

Advanced CSS Training

Advanced CSS Page Layout

Lesson 1, Activity 2: Resetting Styles

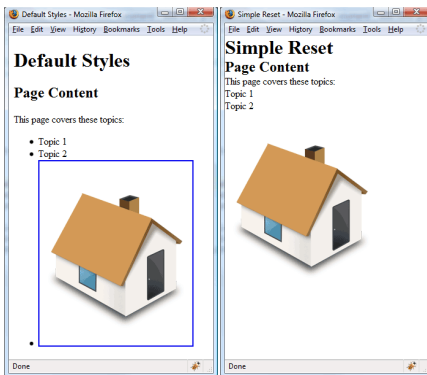
Web browsers make certain assumptions that can sometimes interfere with layouts. Designers often have to undo or override these defaults. It has become a common practice to do so from the outset. This practice has come to be known as "resetting" styles.

A very simple reset that affects page layout is shown below:

```
* {
  margin:0;
  padding:0;
  border:0;
}
```

Note the universal selector. This effectively does away with all margin, padding and borders for every element including the body, headings, lists, and list items. It then becomes the designers responsibility (freedom?) to assign margin, padding and borders.

The screenshots below illustrate the impact of this reset:



Resets can get much more sophisticated. CSS Guru Eric Meyer provides a freely downloadable detailed reset style sheet at <http://meyerweb.com/eric/thoughts/2007/05/01/reset-reloaded/>

Using the simple reset we have shown above gives us a blank slate on which to lay out our page. Now we have to make some decisions:

- Will our design use a fixed width or the full width of the browser?
- How many "rows" and "columns" will we have?
- What sort of positioning should we use: absolute, relative, fixed or static?

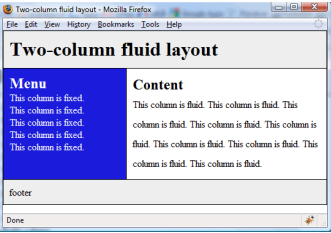
We address these questions in the sections below by looking at specific layouts. For all layouts, we'll assume we want a header and a footer.

Lesson 1, Activity 3: Full Width Layouts

Pages designed to use the full width of the browser window are often described as having a "fluid" or "liquid" layout. That's because one or more of the columns must be automatically resized according to the size of the browser window.

Two-column

We'll start with the simple two-column layout shown below:



As you can see, the page has four content areas. Let's take a look at the HTML first:

Code Sample:

CodeSample\Layout\Demo\fluid-two-column.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<link href="fluid-two-column.css" rel="stylesheet">
<title>Two-column fluid layout</title>
</head>
<body>
<div id="wrapper">
<div id="header">
<h1>Two-column fluid layout</h1>
</div>
<div id="main">
<div id="menu">
<h2>Menu</h2>
<p>This column is fixed.</p>
<p>This column is fixed.</p>
<p>This column is fixed.</p>
<p>This column is fixed.</p>
<p>This column is fixed.</p>
</div>
<div id="content">
<h2>Content</h2>
<p>This column is fluid.</p>
</div>
<div class="clearer"></div>
</div>
<div id="footer">Footer</div>
</div>
</body>
</html>
```

The four content areas have the following ids: header, footer, menu, and content.

The page content is contained in a wrapper div, which we can use to put a border around the content. Later we'll see how we can use it to adjust the width of the whole content. The wrapper div is divided into three "rows": header, main, and footer.

The header and footer rows are simple divs.

The main row is broken up into two "columns": menu and content. It also has a "clearer" div, which we'll discuss shortly.

Now the CSS:

Code Sample:

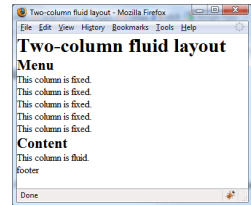
CodeSample\Layout\Demo\fluid-two-column.css

