

## Creating a CSS-Table Layout

In the new age of web development one might think a website created with a simple table defeats the purpose of web standards and designing with CSS. However, tables which have correct structural code and utilize CSS for the layout and presentation of text, images, navigation are just as practical for layout in some cases. In this article, we will examine how to create a simple table layout using CSS.

After the structural code is created, we'll dive in-depth with CSS to control each aspect of the page. We'll place images neatly inside an unordered list which allows for easy maintenance and control. We'll learn how to tile a background image inside table cells using CSS. We'll learn how to create image rollovers without a hint of JavaScript. Finally, we'll learn how to make a "you are here indicator" which lets visitors know which page they are currently visiting.

If you would like to follow along with this article's step-by-step development or even try your hand at creating a CSS Table – Layout, you will find the project files link helpful.

Finished Page

[http://midwestwebdesign.net/tutorials/csstable/index\\_finished.html](http://midwestwebdesign.net/tutorials/csstable/index_finished.html)

Project Zip

<http://midwestwebdesign.net/tutorials/csstable/csstable.zip>

### Open the page

In order to make this article manageable, we'll work with an already made HTML structure and then write CSS to control the layout and format the appearance of our page. Open **index.html** and look at the structure, kick the tires, take it for a spin. Our structure is examined briefly:

- Four rows, three columns
  - First row, first column
    - Header content
  - Second row
    - Four navigation links
  - Third row, three columns
    - Left
      - Navigation
    - Middle
      - Content
    - Right
      - Pictures
  - Fourth row, first column
    - Footer content

We give each column an ID so that we can later set CSS rules in our external style sheet to affect layout and formatting. We'll replace both heading tags in our header and footer with images. We'll make the unordered list in the second row our central navigation using CSS to simulate

JavaScript rollovers. We'll place regular headings and paragraphs in the middle column and in the right column we'll use CSS to format our images which are inside an unordered list.

### Creating our title

We'll give our web page a title, such as *Fighter Planes – Bringing the Fight Home*. Between the opening and closing <head> tags insert the following:

```
<head>
<title>Fighter Planes – Bringing the Fight Home</title>
</head>
```

Using the <title> tag, we give our page a title. We use the ASCII character equivalent of the dash by using the numerical entity. Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_title.html](http://midwestwebdesign.net/tutorials/csstable/index_title.html)

### Attaching the style sheet

In order for our web page to see the styles we write, we need to attach it to our web page. We have provided a blank style sheet in the project files download, **style.css**. Between the opening and closing <head> tags, we link to our style sheet:

```
<head>
<title>Fighter Planes – Bringing the Fight Home</title>
<link href="style.css" type="text/css" media="screen"
rel="stylesheet">
</head>
```

Save your file.

### Setting the width of our layout

Our web page is set to a fixed width of 760 pixels. This means each column and images they may contain must be equal or less to 760 pixels. We'll define these widths via our style sheet. In your text editor, open style.css and add the following:

```
body{
font-family:Arial, Helvetica, sans-serif;
font-size:101%;
color:#000000;
margin-top:10px;
margin-bottom:10px;
padding:0;
background-image:url(images/background.jpg);
background-color:#CC9900;
}
table#layout{
width:760px;
margin:20px auto;
border:0;
```

```
}
table#layout td#header{
}
table#layout td#left{
width:190px;
}
table#layout td#middle{
width:380px;
}
table#layout td#right{
width:190px;
}
table#layout td#footer{
}
```

Let's examine the code in greater detail:

- We use the body selector to set global styles for our web page, such as font family, size and color.
  - We set margin top and bottom to 10 pixels
  - Padding to 0
  - Background image is set to a tiling gold pattern image
  - Background color is set to the same color as our tiling background image. If a visitor is using dial-up, they will be provided a background color which is the same as our image while the image is loading.
- We set our table to a fixed width of 760 pixels.
  - Set top and right margins to 20 pixels and left and right to auto. In a fixed width layout, setting left and right margins to auto will center our layout in modern browsers
- We set our left and right column to a fixed width of 190 pixels
- We set our middle column to a fixed width of 380 pixels

