Creating Your Login Page

The first page we are going to create will be the login page. This will be the first page that users will see, and will allow them to access the rest of the website.

We will design this page, and all of our pages, using a language called HTML. HTML stands for HyperText Markup Language, and this is what tells the browser what controls should be on your page, and what they should do. CSS will style your controls. And JavaScript adds functionality to your controls which we will see later on - but HTML is the key component that defines exactly what we want to see on our page.

Every HTML file takes the same basic structure to start with:

<html> (every html file must start and end with these tags)

<head>

(the head is like the brain of your file. I'll show you what goes here in a bit!)

</head>

<body>

    (This is where your content goes that you see on your site.)

</body>

</html>

You'll see that all of the <**tags**> have a matching <**/tag**> with a slash. These tags identify the **start** and /**end** of a block of code and tell the browser what to expect inside that block of code. Whenever you have a <tag> there must be a corresponding </tag> or your webpage won't render correctly and you will have errors.

Type the above code structure into your index.html file (without the '(comments)', and then go back to your project folder and double click your index.html file. This will open Index in your browser and you will see a blank webpage.

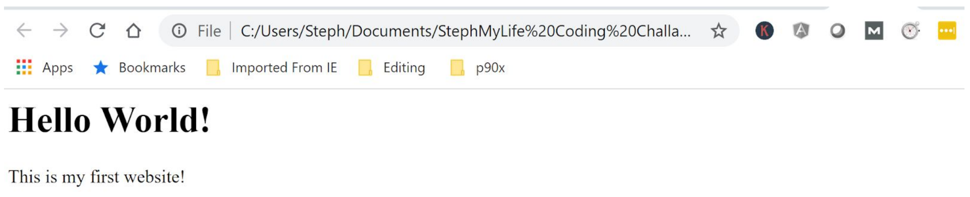
Now, let's put some content in here! (if you have already done the coding challenge, and are familiar with this part, skip to the next page).

Type the following between the <**body> </body>** tags.

<h1> Hello World!</h1>

<p> This is my first website! </p>

Go back to your browser and refresh Index.html (or open it again if you closed it). Now you will see this!



Hello World!

This is my first website!

Troubleshooting tip: If you do not see this, and you see the <tags>, this is because your file has not been saved correctly as HTML. Go back to Visual Studio code, click 'File' -> 'Save As', and save as type HTML with the extension '.html'. If you are not using Visual Studio code, then this is more likely to happen. I would download this application to make things easier for you.

Once your webpage looks like the above, you know your page works and can be rendered by your browser. Remove all of the content within the <body></body> tags and we can start designing your login page.

Designing your page

Think about the items you see on a standard login screen:

- Two textboxes, in which you enter your username and password.
- Next to these textboxes are two text labels which usually say 'username / email' and 'password'.
- A 'Log In' button which you click when you have entered in your credentials.

We will use HTML to create these controls so that the user can enter their details and log in to your website. We are going to create a **HTML Form**, which is used when a page wants to take input from a user and perform an action based on the input. Forms are helpful for login pages, and usually take the following format:

```
<form>
    //Form controls like textboxes and buttons
</form>
```

Between the <body></body> tags, type <form> and press enter. Visual Studio Code will automatically create the closing tab for you.

Next, let's create a label which will sit next to your text boxes to tell the user what they need to type in. Copy or type the following line into your index.html file, between the <form> </form> tags.

```
  <label><b>Username:</b></label>
```

Now, we want to create a textbox next to this, to accept the username. So copy this into the next line:

```
<input type="text" placeholder="Enter Username" id="username" required>
```

Refresh the page, and you can see your label and textbox (Don't worry, we'll style this later using CSS. For now it's just important to have all of the controls on the page).

As you can see, adding a label is very straightforward - you just wrap any text you like in <label></label> tags. You can use these anywhere within the <body> </body> tags and they're super helpful for adding text to your webpages.

For textboxes, there is no **<textbox></textbox>** tag, instead HTML uses **<input></input>** tags for controls that the user can interact with (checkboxes / option buttons etc). There are many different types of input tag, which you can find out more about below. You define which type of input control you would like using something called an **'attribute'**.

