Pre-lab 2: Introduction to client-side programming

The SENG365 course is split into two terms. During the first we will look at how web applications are programmed on the back end. Then, in the second term, we will look at programming front-end, client applications that can communicate with these back ends.

Before we begin with the back-end labs, it is useful to have a basic understanding of how web pages are programmed on the client side. This lab will introduce you to some of the fundamental tools and technologies used in front-end web development. We also take a look at how to write code that caters for different screen sizes and resolutions (desktop, mobile, tablet, etc).

You may work through this lab at your own pace. Those of you who are familiar with HTML/CSS and responsive design may find this lab elementary. Treat this lab as an opportunity to refresh your skills before starting the course.

This lab is split into four sections. Section one introduces Hyper-Text Markup Language (HTML), the language used for specifying the content and structure of a website. Section two introduces Cascading Style Sheets (CSS), which is used for adding style and formatting to a website. Section three discusses responsive web design, which is used for ensuring a website works on all devices and resolutions. And section four concludes with an exercise for you to put all of the previous concepts into practice.

1 Hyper-Text Markup Language (HTML)

The webpage that is presented to the user is usually written in HTML. HTML defines the structure and content of the webpage but doesn't specify the details of how it will look (the presentation).

1.1 Structure of a HTML page

HTML pages are split into two parts, the head section and the body section. The head section stores the majority of the pages meta-data (such as title, description, character encoding etc). The body section stores the content.

Aside from a few exceptions, the page is made up of elements and attributes. Elements are displayed in the page with opening and closing tags (e.g., `<element>Some content</element>`). Attributes are then used to provide additional information about the elements (e.g., `<element attribute="additional-information">Some content</element>`).

```
<!DOCTYPE html>
<html>
    <head>
```

```html
        <title>Page Title</title>
    </head>
    <body>

        <h1 id="main-heading">This is a Heading</h1>    <!--- This is a comment -->
        <p>This is a paragraph.</p>

    </body>
</html>
```

Copy the above code into a new file and save it as `test.html`. Now open the page in your browser.

1.2 HTML elements and how to learn more

There are a lot of HTML elements. Below we outline some of the most popular, the Mozilla Developer Network (MDN) is a good reference to use when looking for new elements to use (https://developer.mozilla.org/en-US/docs/Web/HTML/Element). The following examples are taken from the MDN website.

1.2.1 Links

Links are defined with the `<a>` tag. The `href` attribute gives the address of the page to be linked and the text between the tags is the text that will be presented to the user.

```html
<a href="https://www.mozilla.com/">External Link</a>
```

1.2.2 Images

Images are defined with the `<img>` tag, `src` defines the path to the image on the web server and `alt` provides some alternative text for if the image cannot be displayed.

```html
<img src="mdn-logo-sm.png" alt="MDN" width="104" height="142">
```

1.2.3 Blocks/Sections

The `<div>` tag is used as a container to store blocks of content. These tags can then have individual styles applied to them to structure the content on the page.

```html
<div class="navigation">
    <a href="/home">Home</a>
    <a href="/about">About</a>
    <a href="/services">Our Services</a>
    <a href="/contact">Contact</a>
</div>
```

1.2.4 Forms

Forms allow a user to input some data for the website to use (e.g., for interacting with a back-end API). Below is the code for a simple sign-up form.

```
<form method="POST" action="/subscribe">
  Enter your email to receive updates:<br />
  <input type="email" name="email" value="Enter here..."><br />
  <input type="submit" value="Subscribe">
</form>
```

On the first line, we define the forms method and action.

Method: The method specifies whether your form will use a HTTP POST request on submission of the form, or whether your form will use a GET request, putting the form content as variables in the URL (e.g., mysite.com/subscribe?email=hello@me.com).

Action: The action specifies what resource or function on the web server will manage the form data once it has been submitted.

Our form then contains two elements:

```
<input type="email" name="email" value="Enter here..."><br />
<input type="submit" value="Subscribe">
```

More information on the input element and the different types can be found on MDN.

