HTML / CSS – Lab One – Getting Started

Objectives

1. Ensure that webspace on *public_html* is working and understand how to publish to *homepages.shu.ac.uk*
2. Review editing options
3. Basic HTML structural, block and inline elements
4. Introduce using CSS for styling

Setting up your webspace

Our first simple pages with HTML and CSS do not require to be published to a webserver and can be tested locally. However, it makes sense to start off by checking that your web space at SHU is set up correctly and working as you would expect. You will need this to be in place for your assignments that make use of PHP.

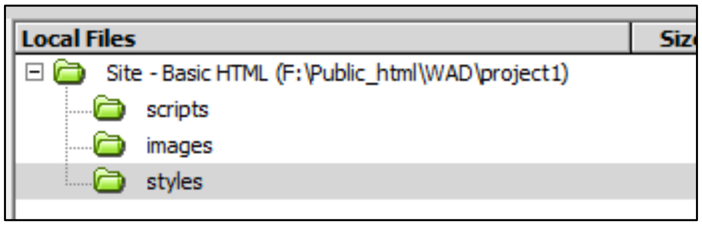
You already have a directory that is mapped to a Webserver so you can access your files. It appears as drive f:\public_html.

As you could be using this for other modules it would help if you created a directory called WAD under f:\public_html.

Structure your files

Good web developers also take time to structure their sites sensibly. It would be useful to create a folder for each week of the module as we will be creating content in all of the labs. Starting with a good naming convention can also help. Generally avoiding mix cases names and working in lowercase is tidy. Also avoid the use of special characters in file and folder names. In particular don't use white space in file or folder names as this will be translated by a browser into *%20*.

Most of the projects we build will include images, style sheets and scripts. It is a good practice to create folders in your site for this kind of content for example an images for all your JPGs, PNGs and GIFs, a scripts folders for your Javascript and a styles folder for your CSS.



Viewing your files

To view pages you create in *f:\public_html* through the University's webserver outside of the University you will need to register your web site with the University's server.

You'll find information on how to do this here:

```
https://students.shu.ac.uk/shuspacecontent/it/registering-your-student-
website-external-access
```

Once registered you can access your page from the Web using a URL like:

```
http://homepages.shu.ac.uk/~<yourid>/WAD/index.html
```

Notice the tilde (~) Put your logon ID after it.

Files on a Webserver can be read by anyone. If a user enters the URL of a directory on a Web site they may see a listing of all the files in that directory. This is unlikely to be what the developer intended. When the name of a directory is given to them Webservers default to searching for files called *index.html*, *index.htm* or, sometimes, *default.htm*.

The University Web server always looks for *index.html*, if it can't find that file it will provide a directory listing of the files. Therefore, whenever you create a new directory you should create an index which points to those files that you want users of your site to see. If you want to hide all of the files in the directory, create *index.html* but leave the body of the page empty.

HTML Editors – The Choice is Yours

To all the file types in this module all you need is a text editor. These can be sophisticated IDE (Integrated Development Environments) like Dreamweaver, freeware like Sublime Text or even basic text editors like notepad. It is recommended that you experiment with different editors to see which suit your workflow. Whichever tool you use it should help you not hinder you in your coding. Modern text editors will provide help with syntax and code checking so it is worth investing time to learn the tricks associated with your chosen tool.

Update Your Checklist File

On Blackboard you'll find a file called checklist.txt. Use this to store information about your *homepages.shu.ac.uk* account.

HTML

Create a boiler plate HTML file

A boiler plate file is a simple started page with common elements needed by all other pages. A really simple boiler plate HTML file would look as follows:

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
      <title>Untitled Document</title>
</head>
<body>
</body>
</html>
```

As we progress through the course you may wish to make your boiler plate file more complex by for example adding a link to a stylesheet and common JavaScript.

HTML Elements

Review the first 18 of the 20 HTML tags listed at *http://www.mustbebuilt.co.uk/html-basics-to-teach/*.

These provide the basic building blocks for HTML pages. There are other tags that we'll pick up along the way.

Key tags are `<div>` logical division, `<p>` paragraph, `<ul>` unordered list, `<li>` list item, `<a>` anchors for links, `<img>` for adding images.

Build some Pages

As a mini-project we are going to create a website for 'Web Application Development'. Download the zip file of resources.

- Create four pages saved as *index.html*, *structure.html*, *presentation.html* and *interaction.html*
- Add appropriate titles to each page.

