Lab 1 - HTML, CSS and Javascript

Graphical User Interfaces[*]

In this lab, we will recap core concepts for HTML, CSS and Javascript. Teaching these technologies in their entirety within the lab itself is impossible - we aim to cover the basics here so that you will be able to explore your interests in more detail outside of the lab. Post questions to the forum outside of lab hours. On completion of this tutorial you will have a greater understanding of:

- Laying out and creating HTML elements.
- Applying CSS styles.
- Arranging HTML elements through CSS.
- Adding interactivity using Javascript.

Students with a strong grasp of HTML, CSS and Javascript already might wish to jump straight to the following exercises:

- 2.3 TASK: HTML Practice
- 3.2 TASK: Create a CSS style sheet and link to our html
- 3.4 TASK: Different properties and values
- 4.1 TASK: Setting up Javascript
- 4.5 TASK: Racing Game

1 Using a Text-Editor

To write HTML, CSS or Javascript code, we will need to use a text editor. There are a number of different choices (you could even use the built in notepad app). Some good options are Sublime Text and Notepad++.

In these lab tutorials, we will use **Visual Studio Code** beacuse it has helpful tools supporting us in the later labs. It can be downloaded from `https://code.visualstudio.com/download`.

[*]Written by Corey Ford 2023/24 for Queen Mary University of London (c.j.ford@qmul.ac.uk)

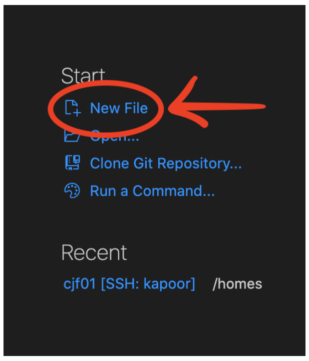
1.1 TASK: Download Visual Studio Code

2 HTML: Hyper Text Markup Language

HTML, or HyperText Markup Language, is a standard markup language used for creating and designing web pages. It is the fundamental building block of web content, allowing the structuring of text, images, links, forms, and other elements on the internet.

2.1 TASK: Make our first HTML webpage

In visual studio click new file, then choose text file.



Press Ctrl+S (or Command+S on Mac) to save the file to a new folder, located anywhere you like. Name the file **index.html**. Saving a file as "index.html" is common practice in web development. When a web server receives a request for a particular directory (e.g., https://example.com/), it looks for a default file to serve – "index.html" is the default homepage for the directory.

HTML files tend to follow a standard boilerplate layout, shown below.

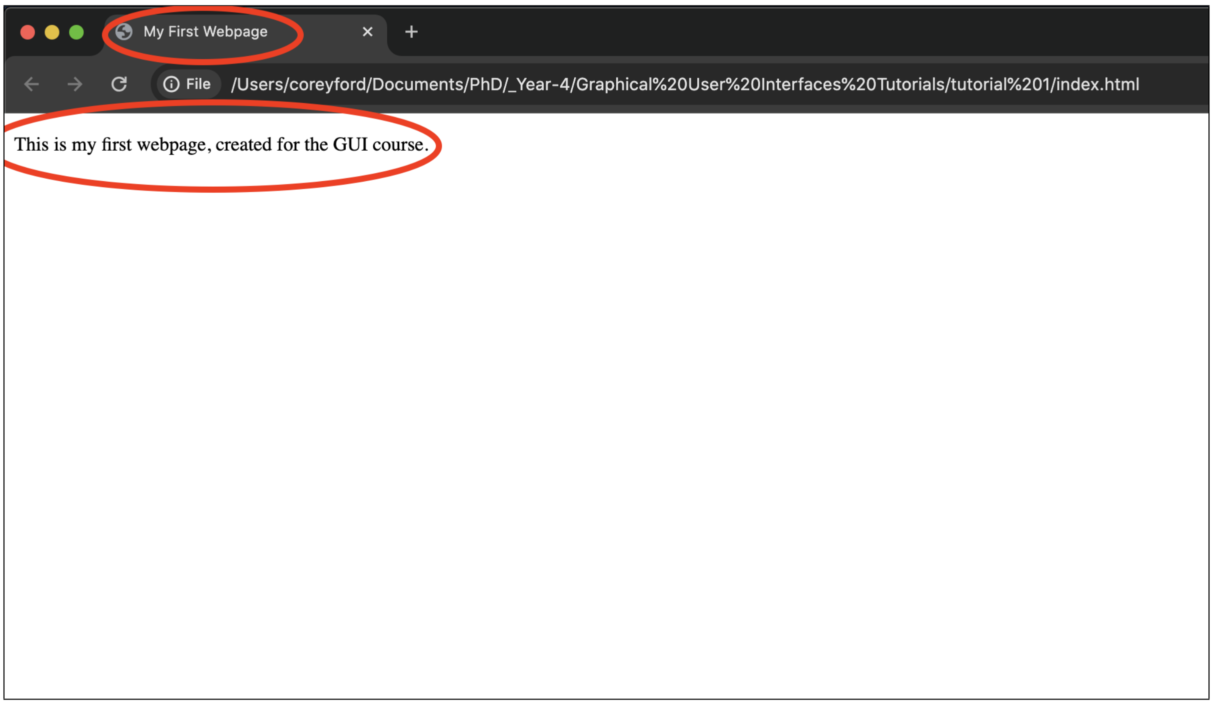
```html
<!DOCTYPE html>
<html>
    <head>
        <title>My First Webpage</title>
    </head>
    <body>
        <p>This is my first webpage, created for the GUI course.</p>
    </body>
</html>

