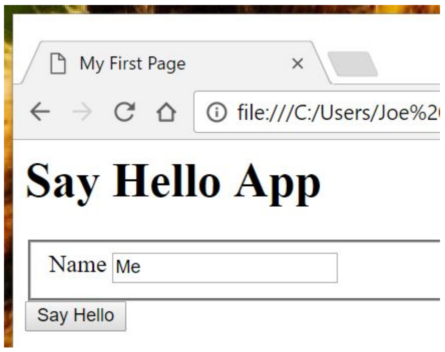
JavaScript Hello World

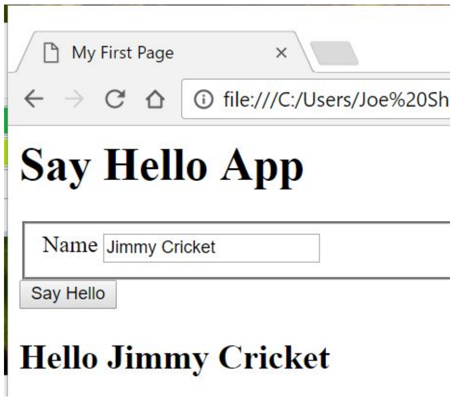
In this project we will do our first bit of JavaScript. JavaScript is the language that drives nearly every dynamic web page you have ever visited. There are a LOT of different libraries and standards for building web pages, but nearly all of them boil down to JavaScript.

We will write a very simple HTML page with a title, text box, output field and a button. When the user clicks on the button the page will print 'hello <name>' to the output field.

The page will look like this, notice that we will be opening this up in a web browser. On the Raspberry Pi I recommend using Chromium.

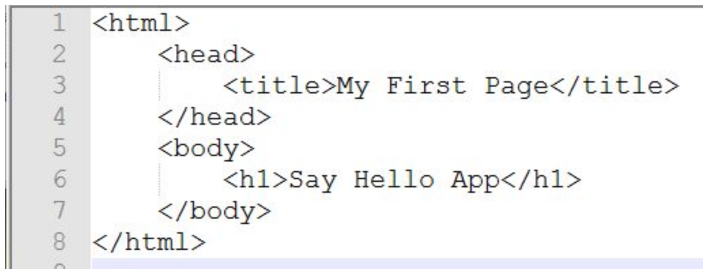


When the user types in a name and clicks on Say Hello it will set the text of the output field as shown.



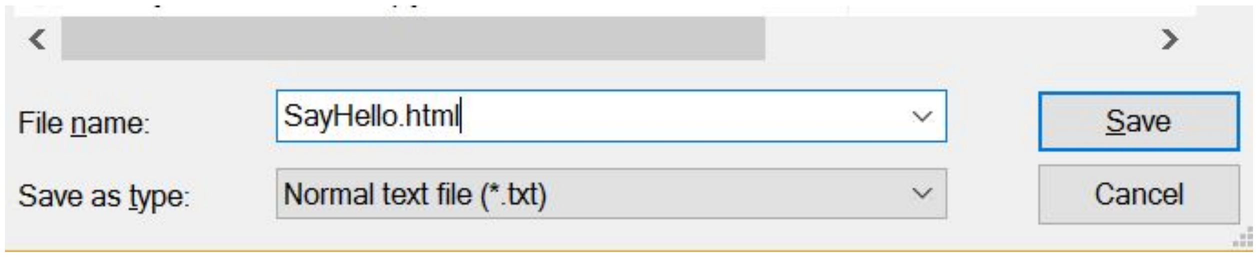
Basic HTML Page

Before we write any JavaScript, we need to write a HTML page to contain it. Open any text editor and type the following into the file.



```
<html>
    <head>
        <title>My First Page</title>
    </head>
    <body>
        <h1>Say Hello App</h1>
    </body>
</html>
```

Save this file into your work directory and give it a filename that ends with **.html**.