2 Cascading StyleSheets (CSS)

CSS is used to apply different styles to our HTML. For example, take the following HTML:

```
<div class="first_section">
  Hello world!
</div>
<div id="second_section">
  Goodbye.
</div>
```

Here, the `<div>`'s can be identified using the class and the ID. We can then apply individual styles to these different identifiers. In CSS, a full stop (`.`) indicates a class and a hash symbol (#) indicates an ID:

```
.first_section{
  background-color: red;
  text-align: center;
}

#second_section{
  font-weight: bold;
}
```

Note: typically, classes are used for applying styles to groups of elements (such as navigation), and ID's are used to apply style to a single element.

We can also add style rules to all elements within a page:

```
div{
  border: 1px solid #000;
}
```

To link our CSS page to our HTML, we place the following inside the HTML's `<head>` section:

```
<head>
<link rel="stylesheet" type="text/css" href="./path_to_CSS/mystyle.css">
</head>
```

2.1 Advanced CSS: something to be aware of

The CSS standard is constantly being updated with new features. However, the different browsers implement these features at different times. This means that when including some of the new features, you need to be careful that you implement for all browsers. As an example, CSS3 (the latest standard) implements gradients on backgrounds. However, this change is yet to be implemented on all browsers. To allow for these different implementations, this is how our code should look (at the time of writing):

```
#grad {
    background: red; /* For browsers that do not support gradients */
    background: -webkit-linear-gradient(red, yellow); /* For Safari 5.1 to 6.0 */
    background: -o-linear-gradient(red, yellow); /* For Opera 11.1 to 12.0 */
    background: -moz-linear-gradient(red, yellow); /* For Firefox 3.6 to 15 */
    background: linear-gradient(red, yellow); /* Standard syntax */
}
```

Websites such as https://caniuse.com allow you to check which browsers a CSS feature will work on. For more information on CSS pages and the different features, check out the MDN resources (https://developer.mozilla.org/en-US/docs/Web/CSS).

3 Responsive Web Design

Responsive web design means designing and coding the front-end in a way that caters for all (or close to all) screen sizes and resolutions. The Google developer's documentation site provides a tutorial of responsive design here:
https://developers.google.com/web/fundamentals/design-and-ux/responsive/

Many CSS frameworks exist for aiding responsive development, the most popular of which is Twitter's bootstrap framework (http://getbootstrap.com/). Bootstrap is a CSS and JavaScript file that you link to from your HTML page. You can then use the Bootstrap CSS classes to make your website responsive.

4 Exercise: Create a personal CV site

The following exercises use the tools and concepts learnt in this lab to create a static website to advertise your CV. The site we create will include the following information:

1. Relevant personal information (name, tagline, address)
2. Academic achievements and qualifications
3. Professional experience
4. A way to contact you and links to any relevant social media (e.g., GitHub, LinkedIn, ResearchGate, etc.).

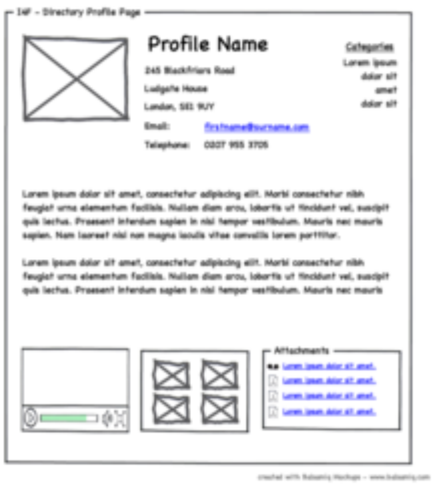
4.1 Exercise 1.1: Designing the site

First draw up a wireframe for each page of the site. A wireframe is an image or set of images which displays the functional elements of a website or page, typically used for planning a site's structure and functionality.

It is always best to start with a simple, high level wireframe so that you can understand how the different elements of the site will work together. Once you are comfortable with your sites layout, you can perform another iteration of wireframing, adding more detail.

Make sure your design considers how your site will look on different screen sizes and at different resolutions (phone, tablet, desktop, etc.)