```
<title>Web Application Development</title>
```

Structuring the page with HTML

Open the new page called *index.html* and add the following HTML to the `<body>` of the page:

```
<body>
<div id="container">
      <div id="header">

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

            </div>
            <div class="sidebar">

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

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

This gives the document a basic structure - notice how the `<div>` logical division elements can be nested inside each other.

Flesh out the sections:

In the header amend the contents by nesting another `<div>` inside it like so:

```
      <div id="logo">
                  <h1>Web Application Development</h1>
      </div>
```

This adds the heading one of which there should only be one. Underneath this add a list of links that will shortly become a horizontal menu bar.

```
<div id="logo">
            <h1><a href="#">Web Application Development</a></h1>
</div>
<div id="nav">
      <ul>
            <li><a href="index.html">Home</a></li>
            <li><a href="structure.html">HTML</a></li>
            <li><a href="presentation.html">CSS</a></li>
            <li><a href="interaction.html">Javascript</a></li>
            <li><a href="contact-us.html">Contact Us</a></li>
      </ul>
</div>
```

Inside `<div id="content">` we'll add some of the filler text and an image.

```
<h2>The 'big' three</h2>
<p><img src="images/ysp-art1.jpg" width="750" height="250" alt=""></p>
<p>Any budding web designer or web developers needs to grasp the concept
of the 'big' three.  These are the core open standard skills that will
enable to build beautiful looking, dynamic web pages as well as mobile
applications.</p>
<p>Web designers ….
```

Notice how `<p>` are used to space out the content.

Fleshing out the `<div class="sidebar">` next add:

```
<div class="sidebar">
      <div>
            <h2>Topics</h2>
            <ul>
                  <li><a href="#">HTML5</a></li>
                  <li><a href="#">CSS Positioning</a></li>
                  <li><a href="#">Cookies</a></li>
                  <li><a href="#">JSON</a></li>
            </ul>
      </div>
</div>
```

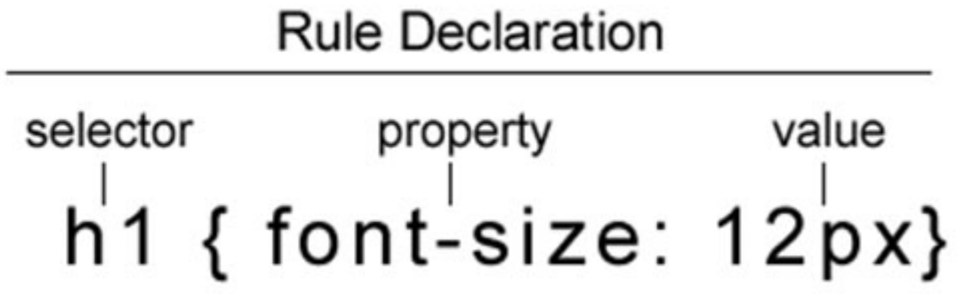
This represents a bulleted list with some dummy links to populate later.

Finally add a copyright message to the footer ie:

```
<div id="footer">
      <p>Copyright &copy; 2015</p>
</div>
```

CSS and HTML

CSS is based on rules. A rule declaration is broken down as follows:



The selector can be a HTML element such as <h1> above. This is known as a HTML selector. The curly brackets enclose a ‘property/value’ pair. More than one ‘property/value’ pair can be added. These are separated by semi-colons and usually for ease of editing placed on a new line for example:

```
h1 {
    font-size: 16px;
    color: #000099;
    font-family: Arial, Helvetica, sans-serif;
}
```

There a number of ways that selectors can be declared. The above is a HTML selector.

HTML Selectors

HTML Selectors use HTML tags to target their content. Any HTML element can be targeted. Commonly used HTML selectors include `<body>`, `<h1>`, `<p>`, `<ul>` but any can be used.

Class Selectors

With class selectors it is up to you to give the selector a name. This should be unique and should begin with a dot (.). An example of a class selector based rule is as follows:

```
.maintext {
       font-size: 12px;
       color: #000;
}
```

In the above example a class of name `maintext` is declared.

To apply the class selector use the class attribute that can be added to any HTML element that you want to assume the style. For example the following applies the rule `.maintext` to the first paragraph but not the second paragraph.

```
<p class="maintext">Paragraph One</p>
<p>Paragraph Two </p>
<p class="maintext">Paragraph Three</p>
```

This gives you more flexibility in page design as the ‘class’ attribute can be added to any tag.