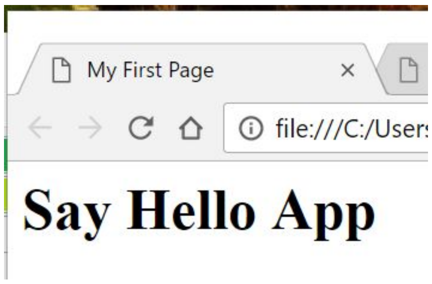


File name: SayHello.html

Save as type: Normal text file (*.txt)

Open the file in your web browser, you can probably just double click on the file in the file explorer and it will default to the web browser. You could also use the CTRL+O shortcut in the browser to open a file dialog.

The page should look like this in the web browser.



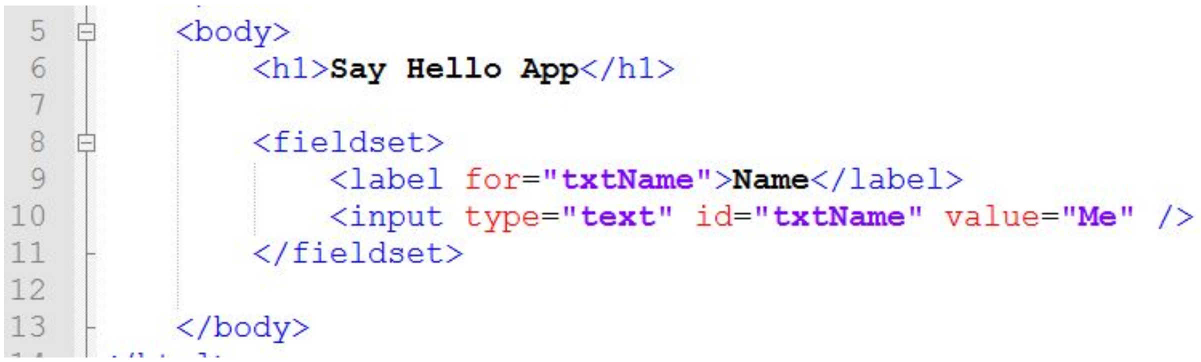
Say Hello App

Notice that the title has been used for the tab, and the heading is within the page itself. This is the difference between the <head> and <body> sections we saw earlier.

The Input Field

We want to capture the name of the user in a text field. We will be using a <fieldset> to contain this form. When we start using stylesheets things like <fieldset> tags will help arrange the page nicely.

Add the input field inside the <body> tag as shown.

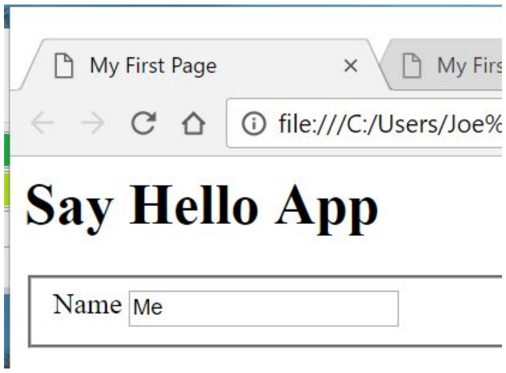


```
    <body>
        <h1>Say Hello App</h1>

        <fieldset>
            <label for="txtName">Name</label>
            <input type="text" id="txtName" value="Me" />
        </fieldset>

    </body>
```

