Introduction to JavaScript Fundamental

What is JavaScript?

HTML controls the content of the webpage and CSS controls the styling of the webpage. JavaScript controls the behaviour/interactivity of a webpage based on user inputs such as clicking, hovering, scrolling etc.
Unlike HTML and CSS which are markup and styling language, JavaScript is a programming language and was initially a front-end language. JavaScript has developed over the years and is also now used in both front-end and back-end development for web applications. There are many JavaScript framework, however, in this tutorial we will only look at vanilla JavaScript to understand the fundamentals first before diving into frameworks as all frameworks uses vanilla JavaScript. A final point — JavaScript is not the same as Java.

How to Include JavaScript on a Website?

To write JavaScript in the HTML we would use the `<script></script>` tags and everything within the tags will be seen as JavaScript.

In the past you needed to put the type in our opening script tag for example `<script type="text/javascript"></script>` however, we no longer need to do this for HTML5.

JavaScript can be embedded in the HTML document and there are two places where we can include our JavaScript code. Where we place the JavaScript in our HTML document will be determined by what the JavaScript is trying to do and the order for loading the code.

JavaScript in the `<head></head>` tags — this will load the JavaScript first before the rest of the website content. You should only do this if it is crucial for the JavaScript to be run before the rest of the website/webapp.

JavaScript in the `<body></body>` tags — this will load the JavaScript after the HTML & CSS content load. The beat practice is to place the JavaScript at the bottom of the <body> tags because it allows the website to load everything else first before the JavaScript which will improve the loading of your website.

The placing of the JavaScript within the `<head>` or `<body>` of the HTML file will depend on the importance of loading the JavaScript first before the HTML content or after. For example: We will add JavaScript in the `<body>` of the HTML because it needs to modify the behaviour of an element which needs to be loaded first before the JavaScript.

JavaScript can be added in a separate JavaScript file for example index.js — the .js extension indicates to the browser that the file is a JavaScript file (*and we do not need the* `<script>` *tag within the .js file*). However, we will need to provide a link in the HTML to locate the JavaScript file as we do for any other linked filed such as CSS/Weblinks/HTML pages etc. for example `<script src="index.js"></script>`

Again the link can be placed in the `<head>` or `<body>` of the HTML file depending on the importance of the load for the JavaScript file i.e. before the HTML content or after.

How to Output JavaScript in the web browser?

There are three different ways to output JavaScript on a webpage or web browser and we will explore the three methods within this section.

1. Alert Boxes

When we alert something in the browser, what we are doing is writing some text that will appear in an alert box in the browser.

To use the JavaScript alert function we will use the syntax:

```
alert("");
```

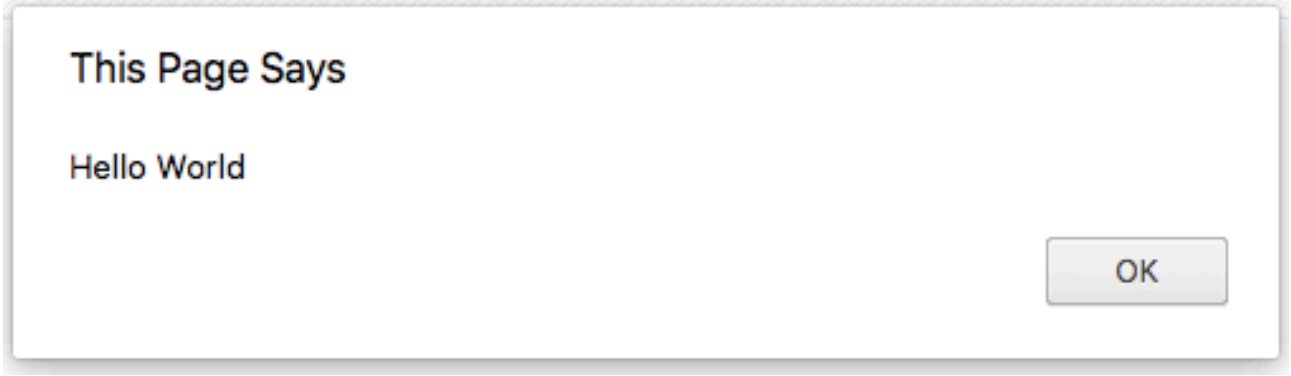
Within the quotation marks we will write our text (a text is a string and a string is a data type that is indicated by single/double quotation marks — we will learn more about data types in the later section). This will generate an Alert Box within the browser - see screenshots below.

index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>Introduction to Javascript</title>
    </head>

    <body>

        <script>
            alert("Hello World");
        </script>
    </body>
</html>
```



2. Writing to the Browser

To write text directly in the browser without an alert box we can use the document.write function to display text within the browser webpage.

To use the JavaScript document.write method we will use the syntax:

```
document.write("");
```

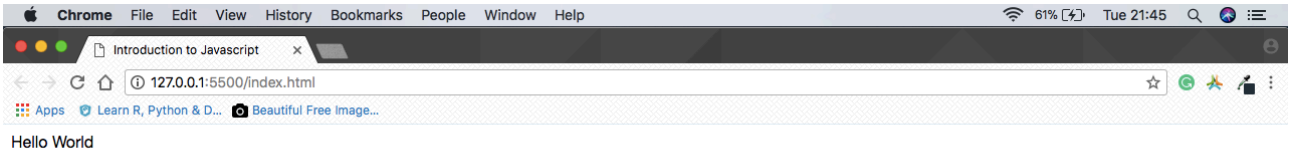
This will write the text string directly in the webpage - see screenshots below.

index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>Introduction to Javascript</title>
    </head>

    <body>

        <script>
            document.write("Hello World");
        </script>
    </body>
</html>
```



The document refers to the html page and we are using the write method to write to the document. We can write to specific elements within our documents by referring to the id of the element using the `getElementById` — for example:

```
<div id="test"></div>
<script>
    document.getElementById("test").innerHTML = "";
</script>
```

The inner HTML will add the text within the opening and closing `<div>` tags — see screenshot below.

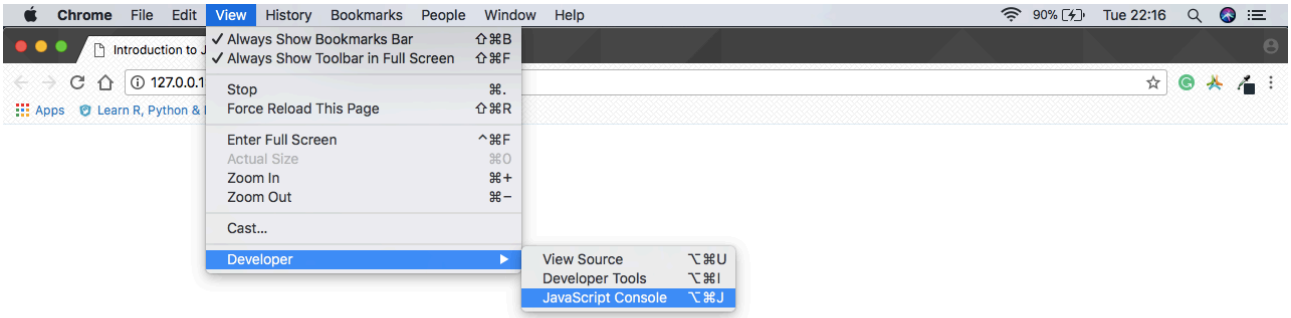
```
Go  Debug  Tasks  Window  Help
index.html — Javascript
index.html ●
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1.0">
6          <meta http-equiv="X-UA-Compatible" content="ie=edge">
7          <title>Introduction to Javascript</title>
8      </head>
9
10     <body>
11         <div id="test"></div>
12
13         <script>
14             document.getElementById("test").innerHTML = "Hello World";
15         </script>
16     </body>
17 </html>
```

3. Logging to the Console

Finally we can output JavaScript code to the JavaScript console of the web browser. To do this we simply use the syntax:

```
Console.log("")
```

This will write to the console of the browser. To view the console in Google Chrome go to `View > Developer > Javascript Console`

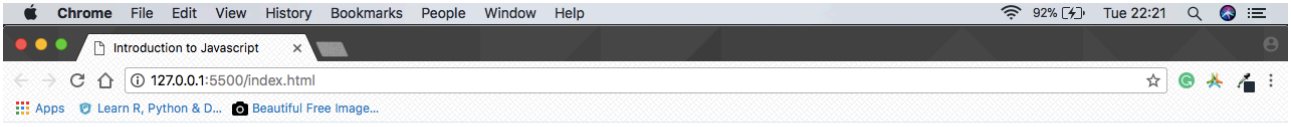


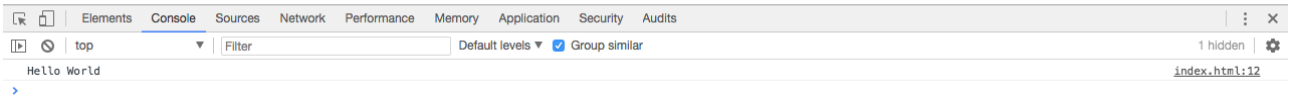
The console will log the string. We can use the console to debug JavaScript and make sure the JavaScript is functioning properly by testing and using the console log to log the results of the JavaScript — see screenshot of the JavaScript console.

index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>Introduction to Javascript</title>
    </head>

    <body>
        <script>
            console.log("Hello World");
        </script>
    </body>
</html>
```





These are the three different methods we can use inside the browser to output JavaScript to the browser (please refer to **Appendix 1 - Output Java.html** as reference for the examples above). We will use the JavaScript console and look at it in more detail in the sections to follow.

How to add Comments in JavaScript?

Adding comments to your code is useful because it allows other developers to read and understand your code. It also helps to write comments as notes for yourself. To write a single line comment in JavaScript we will use double forward slashes — for example:

```
//This is a single line comment.
```

To comment out multi-line we use the same method as CSS — for example:

```
/*Everything within this is commented out.*/
```

How to create JavaScript Variables?

You will use variables all the time in JavaScript and therefore it is important to understand what variables are and how they operate within JavaScript.