HTML5 & Javascript: Lab 6

Objectives of this lab session:

- Use CSS styles to enhance the appearance of a web app
- Use jQuery to simply coding and apply effects to an app

Resources required:

- A current web browser (ideally Google Chrome)
- WebStorm IDE
- jQuery code library: download from - http://jquery.com/download

Part 1: Applying CSS Styles

In this part of the lab, you'll add some CSS style-sheets to a simple web page and apply them to specific elements. Later, you'll use the jQuery code library to effects, and use it to enhance and simplify the Javascript code you write.

Basic Principles

A CSS style-sheet is a statement of style settings that should be applied to specific elements in a document. Elements are identified using the basic DOM selectors that you've used in labs – there are four basic types of selector:

- Individual elements, equicvalent to using document.getElementById(<element-id>)
- Elements with a specific tag name, same as document.getElementsByTagName(<tag-name>)
- Elements that have a specific CSS class, like document.getElementsByClassName(<class>)

Note that only the first of these will return an individual element, since its name starts with ***getElementBy…***. The other three selectors will return ALL of the elements in a document that match the criterion (e.g. document.getElementsByTagName('p') will return all <p> elements as an array).

A CSS rule will effect a selector and apply various types of formatting to specified elements:

All <H1> elements:	H1 {
	  background-color: darkred;
	}

An element with the id "data-table" (the # indicates an id)	#data-table {
	  border: 2px solid red;
	  font-family: serif;
	}

All elements that currently have the CSS class "mainheading" (the . indicates a CSS class)	.mainheading {
	  font-size: 200%;
	  font-style: italic;
	}

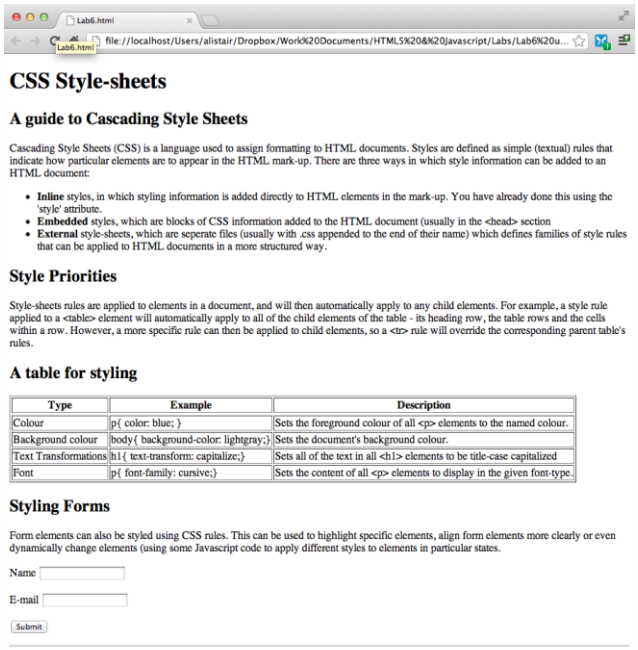
The range of formatting options is extensive – see http://www.w3schools.com/css/default.asp for a good tutorial on CSS rules, selectors and syntax.

A Basic document

1. Start a new project (Lab6)
2. To save having to write an entire HTML document to have something to work on, download

Lab6.html from the Moodle Week 8 folder. This is a simple HTML document with a few different types of element on it. Add the html file to your Lab6 project

3. Load Lab6.html to see how it appears in its raw state – black text on a white background



Adding CSS Rules

We will now add some style rules to this document. Initially, we'll define the rules embedded in the document, but later we'll extract them to a separate CSS file.

4. To have somewhere to place our embedded rules, add a new <style> element to the document's <head> section:

```
<style>
    /* CSS rules go here (this is a comment).*/
</style>
```

5. Add a style-rule for the body element, with a background colour and foreground (text) colour. *Note that as you start to type a colour name, WebStorm will display a pop-up list of matching names – you can select the colour you wish from this setting, and it will be placed in the rule as a 6-digit hexadecimal value. If you continue to type the full name, it will stay as you typed it.* Note the selected colour appears as a small swatch in the left-margin of the WebStorm editor. Be sure to use the American spelling for 'color':

```
<style>
    /* CSS rules go here (this is a comment).*/
    body{
        background-color: papayawhip;
        color: darkred;
    }
</style>
```

6. Refresh the document and you should see that the whole page has been re-coloured. You can change the colour values to try out other combinations.

To continue, you will need to add CSS rules to the <style> element, using the same basic format:

```
rule-target {
      setting-name: setting-values;
      setting-name: setting-values;
}
```

Try adding the following rules:

7. Apply a rule to all <li> elements, with the following settings:

```
color: maroon;
 font-family: cursive;
list-style-type: square;
```

   Once you've added the rule, refresh the page to examine the results.

8. Styling the <table> element on the page is a bit different, since our aim ought to be to make the table structure as clear as possible. That might involve colouring different cells differently and changing the alignment of the content. Add the following rules:
   - ***<th>*** elements should have a ***background-color*** of ***tomato***
   - ***<tr>*** elements which are even-numbered (***tr:nth-child(even)***) should have a ***background-color*** of ***coral***
   - ***<tr>*** elements which are odd-numbered (***tr:nth-child(odd)***) should have a ***background-color*** of ***gold***
   - ***<td>*** elements should have a ***height*** of ***20px*** (pixels), a ***vertical-align*** of ***middle*** and a ***padding*** setting of ***5px***. (vertical align indicates how the text lines-up, padding is the amount of space around the text)

Refresh the page after each step to see how each rule is applied. As an exercise I've left the form at the bottom of the page so that it just adopts the rules that apply to the document body. Have a look at http://www.webcredible.co.uk/user-friendly-resources/css/css-forms.shtml for a tutorial on styling elements in a HTML form – this should be especially useful for styling an attractive user-interface in a web app.