In this code we have set the following

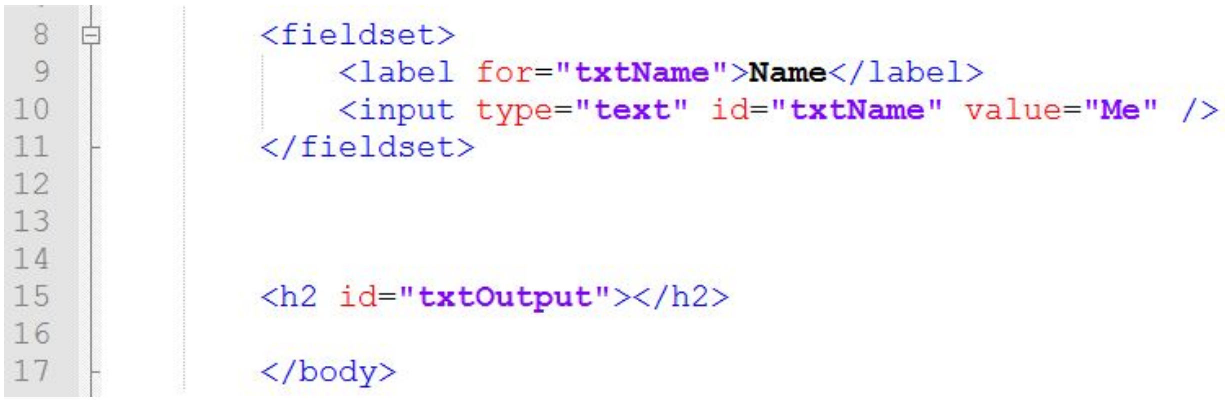
- The **id** of the text field will be used by the JavaScript to reference this text box
- The **value** will be used as a default value when the page is loaded.
- The label is there to give the text field context. I have set the **for** attribute to match the **id** of the input field. This helps people using screen readers.

Before we go further, refresh the page in the browser and it should look like this.



The Output Field

For the output I will be using a heading. Add the following line just after our <fieldset>. Take note that I am leaving a fair bit of space between the <fieldset> and the <h2>. We will put our button in this space next.

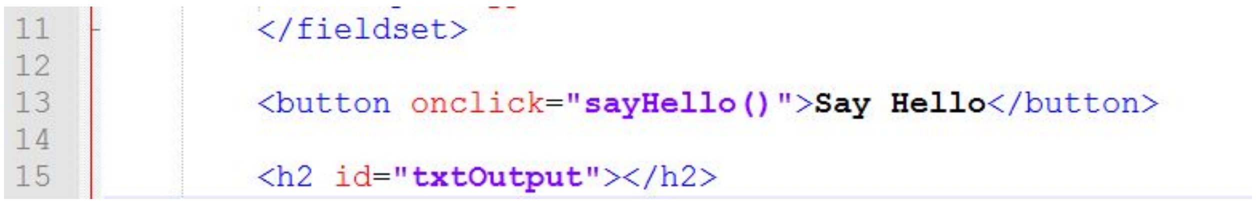


```
    <fieldset>
        <label for="txtName">Name</label>
        <input type="text" id="txtName" value="Me" />
    </fieldset>



    <h2 id="txtOutput"></h2>

    </body>
```

If you refresh the page now, it probably won't look any different. The heading is empty. As before I have given the heading an **id** which will allow me to reach it from the JavaScript later.

The Button

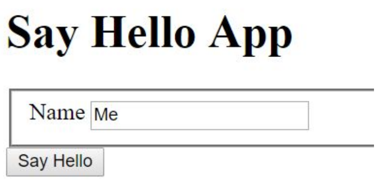
Perhaps the most exciting bit of HTML in our page is the button. It is shown here on line 13



```
11      </fieldset>
12
13      <button onclick="sayHello()">Say Hello</button>
14
15      <h2 id="txtOutput"></h2>
```

Now what is going on here? We have set an **onclick** handler. It refers to a function called **sayHello()** which does not yet exist. We will write this function in the next section.

First just reload the page to confirm that the button is there.

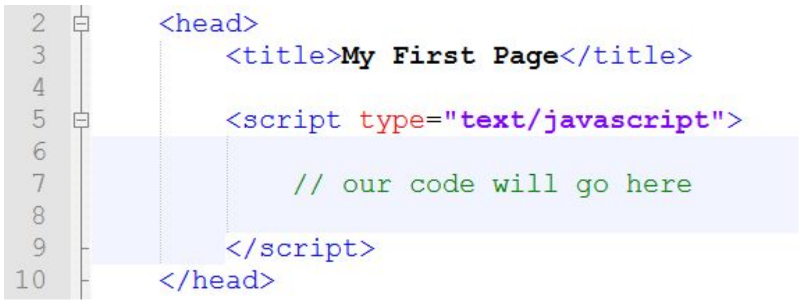


JavaScript Tag

We are now ready to write some JavaScript. But where will it go?

