This guide will attempt to take you step by step, through the process of creating a fully functioning CSS layout. I will try my best to explain the concepts behind each step, but a lot of the time other people have already covered these things better than I can. Because of this there will sometimes be links to more information on external sites.

Please use a modern browser such as **Gecko, Firefox, Google Chrome** or **Safari** for this exercise. Don’t worry though, your site should work in IE at the end of it.

⚠️ Note: This tutorial is out of date. I am in the process of writing a new one to reflect the best practices of the present day.

### Contents

1. Introduction
2. The Design
3.setDefault HTML Template
4. Setting the canvas
5. The major elements
6. Floats
7. Additional Structure
8. Some basic text styles
9. The Header
10. Footer
11. The Navigation (and)
12. Getting it right in IE
The Design
Below is the design we will be using as the basis of this tutorial. Our mission is to slice this puppy up using nothing but XHTML, CSS, and a few images:

First we need to identify the main structural elements of the design, so that we know how to structure our HTML document. The web is very heavily based around rectangles, and we need to remember that when dividing up our design. These main divisions we make will end up being <div> tags. A <div> is basically a rectangular container that we can position using CSS.

The diagram below shows how we will divide the design.
We have identified 5 major elements:

- **Main Navigation**
  The primary navigation for this website. The images will change on hover (when the mouse cursor is on top of it).
  Width: 700px
  Height: 60px

- **Header**
  The website header includes a background image (purely for aesthetics), and the company name.
  Width: 700px
  Height: 100px

- **Content**
  The bulk of the website's content will go here.
  Width: 490px
  Height: Changes depending on content

- **Sidebar**
  This will have second-tier content that isn't as important as the main content.
  Width: 280px
  Height: Changes depending on content

- **Footer**
  Copyright information, credits, and an alternative text navigation.
  Width: 700px
  Height: 66px

This site will also be centered in the browser window. We now have all the info we need to start.

3. Default HTML Template

I have created a basic HTML document that I use as a starting point for all my websites. If you do not understand what a particular line of code means, hold your mouse over that line for an explanation.

Copy the template and paste it into your HTML editor of choice. (Mine is Macromedia HomeSite).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-type" content="text/html; charset=UTF-8" />
<title>Company Name - PageName</title>
<meta http-equiv="Content-Language" content="en-us" />
<meta http-equiv="imagetoolbar" content="no" />
<meta name="MSmartTagsPreventParsing" content="true" />
<meta name="description" content="Description" />
<meta name="Keywords" content="Keywords" />
<meta name="author" content="Enlighten Designs" />
<style type="text/css" media="all">@import "css/master.css";</style>
</head>
<body>
</body>
</html>
```

Save this as index.html in your websites root (htdocs) directory.

The structure of your website directories should be like so:

```
htdocs
|-- css
|-- images
```
4. Setting the canvas

As you’ll notice in the design, everything on our page is 760px wide or less, and nothing floats outside that width. What we are going to do is create a container for our page that is 760px wide, and centered in the middle of the page. Our 5 main elements will be placed inside this container.

Between the `<body>` tags, create a `<div>` with an id of `page-container` attribute:

```html
<div id="page-container">
  Hello World.
</div>
</body>
```

And that's all the HTML we need for our container. Onto the CSS.

Create a new blank text file, and save it as `master.css` in the `/css` directory.

Create a new rule in the stylesheet to select the page-container:

```css
#page-container {
}
```

The `#` in front of the id tells the browser that we are selecting an id. For a class we would use a . instead e.g. `#page-container {}`.

An id is a unique identifier that we use for things that are only going to occur once on the page. So for headers, footers, navigation, etc we use ids, and for any recurring elements like links we should use classes, which can occur multiple times on the same page.

We won't be able to see the changes we are making to this `<div>`, because it is transparent by default. So the first thing we will do is make the background of the div red, to give us a visible indicator of what we are doing.

```css
#page-container {
  background: red;
}
```

You should see something like this across the full width of your browser:

```
Hello World.
```
First we should set a width of 760px on this div.