Below is an example wireframe, taken from: https://en.wikipedia.org/wiki/Website_wireframe



4.2 Exercise 1.2: Creating the HTML

Now that we have decided on the design of our site, we can start to create it.

1. Create a new directory and create a new file for each page of your website. It might be best to create one page with all the features that are common to every page (e.g. title, navigation, footer etc) and then make copies of this file (or copy the HTML to the other files). Each file should be saved with the `.html` extension. Your home page should be called `index.html`. This tells the server that this page is your homepage.

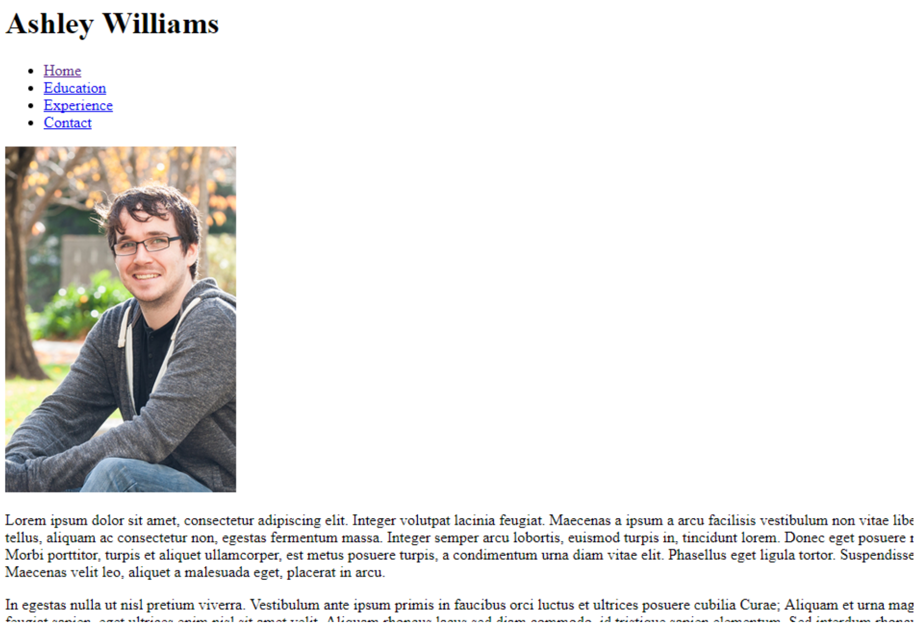
2. Start with the below HTML and create your pages by adding HTML into the body section.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>


    </body>
</html>
```

Add your HTML:

a. Use the `<h1>` tag to create a title for your site
b. Use the `<img>` tag to add an image of yourself
c. Use the `<p>` tag to add text

You can test your HTML by opening the pages in your browser, below is an example home page:



Ashley Williams

- Home
- Education
- Experience
- Contact

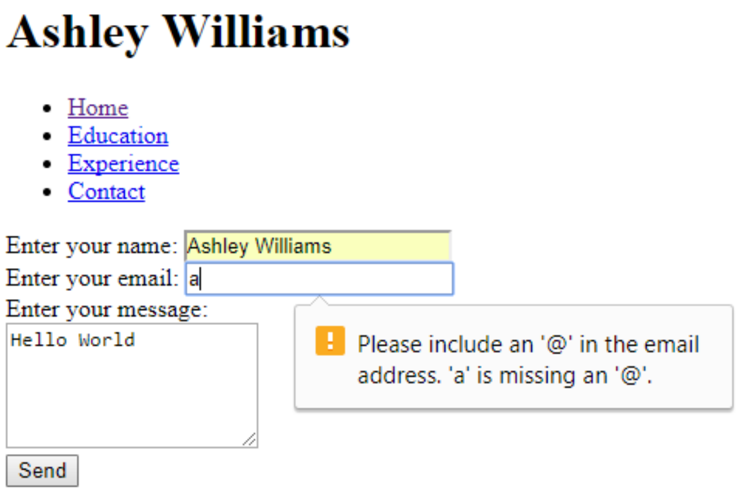
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer volutpat lacinia feugiat. Maecenas a ipsum a arcu facilisis vestibulum non vitae libe
tellus, aliquam ac consectetur non, egestas fermentum massa. Integer semper arcu lobortis, euismod turpis in, tincidunt lorem. Donec eget posuere
Morbi porttitor, turpis et aliquet ullamcorper, est metus posuere turpis, a condimentum urna diam vitae elit. Phasellus eget ligula tortor. Suspendisse
Maecenas velit leo, aliquet a malesuada eget, placerat in arcu.

In egestas nulla ut nisl pretium viverra. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam et urna mag

On your contact page, create a form and set the action to `mailto:<<your_email_address>>`