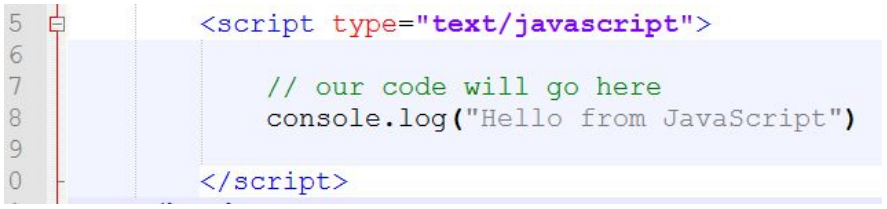
We will add a <script> tag to the <head> section of our HTML page. Larger applications would put the JavaScript in a separate file and reference it in, but for the time being we will embed the script in the page itself.

Add the following <script> tag to the <head> section as shown.



```
    <head>
        <title>My First Page</title>

        <script type="text/javascript">

            // our code will go here

        </script>
    </head>
```

We can add some code just to check that JavaScript is working. Add a **console.log()** call as shown here.

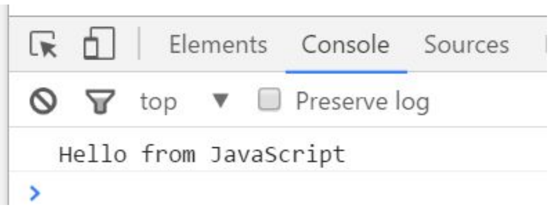


```
        <script type="text/javascript">

            // our code will go here
            console.log("Hello from JavaScript")

        </script>
```

Reload the page...did anything happen? Probably not. The console is a developer function so you will need to open the debug tools in Chromium (or whatever web browser you are using) for this to work. The shortcut key for this is CTRL + SHIFT + I (capital i).

ASK ONE OF THE TEACHERS IF YOU NEED HELP OPENING THE CONSOLE.

With the console open, refresh the page and you should see something like this.



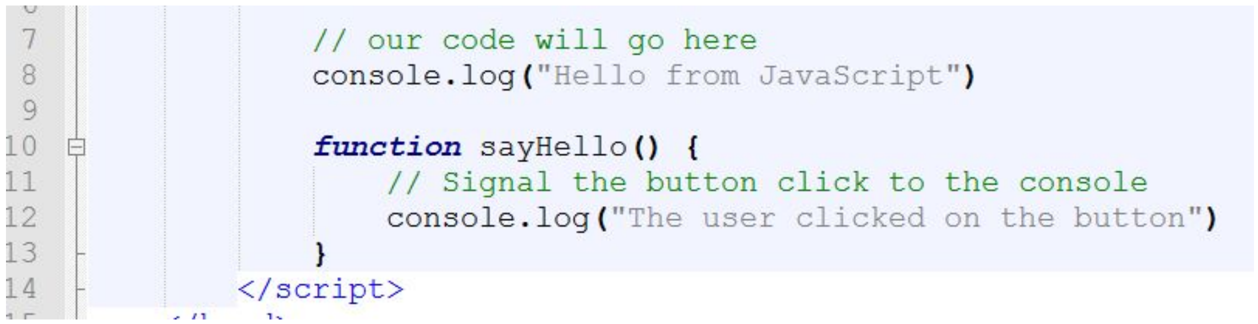
```
Hello from JavaScript
>
```

If this works then you are doing very well!

React to the Button Click

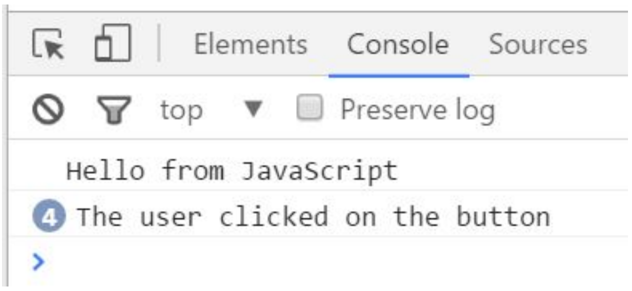
We are now ready to write the function that gets called when the user clicks on the button.