```css
.page-container {
  width: 760px;
  background: red;
}
```

Refresh the page in your browser to see the rule being applied.

Next we want to center this div. This is done by setting the margins on it to auto. When the left and right margins are set to auto, they will even each other out and the div will sit in the center of its container.

```css
.page-container {
  width: 760px;
  margin: auto;
  background: red;
}
```

Now you should have a centered red 760px wide block with “Hello World.” written in it. But its sitting about 8px away from the top/sides of the browser.

This is because the html and body tags have default margins and padding on nearly all browsers. So we need to write a CSS rule to reset the margins and padding on the html and body tags to zero. Add this rule to the very top of your css file:

```css
html, body {
  margin: 0;
  padding: 0;
}
```

A comma in between CSS selectors stands for “or”, so here the rule will be applied to the html tag or the body tag. Because both exist on the page, it will be applied to both.

Brilliant now our box is where it should be. Note that as more content is added to this div, it will automatically change its height to fit whatever content is placed inside it.

Let's move on.

5. The major elements

We need to add 5 divs, all with individual ids that describe their purpose. These divs will correspond to the major areas of the design we identified in Step 2. Replace the Hello World. text with the div’s below. Just for now we’ll also put text inside the divs for easy visual identification when we view the page.

```html
<div id="page-container">
  <div id="main-nav">Main Nav</div>
  <div id="header">Header</div>
  <div id="sidebar-a">Sidebar A</div>
  <div id="content">Content</div>
  <div id="footer">Footer</div>
</div>
```

Your page should now look something like this:

```
Main Nav
Header
Sidebar A
Content
Footer
```

Without CSS, the divs will be arranged from top to bottom, one under the other, in the same order as they are in the code. This is usually referred to as the “flow” of the document.

So let’s use the information we have to make our divs the correct size.

Remove the red background from the #page-container, and add a new rule for #main-nav. Set the background of the main nav to red so we can see it, and set its height to 50px:

```css
#page-container {
  width: 760px;
}

#main-nav {
  background: red;
  height: 50px;
}
```
Notice we didn't specify the width of the div. This is because by default, a div will stretch to fill its parent container, which in this case, is our #page-container div that was set to 700px wide.

Do the same thing for the header div, using the height we got in step one. Make this one blue.

```css
#header {
    background: blue;
    height: 150px;
}
```

While we're at it, let's do the footer. The footer can be orange.

Remember when writing your stylesheet, that you should group your styles together. So all the header styles would go together. All the footer styles would be together, and all the navigation styles would be together. Also I find it helps to structure them in a similar order to the HTML, so header near the top, footer near the bottom.

```css
#footer {
    background: orange;
    height: 66px;
}
```

Now the next 2 divs are slightly different. The heights are descendant on the content that's inside them, so we won't set a height. Let's make them darkgreen, and green. Put the rules in between the header and the footer rules in the css. This makes them easier to find once the stylesheet gets larger.

```css
#header {
    background: blue;
    height: 150px;
}

#sidebar-a {
    background: darkgreen;
}

#content {
    background: green;
}

#footer {
    background: orange;
    height: 66px;
}
```

If all has gone well, you should have a page that looks like this:

![Syled page layout](image-url)
6. Floats

Floats can be a tricky concept to get your head around. Basically, a float is an element that is aligned against the left or right side of its container. (For more detail, read maxdesigns in depth on introduction to floats).

In the case of this website, we are going to float our sidebar-a div to the right, with a width of 280px. Add the following to your CSS:

```css
#sidebar-a {
    float: right;
    width: 280px;
    background: darkgreen;
}
```

You have now successfully floated your first div, and you should now have a page that looks like this:

![Example of floated div layout](image)

Just for testing purposes, replace the text in the content div to this:

```html
<div id="content">
    Duis sodales est ac velit. Nullam rhoncus mauris ac metus. Maecenas vulputate aliquam odio.
    Duis scelerisque justo a pede. Nam augue lorem, semper et, porta eget, placerat eget, purus. Suspendisse mattis nunc vestibulum ligula.
    In hac habitasse platea dictumst.
</div>
```
Notice that the text in the content div wraps around the floated sidebar div, as shown below.

