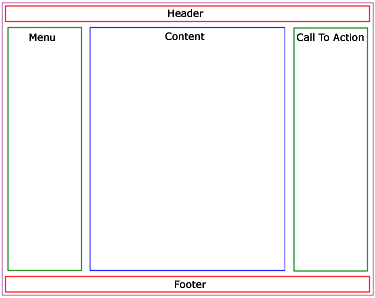
|  |
| --- |
| 3 Column CSS Layout |

[](http://www.vanseodesign.com/blog/3-column-layout.html)

Let’s start by looking at the html structure for the page and then look at the css used to layout the various elements. And once again the html structure is quite simple.

<div id="page">

<div id="header">Header</div>

<div id="menu">Menu</div>

<div id="content">Content</div>

<div id="action">Call To Action</div>

<div id="footer">Footer</div>

</div>

The only new element here is the Call to Action div, so called because often you’ll place your calls to action in this right column. As with the 2 Column Layout we have several divs all enclosed within a container page div. Nothing really exciting here so let’s move on to the css.

|  |
| --- |
| Page Div |

div#page {

border:1px solid purple;

width:755px;

margin:0 auto;

padding:5px;

text-align:left;

position:relative

}

div {

text-align:center;

}

The css for the page div and for the generic div are exactly the same as with the 2 column layout as you might expect with one exception. I’ve added a position of relative to the page div. The relative positioning actually doesn’t affect the page div at all in how it’s displayed, but it will become important later when we introduce our new column to the layout. I’ll save discussion of this positioning until we reach that div.

|  |
| --- |
| Header Div |

div#header {

border:2px solid red;

width:750px;

height:30px;

}

I’ve added nothing to the header div from the 2 column example. You can read that post if you’re confused about anything here. Yawn. When are we going to see something new?

|  |
| --- |
| Menu Div |

div#menu {

border:2px solid green;

width:150px;

float:left;

margin:10px 0 10px 5px;

height:500px;

}

There are two minor changes in the menu div though essentially it’s still the same as the last layout. I’ve reduced the width a little to allow for the third column. I’ve also increased the height just so it extends down the page. You may or may not want to set the height in your design. Again to position the menu I’ve floated it to the left. Like I said nothing really new.

|  |
| --- |
| Content Div |

div#content {

border:2px solid blue;

width:400px;

margin:10px 0 10px 175px;

min-height:500px;

\_height:500px

}

Believe it or not there’s really nothing new here with the content div either. C’mon on already? When are we going to get to the new stuff? Well I did reduce the width to make room for the new column and since out menu column has also been reduced in width I’ve decreased the left margin.

Again keep in mind we don’t want to use any css positioning on the content div since we want it to remain in the normal document flow to help control the position of the footer. A css layout doesn’t always mean using css positioning on everything. Again nothing new from last time. Hang on just a second more. The new stuff is coming I promise.

|  |
| --- |
| Action Div |

Finally something new. You knew it had to be new though since this column wasn’t included in the 2 column layout huh? No fooling you.

div#action {

position:absolute;

top:50px;

right:10px;

border:2px solid green;

width:150px;

margin:0;

height:500px;

}

The action div shares a little in common with the menu div in its width and height. I even used the same border. The margin has been set to 0 however since I used absolute positioning to layout this column and didn’t need the margin to push the div away from the page div border.

Like I just said the new column uses absolute positioning. At least I think that was me. I know I heard it somewhere. The key to css positioning is understanding where your element will sit at the default positon of 0 for the top and the left. When something is positioned in css its origin or (0, 0) position is at the (0, 0) position of its containing positioned element. Huh? Let’s use the specifics of this example to explain it again. Here’s where that relative position on the page div comes in.

I’ve specified a top position of 50px and a right position of 10px. They key question though is 50px and 10px from where. The action div is contained by the page div. The action div is inside the page div so the page div is the container. Since I’ve added a position: relative to the page div the container div also has positioning and the origin in this case is the upper right corner of the page div. It’s upper right since I’m setting the action div at the top and the right. I could also have specified bottom and left in which case the origin would be in the lower left of the page div. I could have positioned the action div using bottom and left just with different px amounts. It’s generally easiest though to position things from the nearest corner.

Now had there been no relative position on the page div the origin would be different. Instead of the upper right corner of the page div the origin would have been the upper right of the body. You can test this by grabbing the source and removing the position from the page div. The action div should move further to the top and right if you do. It will now be 50px and 10px from the corner of your browser window. Remember that the origin will be the closest containing div that is also positioned. If none of the containing div.’s have positioning applied the origin will ultimately be in the body of the page.

|  |
| --- |
| Footer Div |

div#footer {

border:2px solid red;

width:750px;

height:30px;

}

Sadly there’s nothing at all new here. Not even a width change. Maybe I should have changed the color of the border just to have something I could say about the footer. I’ll just remind you that since we’ve kept the content div as part of the normal document flow the footer will always sit just below the content div. The distance below will be the margin-bottom we applied to the content div which here was 10px.

|  |
| --- |
| Conclusion |

We really didn’t need to do a lot to add a third column to our 3 column css layout. We obviously needed to add the new div for the 3rd column and used absolute positioning to place it in our layout. We naturally had to allow for the space this column would occupy so we reduced the width of the other 2 columns. Pretty obvious stuff. The only trick here was to add relative positioning to the page div, which made things easier, but wasn’t necessary since we could have positioned the new div in relation to the body instead.

I hope this 3 column layout as well as the previous 2 column layout has helped to get you started on your own css layouts. Neither layout needs to be complicated nor do I hope I’ve simplified how to build their basic structure. To flesh out your pages you would start adding things inside each of the divs, much in the same way you might nest tables inside of other tables. The difference here is that once you get used to using css layouts you’ll find you have more control over how you can layout your page than you would using tables. You’ll also find you’ll use less code and spacer images as well as being able to make the images you do include smaller. Your pages will load faster and you’ll find your code easier to maintain. So get out there and start practicing your 2 and 3 column css layouts.