

# **XLsuite**

**Template Design Tutorial**

## Creating a template

When I log in to my new XLSuite site, there is a default template set up for me. This is great, and we could easily customize that template but what if we want to build a template from scratch?

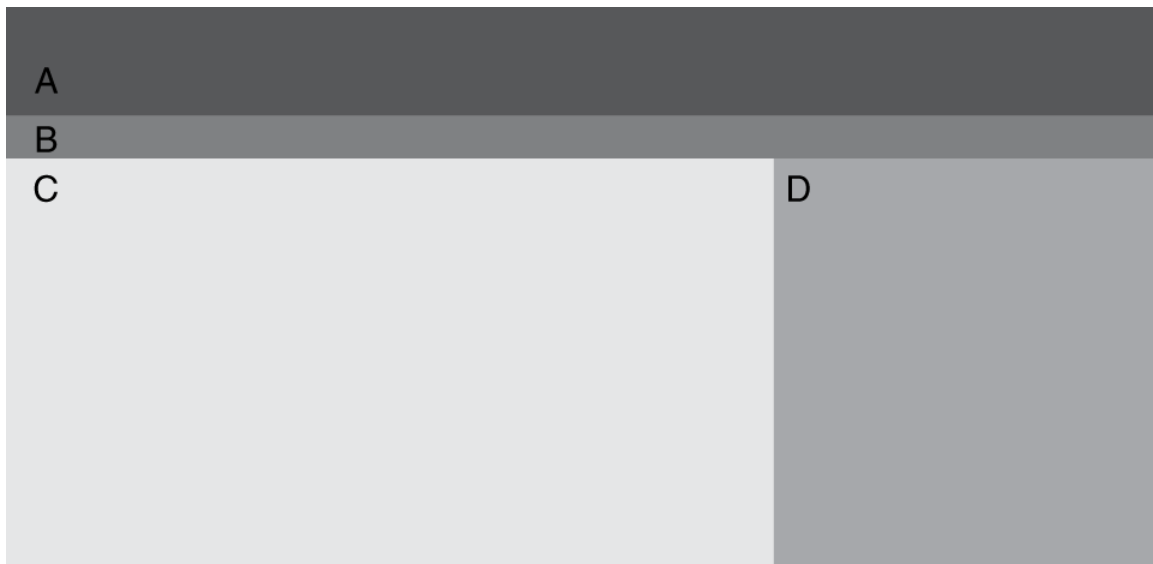
In this tutorial I will show you how to create your own template which will become the base for a hypothetical website; the kind of website you might want to set up for a client, or for yourself.

It's highly recommended that before going through this tutorial you are familiar with the basic functions of the XLSuite CMS (refer to **XLSuite – A Designer's Guide** for more information).

## Planning – Basic Layout

The first thing to do is make a basic sketch out your default layout. You could do this with paper and pen, or using a graphical layout program like Adobe Illustrator or Photoshop.

This is the layout I've decided to start with:



This is a pretty standard web layout. I have space for:

Header

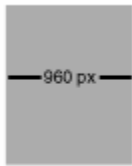
Navigation

Main Content

Sidebar

Basic, but it's a good starting point to build on.

## Planning – Dimensions



For this example I am going to use a fairly rigid grid to keep things in place. My overall page size will be 960px wide and will stretch vertically to fit the content of the page. I can then set this page neatly in the center of the browser window.

I'm also going to use a baseline of 18px to keep a steady vertical rhythm with my type, and this means I will use multiples of 18 when establishing the height of various elements on the page.

My 960px grid can be divided up into 12 separate columns, 68px each. Since I would like to keep lots of space between elements I will consistently use 6px of padding on either side of content blocks

This simple grid is available as **grid.css** in my example template, so feel free to use it for your own designs.

So, using my grid and baseline measurements I've planned:

Header – 960px wide by 90px high (remember I'm using multiples of 18!)

Navigation – 960px wide by 36px high

Main Content – 640px wide (8 columns)

Sidebar – 320px wide (4 columns)

Creating a basic layout

I want to create a brand new layout page which will become the base for my template.

In XLSuite navigate to Resources > CMS > Layouts.