```
#content {
  margin-right: 200px;
  background: green;
}
```

This isn't what we want. We want the content div to sit alongside the sidebar div, with its right edge against the left edge of the sidebar.

An easy way to achieve this in a float layout like this, is to put a right margin on our content div that is the same width as our sidebar, in this case 200px. This will push the right edge of the content away from the right edge of the page-container.

Great, we've almost got the float layout set up. But there's one more thing we need to consider... what happens if the sidebar div is taller than the content div?

Let's see. Copy and paste this text into the sidebar div:

```
```
That's not what we want at all. The reason the footer hasn't moved down is because the sidebar is floated right.

Explanation: By default, any floated element will not push down elements that are below it. This is because floated elements are not considered part of the document flow, i.e., they are on another layer floating above the other elements, and because of this, it can't affect their positions.

What can we do to fix this problem? Introducing the 'clear' css property.

Add this to your stylesheet:

```css
#Footer {
  clear: both;
  background: orange;
  height: 60px;
}
```

When an element has the clear property assigned, if it comes into contact with a float it is placed right below where that float ends. You can specify if it is affected by only left floats or only right floats; in this case, we could use either 'right' or 'both'. We'll use clear: both just to be safe.
7. Additional Structure

Now that we have the base layout divs in place, we can add the rest of the structure that will make up the bare bones of this website.

The main things we still need to add are:

- Navigation Links
- Headings (Site Headings and Content Headings)
- Content
- Footer information (copyright info, credits, and alternative navigation)

In order to start implementing these things without breaking the page layout, we will create a helpful little class called "hidden".

Add this near the top of your stylesheet, after the body tag definition:

```css
.hidden {
  display: none;
}
```

What this means is now we can set any element in the site to have the class 'hidden', and it won't show on the page at all. This will come in handy later. You can forget about it for now.

Let's talk about headings.

Headings in an HTML document are defined by the tags `<h1>` through to `<h6>` in order of importance to the document. For example, `<h1>` for the website name, `<h2>` for the primary headings (ie page name), `<h3>` for secondary headings, etc...

We'll add an `<h1>` inside our Header div, and set it to the name of the company, Enlighten Designs in this case.

```html
<div id="header">
  <h1>Enlighten Designs</h1>
</div>
```

If you refresh your page you will notice that Enlighten Designs has come up in big letters inside the header, but there is also now a lot of white space around the heading. This is caused by the default margins on `<h1>` tags. So we need to strip the margins and padding by doing this:

```css
h1 {
  margin: 0;
  padding: 0;
}
```

Now we'll add the navigation. The ins and outs of how the navigation will work can be rather complicated, and will be addressed fully in its own section later on.
The navigation will be structured as a definition list (<dl>) with individual id's relevant to each navigation item on each definition term (<dt>). These Definition terms will have links to our major sections inside them. If that sounds confusing, just add this code to your main.nav div:

```html
1. <div id="main-nav">
2.   <dl>
3.     <dt id="about"><a href="#"/>About</a></dt>
4.     <dt id="services"><a href="#"/>Services</a></dt>
5.     <dt id="portfolio"><a href="#"/>Portfolio</a></dt>
6.     <dt id="contact"><a href="#"/>Contact Us</a></dt>
7.   </dl>
8. </div>
```

In easy to understand terms, the <dt> acts as a container, the <dt>'s are unique identifiers for each navigation item, and the links are the links.

We use the unique id's later when we come to make this navigation look like it should, with its sexy image rollovers. But more on that later.

If you refresh, you'll notice it looks a bit ugly, so for now, we'll just hide the navigation we added, with the "hidden" class we made earlier.