Attributes define additional characteristics or properties of a control. You can see, label has no attributes, it's simply <label></label> - that's because this label doesn't do anything apart from display text. Input however, requires the user to input data, so it has the following attributes: ***type, placeholder, id,*** *and* ***required***. Here is what each attribute is for:

- **Type**: This specifies the type of input control we are using. An input control requires some type of interaction from a user. There are 22 different types of input, and here we are going to choose 'text' as we want to have a textbox. To do this we add **'type="text"'** to the label tag.
- **Placeholder**: This is a super helpful attribute, that adds placeholder text to your textbox to tell the user what to enter in here. The text is a light grey and disappears when you start to type in the box. You can remove this attribute if you do not want any placeholder text. Or you can change the text and see this on your page!
- **id:** This gives our control an ID which will allow us to reference the control in our Javascript - which is where the functionality is added. You will be using this ID later, and you will see how important it is. Any element on your page that will provide functionality will have an ID attribute.
- **Required**: Adding the Required attribute to your control means that this element is required before you can submit your form. This means that you don't need to write special checks in JavaScript to see if the user has entered a password, so it saves you time! This is one of the benefits of using a HTML form instead of just putting the controls on the page.

Extra Reading: Click here to read more about label controls, click here to read about input controls, and click here to read more about HTML forms and what they can do!

Task:

Now that we have a label and textbox for our username, I want you to create the fields for your password too, directly below the username fields. See if you can do it without looking at the code below - this link will help you to find the input ‘type’ that you need for the password textbox.

When you’re done, your code should look like this:

***I added a <br/> tag between the username textbox and the password label, which is a HTML line break. This means that the password label will appear on the next line. It’s a very helpful tag for adding space between elements. Try it and see!**

```
<form>

      <label><b>Username:</b></label>

      <input type="text" placeholder="Enter Username" id="username" required>

              <br />

      <label><b>Password: </b></label>

      <input type="password" placeholder="Enter Password" id="password" required>

 </form>
```

As you can see, it’s almost exactly the same - except the type changes to ‘password’ for the password textbox. This means that when you type in the password, it shows as ** and not the actual password text. This is another way that attributes save us time writing code to handle things like this!

Type into each of the textboxes and you can see the difference between input types!

Adding the Login Button

Next, let’s add the login button! Add this code after the password textbox, but before the </form> tag, as this button will be part of your form.

```
<button id="submit" type="submit">Login</button>
```

As you can see, you can add a <button></button> tag for this, with additional attributes that control the functionality of the button. Between the <button></button> tags is the text that will be displayed on the button, you can change this and refresh the page to see the wording change.

You can also put a <br/> tag before the <button> tag to put it on a new line from the password text box.

In this tag, the **‘type’** attribute is “=**submit**”. Adding this attribute to a button control creates a button that automatically submits the information that was entered in the HTML Form.

Refresh your webpage and enter in any credentials, and press the Login button. Nothing happens right? That is because we need to add the code that makes all the magic happen when we click these buttons. We do this using another language, called **Javascript**!

Javascript

Javascript is what we use to make web pages interactive. We can add all sorts of functionality to our web pages by adding Javascript functions. HTML can just tell your browser what controls to show, but it can’t control what these controls do. You can’t add functionality to a button using HTML, so to make your pages interactive you will need to learn how to write JavaScript!

Before we start writing our JavaScript, let’s think about what a login screen does.

- Takes the username and password
- Checks if it’s correct and sends the user to the next page if it is.
- If it’s incorrect, show a message to the user to let them know. Does not send the user to the next page.

We are going to create a login.js file and write the code to implement this functionality.

Task: Can you create this file in VS Code and save it to the right folder? The instructions are below if not!

In Visual Studio Code, create a JavaScript file called login.js, and save it into the ‘js’ folder in your project folder. Make sure you save it as type ‘JavaScript’ so that VS Code knows what language you will be typing in here. Within this file, we are going to write the code for the login page.

Variables, Functions, and Conditional statements

I’m going to introduce you to some new concepts here: variables, functions, and conditional statements. I’ll explain them now, and you can see how they work in the code below. Throughout the coming pages you will be learning more about these three concepts and practising how to write them

Variables

These allow us to store pieces of data in our code. We assign a value to a variable, and then we can pass this variable around our code to use it in our functions. Variable means anything that can vary, so the value of a variable can be changed anytime. A variable must have a unique name, and you can assign a value to a variable using equal to (=) operator when you declare it or before using it.

