LAYOUT #2 – A fixed width multi-column layout

During this first part of the tutorial, we are going to make multiple columns using absolute positioning. In the end, your layout should look like this:

For this layout, we will use relative and absolute positioning to create a header and multiple columns (however many that you want – in this case we will create two – a skinny left column and a wider right column). In order to see it centered in your browser we can start where we left off in layout #1.

Before we begin, let’s make sure that we start at the right spot. At the end of the first tutorial (Layout #1) we should have a page that looks like this:
Our html should look something like this:

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Chris's Art 249 Test Site #1</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link href="css/chris-testSite01.css" rel="stylesheet" type="text/css">
</head>

<body>
<div id="container">

<h1>This is a Header 1</h1>
<h2>This is a Header 2</h2>
<h3>This is a Header 3</h3>

<p>This is a paragraph. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc adipiscing molestie neque. Nullam at nisl nec enim pharetra fermentum. Pellentesque pharetra ante at pede rhoncus rhoncus. Praesent vulputate, est eu pulvinar porta, lacus dolor egestas leo, non ultrices purus nisi ut elit. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. In quis felis ut velit sodales interdum. Vivamus tristique elit. Praesent eget diam et leo egestas facilisis. Maecenas adipiscing ligula a mi cursus</p>

</div>
</body>
</html>
```

<a href="#">This is a link</a>

<ul>
<li>This is a list item</li>
<li>This is a list item</li>
<li>This is a list item</li>
<li>This is a list item</li>
</ul>

</div>

</body>
</html>

Notice that all content right now is inside of one DIV tag called “container”.

The css for this file should look something like this:

/* CSS Document */

html {
    height: 100%;
}

body {
    height: 100%;
    margin: 0;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 10px;
    background-color: #CCCCCC;
    text-align: center;
}

img {
    border: 0;
}

h1 {
    font-size: 18px;
    color: #000066;
    margin: 0;
}

h2 {
    font-size: 16px;
    color: #6699CC;
    margin: 0;
}
1. The first thing that we may want to do, is give our “container” DIV a white background color. You can do this via the css by setting the background of container to white
   - `background: #fff;`

   You'll notice that while this worked, it created a scrolling problem. Whenever you resize your browser to be small, then scroll down, the white background does not scroll with the page. This is an annoying bug that pops up whenever you stretch a DIV tag to a height of 100% - which we did in LAYOUT #1.

   A powerful workaround is to use a centered background image instead of setting the DIV background property. To do this, you must create an image that is 1 pixel in height and the exact width of centered column (in our case 750px). Once you do this, and save it into your images folder, you can then set the background of your body to be a background-mage like this:
• `background: #999 url(../images/bg_750px_fff.gif) repeat-y center;`

This sets the background of the entire body to be grey - `#999` - then sets a background image on top if it - `url()` - then tells the browser to only repeat the background image along the y axis - `repeat-y` - then to center it in the browser window - `center`.

2. Now that we have the background color of our page set, we need to make some content for our second column. For this example, we will do this by copy and pasting all html within the DIV tag except for our logo, which we will put inside of its own header tag.

Your new html should look like this:

```html
<div id="container">
  
  <h1>This is a Header 1</h1>
  <h2>This is a Header 2</h2>
  <h3>This is a Header 3</h3>
  
  
  <a href="#">This is a link</a>
  
  <ul>
    <li>This is a list item</li>
    <li>This is a list item</li>
    <li>This is a list item</li>
    <li>This is a list item</li>
  </ul>
  
  <h1>This is a Header 1</h1>
  <h2>This is a Header 2</h2>
  <h3>This is a Header 3</h3>
  
</div>
```
3. Next we need to put the logo inside of a DIV labeled with an ID of “header”, one set of html content inside of a DIV labeled with an ID of “leftcolumn”, and one set of html content inside of a DIV labeled with an ID of “rightcolumn”.

Your final html should look like this:

```html
<div id="container">
  <div id="header">
  </div>
  <div id="leftcolumn">
    <h1>This is a Header 1</h1>
    <h2>This is a Header 2</h2>
    <h3>This is a Header 3</h3>
    <a href="#">This is a link</a>
    <ul>
      <li>This is a list item</li>
      <li>This is a list item</li>
      <li>This is a list item</li>
      <li>This is a list item</li>
    </ul>
  </div>
  <div id="rightcolumn">
    <h1>This is a Header 1</h1>
    <h2>This is a Header 2</h2>
    <h3>This is a Header 3</h3>
    <p>This is a paragraph. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc adipiscing molestie neque. Nullam at nisl nec enim pharetra fermentum. Pellentesque pharetra ante at pede rhoncus rhoncus. Praesent vulputate, est eu pulvinar porta, lacus dolor egestas leo, non ultricies purus nisi
    </p>
    <a href="#">This is a link</a>
    <ul>
      <li>This is a list item</li>
      <li>This is a list item</li>
      <li>This is a list item</li>
      <li>This is a list item</li>
    </ul>
  </div>
```


<a href="#">This is a link</a>
<ul>
<li>This is a list item</li>
<li>This is a list item</li>
<li>This is a list item</li>
<li>This is a list item</li>
</ul>

4. Next, we need to define these three DIV tags that we just created inside of the css.

Inside of your css file, type out empty declarations in the DIV tag section just underneath the container definition:

/* DIV TAGS */

/* DIV TAGS */

#container {
height: 100%;
border: 0px solid black;
width: 750px;
margin: 0 auto;
text-align: left;
}

#header {
}

#leftcolumn {
}

#rightcolumn {
}

5. Since we will be using absolute positioning INSIDE of the DIV “container” we need to set the container DIV the position to “relative”. This will allow any nested DIVs to align themselves in a “relative position” to it.

#container {
position: relative;
height: 100%;
border: 0px solid black;
width: 750px;
margin: 0 auto;
6. Next we need to set the positioning of our three other DIV tags. We will set them all as "absolute;". This will give us the power to place them anywhere on the page that we would like to place them.