```html
1. <div id="main-nav">
2.   <div class="hidden">
3.     <dt id="about"><a href="#"/>About</a></dt>
4.     <dt id="services"><a href="#"/>Services</a></dt>
5.     <dt id="portfolio"><a href="#"/>Portfolio</a></dt>
6.     <dt id="contact"><a href="#"/>Contact Us</a></dt>
7.   </div>
8. </div>
```

"And like "that", it was gone...."

Now we'll jump down to the footer 'cause its relatively easy. There are 2 parts to the footer, the copyright info and credits on the left, and the alternative site nav on the right.

We want the alternate navigation to float right, like we did with the sidebar and the content, so we'll put that in the div first. In theory you should be able to float divs regardless of where they are in the source, but bugs in IE make this difficult, so for now, any floated items should come first in the source order.

Place it in a div with a unique id like so:

```html
1. <div id="footer">
2.   <div id="allnav">
3.     <a href="#"/>About</a> -
4.     <a href="#"/>Services</a> -
5.     <a href="#"/>Portfolio</a> -
6.     <a href="#"/>Contact Us</a> -
7.     <a href="#"/>Terms of Trade</a>
8. </div>
9. </div>
```

Underneath that div, we will add the copyright and credits text:

```html
1. <div id="footer">
2.   <div id="allnav">
3.     <a href="#"/>About</a> -
4.     <a href="#"/>Services</a> -
5.     <a href="#"/>Portfolio</a> -
6.     <a href="#"/>Contact Us</a> -
7.     <a href="#"/>Terms of Trade</a>
8. </div>
9. Copyright © Enlighten Designs
11. </div>
```

And thats the footer done for now. Just to make sure you're doing fine, this is what your site should look like:
Moving onto the main content area, let's add the content. I'm ripping this content directly off the design in step 2. Use `<h2>` tags for the headings "About" and "Contact Us". Enclose the paragraphs in `<p>` tags, and use `<br>` for line breaks.

01 <div id="content">
02  <h2>About</h2>
03  <p><strong>Enlighten Designs</strong> is an Internet solutions provider that specialises in front and back end development. To view some of the web sites we have created, view our portfolio.</p>
04  <p>We are currently undergoing a ‘face lift’, so if you have any questions or would like more information about the services we provide please feel free to contact us.</p>
05  
06  <h2>Contact Us</h2>
07  <p>Phone: (07) 855 6868<br />
08      Fax: (07) 855 6868<br />
09      Email: <a href="mailto:info@enlighten.co.nz">info@enlighten.co.nz</a></p>
10  <p><a href="#">More contact information.</a></p>
11 </div>

Refresh your page you'll notice there is more of that white space popping up around the content div. This is because of the default margins on the `<h2>` tags and the `<p>` tags.

We need to strip their margins and padding. However, we don't want to do this to every single paragraph tag or secondary heading that's going to be on the website. To do this we need to use 'child' CSS selectors.

All elements in HTML have a 'parent, child' relationship to one another. If tag a is inside tag b, then tag b is the parent of tag a. In the code above, our `<h2>` tags and our `<p>` tags are both children of the `#content` div.

If we want to select the child elements of a specific parent, we separate them with a space, like the example below.

```
#content h2 {
    margin: 0;
    padding: 0;
}

#content p {
    margin: 0;
    padding: 0;
}
```

So the above rules tell the browser to apply these styles ONLY to `<h2>`s and `<p>`s that are child elements of the `#content` div.

Next we make the text look a bit better.
8. Some basic text styles

Sweet Jesus! Are you sick of looking at those awful background colours or what?!
Get rid of them. Rip them all out except for the red navigation.

That looks a little better, but the text still looks horrible. Let’s set a global font family, colour, and size to use as a nice base. The font attributes set on the body will automatically inherit down to any other text in the site unless specifically overridden with another style. Make a new CSS rule just before the “hidden” class near the top of the stylesheet.