```
* {
margin:0;
padding:0;
border:0;
}
#wrapper {
border:1px solid #000;
}
#header {
border-bottom:1px solid #000;
padding:10px;
background-color: #eee;
}
#main {
background-image: url(Images/bgMain.gif);
background-repeat:repeat-y;
}
#menu {
width:180px;
float: left;
padding:10px;
border-right:1px solid #000;
color:#fff;
}
#content {
margin-left:200px;
border-left:1px solid #000;
padding:10px;
line-height:2em;
}
.clearer {
clear:both;
}
#footer {
border-top:1px solid #000;
background-color: #eee;
padding:10px;
}
```

1. We'll start by resetting margins, padding, and borders with the universal selector:

```
* {
margin:0;
padding:0;
border:0;
}
```

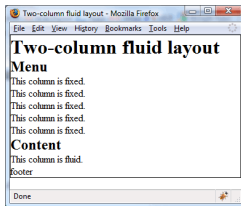
The result:



2. Next let's add a border the wrapper div.

```
#wrapper {
border:1px solid #000;
}
```

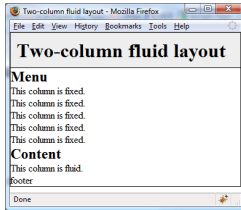
The result:



3. Next let's add a bottom border to the `header` and give it some padding and a background.

```
#header {
  border-bottom: 1px solid #000;
  padding: 10px;
  background-color: #eee;
}
```

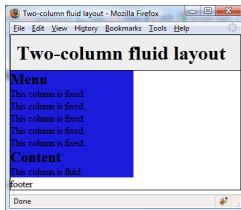
The result:



4. Next let's add a background image to the `main` `div`. The background image is 1px high by 200px wide. This is a trick to apply a background color to the left column that extends all the way to the bottom of the column. If we put the background image or a background color on the `menu` `div` itself, the background would end at the bottom of the content in that `div`. By putting it on the containing `main` `div`, we make sure that the background extends to the bottom of the whole column.

```
#main {
  background-image: url(Images/bgMain.gif);
  background-repeat: repeat-y;
}
```

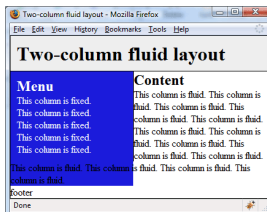
The result:



5. Now we need to float the `menu` `div` left so that the `content` `div` will come up to its right edge. We'll also add some other styles to make it more readable and give it a width of 180px and padding of 10px, which gives it a de facto width of 200px: 180px + (2 x 10px):

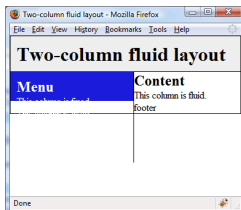
```
#menu {
  width: 180px;
  float: left;
  padding: 10px;
  border-right: 1px solid #000;
  color: #fff;
}
```

The result:



What happened there??? The text in the `content` `div` is wrapping underneath the menu. We don't want that.

6. And there's another problem - watch what happens when we shorten the text in the `content` `div`:

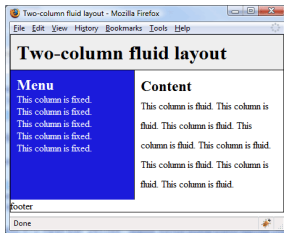


The height of the `main` `div` only extends to the bottom of its content, which it considers to be the `content` `div`. So the background only tiles down that far.

7. The solution to the wrapping problem is to set the left margin of the `content` `div` to the de facto width of the `menu` `div`:

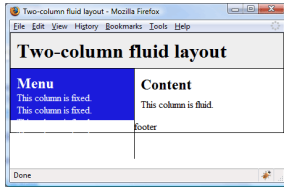
```
#content {
  margin-left: 200px;
  border-left: 1px solid #000;
  padding: 10px;
  line-height: 2em;
}
```

The result (Notice we added the content back to the `content` `div`):



That looks good.

8. But what happens when we remove content from the `contentdiv`:

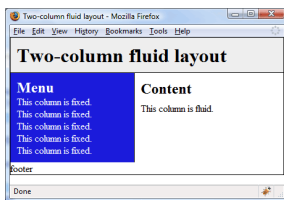


This is the same problem we mentioned earlier. The height of the `maindiv` only extends to the bottom of its content.

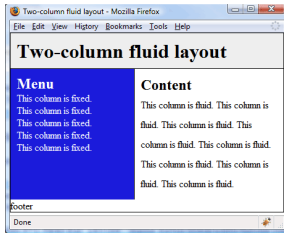
9. This is where the clearer comes in. By putting the clearer `div` at the bottom of the `maindiv`, we force its height to extend:

```
.clearer {
  clear:both;
}
```

The result:



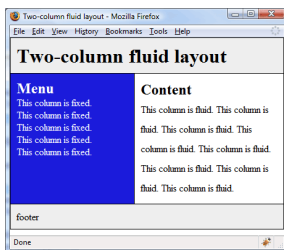
Good! We're almost there. And it even works when the `contentdiv` grows:



10. Now we just need to style the footer a little to visually separate it from the rest of the page:

```
#footer {
  border-top:1px solid #000;
  background-color: #eee;
  padding:10px;
}
```

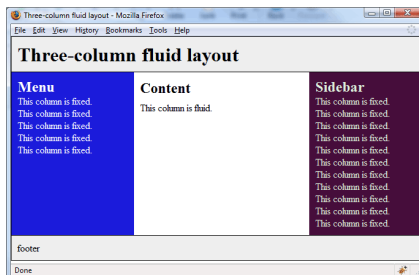
The result:



And we're done. Phew.

Three-column

Let's see how we would add a third column with the right and left columns fixed and the center column fluid:



First the HTML:

Code Sample:

[CeeBopel layout/Demos/fluid-three-column.html](#)

----- CODE OMITTED -----

```

<div id="wrapper">
  <div id="header">
    <h1>Three-column fluid layout</h1>
  </div>
  <div id="main">
    <div id="inner">
      <div id="menu">
        <h2>Menu</h2>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
      </div>
      <div id="sidebar">
        <h2>Sidebar</h2>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
        <p>This column is fixed.</p>
      </div>
      <div id="content">
        <h2>Content</h2>
        <p>This column is fluid. This column is fluid.
        </p>
      </div>
      <div class="clearer"></div>
    </div>
  </div>
  <div id="footer">footer</div>
</div>
</body>
</html>

```

Things to notice:

1. We've added a sidebar

. Notice the location: it comes before the content

- 2. We've added an inner

Now the CSS:

Code Sample:

[Go Back to layout Demos / fluid-three-column.css](#)

```

---- CODE OMITTED ----
#inner {
  background-image: url(images/bgMainInner.gif);
  background-position:right;
  background-repeat:repeat-y;
}
#menu {
  width:180px;
  float: left;
  padding:10px;
  color:#fff;
}
#sidebar {
  width:160px;
  float: right;
  padding:10px;
  color:#ded;
}
#content {
  margin-left:200px;
  margin-right:200px;
  padding:10px;
  line-height:2em;
}
---- CODE OMITTED ----

```

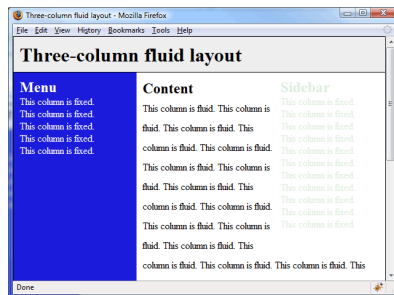
1. First, we need to do is to float the sidebar to the right:

```

#sidebar {
  width:160px;
  float: right;
  padding:10px;
  color:#ded;
}

```

The result:



Notice how the content in the center flows under the side bar.

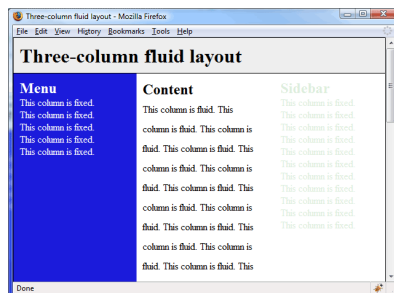
2. To fix that we need to give the content