```

Write the code above into the "index.html" file in visual studio. The elements of the code are as follows:

- **Line 1:** declares that we are writing html code, and adheres to html5 standards. It ensures consistent rendering across browsers and enables validation of HTML code.
- **Line 2 and Line 9:** are tags. Open tags use no slash, whereas closed tags used a "/". HTML tags are the root of a html document. All content nested within them will appear on the webpage.
- **Line 3 and Line 5:** are the header tags, where we place metadata related to our webpage. For example, on line 4 we add <title> tags, and inside these write "My First Webpage". This means that on our web browser tabs we will see this web page named "My First Webpage" (see image below).
- **Line 6 and Line 8:** show the <body> tags. Everything inside of these tags are displayed as the main content of our webpage. Here we use the <p> tag on line 7 – this means the content inside of this p tag is a paragraph, so we add some text.

If you now open "index.html" from your folder in a web browser (e.g. Google Chrome), you should see a website that looks like this:


My First Webpage
File
/Users/coreyford/Documents/PhD/_Year-4/Graphical%20User%20Interfaces%20Tutorials/tutorial%201/index.html

This is my first webpage, created for the GUI course.

2.2 Different Tags

Above we have shown the use of the <p> tag to add text onto our webpage. There a numerous different tags that we can use.

Heading tags. Instead of using the <p> tags, we can use <h1>, <h2>, <h3> etc... up to <h6>, to have different levels of heading text.

Lists. We can make both bullet point lists using <ul> and numbered lists using <ol>.

```
<ol>
    <li>Corey</li>
    <li>Ulfa</li>
    <li>Nick</li>
</ol>
```

Links. Links can be created by wrapping the content (e.g. text) in <a></a> tags. We also add an *attribute* to the opening tag, before its closing angular bracket. This attribute is "href" and points to the url of the file to link to, placed in quotes "".

```
<a href="https://example.com/">Click this link here!</a>
```

Images. Images can be added using a self closing tag, using the syntax below. The image source can be a filepath or URL.

```
<img src="ADD IMAGE SOURCE HERE - e.g. from google or files">
```

