

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 best 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 files 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.

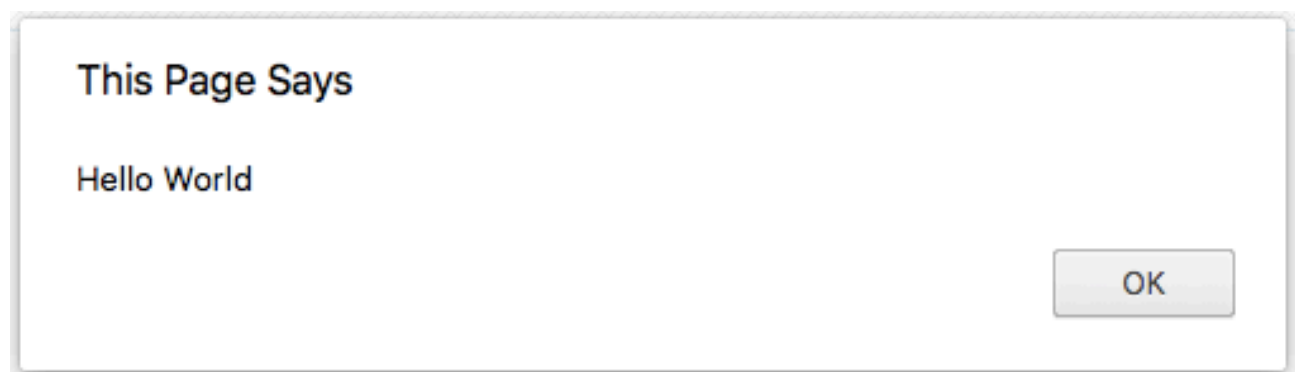
To use the JavaScript alert function we will use the syntax:

```
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.



```
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
12     <script>
13       alert("Hello World");
14     </script>
15   </body>
16 </html>
```



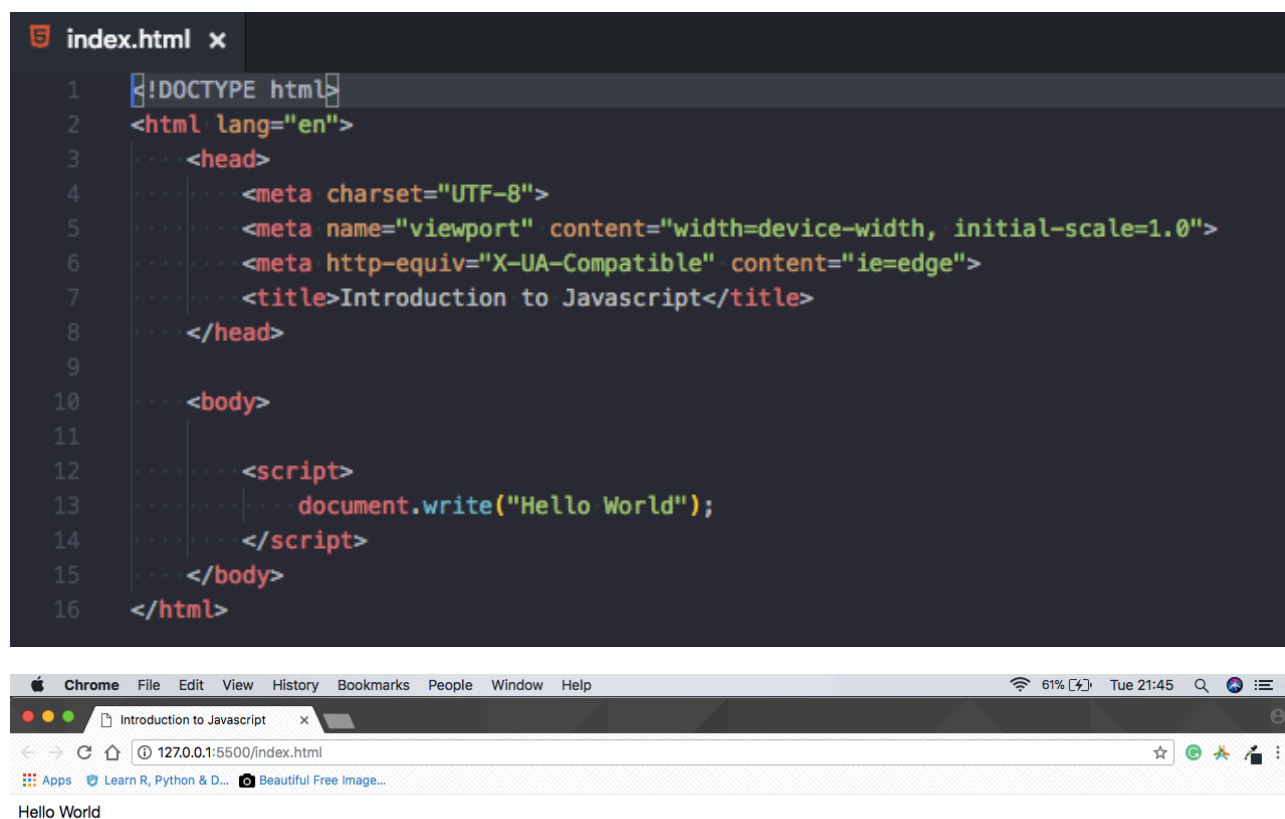
2. Writing to the Browser

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("");
```

