Lab 4 – Responsive layouts and introduction to JavaScript

1 Introduction

Welcome to Week 4!

So, by now we can create web pages and make them pretty. That's enough for 50% of the assignment. Today we will be looking at responsive design and scalable layouts.

Following that, we will start looking at JavaScript (the most popular programming language in the world and language of the web). We'll begin with the basics today and then over the next couple of weeks we will build up our knowledge necessary for the final 45% of the assignment.

While this lab looks a lot shorter than what we have done in previous weeks, some of the exercises will take you a lot longer to complete. These are important exercises however as they will be needed for the assignment.

2 Learning Objectives

- To understand, and practice responsive design and how we can create layouts and structures that work across all devices
- To understand, and practice using, some JavaScript basics

3 Responsive Design

Responsive design means creating a style and layout for your website that works across all screen sizes and devices. Thinking about responsive design while making websites leads to writing code once and it working everywhere. Most of the web traffic today originates from mobile devices, so many even advocate the idea of mobile-first design to encourage developers to approach web projects with a responsive mindset.

3.1 Exercise 1: Getting started

To get started with responsive design, we need to set the viewport size. The viewport is the part of the screen/device/browser that will show the content of our site.

1. Create a HTML page using the template we've been using in previous weeks.
2. Download any image from the internet and add it into the body of your page using the <img> element
3. Open up your browser's developer tools and experiment with how the image looks over different screen sizes. Try a very large screen size and a very small screen size.
4. Now set the viewport size to be the same as the device's size. This can be done by adding the following <meta> tag into the <head> section of your web page:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
ead>
```

5. Experiment again with the different device sizes in the developer tools. Can you see how the image now adapts to the different devices?

3.2 Exercise 2: Responsive sizing

One step we can make towards creating responsive layouts is to avoid using fixed dimensions on elements (e.g. using pixels for images and divs). Instead we can use percentages and the vw and vh (view width and view height) to make layouts proportional to the device being used.

1. Take your solution to exercise 1 and using CSS, set the width of your image to 300px.
2. Using the developer tools again, experiment with different screen sizes so that you can see how it behaves.
3. Try changing the width to instead be 100%. The image should now scale with the size of the screen.
   Note: that the width is inherited from the parent element. If the image was placed inside a div with a width of 1000px, then 100% will be 1000px. To use the width of the viewport, we can instead use 100vw (and 100vh for height).
4. One issue with setting the width to 100% is that on large screens, the image may scale too high up and become grainy and pixelated. We can set the max-width property of the image to 100%. This means that the image will scale like before, but won't scale higher than its original size.

3.3 Exercise 3: Learning flexbox with FlexboxFroggy

Develop your understanding of Flexbox by completing the exercises on the Flexbox froggy website: https://flexboxfroggy.com

3.4 Exercise 4: Using flexbox to create complex layouts

1. Using a piece of paper, or any graphical software (MS Paint, GIMP, even MS Word), draw a rectangle in landscape orientation that is about the same dimensions as a typical web browser.
2. Fill the rectangle by drawing at least 8 smaller squares and rectangles of different shapes and colours.
3. Now create HTML webpage and link it to a CSS file.
4. Using flexbox and a series of elements, can you mimic your design in the browser?
5. Using the browsers developer tools, test your design on a simulated phone and tablet. Has it broken the design of the layout in a way in which you were not expecting?
6. Fix the code so that it displays correctly on all screen sizes.
7. Try again with a different screen size (e.g. iPhone 5 has a very small screen size).

4 Intro to JavaScript

4.1 Exercise 5: Getting started with JavaScript

1. Create a HTML page using the basic template that we have used in previous weeks
2. Create a new file with a .js extension. This will be where we store our JavaScript code

3. In the <head> section of the HTML document, link to the JS script using the following line of code:

```
<title>Week 3 exercises</title>
<script src="week3_ex.js"></script>
</head>
```

4. That's all you need for linking your HTML to the JS script. However, for the remaining exercises, also add the following elements:

```
<body>
    <h1 id="header">Week 3 Exercises</h1>
    <p id="paragraph">Welcome to JavaScript</p>
    <button id="myBtn">Click here</button>
</body>
```

4.2 Exercise 6: The basics – variables, if's, loops, functions

Using W3Schools and the lecture slides to help you, work your way through the following exercises.

Note: place all of your code inside window.onload to ensure that the HTML page loads before the JS is executed:

```
window.onload = () => {

    // code here

}
```

1. Change the contents of the <h1> tag so that it says "Welcome to my website"
2. Change the contents of the <p> tag so that it displays the current date and time
3. Write a function that calculates the area of a circle for any given radius. Update the contents of the <p> tag so that it displays the area of a circle with a radius of 4.
4. Write a function that accepts a student's grade as a parameter and returns their classification using the below table:

Mark	**Classification**
81 – 100	Fantastic
61 – 80	Very Good
41 – 60	Good
<40	Poor

5. Write a function that accepts three integers as parameters and returns the largest.
6. Write a function that capitalises the first letter of every word in a sentence (Hint: use the split() function to convert a sentence into a list of words)

4.3 Exercise 7: Arrays and objects

We often use JavaScript to send and retrieve data to/from a server. This data is usually structured in JSON objects (JavaScript Object Notation). JSON objects are essentially a set of key value pairs wrapped in curly brackets. For example, look the car object here:

```
let car1 = {
    brand: "Tesla",
    model: "Model 3",
    year: 2021,
    price: 40000
}
```

Using W3Schools and the lecture slides to help you, work your way through the following exercise.

1. Create three student objects. Each object should hold the students name, email, and average grade (just use dummy data)
2. Create an array called “students”. Add each of your three objects to the array.
3. The remaining exercises require you to loop through the array and perform operations on the data (you can print to the console like we did in the lecture):
    a. Loop through the array and print out the names of each student
    b. Using your function from exercise 6, print the classified grades for each of the students
    c. Print the name of the student who has the highest grade
    d. Calculate the average grade