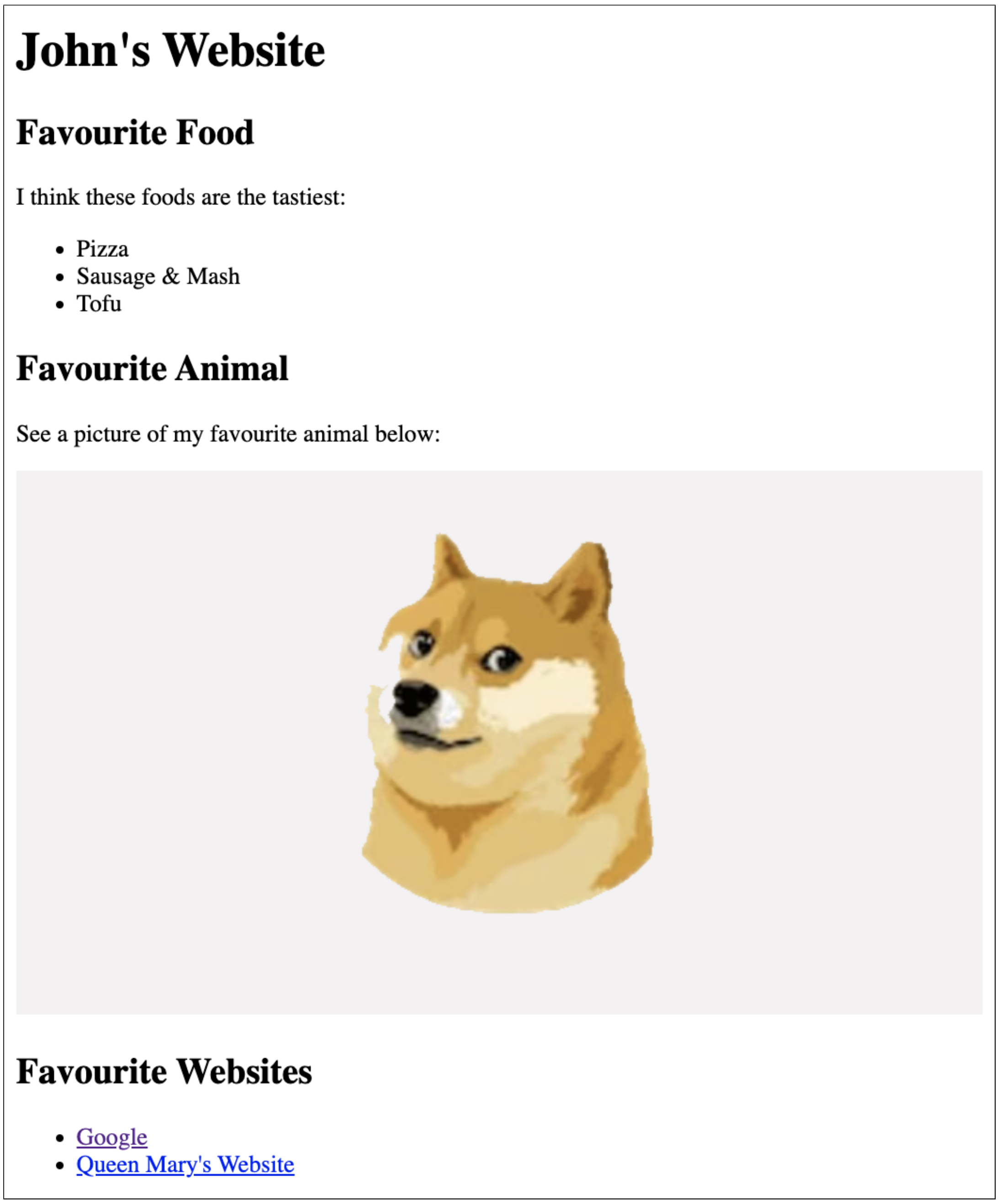
Div. The <div> tag acts like a container, grouping content together. It doesn't have a visible effect on its own, but when you learn CSS later, you can use <div> to organize and style different sections of your webpage. E.g. you could place images and text within one div, and different images and text in another, and place them besides eachother.

Comments. To add comments to HTML (text which is not shown on the webpage but that can be useful for documentation), we open the comment with **<!- -** , write our text, and then close with **- ->**.

Others. The best students may wish to go away and learn about HTML tags for Tables and Forms. The links at the end of this lab worksheet will help you.

2.3 TASK: HTML Practice

Try to recreate the website shown below. You might even like to customise it to show information about your own hobbies and interests, and find your own images.



John's Website

Favourite Food

I think these foods are the tastiest:

- Pizza
- Sausage & Mash
- Tofu

Favourite Animal

See a picture of my favourite animal below:

Favourite Websites

- Google
- Queen Mary's Website

3 CSS: Cascading Style Sheets

CSS, or Cascading Style Sheets, is a stylesheet language used to describe the presentation and visual styling of HTML documents. It enables the separation of content from design, allowing web developers to control the layout, colors, fonts, and other visual aspects of a website.

If HTML is the structure of our page, CSS is the styles that get **applied** to the structures of our page.