```css
#header {
    position: absolute;
}

#leftcolumn {
    position: absolute;
}

#rightcolumn {
    position: absolute;
}
```

7. Whenever you use absolute positioning, you should always set the top and left attributes of the block level element (the DIV). In this case, we need to do a little math. The header is usually jammed into the upper left hand corner of the layout, or in this case, the “container” DIV. Therefore we can set it to `top: 0;` and `left: 0;`

But, after doing this, we realize that the logo is too far over to the left hand side – really it should be 20px over to the right. We can resolve this in multiple ways: one way is to set the left attribute to 20px. We do this by changing `left: 0;` to `left: 20px;` the other way is to give it a left margin of 20px. Either will work.

Your header should now look like this:

```css
#header {
    position: absolute;
    top: 0px;
    left: 20px;
}
```

8. Next we need to do some math in order to set the layout and margins of our two columns. Instead of using the margin property to define our margins, we are instead going to use absolute positioning.

Both columns should be 110px from the top (you can figure this out either in Photoshop or by trial-and-error), so we can set both of their top attributes to 110px.

But now we need to figure out where to place each block. If we had two columns, then there will be three hypothetical gutters in between them. Therefore, if we want the left gutter (or margin) to be 20px, the right gutter (or margin) to be 20px, and the center gutter (or space in between the two columns) to be 20px, then we need to control the width of each column and do the math so that it will all add up to 750px.

If we make the left column 200 pixels, then we can add 200 + the sum of all gutters (60) to get 260. By subtracting this number from 750 (the width of our central column), we will get the width of the right column, which should be 490.

So next we should set all of these properties inside of our css:
9. Your final CSS should look like this:

/* DIV TAGS */

#container {
    position: relative;
    height: 100%;
    border: 0px solid black;
    width: 750px;
    margin: 0 auto;
    text-align: left;
}

#header {
    position: absolute;
    top: 0px;
    left: 20px;
}

#leftcolumn {
    position: absolute;
    top: 110px;
    left: 20px;
    width: 200px;
}

#rightcolumn {
    position: absolute;
    top: 110px;
    left: 240px;
    width: 490px;
}

And your final layout should look like this:
LAYOUT #3 – horizontally and vertically centered rectangle.

The goal of this layout is to center a fixed width and fixed height box into the center of our browser window.

To achieve this, we use a table, instead of a div block element, in order to avoid using a css browser hack. We will use one table with a 100% width and 100% height with one row and one cell. Then, just like we did in Layout #1, we center a div tag inside of this cell will use a combination of margins set to “auto” and text-alignment set to “center”.

The layout will look like this:

1. First we need to start a completely new html file and a completely new css file.
2. Next, we need to set up the basics. In the html file, let's add a small amount of content:

```
```


3. In the css file, since we do not need margins and we will want our table to stretch to the width and height of the browser window, we need to set the html and body height attributes to 100%. We can also set the font and background color.

/* CSS Document */

html {
  height: 100%;
}

body {
  height: 100%;
  margin: 0;
  font: 12px Verdana, Arial, Helvetica, sans-serif;
  color: #000;
  background: #999;
}

4. Next we need to add an empty table into our html file, with no properties at all.

<table>
  <tr>
    <td>
      ← our box and all content will go here →
    </td>
  </tr>
</table>

5. Next we need to set the height, width, and alignment properties for our table and the single td cell. The vertical-align: middle will make all content in the cell align in the middle vertically. The text-align: center will center all content in the cell into the center vertically only in IE. We will fix how it appears in Firefox later on.

table {
  width: 100%;
  height: 100%;
}

td {
  vertical-align: middle;
  text-align: center;
}

6. Next we need to place all of our content (our google image and the one paragraph) into an empty div. We will give this div an id called “box”.

   •  <div id="box"> ←content → </div>

7. Next we need to declare this “box” div tag. We need to set its height and width here, along with its background color. In order to fix the horizontal centering in Firefox, we
will give it a left & right margin of “auto”.

```css
#box {
  margin: 0 auto;
  height: 400px;
  width: 700px;
  background: #fff;
}
```

8. Once you are done, your final page should look like this:

![Google search page](image)

9. We’re not done yet – if you test this page on a Mac using IE 5.x, you’ll notice that our box is not centered, and appears broken. To fix this, we need to back up and try a completely new method: negative margins and absolute positioning.

It sounds odd, but try this method and you will see for yourself.

First, inside the css file, you need to add 3 lines to your surrounding “box” or “container”:

```css
#box {
  position: absolute;
  top: 50%;
  left: 50%;
  margin: 0 auto;
  height: 400px;
  width: 700px;
  background: #fff;
}
```
10. Since the size of our rectangle is 700x400, the next thing we need to do is add a negative top margin and a negative left margin of half the height and width.

So we need to change:

```
margin: 0 auto;
```

to this:

```
margin: -200px 0 0 -350px;
```

11. Your final css for the box div should look like this:

```css
#box {
    position: absolute;
    top: 50%;
    left: 50%;
    margin: -200px 0 0 -350px;
    height: 400px;
    width: 700px;
    background: #fff;
}
```

12. Since we are using absolute positioning in combination with negative margins, we no longer need to use a table to help center our box, therefore, **in the html you can delete out all table code and in the css you can delete out all table attributes.**

13. Last step – test it on all browsers – including Mac IE 5.x, of course.