Challenges

Here is a series of challenges for you to play with, they details what is expected and how to achieve this, if you have any questions or get stuck at any point, do ask for help – there might be different ways to achieve the same thing too so no question is a bad question.

Let’s have some fun!

Challenge 1 – add some styles and elements to your web page

- You could add some other CSS styles - why not change the style of the text (color, size etc…), or give the image a border?

- Add another input text box and position this text at the top of the meme, so you can have text on both the top and bottom.

Challenge 2 – separate your CSS code

When using multiple web pages on a website, they usually follow the same style rules. These rules are saved within a CSS file separate from you HTML yourfile.html.

- Create another file called “styles.css” and save it in a new folder called “css” at the root of your project, where the html yourfile.html file is saved. This will be our CSS file where all the style rules will be defined; we will then call this file from the html files where we want to apply those rules.

- Now that we have a CSS file of its own, we will need to link the HTML file to this new CSS file. In order to do this, replace your current <style></style> tags and all in contents with the following:

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

- Refresh your web page in the browser and make sure that the styles are still applied

- Well done, you have created your first CSS file!

Challenge 3 – separate your JavaScript code

Same as with the CSS styles, the behavior in JavaScript can be shared between web pages and are usually stored within a JS file separate from you HTML yourfile.html.

- Create another file called “app.js” and save it in a new folder called “js” at the root of your project, where the html yourfile.html file is saved. This will be our JS file where all the behavior will be defined; we will then call this file from the html files where we want to apply those rules.

- Now that we have a JS file of its own, we will need to link the HTML file to this new JS file. In order to do this, replace your current <script></script> tags and all in contents with the following:

```
<script src=”app.js”></script>
```

- Refresh your web page in the browser and make sure that the behavior is still applied

- Well done, you have created your first JS file!

Challenge 4 – add a new web page and link to it +_ image in the page with anchor

- Open a new file in your text editor and save it as yourfile2.html at the root of your project.

- Add new elements to your page such as headers, paragraphs or images.

- When you are happy with it, we will add a link to the first page in it.

```
<a href="yourfile.html">My link</a>
```

- Try to make your link open in a new window.

- Well done, you have created your first link!

- Add a link back to your second page in your first page so you can navigate between the two.

Challenge 5 – create navigation links to navigate between your web pages

- We now have two links, one in each page to navigate between the two. We want to create a navigation bar that will contain both of those links in order to navigate between those two pages in a similar manner.

- We can do that by creating a new area on the page, which will contain a list of the two links and apply some style so that it displays as we wish. Let's start with the navigation area, right after the <body> tag:

```
<div class="navigation">
    <ul>
      <li><a href="yourfile.html">Link 1</a></li>
      <li><a href="yourfile2.html">Link 2</a></li>
   </ul>
```

```
        </div>
```

- Make that navigation bar stick to the top of the page with some styling (for eg: full width, fixed position, elements of the list in one row with the display attribute)

Other Challenges:

- Make your website responsive with the Bootstrap library

- Add some behavior with the JQuery library

- Make sure you don’t lose any work by saving each version of your files with git