From the Layout index page, select "New Layout"

I'll call this layout "MyLayout" and enter that into the TITLE field. In the CONTENT TYPE field I see that text/html is already entered and will leave that alone since I am building an HTML page. I will also leave the ENCODING set to UTF-8:

re | Close

Title:	MyTemplate
Content Type:	text/html
Encoding:	UTF-8
Body:	

Next I will enter my HTML code into the BODY field:

Declare the DOCTYPE:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
```

I have chosen XHTML 1.1 as my language, but you are free to use any flavor you like

Set up the HEAD of the document:

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>{{page.title}}</title>
</head>
```

Now, notice that for in the TITLE, I've used a Liquid Tag. This will automatically add the correct page title to any page using this layout. The page title will come from the Pages layout itself, which we will look at shortly.

Build the BODY:

```
<body>
  <div id="wrap">
    <div id="header">
      Header is here
    </div>
    <div id="navigation">
      Navigation is here
    </div>
    <div id="content">
      Content goes here:
      {{page.body}}
    </div>
    <div id="sidebar">
      Sidebar is here
    </div>
  </div>
</body>
```

Note two things here:

I'm using some placeholder text ("header is here") just to make sure everything is where it should be.

I'm using another Liquid Tag – this one tells XLSuite where to render the page content. This tag must be used or else any pages you build will fail to load.

Finally I close the page with an `</html>` tag – I’m not going to worry about Domain Patterns and Groups (for now) - and I will **SAVE** (Click on SAVE in the top left, or bottom left corner) this page for the time being. Here is all of the code I’ve just entered.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>{{page.title}}</title>
</head>

<body>
  <div id="header">
    Header is here
  </div>
  <div id="navigation">
    Navigation is here
  </div>
  <div id="content-wrap">
    <div id="content">
      Content goes here:
      {{page.body}}
    </div>
    <div id="sidebar">
      Sidebar is here
    </div>
  </div>
</body>

</html>
```

## Preparing XLsuite’s default styles

I’m going to leave the layout alone for a bit and get some default styles into place. Most of the stylesheets described below are available, by default, in XLsuite, so feel free to use and modify them for your designs.

These stylesheets are found in **RESOURCES > CMS > Pages**

Title	Status	Behaviour	Slug	Actions
Home	Published	Plain Text		<a href="#">Add child</a>
Assets	Published	Plain Text	assets	<a href="#">Add child</a>
Contact us	Published	Plain Text	contact	<a href="#">Add child</a>
CSS	Draft	Plain Text	CSS/color.css	<a href="#">Add child</a>
cart.css	Published	Plain Text	CSS/color.css/cart.css	<a href="#">Add child</a>
color.css	Published	Plain Text	CSS/color.css/color.css	<a href="#">Add child</a>
forms.css	Published	Plain Text	CSS/color.css/forms.css	<a href="#">Add child</a>
forums.css	Published	Plain Text	CSS/color.css/forums.css	<a href="#">Add child</a>
grid.css	Published	Plain Text	CSS/color.css/grid.css	<a href="#">Add child</a>
mycart.css	Published	Plain Text	CSS/color.css/mycart.css	<a href="#">Add child</a>
products.css	Published	Plain Text	CSS/color.css/products.css	<a href="#">Add child</a>
profiles.css	Published	Plain Text	CSS/color.css/profiles.css	<a href="#">Add child</a>
style.css	Published	Plain Text	CSS/color.css/style.css	<a href="#">Add child</a>
type.css	Published	Plain Text	CSS/color.css/type.css	<a href="#">Add child</a>

The two stylesheets I will definitely want to use are a RESET sheet (**reset.css**) and my GRID (**grid.css**).

**Reset.css** strips any default CSS rules that are enforced by the browser, and generally gives us a “level playing field” upon which to build our own CSS.

**Grid.css** facilitates quick layout by using predetermined measurements for “blocks” and “cells” of content. I’ll use the default width of 960px (as planned) using a class called “block”:

```
.block {  
margin: 0 auto;  
position: relative;  
width: 960px;*  
}
```

\*You can change this width if you’d like but keep in mind that you will also need to re-calculate the column widths found at the end of the sheet.

Any element with that class will be constrained to 960px, which will be useful for stacking layers of content, which I will demonstrate when we start building pages.

Next I would like to have some CSS rules in place for TYPOGRAPHY (type.css) and for COLOR (color.css) (though I will almost certainly be customizing these two sheets later on. For now the most important thing is to make sure that the default paragraph elements fall in line with the 18px baseline I planned:

```
html>body {font-size: 12px;}  
  
p {  
font-size: 1em;  
line-height: 1.5em;  
margin-bottom: 1.5em;  
margin-top: 1.5em;  
}
```

Yep – it does (where 1em = 12px, 1.5em = 18px).

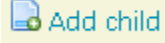
I’m going to add some rules to the color.css to make our layout a little clearer as we build our template:

```
/* Some background colors to help visualize page layout */  
  
#header {background: #666}  
#navigation {background: #999}  
#content {background: #eee}  
#sidebar {background: #9f9f9f}
```

## Creating a new stylesheet

First, I'll create a new stylesheet which will be the base CSS for the *MyTemplate* layout we just created.


Go to RESOURCES > CMS > Pages

Find the page named CSS and click "Add child" from the  options on the right side of the table.


I'll call this page "*MyStyles*" (in the TITLE field), and then I'm going to add the following code, which instructs this sheet to import rules from the default styles that we looked at previously. This goes in the BODY field:


```
/*import default XLSuite styles*/

@import url(color.css);
@import url(grid.css);
@import url(reset.css);
@import url(type.css);
```

**Behavior:** Plain Text 

**Fullslug:** CSS/MyStyles.css

**Layout:** stylesheet 

**Status:** Published 

**Require Ssl**

Now I'll fill in some technical information for the page. In the FULLSLUG field I want to make sure that I've got the path correct; in this case it needs to be CSS/MyStyles.css. For LAYOUT, I select "stylesheet" (for obvious reasons), and STATUS needs to be set to "Publish", to make this CSS public. I'll leave the rest of the fields as they

are, for now, and **SAVE** the page.

## Linking the layout to the stylesheet

Now I return to the layout I saved as "*MyTemplate*" (RESOURCES > CMS > Layouts); luckily the tab is still open so I can flip back to it quickly.

In order to link the base CSS page I just created I place the following code in the <HEAD> of the document:

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>{{page.title}}</title>

  <link href="/CSS/MyStyles.css" rel="stylesheet" type="text/css"
  ↪ media="screen" />

</head>
```

This is more or less the standard way we link stylesheets to HTML pages – note that in this case the HREF has a leading slash.

I'm done here for now, so I **SAVE** this layout.

Adding classes to the layout

In *MyTemplate*, I'm going to add some CSS classes in order to lay out the various divisions of the page.

To start with I will assign to the *header*, *navigation*, and the *content-wrap* DIVs the **block** class from **grid.css**:

```
.block {  
margin: 0 auto;  
position: relative;  
width: 960px;*  
}
```

Now I've got the following rules set for those elements:

- **Margins** are set at 0 and auto, centering the DIVs in the browser window
- The **position** property is set to **relative**, which comes in handy if I need to position another element absolutely within each DIV
- The DIVs are set at a fixed **width** of 960px

Here is the code:

```
<div id="header" class="block">  
Header is here  
</div>  
<div id="navigation" class="block">  
Navigation is here  
</div>  
<div id="content-wrap" class="block">  
<div id="content">  
Content goes here:  
{{page.body}}  
</div>  
<div id="sidebar">  
Sidebar is here  
</div>  
</div>
```

Next I will set the widths for the columnar sections using some predefined styles:

```
<div id="content" class="cell span8">  
Content goes here:  
{{page.body}}  
</div>  
<div id="sidebar" class="cell span4">  
Sidebar is here  
</div>
```

The **cell** class sets the following CSS rules, giving me some space between the columns:

```
.cell {  
float: left;  
margin-left: 6px;  
margin-right: 6px;  
}
```

The **span8** and **span4** rules set the exact width of the cells:

```
.span4 {width: 308px;}  
.span8 {width: 628px;}
```

These column widths, along with the margins in the cell class will give me a total width of 960px.