We arrive at the following mathematical calculation:

```
Table layout (760px) = Left (190px) + Middle (380px) + Right (190px)
```

The CSS rules for header and footer were omitted for the time being. Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_widths.html](http://midwestwebdesign.net/tutorials/csstable/index_widths.html)

### Header and footer images

We'll use the images which are provided from the project files download. However, if you're using Dreamweaver for this task, please insert the image via code view. The reason for this will be explained shortly. Follow these steps to insert the images:

- For the header and footer image, remove the heading two tags
- To insert our header image, add the following code:

```
<tr>
<td id="header" colspan="3">

</td>
</tr>
```

Next, to insert our footer image, add the following code:

```
<tr>
<td id="footer" colspan="3">

</td>
</tr>
```

To help in accessibility, we provide an ALT attribute so that assistive reader technology can notify handicapped persons there are images residing in our page. In our style sheet, add the following to our existing header and footer rule:

```
table#layout td#header{
width:760px;
height:100px;
}
table#layout td#footer{
width:760px;
height:100px;
}
```

Remember that we didn't add a width or height attribute inside our image tag. Our reason is simple – these images will be used in multiple pages for a website. If a change is made to the width or height, you would need to change the width and height attributes for each image tag in each page. Assigning the width and height via CSS alleviates this problem. If a change is made to the width or height then we change it in one place and we're done. If you insert the image via Design view in Dreamweaver, width and height are automatically entered. Save your file and preview the changes.

[http://midwestwebdesign.net/tutorials/csstable/index\\_images.html](http://midwestwebdesign.net/tutorials/csstable/index_images.html)

### **Moving left, middle, and right column content up**

You'll notice that our left and right cells have their content pushed down. The reason is because the middle cell has more content and is adjacent to these cells. We add the following CSS rule:

By setting vertical-align to top, we force our left and right cell content to move to the top of the cell. Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_moveup.html](http://midwestwebdesign.net/tutorials/csstable/index_moveup.html)

## Left and right cell – tiling background

Let's create a tiling background which will repeat down the left and right cells. Since we have an ID assigned to both, it's easily done by adding the following:

```
table#layout td#left{
width:190px;
vertical-align:top;
background-image:url(images/background_tile.jpg);
}
table#layout td#right{
width:190px;
vertical-align:top;
background-image:url(images/background_tile.jpg);
}
```

We use the background-image property and set the value to a small gradient texture. Setting a repeating value isn't necessary since the background will tile as additional content is added in left, middle or right cells. Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_tiling.html](http://midwestwebdesign.net/tutorials/csstable/index_tiling.html)

## Middle cell – background color

Since our content is in the middle, add the following rule to our existing middle ID:

```
table#layout td#middle{
width:380px;
vertical-align:top;
background-color:#FFFFFF;
}
```

We set background color of our middle cell to white to make our text readable.

[http://midwestwebdesign.net/tutorials/csstable/index\\_middle.html](http://midwestwebdesign.net/tutorials/csstable/index_middle.html)

## Formatting left and right heading three tag

Let's touch up our heading three tags which are in left and right columns. Add the following rule:

```
h3{
margin:10px;
font-size:1.2em;
text-align:center;
}
```

We use our heading three as our selector and use the following property-values:

- Margin
  - Set to 10 pixels, on all sides
- Font-size
  - Set to a proportional 1.2em
- Text-align
  - Set to center

Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_h3.html](http://midwestwebdesign.net/tutorials/csstable/index_h3.html)

### **Format our heading two - middle column**

Let's touch up our heading two tag in our middle column. Add the following rule:

```
h2{
margin:10px;
font-size:1.6em;
}
```

We use our heading two as our selector and use the following property-values:

- Margin:
  - Set to 10 pixels on all sides
- Font-size:
  - Set to a proportional size of 1.6em

Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_p.html](http://midwestwebdesign.net/tutorials/csstable/index_p.html)

### **Creating space - middle column**

Currently, the paragraph text in the middle column is very tight, that is, against the left edge of the left column. Add the following CSS rule to provide a bit of space:

```
p{
padding:8px;
font-size:.8em;
line-height:1.5em;
}
```

We set all paragraphs to have the following property-values:

- Padding
  - Set to 8 pixel on all sides
- Font-size
  - Set to a proportional size of .8em
- Line-height:

- Set to a proportional size of 1.5em

Setting padding provides horizontal and vertical spacing between the left column and paragraph. Some people feel as though setting a line-height on paragraphs makes for easier reading. Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_p.html](http://midwestwebdesign.net/tutorials/csstable/index_p.html)

### Formatting our photos – right column

Since our photos are in an unordered list they are victim to default margins, padding, and bullets. First, let's give our unordered list an ID in HTML:

```
<ul id="planes">
<li></li>
<li></li>
<li></li>
<li></li>
</ul>
```

Let's turn off margins and padding of list items:

```
ul#planes{
margin:0;
padding:0;
}
```

We need to specify both margin and padding because indentations of list items vary between browsers. Padding satisfies Internet Explorer and Opera, while margin satisfies Mozilla. Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_ulindentation.html](http://midwestwebdesign.net/tutorials/csstable/index_ulindentation.html)

Let's remove bullets from appearing and create a vertical offset between each image:

```
ul#planes li{
list-style-type:none;
margin-top:10px;
}
```

We use a descendent selector. A descendent selector describes an element which is a child of a parent element. In our case, we target all list items inside their parent, unordered list. Setting list style type to a value of none turns off bullets in Opera and Mozilla. Setting margin top to a value of 10 pixels forces each image inside the list to have a bit of vertical offset. Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_ulbullets.html](http://midwestwebdesign.net/tutorials/csstable/index_ulbullets.html)

## Setting width and height of our fighter planes

You will notice in our HTML code, we have no width or height set. The reason for this is the same as with our header and footer. Therefore, we'll create a unique ID for each image and then add CSS rules. First, add the following ID's to each image:

```
<ul id="planes">
<li></li>
<li></li>
<li></li>
<li></li>
</ul>
```

In our style sheet, add the following rules:

```
ul#planes li#planeone{
width:190px;
height:128px;
}
ul#planes li#planetwo{
width:190px;
height:128px;
}
ul#planes li#planethree{
width:190px;
height:128px;
}
ul#planes li#planefour{
width:190px;
height:128px;
}
```

For each ID in HTML, we write a corresponding rule in CSS. We set a width and height. Do note each image's physical dimension's is precisely 190 pixels wide by 128 pixels tall. Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_imageid.html](http://midwestwebdesign.net/tutorials/csstable/index_imageid.html)

## Formatting our menu

We have saved the best part for last, using an unordered list to create our main navigation. Using two images, we'll write CSS rules which can create image rollovers without JavaScript. This saves on:

- Amount of code needed
- Reduces maintenance time if a change is needed
- If JavaScript is disabled in a visitor's browser, the navigation still works

First, we'll assign our unordered list an ID in HTML, which can then be used in CSS to create the image rollover affect. Inside the opening unordered list tag, in our second row enter the following:

```
<ul id="navigation">
<li><a href="javascript:;">menu one</a></li>
<li><a href="javascript:;">menu two</a></li>
<li><a href="javascript:;">menu three</a></li>
<li><a href="javascript:;">menu four</a></li>
</ul>
```

We remove default indentation of our lists using margin and padding:

```
ul#navigation{
margin:0;
padding:0;
}
```

Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_navigationidbullets.html](http://midwestwebdesign.net/tutorials/csstable/index_navigationidbullets.html)

## CSS rollover images

In order to create rollover images, we'll write four classes in our style sheet targeting our links. Anytime you swap images in a navigation menu you inevitable run into loading issues. For example, when you load web pages which use a swap image method, the over state of the graphic will often flicker while it's retrieved from the server. We alleviate the problem by preloading our images. In CSS, if we are creative and use a little bit of ingenuity, we can get our images, specifically our rollover image to show without delay.