```css
body {
  font-family: Arial, Helvetica, Verdana, Sans-serif;
  font-size: 12px;
  color: #666666;
  background: #ffffff;
}
```

The stuff above is pretty self-explanatory.

If everything is going according to plan, you should be looking at something like this:

![Enlighten Designs](image)

What this needs is some padding to separate those blocks of content from each other.

According to the design, the gap below the content headings is roughly 15px, and the gaps below each paragraph are around 15px. So let’s grab those 2 CSS rules we made earlier and apply padding-bottom rules to them:

```css
#content h2 {
  margin: 0;
  padding: 0;
  padding-bottom: 15px;
}

#content p {
  margin: 0;
  padding: 0;
  padding-bottom: 15px;
}
```

We also need 25px of padding around the whole content div, and the whole padding div, giving them some space to breathe.

This part SHOULD be easy. In theory you would just set padding: 25px; on the divs, but sadly, due to problems in Internet Explorer, we can’t do this.
There are 2 possible ways to tackle this problem, one involves writing some funky CSS ‘Hacks’ to hide certain css rules from one browser, while showing it to another, but because padding is something we use a lot, we’re going to do it the other way.

The other way is to insert an additional div inside the divs which we want padded, and set their class to “padding”. Padding is the only thing that will be applied to these padding divs.

The reason this works is that the padding divs don’t have a set width. As a rule, try not to add padding and a static width or height on the same element.

```html
div id="sidebar-a">
  <div class="padding">
    Lorem  ipsum dolor sit amet, consectetur adipiscing elit. Nullam gravida enim ut risus.
  </div>
</div>
```

Do the same for the content div.

Now in the stylesheet we will create 2 new rules, shown here in green:

```css
#sidebar-a {
  float: right;
  width: 200px;
}
#sidebar-a .padding {
  padding: 25px;
}
#content {
  margin-right: 200px;
}
#content .padding {
  padding: 25px;
}
```

Using the same method as we did before, we have selected only the elements with a class="padding" that are children of the #sidebar-a, or #content divs.

The leading (vertical space between lines of text) on the content text and sidebar text should be larger, according to the graphic draft. In CSS, leading is set with the line-height attribute. Let’s add a line height of 16px:

```css
#sidebar-a {
  float: right;
  width: 200px;
  line-height: 16px;
}
#content {
  margin-right: 200px;
  line-height: 18px;
}
```

Moving on, the heading <h2>’s that we added look pretty ugly. Because the font they are using isn’t a web font, we are going to have to replace them with images. Create 2 images like the ones below, and put them in the images/headings/directory

![About](images/headings/about.gif)

![Contact Us](images/headings/contact.gif)

Replace the heading text with these images, but remember to keep the <h2> tags around the image tags, and remember to put all attributes on the images. All attributes are designed to display as an alternative to the image if a user is viewing the page in a browser that does not support images, or has images turned off. It is also useful for search engine spiders, as they can not understand images.

```html
<h2><img src="images/headings/about.gif" width="54" height="14" alt="About" /></h2>
<h2><img src="images/headings/contact.gif" width="58" height="14" alt="Contact Us" /></h2>
```

It’s starting to take shape. You should be looking at something like this:
9. The Header

To implement the header, we need to get the background image applied to the header div, replace the “Enlighten Designs” heading with a graphical logo, and position it in the correct place on the header (the dark grey bar to the right).

First create 2 images like the ones below (or just save these ones):

![Images/logo_enlighten.png](images/logo_enlighten.png)

![Images/header/about.jpg](images/header/about.jpg)

The first part is easy. Set a background image in the CSS using the format below:

```css
#header {
  height: 150px;
  background: #d55118 url(../images=headers/about.jpg); 
}
```

The background property that we just used is actually a shorthand property which allows us to specify the background colour, image, image position, and how the image repeats, all in one property. We’ve set the background to the same shade of orange as the header uses just so that the page doesn’t look too bland if the user doesn’t have images enabled in their browser. Paths to images in your CSS file are relative to the CSS file, not the HTML page. This is why the ../ is required in the path above.

Now replace the “Enlighten Designs” text with the logo image. Again, remember to keep the `<h1>` and put a descriptive alt attribute.
Now we have to position it on the right where it should be. We'll do this by floating the <h1> to the right, and then using "margin-top" and "padding-right" properties to get the position exact. Add the following to the <h1> rule in your stylesheet:

```css
h1 {
  margin: 0;
  padding: 0;
  float: right;
  margin-top: 57px;
  padding-right: 31px;
}
```

The reason we used padding-right instead of margin-right is because margins can often trigger weird bugs in IE if used in certain places.

And that's the header done.
10. Footer

First we need to make the text look right. The design shows very light grey (#e9e9e9) Tahoma 10px text. I’m sure you can figure this one out yourselves, but for the sake of copy/paste:

```html
#footer {
  clear: both;
  height: 66px;
  font-family: Tahoma, Arial, Helvetica, sans-serif;
  font-size: 10px;
  color: #e9e9e9;
}
```

And to change the link colour (and remove the link underline) we add this:

```html
#footer a {
  color: #e9e9e9;
  text-decoration: none;
}
```

But the links need some way to stand out when you mouse over them, so we’ll make them turn orange on hover:

```html
#footer a:hover {
  color: #d6d616;
}
```

We’ve also got to add a 1 pixel top border on the footer div, set some padding, and make the line-height 18px (increasing the leading). Because we are setting padding on the footer div, we will remove the height property to stop the padding/width/height bug I mentioned earlier. We don’t really need height on this div anyway:

```html
#footer {
  clear: both;
  font-family: Tahoma, Arial, Helvetica, sans-serif;
  font-size: 10px;
  color: #e9e9e9;
  border-top: 1px solid #eefef2;
  padding: 13px 25px;
  line-height: 18px;
}
```

The last thing left to do is float the alternate navigation to the right. Note that floated elements must have a width specified to work properly, so set the width to slightly larger than the nav actually needs, and set the text alignment to right so the text sits where it should:

```html
#footer #altnav {
  width: 350px;
  float: right;
  text-align: right;
}
```

Tada! We have a footer.
11. The Navigation (arg!)

There's a lot of funky CSS in this chapter, it's not imperative that you understand exactly what each bit of CSS does, just that you are able to modify this CSS to do what you want for other websites, which is basically changing heights, widths and images. However I'll do my best to explain the code.

Let's start out easy. Remove the red background in the CSS, and show the navigation by removing the "hidden" class on the definition list.

The method of image rollovers we are going to use for this menu is a 100% CSS solution. The basic premise behind it is to position the items in a definition list next to each other (side by side), hide the text on them, and use CSS to change the background image depending on what state the button is in (rollover, normal, or selected).

For each of the 4 nav items we need to create an image like the one above. The first third of the image is the normal state, the second is the mouseover state, and the third is the selected state. The animated gif below shows how this will work:
Hopefully you have some kind of understanding of how this works from the diagram above. Let's make our 4 nav images.

Save them in images/nav/
Now I'm going to do my best to explain the CSS behind this block by block, bare with me.

Replace your #main-nav rule with the code below:

```css
/* Main Navigation */
#main-nav { height: 50px; }
#main-nav dl { margin: 0; padding: 0; }
```

This sets the main-nav div height to 50px and strips all margins from the datalist.

```css/* IE8 Mac Hack */
#main-nav { padding-left: 11px; }
/* */
#main-nav { padding-left: 11px; overflow: hidden; }
/* End Hack */
```

This is a hack that does 2 things, sets the left padding of the main-nav to 11px (so it's bumped in slightly like the design shows), and fixes a bug on IE8/mac.

```css
#main-nav dt { float: left; }
```

This sets the definition titles (our individual nav item containers) to float left, which stacks them left to right, instead of one under the other.

```css
#main-nav dt a { display: block; height: 0px !important; height:/**/30px; /* IE 5/Win hack */ padding: 50px 0 0 0; overflow: hidden; background-repeat: no-repeat; }
```

Sets the link to the same dimensions as its surrounding container, and hides the text using the overflow property.