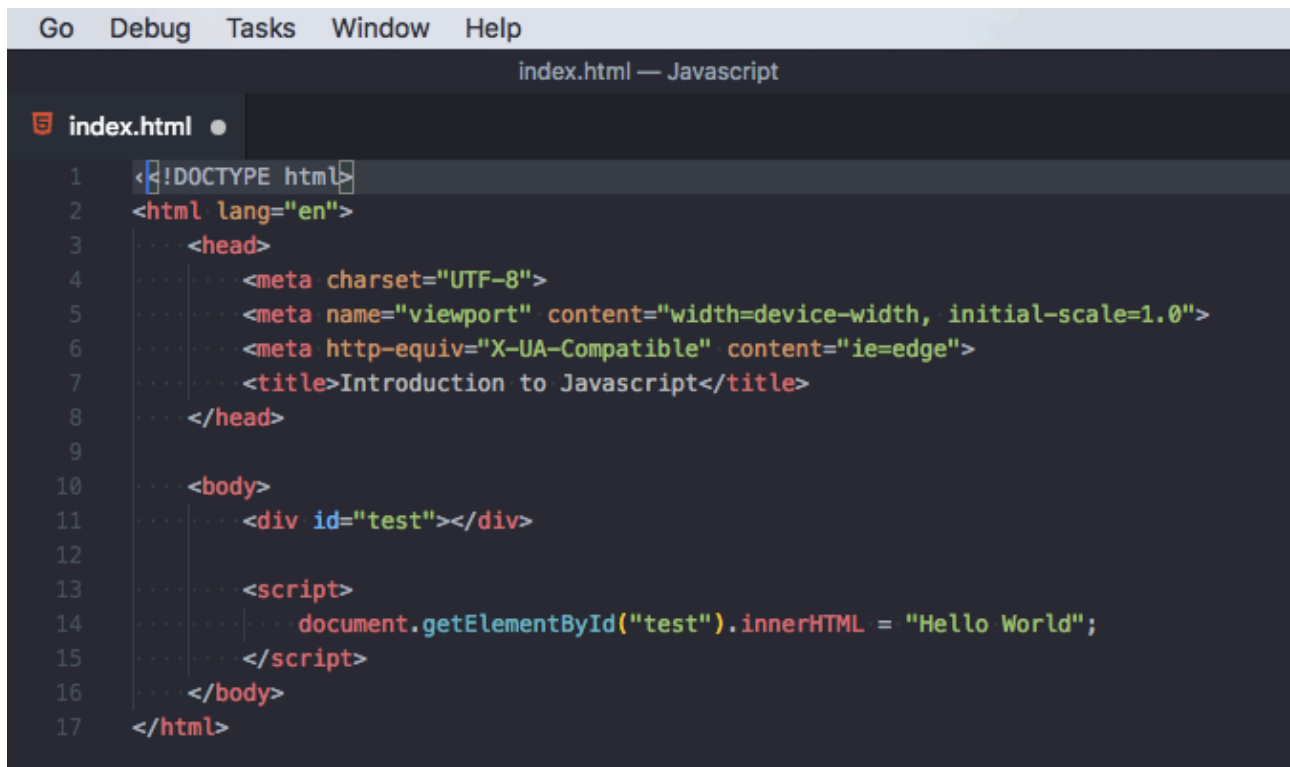
This will write the text string directly in the webpage - see screenshots below.