Here's how it works: We'll assign our rollover image to the background of each list item and assign the normal image to the anchor. Then, using the hover-pseudo state, we'll set the background of our anchors to transparent, which will show the image underneath because it's already there. This will ensure the fastest possible swap method and is probably more efficient than the background-position solution in some cases.

Follow these steps to create the swap image method:

- Remove the textual link from each of the four links
- Inside the unordered list, assign a class to each list item as follows:
  - List item one
    - Home
  - List item two
    - Services
  - List item three
    - About
  - List item four
    - Contact

In HTML, our unordered list looks like this:

```
<ul id="navigation">
<li class="home"><a href="javascript:;"></a></li>
<li class="services"><a href="javascript:;"></a></li>
<li class="about"><a href="javascript:;"></a></li>
<li class="contact"><a href="javascript:;"></a></li>
</ul>
```

In your style sheet, add the following code:

```
ul#navigation li.home, ul#navigation li.home a{
display:block;
background-image:url(images/home_over.jpg);
width:190px;
height:20px;
}
ul#navigation li.home a{
background-image:url(images/home.jpg);
width:190px;
height:20px;
}
ul#navigation li a:hover{background:transparent;}
```

Let's examine the rule in greater detail:

- First, we set the hover state of our image to the background of the list as well as display set to block which makes our link "hot" or "clickable"
- Next, we set the normal state of our image to the anchor
- We set width and height
- For the hover state of each list item, we set the background to transparent to hide the default image and show the hover image

Complete the image rollover steps by writing three more rules for services, about and contact by replacing the background image, width and height and class names. Once completed, your navigation should work as shown at the below page address.

[http://midwestwebdesign.net/tutorials/csstable/index\\_navigationidlinks.html](http://midwestwebdesign.net/tutorials/csstable/index_navigationidlinks.html)

To make our navigation fit snug like a bug in our layout, add the following rule toward the top of your style sheet, preferably right after the original **table#layout** rule:

```
table#layout td{
padding:0;
}
```

Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_nopadding.html](http://midwestwebdesign.net/tutorials/csstable/index_nopadding.html)

## Create a “you are here” marker

Most websites today use some sort of visual indicator which informs the visitor which page they’re currently viewing. Adding this functionality to our page is a cinch and is done by adding the following CSS rule:

```
ul#navigation li a.current{
background-image:url(images/home_over.jpg);
width:190px;
height:20px;
}
```

We create a class named **current**, which is specific to anchor tags, inside a list item, in an unordered list with an ID of navigation. We simply assign our hover image for the currently page. Lastly, locate the first menu item and add the class to the anchor:

```
<ul id="navigation">
<li><a href="javascript:;" class="current">menu one</a></li>
<li><a href="javascript:;">menu two</a></li>
<li><a href="javascript:;">menu three</a></li>
<li><a href="javascript:;">menu four</a></li>
</ul>
```

Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/csstable/index\\_youarehere.html](http://midwestwebdesign.net/tutorials/csstable/index_youarehere.html)

## Summary

In this article you learned how create a stable and practical layout using a table. You also learned how to control the layout and formatting using CSS. You also learned a bit about:

- Controlling the layout with CSS
- Formatting the appearance of our page with CSS
- Formatting images inside an unordered list with CSS
- Create image rollovers with CSS and unordered lists
- Create a custom class selector to indicate a “you are here” marker

From this article, you should come to the understanding that creating a table and using CSS to control the layout and formatting is still a valid and stable approach in some instances for websites, no matter what the CSS experts say.

If you have questions, please follow the link below.

<http://midwestwebdesign.net/tutorials/contact.aspx>