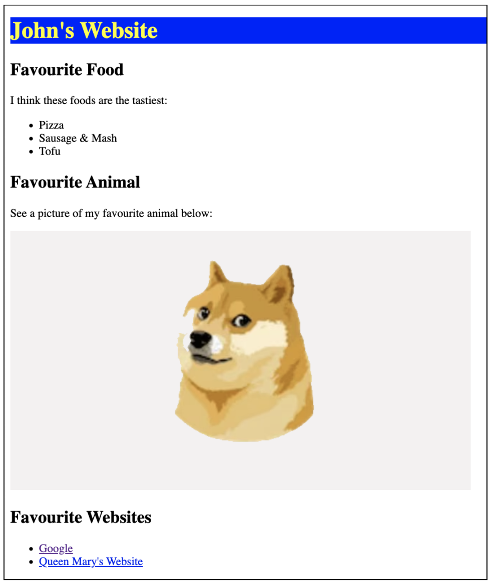
3.1 Our First Styles

We can modify CSS in two ways: i) inline with our HTML code, and ii) using a CSS style sheet.

For the first method (inline), lets take our previous exercise to add colours to our h1 tag. Specifically, lets add a style attribute, with the following properties:

```
<h1 style="color: yellow; background-color: blue;">John's Website</h1>
```

This will make our website look like this:



John's Website

Favourite Food

I think these foods are the tastiest:

- Pizza
- Sausage & Mash
- Tofu

Favourite Animal

See a picture of my favourite animal below:

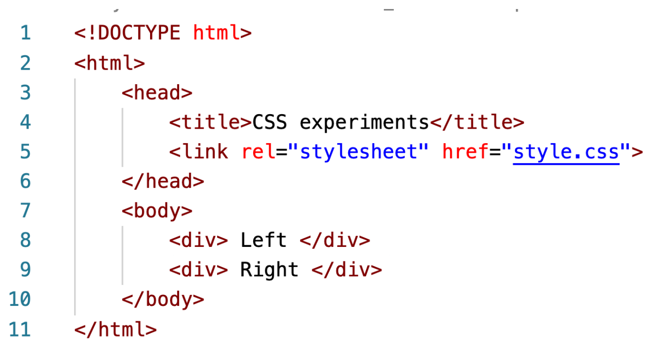
Favourite Websites

- Google
- Queen Mary's Website

Applying this style would become annoying if we had to do this many times. So it is better to move this into another file. Lets set this up.

3.2 TASK: Create a CSS style sheet and link to our html

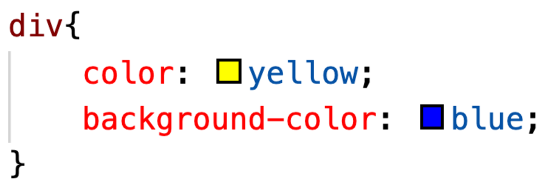
a. Create a new folder and create a new "index.html" file. In this new index file, add the code below:



```html
<!DOCTYPE html>
<html>
    <head>
        <title>CSS experiments</title>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <div> Left </div>
        <div> Right </div>
    </body>
</html>
```

b. Next to "index.html" in our new folder create a new file called "style.css".

c. Add the code below to "style.css". As we use div on line 1, all attributes within the brackets will be applied to any div tag on our webpage.



```css
div{
    color: yellow;
    background-color: blue;
}
```

Note that "color" and "background-color" are known as **properties** in CSS, whilst yellow and blue are **values**.

d. We link the style sheet to index.html where we added a <link> tag within the <head> tags. The link is self contained and with two attributes. **<link rel="stylesheet" href="style.css">**.

3.3 Applying CSS

In our CSS we can add lots of different properties and values, and assign these to different elements in our HTML.

At the moment, in our previous exercise, we are overriding all DIV tags to look a certain way. Usually, we will only want to apply a certain style to one element, or create a **class** which applies to multiple elements. We can create our own IDs for CSS elements by starting our CSS objects with either a # (for a style only used once) or a **.** (for a style to be applied across multiple elements).

```css
#myStyleForOneElementOnly{
    color: yellow;
    background-color: blue;
}

.myStyleForManyElements{
    color: yellow;
    background-color: blue;
}
```