Consider this piece of JavaScript:

```
var x = 3;

var y =  4;

var z = x + y;
```

Here I am creating three variables. The first two variables are being assigned numerical values, 3 & 4. The third variable is declared next and is using the values of the previous two variables to set its own value. The value of var z will be ‘7’ when the page loads.

Functions

A function is a block of JavaScript code that performs some type of task. It’s good practise to create a different function for each different piece of functionality your page has - this makes your code easier to read and understand. A function allows you to define a block of code, give it a name and then execute it as many times as you want.

Functions often use variables to perform their tasks, as you will see in the code below. Sometimes a variable will just be used by one function, so it will be declared inside the function. However sometimes a variable will be used by many functions, so it will be declared at the top of the JavaScript file, outside of all functions.

One function can be used many times, so they can save a lot of time writing code!

A function is written like this:

```
function task() {
                    //code to perform the task
}
```

Conditional Statements

Conditional statements are used within functions to specify if a block of code should be executed or not. They wrap around a piece of code, and they specify a condition which has to be met for the block of code to be executed.

Think of it like **'If this is true, then do that'** so for our page **'if the username and password is correct, go to the next page'.** If the condition is not met, then the code within the statement will be executed! These are super helpful and are used a lot in software development

And 'if' statement is usually, but not always, followed by an 'else' statement. This contains an alternative block of code to execute if the condition is not met.

They take the following format:

```
if (vegetarian== true) {
    // return veggie menu
}
else {
    // return full menu
}
```

It's ok if you are still not super clear on these concepts, we are going to use them to create the functionality for our login page so you can see how to use them in a real example!

Extra reading: Learn more about variables, functions, conditional statements

Putting these concepts into practice

In our login.js code, we are going to create one **variable**, which will be used to track how many remaining attempts the user has to log in. We will give this variable a value when the page is first loaded, let's say 3.

In our **function** we will take the username and password that the user has entered and check **IF** these match the correct login credentials. We will use a **conditional statement** to check if they do match, and if so we will show them a pop up alert to tell them they have been successful and send a 'true' response to our form to let it know that the user has been successful.

If the credentials do not match, we will reduce our variable value by 1 every time the user gets the credentials wrong. This means changing the variable value within the function.

After three wrong attempts, we will disable all text boxes and the login button - effectively locking the user out of the login page.

Task

On the next page is the code that performs these actions. Read through it and see if you can spot the following:

- Where do we use variables?
- Can you identify our function?
- Can you identify our conditional statements? (There are two)
- Do you notice anything about how we get the value of the username and password? If you look at the HTML can you see how we identify the controls we want to access in the JavaScript?
- Can you see how we disable controls?
- Can you see where we change the value of our variable?
- Can you see how we send popup messages to our user?

Everything is answered below, but I would love for you to see how much of this code you can understand.

Don't forget that right now you are learning a new language - it's not easy to do, so anything that you can recognise and identify is great!

I don't expect anyone new to JavaScript to be able to answer all of these questions, but reading code is a skill and it's good to practice it.

```
var attempt = 3;
function validate() {
    var username = document.getElementById("username").value;
    var password = document.getElementById("password").value;
    // Checking the username and password
    if (username == "admin" && password == "admin123") {
        alert("Login successful");
        return true;
    }
    else {
        attempt--;// Decrementing by one.
        alert("You have left " + attempt + " attempt;");
        // Disabling fields after 3 attempts.
        if (attempt == 0) {
            document.getElementById("username").disabled = true;
            document.getElementById("password").disabled = true;
            document.getElementById("submit").disabled = true;
            return false;
        }
        return false;
    }
}
```

Copy the above code into your login.js file in VS Code, and let's run through what it does.

Step 1:

We have created a variable called 'attempt' which has a value of 3. This means the user has 3 attempts to log into the system.