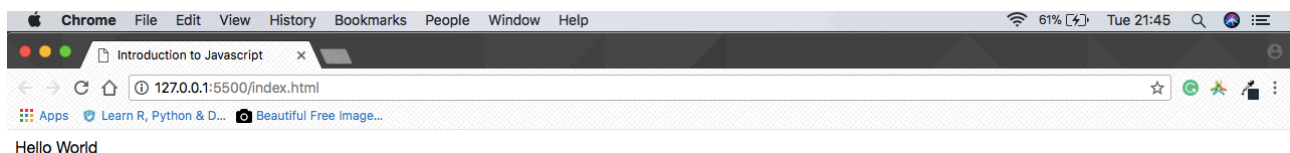
The document refers to the html page and we are using the write method to write to the document. We can write to specific elements within our documents by referring to the id of the element using the `getElementById` — for example:

```
<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>
```

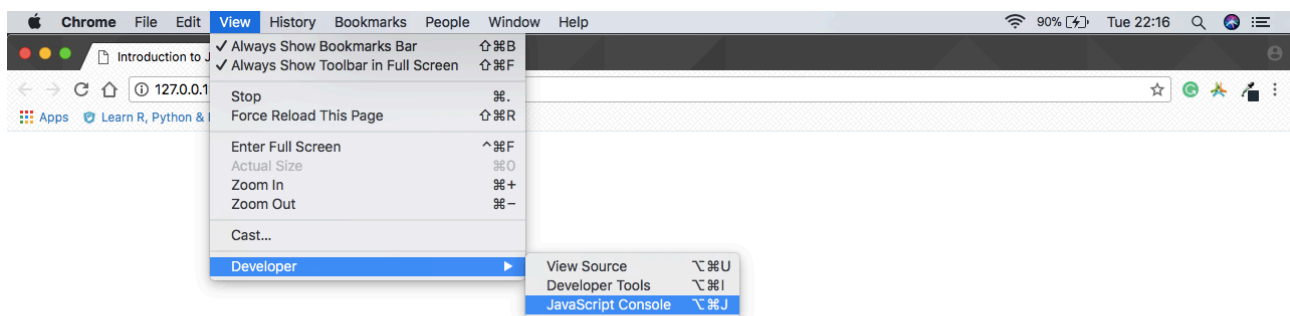


3. Logging to the Console

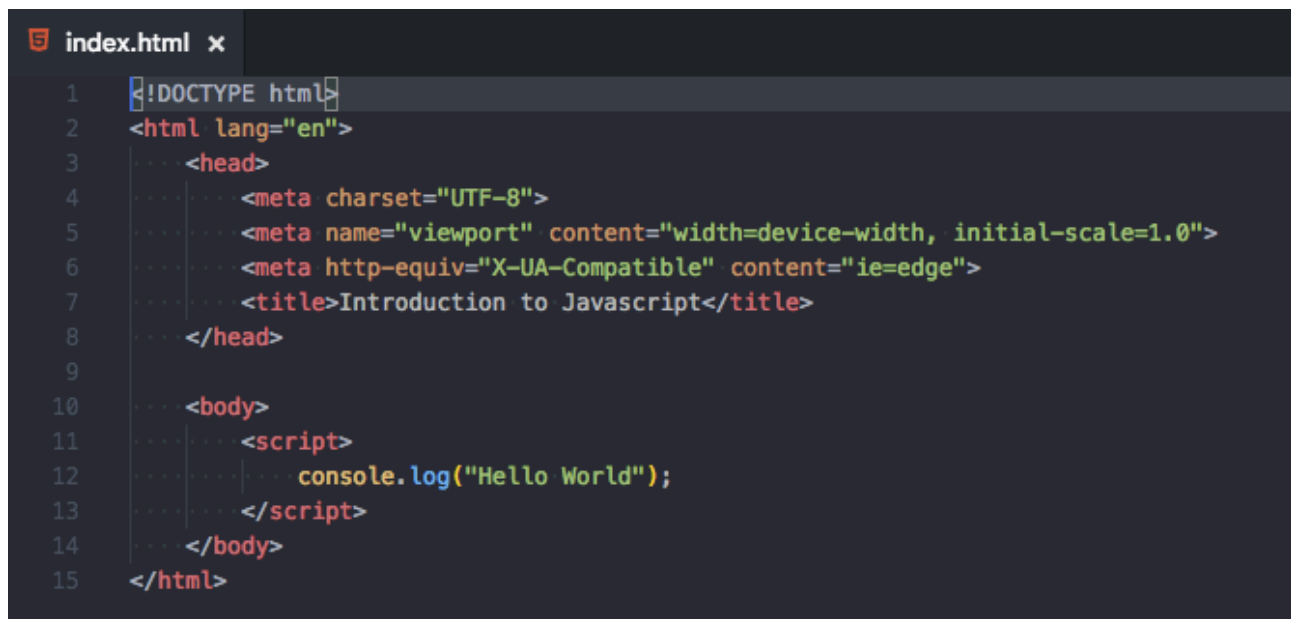
Finally we can output JavaScript code to the JavaScript console of the web browser. To do this we simply use the syntax:

```
Console.log(“”)
```