To attach the CSS to elements, within a HTML tag, we either use the ID attribute (for a style used only once) or class attribute (for an style used many times).

```html
<div id = myStyleForOneElementOnly> Left </div>
<div class = myStyleForManyElements> Right </div>
<div class = myStyleForManyElements> Right </div>
<div class = myStyleForManyElements> Right </div>
<div class = myStyleForManyElements> Right </div>
```

3.4 TASK: Different properties and values

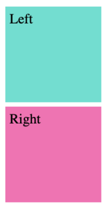
There are a range of different properties and values that we can apply. In our new project, let's add the following to our "style.css" file:

```css
1   #leftDiv{
2       background-color:  turquoise;
3       width: 100px;
4       height: 100px;
5       margin: 5px; /* Spacing around the box */
6       padding: 5px; /* Spacing within the box*/
7   }
8
9   #rightDiv{
10      background-color: hotpink;
11      width: 100px;
12      height: 100px;
13      margin: 5px;
14      padding: 5px;
15  }
```

Line 2 and 10 specify different background colours. **Line 3, 4, 11 and 12** we set the width and height to 100 pixels. **Line 5, 6, 13 and 14** we apply some padding so that the boxes do not touch directly next to each other. Margin is the space between the outside of the box, whereas padding is the space added within the box, between the box itself and its text.

Next, apply these CSS styles to the two div tags (left and right) using the ID attribute in "index.html" – apply the knowlege explained in the previous section to do this.

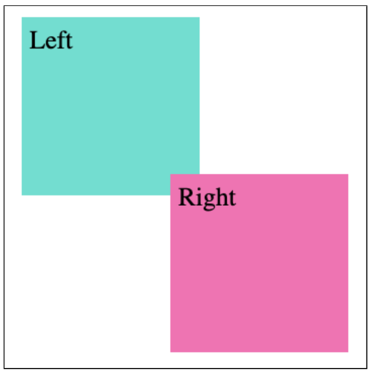
You will see the following on the web page.

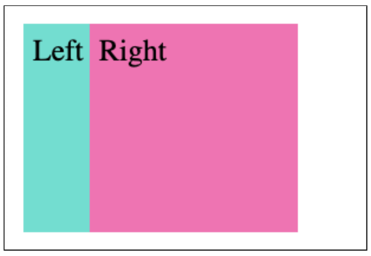


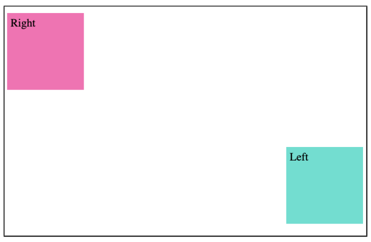
By default, our CSS styles apply the **relative** value to the **position** property – this means our DIV tags will be placed where they appear in our HTML code. We can change this position property to **absolute** if we want to specify where it should go on our webpage using the CSS code itself. This allows us to choose where to place our boxes from the top and left side of the screen, e.g:

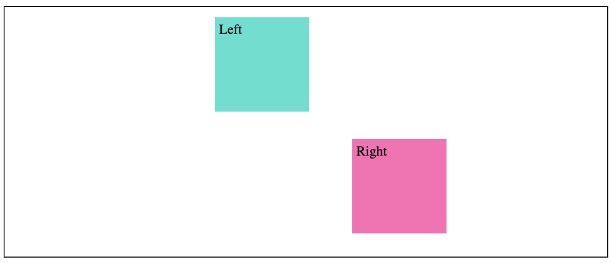
```
position: absolute;
top: 100px; /*from the top*/
left: 100px; /*from the left*/
```

Try to recreate the four different images below by manipulating the CSS code only, using absolute positions.









Using absolute CSS positioning can be advantageous when given prototypes of interfaces from designers (using tools such as Figma which we explore next week). It simplifies the process of positioning elements as you can experiment through trial and error. This makes it super helpful for the coursework on this module. However, it's worth noting that this approach does not lead to a responsive web design – aspiring web developers should proactively explore and learn additional techniques for responsive design beyond this approach (such as here `https://www.w3schools.com/css/css_rwd_intro.asp`).

4 Javascript

JavaScript is a versatile programming language used to add interactivity and dynamic behavior to websites. It is essential for creating responsive and engaging user interfaces, enabling features like form validation, animations, and real-time updates on web pages.

If HTML is the structure of our page, CSS is applying styles to that structure, and Javascript is for adding interactivity to our style and structure.