## Create test page for the main content:

Go to RESOURCES > CMS > Pages

Find the page named HOME and click “Add child” from the



I’ll give this a TITLE of “MyTemplate – Home” and add some dummy content to the BODY:

```
<h1>Lorem Ipsum</h1>
```

```
<h1>Dolor Est</h2>
```

```
<h3>Sicut Valor</h3>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum in odio. Fusce suscipit. Phasellus quis mauris et eros aliquet bibendum. Vestibulum vitae est consequat velit adipiscing condimentum. (etc., etc.)</p>
```

```
<h3>Vitae est consequat</h3>
```

```
<p>Mauris venenatis augue et nunc. Proin tristique pharetra arcu. Nulla facilisi. Cras rhoncus. In sed tellus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Vivamus at metus. Morbi dignissim libero nec nunc. Curabitur luctus lectus sed eros. Vivamus lacus dui, ultricies eget, mollis et, dapibus ut, eros.</p>
```

I need to indicate the correct FULLSLUG in this case *MyTemplate-Home*, and now all I have to do is apply my new template to this page by ensuring that the LAYOUT selection is set to “*MyTemplate*”

Behavior:	Plain Text	▼
Fullslug:	MyTemplate-Home	
Layout:	MyTemplate	▼
Status:	Published	▼

I click on **SAVE** and now have some text for our “content” section.

## Create navigation

I will now create some navigation links to test out in our new template. I'll use a **snippet** for the navigation, which makes it easier for me to edit the links further on down the road.

Navigate to RESOURCES > CMS > Snippets

Click on "New Snippet"

I will give this snippet a TITLE of "MyTemplate\_navigation" and enter the following HTML into the BODY field:

```
<ul class="inline">
  <li><a href="/">Home</a></li>
  <li><a href="#">Page One</a></li>
  <li><a href="#">Page Two</a></li>
  <li><a href="#">Page Three</a></li>
</ul>
```

That's all I need to do for now. Notice that because this is going to be horizontal navigation I've given the UL element a class called **inline**, which links to a handy little style rule in **type.css**:

**\*\*\*Lee4Riel\*\*\* re: Inline – do we have a “handy little style rule” for a vertical menu?**

Now I can **SAVE** this snippet.

## Create some sidebar content

Now I am going to place some dummy text into the sidebar – I will be adding something more substantial later on, but this will give me some idea of what the template will look like.

Navigate to RESOURCES > CMS > Snippets

Click on "New Snippet"

I'm going to call this snippet "MyTemplate\_sidebarContent" and will enter some dummy text as a couple of paragraphs:

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec at
↳ felis ut leo accumsan porta. Proin scelerisque velit id ipsum.</p>
<p>Sed augue libero, nonummy ac, faucibus vel, accumsan in, velit. Aenean
↳ posuere, turpis quis pellentesque fringilla, dolor diam venenatis
↳ lorem.</p>
```

There, now I have some content in the sidebar and I can **SAVE** this snippet.

**\*\*\*Lee4Riel\*\*\* make it the links to these documents.**

Link the snippets to the template

Going back to my **layout** "MyTemplate" I tell XLSuite to render the snippets I just created.

First the navigation, replacing the placeholder text with a liquid tag:

```
<div id="navigation">
  {% render_snippet title="MyTemplate_navigation" %}
</div>
```

Next, the sidebar:

```
<div id="sidebar" class="cell span4">
  {% render_snippet title="MyTemplate_sidebarContent" %}
</div>
```

Then I'll **SAVE** the layout and add a few more elements before testing the page to see how it looks.

## Add some content to the header and add a footer to the layout.

I add a logo to the header:

```
<div id="header" class="block">
  <h1>MyTemplate</h1>
</div>
```

I add a footer to the layout, *underneath* the “content-wrap” element giving it the **block** class so that it spans the entire width of the layout and a clear class to force it below the rest of the content.

My code for this layout, with my new footer, now looks like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>{{page.title}}</title>
  <link href="/CSS/MyStyles.css" rel="stylesheet" type="text/css"
media="screen" />
</head>

<body>
  <div id="header" class="block">
    <h1>MyTemplate</h1>
  </div>
  <div id="navigation" class="block">
    {% render_snippet title="MyTemplate_navigation" %}
  </div>
  <div id="content-wrap" class="block">
    <div id="content" class="cell span8">
      Content goes here:
      {{page.body}}
    </div>
    <div id="sidebar" class="cell span4">
      {% render_snippet title="MyTemplate_sidebarContent" %}
    </div>
    <div id="footer" class="clear block">
      My footer content goes here
    </div>
  </body>
</html>
```