Externalizing rules

All of your CSS rules can be moved to an external CSS file that the <style> elements links to. To do this:

9. Create a new file in the project called styles.css
10. Move the rules that are currently in the HTML <style> section into this document
11. Remove the <style> heading, and instead insert:

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

Major Exercise

The Appointments application could do with some CSS to improve its appearance. One area that you'll need to think about carefully is how to apply style-sheets to the table that displays appointments, without this affecting the table that tidies up the user-interface. You could do this in a number of ways (e.g. remove the first table and use style-rules as described in the link above (webcredlble.co.uk), but possibly the most straightforward would be to single out only the table elements in the second table – you can do this by adding an identifier (e.g. 'tbl') to the table element (which is created in Javascript code so you'll need to ad this in the updateList() method), and then set the table rule selectors as, e.g. ***#tbl { … }***, ***#tbl th { … }***, ***#tbl tr:nth-child {even}*** etc.

jQuery

jQuery is a well-used library for Javascript programmers that simplifies a lot of the coding needed for styling documents, working with the HTML DOM, adding animations, handling events and handling asynchronous web requests (we'll come back to that point later). It is an industry standard used in many large websites. We'll use it in this lab to illustrate how to select and manipulate elements.

Adding jQuery to an application

jQuery is a library of Javascript code, so to add it to a project, you simply need to provide a link to the jquery.js file. There are two ways to go about this:

- Download the jquery code file, add it to your project and then add a link to it
- Don't download the code file, but simply add a link to it at one of the servers it is hosted at.

There are advantages and disadvantages to both methods. By downloading the file and adding it to your project, you can guarantee that it is always available, even in the absence of a web connection. However, it is almost certain that it will download faster from one of the standard hosting sites (e.g. developers.google.com) and this will probably make your application load more quickly on a user's machine.

1. Add jQuery to your project, by ***EITHER***:

   Downloading it from jquery.com, placing it in your project folder and then adding a script link to it:

   ```
   <script type="text/javascript" src="js/jquery-1.8.2.min.js">
   ```

   OR:

   Adding a link to:

   ```
   <script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
   ```

   Note that ***jquery-1.8.2*** is the current version (at time of writing) and that ***jquery-1.8.2.min.js*** is a minified version (the size of the code has been reduced to make it download and execute more quickly.

We can now add some jQuery constructs to the code in Lab6.

1. The first piece of jQuery code that usually gets put into a document's Javascript is the document-ready handler. This is used to replace window.onload and is shorter and quicker to write, but also better since it waits until the full document, including links to images, is downloaded. Add a new file (lab6.js) to the project and add this code to it:

   ```
   $().ready(function () {
       alert("Loaded");
   });
   ```

   Note - $ is an alias for jQuery, and the default operation of the jQuery function takes the current document as a parameter, so the first line is equivalent to a call to ***jQuery(document).ready();***, which is the function that jQuery calls when a document has completely loaded. We've simply added an anonymous function to this.

2. Refresh the page and the alert() message should be displayed.
3. We'll try out some jQuery document manipulation. Add an id to the <h2> element above the table (id='tbl'), and then change the code in ***$().ready()*** to:

   ```
   $().ready(function () {
       $('table').hide();
       $('#tbl').click(function () {
           $('table').fadeToggle();
       });
   });
   ```

   Note that several things are going on here. In the call to ready(), we've first hidden the table on the page, but then have set up an event handler for a click on the heading for the table so that the table can be made to appear and disappear by clicking on the heading (***fadeToggle()*** is a function that fades an element in or out).

4. We can use jQuery to quickly and easily change HTML content in a document. Try the following:
    - Add code to the ready function so that the heading for the table says "A table for styling (click to reveal)" – use the ***.text()*** method of the '#tbl' element to do this.
    - Add code to the click() method of the table to change the text back to "A Table for Styling".
5. We can also affect the content of form elements. The name box in the form on the page is currently empty. It has an id of 'name'. Add a click event (in the ready() function) to this so that when you click on the element, the name "Fred Bloggs" (or whatever) is added to the text box (***.val("Fred Bloggs");***)
6. jQuery allows you to manipulate styles dynamically – i.e. in an event handler you can change an existing style, or you can apply a different style to an element. For example, to update the style settings for a single element (in this case, the <li> element clicked on):

```
$('li').click(function (event) {
    $(this).css('font-family', 'sans-serif');
});
```

Add this to the .ready() function, and then click on the <li> elements at the top of the page.

You can view the range of facilities for using jQuery in Javascript code to manipulate documents at http://www.w3schools.com/jquery/jquery_examples.asp.

Major Exercise

Add jQuery to the Appointments application. This will then allow you to:

- Change the selectors – instead of using ***document.getElementById('id')***, you can use ***$('#id')[0]*** (note, the [0] at the end is necessary because jQuery will return all of the elements that match a selector as an array, even if there is only one)
- Assign event handlers to elements using jQuery syntax (e.g. ***$("#ok").click(add);***)
- Add styles to elements dynamically (i.e. inside event handlers) so that elements only appear when they are needed (e.g. the Add/Update form can be hidden until either a new element is added or an existing one is updated)
- Use the ***$().ready()*** function to perform the application initialization (e.g. assigning elementsreferences to variables, loading appointments from localStorage and displaying them in the table).

End of Lab 6