Javascript does not directly change our HTML code, but instead changes the "Document Object Model" of our HTML code. Put overly simply, the DOM is a representation of the HTML code that Javascript is able to tweak – they both are identical where Javascript updates the DOM to display changes in the HTML. Don't worry about this too much for now, however, we will explore this more deeply when we start using React.

4.1 TASK: Setting up Javascript

Create a new file alongisde our most recent "index.html" and "style.css" code, named "script.js". This is where we will write our javascript code.

In our "index.html" we need to then link to our javascript code by adding the following code:

- **<script src="script.js"></script>**

Add this above the </body> tag, but below all of the main page content. This is because Javascript needs to know what is contained in our HTML file in order to modify it (via the DOM).

4.2 Console & Dev Tools

To print output to the console in Javascript use the command: **console.log("insert text here")**. This can be useful for debugging.

To see the output of the console in Google Chrome go to **View** >> **Developer** >> **Javascript Console**.

This is very useful to have open when developing web applications.

4.3 Variables, Conditionals, Loops, Lists, and Functions

JavaScript shares key programming features with languages like Java. Here's a breakdown of essential concepts:

Variables can be declared using **let** or **var**. The key difference is in how they behave. With let, you create a variable that is only known and usable in the part of the code where you create it (e.g. within a function). This can help prevent accidental mistakes. On the other hand, var might sometimes cause unexpected issues because it has a broader reach in the code. For modern JavaScript, **let** is recommended. Experienced programmers might like to search up and learn when to use **const**.

Conditionals such as if-statements can be used. Be careful to use === instead of == which appears in other programming languages.

Loops are valuable. E.g. a for loop is written like **for(let i = 0; i < maxValue, ++i)**. Other loops such as while loops are also available.

Lists can be declared in Javascript and are similar to arrays. The .length property can be used on a list to get the maximum number of items in a list.

Functions can be declared and called using the **function** keyword. Arguments do not need a specified type, nor do return values.

An example is shown below which uses all of the above features. They are best to learn as you go about your own projects, building upon your previous programming experiences.

```javascript
// make a list of numbers
let numbers = [1, 2, 3, 4, 5];

// make a function to say if numbers are even or odd
function isEven(arr) {
  for (let i = 0; i < arr.length; i++) {
    if (arr[i] % 2 === 0) { //< note the triple equals sign
                            //< the percent sign means mod, which returns the remainder of a division
      return true; // if it is even
    } else {
      return false; // if it is not even
    }
  }
}

isEven(numbers); //< call the function
```

4.4 Accessing the DOM & Event Listeners

Javascript really becomes useful when we are able to manipulate our HTML elements. We can do this by using the **document** keyword.

For example, **document.getElementById("leftDiv");** can be used to set information about our leftDiv HTML element. For instance, **document.getElementById("leftDiv").style.left="400px";** will set out leftDiv to 400 pixels from the left hand side of our screen.

We can also apply functions to write code which happens when interacting with objects through the **document** keyword. Below we add an event listener, with the "click" argument. This means that whatever code we add on line 4 in the image below will be run when we click on the "leftDiv" object.

```javascript
3   document.getElementById("leftDiv").addEventListener("click", function(){
4     // Insert function information here...
5   })
```

4.5 TASK: Racing Game

Let’s turn our two DIV boxes into a racing game. Pressing ’f’ will move the left square to the right, whilst ’j’ will move the right square to the right. A friend can then press one key whilst you press the other, until a a winner moves their square off-screen. This will illustrate how to implement different types of interactivity to control our DOM, which will be useful for the coursework.

In our JS file, lets declare two variables which will store the staring values for our boxes.

```
// Variables
let leftStart, rightStart;
```

Next, we write a function to reset our values to their defaults. We set our variables created above, and then using the **document** keyword set the left and top properties of our DIVS.

```
// Function to move to the staring point
function startingPoint(){
    // set variables
    leftStart = 10;
    rightStart = 10;

    // set left
    document.getElementById("leftDiv").style.left = leftStart + "px";
    document.getElementById("rightDiv").style.left = rightStart + "px";

    // set right
    document.getElementById("leftDiv").style.top = "0px";
    document.getElementById("rightDiv").style.top = "150px";
}
startingPoint(); // call function for the first time.
```

We then attach on click listeners to this **startingPoint()** function on the left and right div. This means that when we click on our boxes, they will move back to the beginning.

