BDBAB8;
        padding-top: 10px;
}
```



Congratulations, you have completed your first CSS design!

CONCLUSION

Hopefully this has shown you how clean and efficient it is to layout a site using CSS versus HTML formatting. The best way to get better at CSS is to practice. You may also want to investigate the Photoshop for Web class too.