```css
#main-nav dt a:hover { background-position: 0 -50px; }
```

Sets the background position to move up 50px when a link is hovered.
Sets the individual widths of each nav item, and the paths to each image.

Now if all your images are named as they are above, and are saved in the correct place, your navigation should work.

Last thing we need to do to make the navigation work, is to get the selected button states to show up when you are on the corresponding page.

We need to add some new css, and modify some existing css to achieve this. Add this CSS below the rest of your navigation CSS:

```css
body, about dt#about, 
body, about dt#about a, 
body, services dt#services, 
body, services dt#services a, 
body, portfolio dt#portfolio, 
body, portfolio dt#portfolio a, 
body, contact dt#contact, 
body, contact dt#contact a {
  background-position: 0 -100px;
}
```

This large CSS selector checks to see what class the body tag has, and then sets the background position of the correct navbar. So if you wanted the about navbar to be selected, you would set a class="about" on the body tag. Let's do that now:

Now the problem we have, is that we also want the header image to change based on what section we are viewing. So we need to modify the #header rule like so:

```css
body, about #header {
  height: 150px;
  background: #b6b616;
  url(../images/headers/about.jpg);
}
```

Now when you create pages for your other sections, you'd just change the class on the body from about to say, contact, set up a css rule pointing to the correct header image, and you're done.
12. Getting it right in IE

Let's start the hacks at the current problem child, IE5.

Load up your IE5 browser. There's 2 things that I notice instantly that are wrong. The first is that the page isn't centered in the browser like it should be, and the second is that the footer has a weird alignment issue.

The alignment issue is well known, so we'll tackle that one first.

IE5 and 5.5 do not recognise the margin: auto;css property like they should. To get around this we need to use the text-align: center; property on the body which will center the container div.

```html
body {
  font-family: Arial, Helvetica, Verdana, Sans-serif;
  font-size: 12px;
  color: #000000;
  text-align: center;
}
```

This will center your container div, but will also center all the text inside that div. We don't want that, so we need to override the text-alignment inside the container div.

```html
#page-container {
  width: 768px;
  margin: auto;
  text-align: left;
}
```

That solves the centering issue. Now the weird footer.

I couldn't actually find any references to this particular bug with a brief google search, so I just set out to figure out how to fix it. I guessed that the bug had something to do with the floated "#nav" div, but couldn't work out what the problem was exactly. So eventually I tried putting a div around the copyright into, and that made the bug disappear.

```html
<div id="copyright">
  Copyright & Enlighten Design<br />
</div>
```

This triggerd some padding-top issues on the footer, so I removed the padding-top: 13px; attribute from the #footer rule, and added it to both the #copyright div and the #nav div.
There's one last IE bug that I can see, and that is that when you mouseover the selected nav item, it reverts to the white background mouseover as if it wasn't selected. We don't want the selected item to change on mouseover.

If we add a few hover rules to our big rule that sets the nav selection, that will fix our problem.

And there you have it. Hopefully you've learned something about CSS layouts. My recommendation is to have a look at some major CSS sites, like the ones listed below, and have a look through their HTML/CSS to see how they've done things. If in doubt about something, Google it.

Feel free to suggest changes or improvements to this tutorial, or ask for tutorials on other topics (just not dropdown navigation menus yet).

**Final Example**

After half a million requests for example HTML/CSS files, I've put some up:


---

**About**

Enlighten Designs is an internet solutions provider that specialises in front and back end development. To view some of the web sites we have created view our portfolio.

We are currently undergoing a face lift, so if you have any questions or would like more information about the services we provide please feel free to contact us.

**Contact Us**

Phone: (07) 853 6086
Fax: (07) 853 6060
Email: [info@enlightenden.co.nz](mailto:info@enlightenden.co.nz)
P.O Box 14159, Hamilton, New Zealand

[More contact information...](#)