Look once at again some of the Liquid Tags I've used in this layout, and what happens when I attach this layout to any pages I create:

<code>{{page.title}}</code>	Picks up whatever content is in the TITLE field of a page
<code>{{page.body}}</code>	Picks up the content from the BODY field of a page
<code>{% render_snippet title="MyTemplate_navigation" %}</code>	Looks for, and renders the snippet entitled "MyTemplate_navigation"

Now I can **SAVE** this layout and I'm ready to test out my template:

It's not ready to win any design awards just yet, but I can see that it's fitting in to my original sketch and there are some features that will come in handy when I publish this sucker: the navigation and sidebar can be adjusted independently of the template itself, and the page content can be kept separate from the layout.

I want to make some tweaks to this though, so when you're ready, let's move on to Part II of the tutorial

**MyTemplate**

[Home](#) [Page One](#) [Page Two](#) [Page Three](#)

Content goes here:

## Lorem Ipsum

### Dolor Est

#### Sicut Valor

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum in odio. Fusce suscipit. Phasellus quis mauris et eros aliquet bibendum. Vestibulum vitae est consequat velit adipiscing condimentum. Integer neque massa, pharetra volutpat, interdum quis, ultrices quis, nulla. Vivamus at tellus imperdiet quam mollis commodo. Duis accumsan nisi non lorem. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Duis fringilla interdum lectus. Fusce augue sapien, fermentum eu, elementum ac, commodo id, ante. Integer fringilla pede. Sed scelerisque urna nec lacus. Suspendisse varius, urna eu vulputate pharetra, lectus urna ultrices lectus, id volutpat eros lacus placerat ligula. Phasellus venenatis purus ac risus. Aliquam semper blandit magna. Pellentesque vehicula fermentum pede. Suspendisse ornare, quam vitae sollicitudin feugiat, metus mi lacinia enim, quis facilisis tellus diam ut ligula. Nulla varius ornare arcu. Sed rhoncus, nisi a laoreet laoreet, velit mauris mattis metus, id commodo diam nulla id purus.

#### Vitae est consequat

Mauris venenatis augue et nunc. Proin tristique pharetra arcu. Nulla facilisi. Cras rhoncus. In sed tellus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Vivamus at metus. Morbi dignissim libero nec nunc. Curabitur luctus lectus sed eros. Vivamus lacus du, ultrices eget, mollis et, dapibus ut, eros.

My footer content goes here

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec at felis ut leo accumsan porta. Proin scelerisque velit id ipsum.

Sed augue libero, nonummy ac, faucibus vel, accumsan in, velit. Aenean posuere, turpis quis pellentesque fringilla, dolor diam venenatis lorem.

## Adding CSS



I can clearly see that some areas need improvement here. The grid.css and type.css are really just defaults to help me get started so let's see what can be done to improve the situation.

## Adding background images to block-level elements

Let's start with the header. I want to see this as the background:



That's an image that's 960px by 90px, which will fit nicely and so I'm going to upload it into XLSuite by going to RESOURCES > Content > Files. From there I click on "Add a file", browse to the image I want to upload, and fill in any relevant data:

or

File:

Title:

Tag List:

Description:

### Template Header

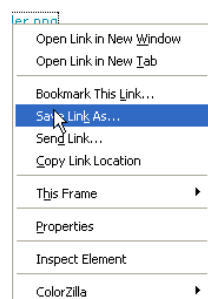


Tags [gif](#)

Filename [sample\\_header.png](#) 

Size 73.3 KB

Now I click "Upload" (to upload the file). Once the file has been uploaded and I am viewing it, I **RIGHT-Click** (PC) on the link next to "Filename" and select "**Copy Link location**" to save the URL of the image onto the clipboard.



## Referencing saved images

Now I navigate to the CSS file I'm using for this exercise, MyStyles.css, (RESOURCES > CMS > Pages – or click on the “Pages | Index” tab if it's already open) and create a rule for the header element that uses the URL I just copied in the previous section:

```
#header {  
  background:  
    ↪url(http://james.template.xlsuite.com/admin/assets/43435/download)  
    ↪no-repeat 0 0; height: 90px;}
```

Notice also that I have added a definite height (90px;) to the header element; because I gave the element a class of **block** in the previous chapter, it will already display as a block-level element, and have a defined width.