```

#content {
  margin-left:200px;
  margin-right:180px;
  padding:10px;
  line-height:2em;
}

```

The result:



3. Finally, to get the background, we use the same trick with our new inner wrapper

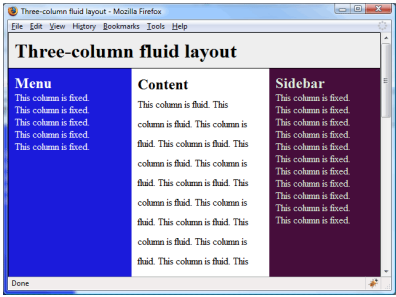
```

#inner {
  background-image: url(images/bgMainInner.gif);
  background-position:right;
  background-repeat:repeat-y;
}

```

)

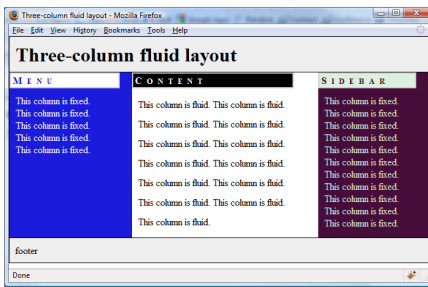
The result:



Lesson 1, Activity 6: Positioning the Headings

Duration: 15 to 25 minutes.

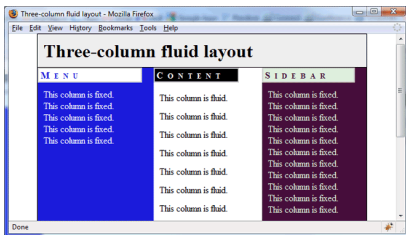
In this exercise, you will position the heading elements within the columns, so the page looks like this:



1. Open [CssPageLayout/Exercises/fluid-three-column.html](#) in your browser. It should look familiar.
2. Open [CssPageLayout/Exercises/fluid-three-column.css](#) in your editor. Modify the code so that the page looks like the screenshot above.
3. Although it's not required, you may find that it helps to modify the HTML file as well.
4. Save your changes and test your solution in the browser.

Challenge

See if you can change the page so that the design is centered takes up 85% of the width of the screen:



You'll only need to change the CSS.

Solution:

CssPageLayout/Solutions/fluid-three-column.html

Solution:

CssPageLayout/Solutions/fluid-three-column.css

```
%
margin:0;
padding:0;
border: 0;
}
----- CODE OMITTED -----
.column h2 {
position:relative;
left:-10px;
top:-10px;
padding:3px;
line-height:1em;
border:2px solid #ccc;
font-weight:normal;
font-size:small;
letter-spacing:.5em;
}

#menu h2 {
background-color: #fff;
color:#1b1b1b;
}

#sidebar h2 {
background-color: #dede;
color:#460d3b;
}

#content h2 {
background-color: #000;
color:#fff;
}
----- CODE OMITTED -----
```

Challenge Solution:

CssPageLayout/Solutions/fluid-three-column-challenge.css

```
* {
margin:0;
padding:0;
border:0;
}
body {
min-width:600px;
}
#wrapper {
border:1px solid #000;
width:85%;
margin:auto;
}
--- CODE OMITTED ---
```

When you set `margin` to `auto` it makes the left and right margins equal. You might have tried setting each margin to 7.5%. Theoretically, that should work, and it does in Firefox and Safari, but it's buggy in IE7. Use `auto` instead.

Another rule we've added to this solution is:

```
body {
  min-width: 600px;
```



```
    }
```

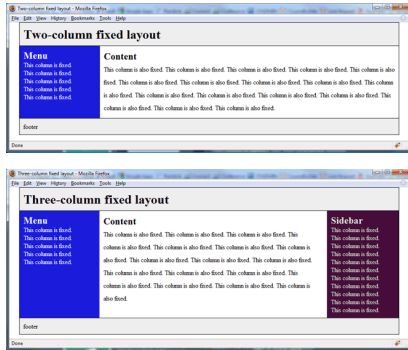
This prevents the columns from wrapping when the browser window is made really small. IE6 doesn't support `min-width`, but it works for all other modern browsers.

Lesson 1, Activity 7: Fixed Width Layouts

Changing our fluid layouts to fixed layouts is easy. In fact, the challenge to the last exercise gets us most of the way there. Instead of setting the width to a percentage value, we set it to a fixed-width value in pixels and we set `margin` to `auto`:

```
#wrapper {
border: 1px solid #000;
width: 950px;
margin: auto;
}
```

The screen shots below show the results:



The full code can be found in [CssPageLayoutDemoFixedTwoColumn.css](#) and [CssPageLayoutDemoFixedThreeColumn.css](#).

A Recommendation on Positioning

When designing, we recommend the following general principles with regards to positioning choices:

1. Try to layout the page using static positioning, which is the default. In this case, you'll rely entirely on `float` and the box model (`padding`, `margin`, and `border`) to lay out your page.
2. If static positioning doesn't work (e.g. you need to have overlapping elements), try relative positioning first.
3. Finally, use absolute positioning in conjunction with the other forms of positioning and only when you need to have an element positioned in an exact location.

Other Methods of Layout

There are many other ways to create multi-column layouts, each with its own advantages and disadvantages. We feel that the method we've laid out here is a good choice; however, it does have its disadvantages:

- **Meaningful markup:** One of the advantages of CSS is that it makes it possible to structure HTML in a logical way. Ideally, you shouldn't have to throw in extra tags for formatting purposes. The methodology we show here makes a small compromise on this in two ways:
 1. The wrapper `div`s used to create "rows" and to apply backgrounds to columns are there for formatting purposes only. There is no other reason to structurally combine the individual columns in the "row".
 2. The "clearer" `div` contains no content at all. It's only there for formatting purposes.
- **Content order:** In the three-column layout, we put both the menu and the sidebar before the main content. This is counter-intuitive as they don't show up on the page in that order. If a screenreader were to read the page, it wouldn't know to read the main content area before it read the sidebar. Likewise, search engines might place more importance on words that show up earlier in the source code. We've had to push those words down the page to achieve the layout we want.

The good news is that the other methodologies use the same basic CSS properties you have learned to use here. So if you decide that you can't deal with either of the two compromises we've made above, you should be able to pick up the other methodologies relatively easily.

One that we recommend is known as the [Holy Grail](#). This methodology uses less meaningless markup (i.e. markup for formatting purposes only) and it doesn't compromise on content order. However, it makes other compromises:

1. It requires CSS hacks to make it work in Internet Explorer 6 and 7. See <http://www.geert-rieselmann.net/development/the-holy-grail-css-layout-fix-for-ie2/> for a cross-browser working example.
2. It doesn't allow for backgrounds that flow to the bottom of the "row".
3. It's more difficult to modify and maintain. For example, to add padding to one column, you need to adjust multiple declarations in the code. With the methodology we show here, you simply change the value of the appropriate padding property.


```

padding-left: 15px;
color: #666;
text-decoration: none;
}
---- C O D E   O M I T T E D ----
#mainmenu a {
display: block;
height: 40px;
padding-right: 25px;
padding-left: 25px;
float: left;
text-decoration: none;
padding-top: 15px;
background-attachment: url(Images/menudivider.png);
background-repeat: no-repeat;
background-position: 0px 3px;
color: #5B920A;
}
---- C O D E   O M I T T E D ----
#homeleft {
width: 480px;
float: left;
}
---- C O D E   O M I T T E D ----
#homeright .column1 {
width: 31%;
float: left;
padding-right: 5px;
}
#homeright .column3 {
float: left;
width: 32%;
padding-left: 8px;
}
#homeright .column2 {
float: left;
width: 33%;
}
---- C O D E   O M I T T E D ----
#submenu a:visited, #submenu a:active {
background-image: url(Images/submenudivider.png);
background-repeat: no-repeat;
background-position: left;
display: block;
float: left;
height: 16px;
padding-top: 2px;
padding-right: 15px;
padding-left: 15px;
color: #666666;
text-decoration: none;
}
#submenu a:hover {
background-image: url(Images/submenudivider.png);
background-repeat: no-repeat;
background-position: left;
display: block;
float: left;
height: 16px;
padding-top: 2px;
padding-right: 15px;
padding-left: 15px;
color: #000000;
text-decoration: none;
}
---- C O D E   O M I T T E D ----
.productimage {
float: left;
}
---- C O D E   O M I T T E D ----

```