Step 2:

Below this, we have created a function and given it a name 'Validate'. When you create a function, this is the syntax:

```
function myFunction() {

    // Do some stuff here

}
```

Step 3:

Within the function, the very first thing we do is get the values that the user has entered into our textboxes. We assign these values to variables so that we can use them elsewhere in the code easily. Here is how we do this:

```
    var username = document.getElementById("username").value;

    var password = document.getElementById("password").value;
```

document.getElementById("elementID").value, will return the value that is in the specified element in your HTML. As you can see in the HTML file, the username textbox has an ID of **'username'** and the password textbox has an ID of **'password'**

In these two lines of code, we are getting the value of these two textboxes and assigning them to two variables: username and password.

Now every time we want to find and use the username and password, we can just type **'username'** and not **'document.getElementById("username").value'**. It makes our code easier to type and also easier for someone else to understand!

Step 4:

Next, we create our conditional logic. We want to check that our username and password are correct right? So let's write that in code. The syntax for a conditional statement is:

```
If (condition) {
    // do this thing
}
else {
    // do this other thing
}
```

```
 if (username == "admin" && password == "admin123") {
        alert("Login successful");
        return true;
    }
else {
        attempt--;// Decrementing by one.
        alert("You have " + attempt + " attempts left;");
        // Disabling fields after 3 attempts.
        if (attempt == 0) {
            document.getElementById("username").disabled = true;
            document.getElementById("password").disabled = true;
            document.getElementById("submit").disabled = true;
        }
        return false;
    }
```

Here, we are checking if the username and password match what we want them to be ('admin' and 'admin123'), and if so we are creating an alert to say the login is successful and returning a 'true' value to our HTML form.

If not (‘**else {}**’), we are reducing the value of our attempt variable by 1 (‘**attempt--**’), and showing an alert to show how many attempts the user has left. You can type:

attempt = attempt -1;

To reduce the attempt value by 1, but JavaScript provides a shortcut and allows you to just type **‘attempt--’** instead! It’s much quicker.

Within our ‘else’ statement, we have another **conditional statement**. This conditional statement checks if the number of attempts is 0 (‘**if (attempt == 0)**’), and if so, it locks the user out of the controls by setting them to disabled.

You can see that to disable a control, we use **document.getElementById.disabled** for each control and set this to be true.

We then return a ‘false’ value to our HTML form, telling it not to proceed to the next page.

Read the above code again and see if things make more sense to you. Here are a few things to know:

- To assign a value to a variable you use ‘=’. To check if a variable equals a particular value, use ‘==’.
- An ‘else’ statement can only come after an ‘if’ statement, but not every ‘if’ statement needs to have an ‘else’ statement. You can see the second ‘if’ statement doesn’t have an else statement.

Adding this functionality to our HTML Form

Now that the javascript is in place, refresh your webpage and click the ‘Log In’ Button.

Nothing happens right?

That’s because you need to link your html file with your Javascript. You need to add a line of code to tell your html file to go look for your Javascript file when it starts loading. You do this in the **<head></head>** tags - paste this bit of code between the head tags.

<script src="js/login.js"></script>

This tells your browser to go into the js folder, and find the login.js Javascript file when it loads our HTML page, because our HTML needs to use it to do its job.

Next, go the **<form>** tag, and change it so it looks like this:

```
<form onsubmit="return validate()" method="POST">
```

This tells the browser to run the **validate()** function in the **login**.**js** file when the ‘Log in’ button is clicked (remember we gave the button a ‘submit’ attribute? This means that when this is clicked, the form is being submitted and this calls the ‘validate’ function.

Save this change, refresh your browser and enter in the wrong password. Click the button and you will see the alert to say how many attempts you have left. Enter it in wrong twice more and you will see that all of the fields are disabled. This means that your validate() function is being picked up.

Now refresh the page, and enter in the correct username and password. You will see a message telling you that your login attempt was successful. Next we need to tell our form to send the user to the next page if their login is successful. You do this by adding an ‘action’ attribute to the **<form> </form>** tag.

Adding an action to a HTML Form

The action we want to take if a user is successful is to open a new page. The next page we will create will be called ‘guessing-game.html’. So we will set our action attribute to be the page we want to to open. Update your form tag to add this attribute - so it looks like this:

```
<form action="guessing-game.html" onsubmit="return validate()" method="POST">
```

Create a new HTML file in your main project folder called ‘guessing-game.html’ and save it as type ‘HTML’.