ID Selectors

ID selectors are very similar to class selectors in that they can be applied to any element in a HTML file. ID selectors are flagged with a hash (#) and can be given any user-defined name as appropriate.

```
#footer {
    color: #0066CC;
    background-color: #CCCCCC;
    font-size: 10px;
    text-align: center;
}
```

To apply this ID to an element in a HTML document use the ID attribute that can be added to any tag. Add the value of the user-defined ID selector name as follows:

```
<div id="footer">Page Maintained by mustbebuilt.com</p>
```

The main difference between ID and class selectors is that ID selectors are only supposed to be used once per document. They can be used on any HTML element in the document but should only appear once. In the example above a rule is declared for a section of the document that will be identified as the footer of the file and styled accordingly.

The document will only have one footer. When we consider CSS positioning we will see how this ability to single out an element for styling lends itself to page layout with CSS.

We'll style our basic document with a combination of HTML, class and ID selectors.

Use the `<span>` tag to select a word in the your document for example:

```
<span style="color:#ff0000;">ipsum</span>
```

This is an inline style applying a red colour to the text inside the <span>.
Remove this inline style and replace it with a document level style. In the head of the document you will need to create a class selector.

```
<style>
.highlight{
      color:#FF6600;
}
</style>
```

You will need to remove

```
<span style="color:#FF6600;">ipsum</span>
```

... and replace it with

```
<span class="highlight">ipsum</span>
```

Using body as a HTML Selector

Add some html selectors to your stylesheet.

Start with a rule for the body tag. This will set defaults like font style, font color and background colour.

```
body{
      font-size:12px;
      font-family:Arial, Helvetica, sans-serif;
      color:#000099;
      background-color:#FFFFCC;
}
```

Create html selectors to target the documents headings.

```
h1{
      font-size: 18px;
      text-transform: uppercase;
      font-family:"Times New Roman";
      background-color:#cccccc;
}
```

A Container with an ID Selector

Create an ID selector to target the `<div>` element that is acting as the document container. Assign the ID selector to the `<div>` as follows:

```
<body>
<div id="container">
```

Your CSS may look as follows:

```
#container{
      width:1200px;
      margin: 50px auto;
      background: #FFFFFF;
}
```

These properties will set the container's width to 1200 pixels. By setting the margin to 'auto' this will tell the browser to allocate margin equally to both the left and right sides.

Pseudo Classes

Create specific rules using pseudo-classes to target the links in the bulleted list. You may approach this a number of different ways. For example by declaring an ID and assigning it to the `<ul>` that contains the list ie

```
<ul id="nav">
            <li><a href="index.html">Home</a></li>
            <li><a href="structure.html">HTML</a></li>
            <li><a href="presentation.html">CSS</a></li>
            <li><a href="interaction.html">Javascript</a></li>
            <li><a href="contact-us.html">Contact Us</a></li>
</ul>
```

The CSS could look as follows:

```
#nav a:link{
      text-decoration:none;
      color:#000000;
}
#nav a:visited{
      text-decoration:none;
      color:#CCCCCC;
}
#nav a:hover{
      text-decoration:underline;
      color:#66CC00;
}
#nav a:active{
      text-decoration:underline;
      color:#CC6600;
}
```

External Stylesheets

Upgrade your document level stylesheet to an external stylesheet. To do so copy the rules you have created into a new stylesheet file saved with a CSS extension. If you haven't done so already create a folder for this in your site structure called ‘styles'.

Remove the `<style>` tag and its enclosed code from your HTML page and replace it with a call to the external stylesheet. Your HTML should appears as follows:

```
<link href="styles/main.css" rel="stylesheet" type="text/css">
```

Your external stylesheet may look something like this:

```
body{
      margin: 0px;
      padding: 0px;
      background: #15384C;
      font-family:Helvetica, Arial, sans-serif;
      font-size: 15px;
      color: #545454;
}
h1, h2, h3{
      margin: 0;
      padding: 0;
      text-transform: uppercase;
      font-weight: 500;
}

h2{
      padding: 0px 0px 20px 0px;
      font-size: 1.50em;
}

p, ol, ul{
      margin-top: 0px;
}

p{
      line-height: 180%;
}

strong{
      color: #2C2C2C;
}
```

Try attaching this stylesheet to the other files in the sequence with the `<link>` tag. Notice that you will need to add ID and classes as attributes to see those rules been applied.