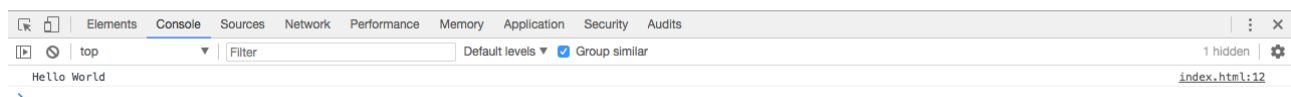
This will write to the console of the browser. To view the console in Google Chrome go to View > Developer > Javascript Console



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.



```
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    <script>
12      console.log("Hello World");
13    </script>
14  </body>
15 </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.*/
```

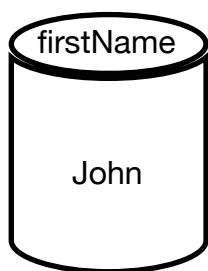
How to create JavaScript Variables?

In this section we will be looking at variables as we will be using variables frequently within JavaScript and therefore it is essential to understand what variables are and how they operate in JavaScript.

A variable is like a container that holds a data. To use a variable we first need to declare a variable in JavaScript in order to use the variable. To declare a variable we will use:

```
var firstName;
```

This will create a variable called firstName; however, this variable does not contain any data/value. We can assign data/value in two ways using the = sign followed by the data:



Method 1 - declare the variable first and then assign the data to the variable:

```
var firstName;  
firstName = "John";
```

Method 2 - declare the variable and assign data at the same time:

```
var firstName = "John";
```

We are storing information in the variables and we can use these values later on in our code by calling/referencing the specific variable name for example:

```
document.write(firstName);
```

Note that we did not need to use var as we only use var when declaring a new variable.

Data Types in JavaScript.

A data type refers to the type of information. There are seven data types:

- Boolean
- Null
- Undefined
- Number
- String
- Symbol
- Object

1. Boolean

Standard across all programming language, booleans are true and false and are often used for conditional statements.

2. Null and Undefined

Undefined means a variable has been declared but has not yet been assigned a value. On the other hand, null is an assignment value. It can be assigned to a variable as a representation of no value.

3. Number

The number data type covers integers and floats. The number data type can handle positive, negative and decimal numbers (*compare this with many other languages that have multiple data type to support different type of numbers*).

4. String

A string represents a grouping of characters. Each element in the String occupies a position in the String. The first element is at index 0, the next at index 1, and so on. The length of a String is the number of elements in it.

To write a string we simply use single or double quotation. Numbers within quotations will be treated as string and cannot be computed using math operators.

5. Arrays

We can create an array of data types within a variable by using the opening and closing square brackets for example:

```
var people = ["Andy", "Bella", "Cathy", 1, 5, 7];
```

Arrays allows for different types of data to be stored in a single variable.