Save all of your files, and refresh index.html in your browser. Enter ‘admin’ as username and ‘admin123’ as password and click ‘Log In’. You will now be directed to your new **guessing-game.html** page!

This means you are logged in and your functionality is working!

Hurrah! You have created your working login page!

If you are having any issues, just go through the above HTML and Javascript files and make sure that they are the same as the above. Don’t forget you can go to the completed project folder to get the working files for comparison too. Here are some things to check:

- Are you referencing your Javascript file in your HTML?
- Do you have the location correct for the javascript file if you have referenced it?
- Have you added the onSubmit and action attributes to your Form tag?

Now let’s get this page looking better…

Styling your login page

Adding Javascript has provided the functionality to your page, but it doesn't look great does it? The controls are kind of all grouped in together to the top left of your page, and it's very basic looking.

This is where CSS comes in! CSS is what makes our web pages look great - it controls how all of the elements on our HTML page will look and makes it really easy to create great looking websites. CSS tells your browser - make this background green, make this button blue, and move all of these controls so they're in the right position.

CSS is so important when you're building a website, and it's really fun to play around with CSS to see what it can do.

In Visual Studio Code, create a new file and save it in your 'css' folder, with the name 'style.css' and the type 'CSS', or 'cascading style sheet'.

Just like the JavaScript file, we need to tell our browser to go look for our **css** file when it loads the page. Otherwise, the styling changes won't be applied.

To do this, go to your Index.html file and paste this bit of code in between the **<head> </head>** tags, just above your script tag:

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

This will pick up your css file when your webpage loads, and apply all of the styling to your webpage next time it is loaded.

In your index.html file, your head section will now look like this:

```
<head>

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

    <script src="js/login.js"></script>

</head>
```

How CSS works

Within a CSS file, we create a series of classes. Each class defines the different styling attributes we want to add to a specific element.

We then apply this class to an element by adding a **‘class’ attribute** to this element in the html. You can apply the same class to many elements, so it makes it easy to have a consistent style throughout your website. Every page on our website will use the same css file, so they will have the same look and feel.

The first thing I want to do here is to get our login controls into the middle of the screen, instead of leaving them bunched up on the left of the screen.

I’m going to do this by placing these HTML elements within another element called a ‘DIV’. A div is a HTML element that is used to group other elements together and apply styling to them. You can’t ‘see’ a div as it’s not a control like a textbox or label, it’s more of a container for elements that you can see.

To put elements in a DIV, you put the following tag around the elements: <div></div>

We’re going to wrap our <form></form> tags in <div></div> tags.

- Put a <div> tag BEFORE the <form> tag in your index.html
- Put a </div> tag AFTER the </form> tag.

Now, instead of applying styling to each element to move them into the middle of the page, I can just apply styling to the div , and this will move everything within it to the middle of the page. It’s really handy!

Now lets style our <div> . Copy or type the following into your style.css :

```css
.main {

    border: 3px solid #f1f1f1;

    background-color: white;

    margin-left: 30%;

    margin-right: 30%;

    margin-top: 10%;

    padding: 20px;

    text-align: center;

  }
```

First, we have set a border which will be 3 pixels thick, and a solid line which will be light grey (#f1f1f1 is a HTML colour code which is used in css. **You can find different codes here**.)

After this, we're setting a margin on the left and right of our div. Setting a margin says 'I want to have this much empty space to the left and right of my div'. Since I want my div to be in the middle, I will put a margin of 30% on the left and right which will center the div.

I also don't want the login box to be right at the top of my page, so I will add a margin of 10% to move the login box away from the top of the page.

Padding is next. While a margin controls the space outside an element, padding controls the space inside the element. I don't want my controls to be stuck right to the edges of my element, so I will add some padding to move them away from my border. I have added 20 pixels to the top, bottom, left and right of my element.

Finally, I want everything in my div to be in the center, so I will add 'text-align: center;' to do this.