```
<form method=POST action="mailto:ashley@mywebsite.com">
       Enter your name: <input type="text" name="name" /><br />
       Enter your email: <input type="email" name="email" /><br />
       Enter your message:<br />
       <textarea name="message" rows="5">
       </textarea>
       <br />
       <input type="submit" value="Send" />
</form>
```

Open your contact page in your browser and test your form. Below shows the browser reacting to a user submitting an invalid email address.



Ashley Williams

- Home
- Education
- Experience
- Contact

Enter your name: Ashley Williams
Enter your email: a
Enter your message:
Hello World

Please include an '@' in the email address. 'a' is missing an '@'.

4.3 Exercise 1.3: Adding style

Now that we have the site's content, it's time to make it look presentable, and make it responsive. In this exercise we will show you a method for styling your navigation as an example, and then leave it up to you to style the rest of your site.

Styling navigation

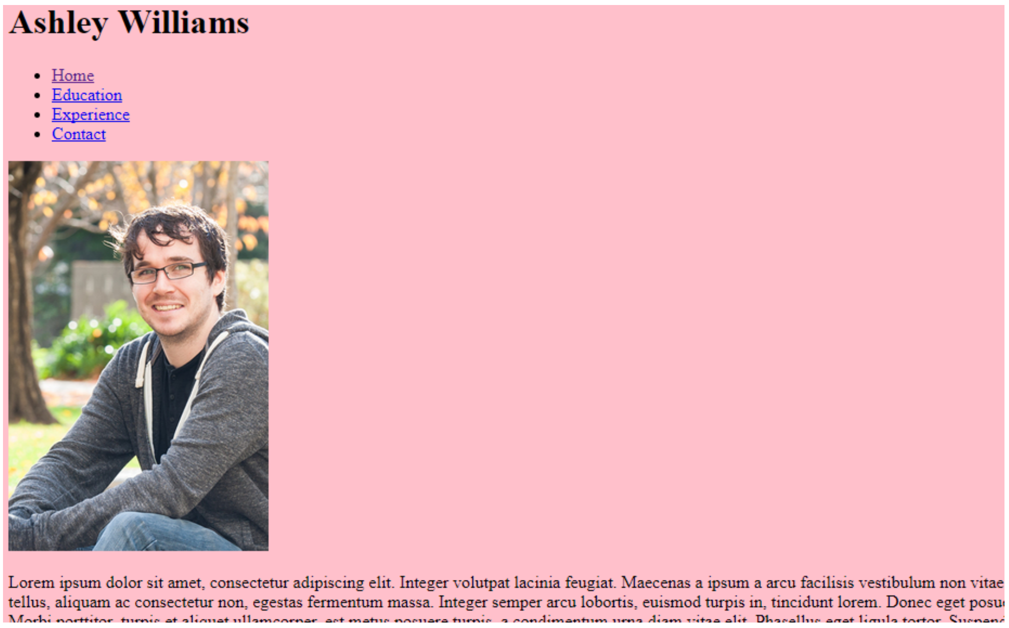
1. First, we create a style sheet, we will call it `style.css`. This file will store all of our site's stylings. For now, all that we are going to put in our stylesheet is the following code for testing:

```
body {
    background-color: pink;
}
```

2. Now, in the head section of our HTML file, we are going to add a link to our stylesheet.

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

3. Open your HTML file in your browser, if your page has linked to the stylesheet successfully, then you will have a pink background.



Ashley Williams

- Home
- Education
- Experience
- Contact

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer volutpat lacinia feugiat. Maecenas a ipsum a arcu facilisis vestibulum non vitae tellus, aliquam ac consectetur non, egestas fermentum massa. Integer semper arcu lobortis, euismod turpis in, tincidunt lorem. Donec eget posu
Morbi porttitor, turpis et aliquet ullamcorper, est metus posuere turpis, a condimentum urna diam vitae elit. Phasellus eget ligula tortor. Suspend

4. Once you know that the stylesheet is working, you can remove the background. Open your stylesheet and delete its contents. Now we can style our navigation.
5. We will start with the below navigation HTML. We have put our navigation into its own `div` and given it a class. This means that we can make changes to just the particular list elements inside of this `div`, and not have our styles applied to every list on our site.

```
<div class="navigation">
       <ul>
              <li><a href="/">Home</a></li>
              <li><a href="/education">Education</a></li>
              <li><a href="/experience">Experience</a></li>
              <li><a href="/contact">Contact</a></li>
       </ul>
</div>
```

6. At the moment, the navigation is listed as bullet points. We can get rid of this by setting the `list_style_type` to none (setting the `margin` and `padding` to `0` removes any pre-set browser styles). In our stylesheet, we add the following code:

```
.navigation ul {
       list-style-type: none;
       margin: 0;
       padding: 0;
}
```

7. Save the stylesheet and open the HTML file in the browser, the bullet points have gone. Now, we want each link to appear as a blue rectangle button instead of a string of text. To do this, we set the width and height of the link element (`<a>`) and set the `display` to be a `block`. In our stylesheet, we add the following code:

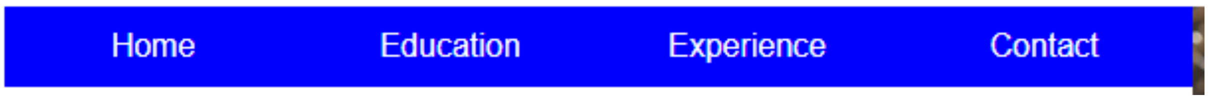
```
.navigation li a{
	display: block;
	width: 150px;
	height: 40px;
	background-color: blue;
	color: white;
}
```

8. Again, save the stylesheet and open the HTML in the browser. Each link is now a block. We want to make our navigation list a horizontal row, so we are going to set the `display` of each list element to be `inline` and set the `float` to `left`.

```
.navigation li{
	display: inline;
	float: left;
}
```

9. Now after we save and refresh the page, the links appear in a row, rather than one underneath the other. We can clean up the navigation using the following properties:
   a. `font-family` changes the font
   b. `text-decoration` removes the underline
   c. `text-align` centres the text horizontally within the link block
   d. `vertical-align` and `line-height` centres the text vertically within the link block

   Our navigation now looks like this:



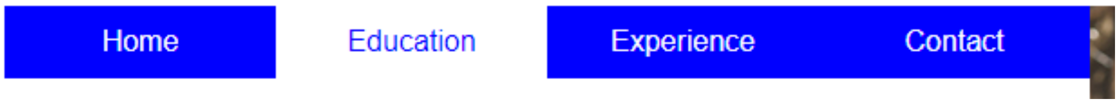
   And the stylesheet like:

```
.navigation ul {
	list-style-type: none;
	margin: 0;
	padding: 0;
}

.navigation li a{
	display: block;
	width: 150px;
	height: 40px;
	background-color: blue;
	color: white;
	text-decoration: none;
	font-family: helvetica;
	text-align: center;
	vertical-align: middle;
	line-height: 40px;
}

.navigation li{
	display: inline;
	float: left;
}
```

10. Finally, we want to set the `hover` property so that the link changes color if our mouse cursor hovers over it. Add the following code to your stylesheet:

```
.navigation li a:hover{
       color: blue;
       background-color: white;
}
```

11. Save your code and test in your browser. Hovering over a link with your cursor will change the link’s color.



Now style the rest of your website, using resources available on the web to help you. You may use CSS frameworks such as Bootstrap, or write your own styles as we have done with the navigation example above. Your finished website should end up looking like the wireframes you created.