Adding background images to text

I'm going to upload an image that I can swap for the H1 element in the header, but first we need to add a class to that element in the layout.

I'll navigate to the MyTemplate layout, if it's not already open in a tab, and add the class **logo** and the class **logo\_imgRep** (for *Image Replacement*) to the H1 element, and then **SAVE** the template:

```
<div id="header" class="block">  
  <h1 class="logo_imgRep"><a href="/">MyTemplate</a></h1>  
</div>
```

I've created this in Adobe Photoshop:



I've uploaded it into the XLSuite file manager, and copied the image location to my clipboard by right-clicking on the name. (PC).

I'm going to use this as a CSS-based rollover image using the **:hover** pseudo-class that's available for the `<a>` element (which is nested within the H1 element in my layout).

In the stylesheet **type.css** there is a rule governing image replacement for situations like this:

```
h1.logo_imgRep a {  
  display: block;  
  overflow: hidden;  
  text-indent: -9000px;  
}
```

That sets up the general attributes I need but I'll need to define the width and height of my <a> element and refer to the image location. I can do this by putting the following code in the **MyStyles.css** file:

```
#header h1.logo_imgRep a {
  background:
    ↪url(http://james.template.xlsuite.com/admin/assets/43440/download)
    ↪no-repeat 0 0;
  height: 90px;
  width: 316px;
}
```

I also set up the **:hover** rule in the same file, moving the background image up by exactly the height of the container to achieve the rollover effect:

```
#header h1.logo_imgRep a:hover {
  background:
    ↪url(http://james.template.xlsuite.com/admin/assets/43440/download)
    ↪no-repeat 0 -90px;
}
```

There...now I've got a nice graphic logo with a hover effect when the cursor is over top of it. My template's looking better already:



## UPDATE: Adding Images En-Masse

The XLSuite **Assets Manager** has a handy feature that allows a user to upload multiple image files all at once, in a ZIP file (.zip) that will be automatically unpacked on the server and assigned to a folder.

This is particularly useful for designers who build sites using a packaged application, such as Adobe Creative Suite.

### File:

C:/MySite\_images.zip

Unzip after uploading

Files unpacked by XLSuite will be moved into the “/z” directory. If, for example, you are building a site locally – using Adobe Dreamweaver – you would place all your images (including CSS background images) into a folder named “z” and then reference that folder and the images within from your HTML using a site-relative URL:

```

```

Once you are ready to deploy your site package all of your images into a ZIP file (.zip) and upload them to XLSuite with the “Unzip after uploading” option ticked. Your images will now be stored in the “z” directory of the site.

## Adding CSS rules

Now I can start adding CSS rules as I please to the keep building on the *MyStyles.css* file (which is linked from the *MyTemplate* layout), and altering the default stylesheets as necessary.

I'll start by improving the look of the navigation and adding some rollover effects.

First I'll add some space to the <ul> element and to the links:

```
#navigation ul {
  margin-left: 10px;
}

#navigation ul li a{
  display: block;
  padding: 9px;
}
```

Then change the colors (in **color.css**):

```
#navigation {
  background: #77a35a;
  border-bottom: 6px solid #325C8A;
}

#navigation a {
  color: #fff;
}

#navigation a:hover {
  background: #fff;
  color: #2B4F78;
}
```

That's looking much better:



## More improvements

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec at felis ut leo accumsan porta. Proin scelerisque velit id ipsum.

Sed augue libero, nonummy ac, faucibus vel, accumsan in, velit. Aenean posuere, turpis quis pellentesque fringilla, dolor diam venenatis lorem.

I like the layout of the **sidebar** and the margins are effective here, but the background color isn't going to work out very well. The margins push the background inwards and the color doesn't reach all the way to the bottom, as the text in the **content** division takes up more vertical space than the text in the sidebar.