Save this class, and now we need to apply this class to the div control so that it can look how we want. To do this, you give your opening div tag a class attribute which is the name of the css class you just created "main".

```
    <div class="main">
```

Go back to the style.css file, and let's change the background colour of the entire page:

```
body {

    font-family: Arial, Helvetica, sans-serif;

    margin: 0;

    background-image: linear-gradient(to bottom right, #667eea, #764ba2);

}
```

This changes the fonts that we'll use on the page, and changes the background to be a nice gradient which flows from left to right. You can swap out these hex colours for other colours and make your own gradient!

Refresh your page. See how different it looks?

Task

Play with this class, and see how changing the numbers will affect your page. Increase and decrease your margin and padding, and remove the text-align property and see what happens.

You can add new properties like changing the background color of the box! Here is a list of different properties: https://www.w3schools.com/cssref/default.asp

Next, let's style our text boxes: Add this class to the css file, below ‘main’.

```css
 .loginInputs {
   width: 60%;
   padding: 12px 20px;
   margin: 8px 0;
   display: inline-block;
   border: 1px solid #ccc;
   box-sizing: border-box;
 }
```

Then add the class ‘loginInputs’ to **both** of your input tags in your HTML.

```html
<input class="loginInputs" type="text" placeholder="Enter Username" id="username"
required>
```

Let’s make the button nicer too! Add this class to your css file

```css
 .loginButton {
   background-color: #764ba2;
   color: white;
   padding: 14px 20px;
   margin: 8px 0;
   border: none;
   cursor: pointer;
   width: 100%;
}
```

```
  button:hover {

    opacity: 0.8;

  }
```

And add the ‘loginButton’ class to your button in the HTML.

```
<button class="loginButton" id="submit" type="submit">Login</button>
```

Refresh the page and see that the button looks much better! We have set the colour of the button using the property ‘background-color’ and used a HTML colour code here. Feel free to change this!

The **button: hover** class is a class in our css that is added to any button and changes the opacity. Hover your mouse over the button to see how this works.

The last thing I am going to do is add an avatar to the page, just so it’s not so plain.

Copy the below classes into your style.css file:

```
.imgcontainer {
    text-align: center;
    margin: 24px 0 12px 0;
}

img.avatar {
    width: 30%;
    border-radius: 50%;
}
```

And copy the following into your index.html file, just under your opening <form> tag.

```
    <div class="imgcontainer">

        <img src="img/login-avatar.jpg" alt="Avatar" class="avatar">

     </div>
```

Refresh the page, and now you have an avatar image for your login page! As you can see from the HTML above, the <img> tag is used to display an image on your page. Our 'img' tag has three attributes:

- **Src**: This tells the browser where to find the image. As you can see, ours is in the img folder and is called avatar.jpg. If you want to change the image, you would put the new image name here.
- **Alt**: this specifies alternative text for the image if a user for some reason cannot view it (because of a slow connection, an error in the src attribute, or if the user uses a screen reader).
- **Class**: This applies the style class you just created in **style.css** to your image.

Task:

Style your login page further to get used to creating and using classes.

- Replace the avatar image.
- Change the colour of the button.
- Change the background colour of the page or of the login box (you can give the <body> tag a class if you want to change the whole page and not just a div or element).
- Change the font style or colour.
- Use this link to find different properties to apply to each class you have already created.

What we have learned so far

- Websites use HTML to render elements, JavaScript to provide functionality, and CSS for styling.
- You need to reference your JavaScript and CSS files in the <head> tags in your HTML file.
- HTML elements have attributes which add characteristics to an element. These are added to the opening tag of an element and take the form attribute="att".
- JavaScript variables are used to store certain information in your browser that is used by your webpage.
- JavaScript functions are blocks of code that make a task happen. They often use variables to do what they need to do.
- Conditional statements help us to make certain tasks inaccessible if a condition is not met. They ensure that the right parts of the code are run at the right time.
- CSS classes are applied to elements by adding the 'class' attribute to the element.
- It is better to use percentages for defining where a control will go on the screen.
- It is better to wrap controls in a <div> container if you want to style multiple controls at once.

If you cannot get anything working, don't forget that there are working versions of these files in the 'completed project' folder. Just open these and compare them with what you have written.

Now it's time to move to your first game: Guessing Game!

Open StephMyLife Beginners Bootcamp Guessing Game to begin.