Add the following function to your JavaScript code. Lines 10 to 13 show the new code.



```
 7              // our code will go here
 8              console.log("Hello from JavaScript")
 9
10              function sayHello() {
11                  // Signal the button click to the console
12                  console.log("The user clicked on the button")
13              }
14          </script>
```

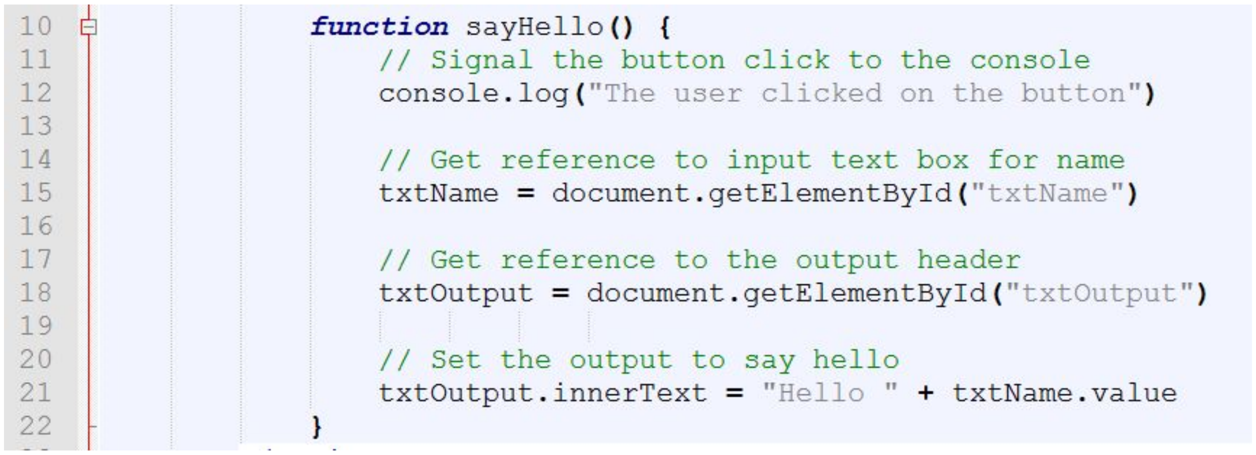
Refresh the page and click the button, you should start to see this message appearing in the console. If you click it multiple times, Chrome cleverly just tells you how many times the same message has been seen. It should look like this in the console.



```
  Hello from JavaScript
4 The user clicked on the button
>
```

Finish the sayHello() Function

We are now ready to interact with our HTML page. Add the following code to your sayHello() function.



```
10  function sayHello() {
11      // Signal the button click to the console
12      console.log("The user clicked on the button")
13
14      // Get reference to input text box for name
15      txtName = document.getElementById("txtName")
16
17      // Get reference to the output header
18      txtOutput = document.getElementById("txtOutput")
19
20      // Set the output to say hello
21      txtOutput.innerText = "Hello " + txtName.value
22  }
```

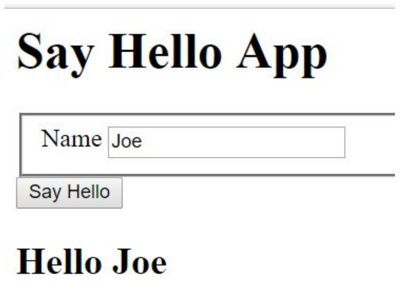
Now there is a lot going on here, here is the explanation of the new lines.

Line 15 - This uses the **id** of the input field to get a **reference** to the input field element.
Line 18 - This uses the **id** of the output heading to get a **reference** to the output

With these references we can now manipulate the page. We can retrieve the value of the input text field and change the contents of the output heading.

Line 21 - This shows us setting the **innerText** of the heading tag to a string we have built using the **value** of the **txtName** field.

Refresh the page and it should now look like this!



Say Hello App

Name Joe

Hello Joe

Experiment with different names, feel free to modify the message and generally play around with the code. Welcome to JavaScript!

CONGRATULATIONS - YOU HAVE FINISHED THE JAVASCRIPT HELLO PAGE