What I do here is make the background colors within the **content-wrap** element transparent and add a 960px background image, tiled

vertically which will sit behind both divisions (see: <http://www.alistapart.com/articles/fauxcolumns/>)

Here is my background image, which I'll upload and copy the location:

And the CSS to make it happen (in the **color.css** file):

```
#content-wrap {
  background:
  url(http://james.template.xlsuite.com/admin/assets/43445/download)
  repeat-y 0 0;
}

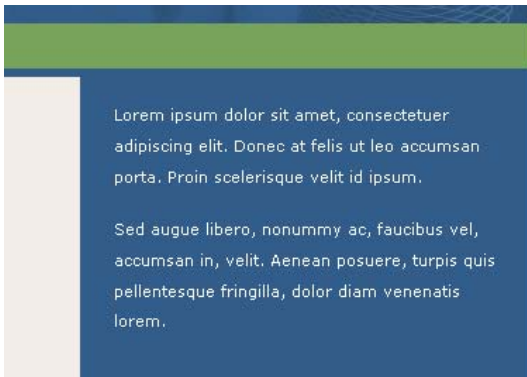
#content-wrap > div {
  background: transparent;
}

#content-wrap #sidebar {
  color: #fff;
}
```

I'll also adjust the size of the font, and the line-height, and add some padding to the sidebar text – overriding the default body style. I could do this in the **MyStyles.css** file, but, to keep things organized, I'll add this rule to the **type.css** file instead:

```
/* sidebar text */

#sidebar p {
  font-size: .9em;
  line-height: 2em;
  padding: 0 1.5em;
}
```



I've thrown the baseline off here, but you should get the basic idea...

I'll stick in a fairly basic footer to square things up. I've created a snippet and added a liquid tag into the layout for content:

```
{% render_snippet
title="MyTemplate_footerContent" %}
```

Some CSS to give it shape:

```
#footer {
  height: 45px;
  margin-top: 36px;
}
```

## Adding CSS to server-produced code

XLsuite ships with many useful modules that can add a whole range of dynamism to a site. Almost always these modules produce code from the server side which is then churned out as HTML.

In order to attach CSS to this code you should always view the source code that a module produces. In most cases the HTML is very basic and easy to assign CSS to, or there are “handles”, i.e. Classes or IDs within the HTML that allow a designer to add style.

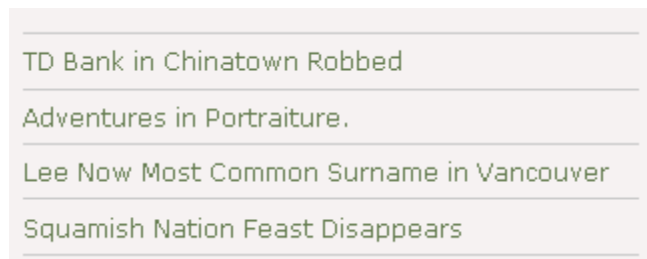
In this example I've rendered a feed using XLsuite's Feed Manager (see XLsuite: A Designer's Manual for details on the various aspects of the XLsuite CMS). Viewing the source code shows me this pattern:

```
<ul class='feedItems'>
  <li>
    <a href='http://feed1.com/article01'>Title 01 </a>
  </li>
  <li>
    <a href='http://feed2.com/article02'>Title 02 </a>
  </li>
  <li>
    <a href='http://feed3.com/article03'>Title 03 </a>
  </li>
</ul>
```

Notice that each title and link is displayed in an unordered list which has been assigned the class “feedItems”. To style this I might add the following to one of my CSS files:

```
ul.feedItems {
border-top:1px solid #CCCCCC;
}
ul.feedItems li {
border-bottom: 1px solid #CCC;
}
ul.feedItems li a {
display:block;
padding:0.6em 0pt;
}
```

Which would style the HTML to something similar to this:



Generally XLsuite modules produce clean, usable HTML with lots of handles that allow for quick and easy CSS to be applied.

## **Adding additional content using the XLSuite CMS**

Once you have a basic layout/page/snippet system set up you are ready to add pages and start publishing content on your site.

An understanding of how XLSuite applies CSS to pages and layouts, as well as a familiarity with linking images using XLSuite's file manager, will give the designer enough knowledge to style some of the myriad modules that ship with XLSuite such as:

Forums

Contact and Profile forms

RSS Feeds

Product Catalogue pages

...and more.

Many of the specific how-to's and parameters of the XLSuite CMS are covered in the Designers' Guide to XLSuite, available at <http://template.xlsuite.com> and on the XLSuite community wiki site located at <http://wiki.xlsuite.org/>.

Good luck and have fun managing your content with XLSuite!