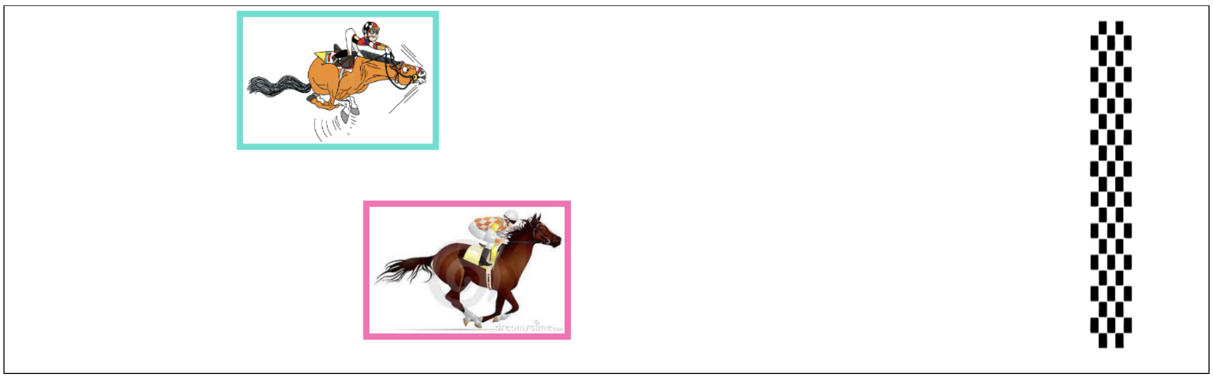
```
// If either DIV clicked move to the beggining
document.getElementById("leftDiv").addEventListener("click", startingPoint);
document.getElementById("rightDiv").addEventListener("click", startingPoint);
```

Lastly, we setup an event listener for the ’f’ and ’j’ keys. We increase our leftStart and rightStart variables, and update our DOM accordingly to move our DIV boxes.

```
// Add event listener on keydown
document.addEventListener('keydown', function(event) {
    if (event.key == 'f') {
        leftStart += 10;
        document.getElementById("leftDiv").style.left = leftStart + "px";
    }

    if (event.key == 'j') {
        rightStart += 10;
        document.getElementById("rightDiv").style.left = rightStart + "px";
    }
});
```

Extension: Why not add some images to make our game more fun. We could also add other mechanisms like randomess, and a score-board?



5 Summary

In this tutorial, we navigated the core concepts of web development, focusing on HTML, CSS, and JavaScript. Commencing with HTML, we established the groundwork for structuring web content. CSS was introduced with stylesheets customising the visual aspects of our HTML. JavaScript was then added to our technology stack to provide the means to dynamically manipulate the Document Object Model (DOM) for enhanced interactivity. The tutorial culminated in a practical exercise, implementing a racing game to showcase the synergy of HTML, CSS, and JavaScript in crafting responsive and dynamic web applications. Armed with this foundational understanding, you are well-equipped to navigate the intricacies of web development. Next week's exploration of Figma will further enrich our understanding by delving into the graphical user interface (GUI) design process.

6 Reading

Check the following websites with HTML and CSS tutorials. Both contain good and understandable information to refresh your knowledge for websites' layout using these two tools. Both tools will be useful in the implementation part of the coursework.

- `https://learn.shayhowe.com/html-css/`
- `https://www.w3schools.com/html/`
- `https://www.w3schools.com/css/`

On Javascript:

- This website has a good 'Hello World' example. You should start with this one if you are a beginner or have not used JavaScript for a while. `https://developer.mozilla.org/enUS/docs/Learn/Getting_started_with_the_web/JavaScript_basics`
- 'JavaScript Guide' gives you a lot of information about JavaScript. It has many examples for you to check. Go step by step to master JavaScript. `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction`
- There is a very famous JavaScript tutorial on w3schools.com. With the 'Try it Yourself' editor, you can edit the source code and view the result. It would be best if you went through at least until the lesson 'JS Array Iteration'. `https://www.w3schools.com/js/`
- You can find another one, similar to the previous one, with good structure and many examples for you to try. Useful if you check the sections, JavaScript Basic Tutorial and JavaScript Objects. `https://www.tutorialspoint.com/javascript/index.htm`