6. Objects

Objects are essentially variables but contain many values (*note an array is an object*). The values are written as name:value pairs within curly braces — for example:

```
var car = {carMake:"BMW", carModel:"4 series",  
colour:"Black", carYear: 2018};
```

The car is the object and objects have properties such as the make, model, year, colour and year. Essentially objects are containers for named values.

Objects are more complicated and we will revisit this in the later chapters.

What are Arithmetic Operators in JavaScript?

Arithmetic Operators allows us to perform basic maths on number data types.

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Percentage

Loop Arithmetic Operators

x++	Adds 1 to the number x. When we loop the first number will be x and then the next number will be x + 1
x--	Subtracts 1 to the number x. When we loop the first number will be x and then the next number will be x - 1
++x	When we loop the first number will be 1+x.
--x	When we loop the first number will be x-1.

Loop examples:

5++ in a loop will = 5, 6, 7, 8 etc.

5-- in a loop will = 5, 4, 3, 2 etc.

++5 in a loop will = 6, 7, 8, 9 etc.

--5 in a loop will = 4, 3, 2, 1 etc.

What are Assignment Operators in JavaScript?

Assignment Operators allows us to assign a value to an existing value for example:

```
var x = 10;  
x = 10 + 5;
```

This will add 5 to the existing value of x. This can be written in an alternative (shorter syntax):

```
var x = 10;  
x += 5;
```

This adds 5 to the existing value of x which is 10, essentially doing the same as the above code but in a shorter syntax form.

We can use the assignment operators with any arithmetic operator such as =, +=, -=, *= etc. and this will assign the new value to the existing variable value. Note this will only work with number data types.

What are Functions in JavaScript?

In Javascript there are two types of functions:

1. Predetermined functions

```
document.write()
```

Document is the object and write is the function. The write function is a predetermined function because it already exists within the JavaScript language and we do not need to create the write function.

2. User-defined functions.

```
function myFunction(){  
    document.write("Hello World");  
}
```

This is a function that we have create ourselves and does not exist within the normal JavaScript language i.e. not predetermined. To execute this function we will write

```
myFunction();
```


To recognise a function we normally see the syntax of the name of the function followed by the opening and closing parenthesis/brackets after the name e.g. `functionName()`

What are Scopes in JavaScript?

Scopes are an important concept to understand within JavaScript. Scopes affect variables and whether they can be used within and outside of functions. There are two types of scopes, Global and Local scopes.

1. Global Scopes

A global scope variable is a variable that is declared outside a function and therefore the variable is outside in the global scope. The variable can be called multiple times outside and inside different functions.

2. Local Scopes

A local scope variable is a variable that is declared within a function and therefore the variable is inside the local scope of the function. This means that the variable can only be used within that specific function and cannot be used outside the function or within another different function.

If you want to create variables that you wish to use in multiple functions, the variable must be outside in the global scope. If you want a variable or data that you want to use within a certain function then the variable can be created within the local scope of the function.

Refer to Appendix 3 - Global and Local Scope.html.

What are Events in JavaScript?

JavaScript events are things that happen to a HTML element i.e. JavaScript code is triggered and interacts/changes the behaviour of a HTML element. Event examples include:

```
onclick - user clicks on a HTML element.  
onchange - An HTML element has changed.  
onload - The browser has finished loading.  
onkeydown - the user pushes a keyboard key.  
onmouseover - the user moves the mouse over an HTML element.  
onmouseout - the user moves the mouse away from an the HTML  
element.
```

There are many events and you should google the different types of JavaScript events to understand what they do and how to write the syntax for the event.

Important Note: unload only works inside specific HTML elements for example in the body tag, image tag and script tag but it will not work in a div tag as an example.

To call an event on a HTML elements we have to write the event in the opening HTML element tag followed by the function we wish to run — for example:

```
<body onload="myFunction()">

    <script>
        function myFunction() {
            alert("Hello World")
        }
    </script>
</body>
```

When everything within the body script has been loaded in the browser this will trigger the event and call on the function to alert the user with an alert message box with the text Hello World.

Objects and Properties in JavaScript?

In the previous section we saw how to create a variable. Variables are essentially an object. The variable `var person = "John";` is an example of a very basic object of person where do not know of any properties.

To create an advanced object by assigning properties we simply use the curly braces — for example:

```
var person = {firstname: "John", lastname: "Doe", age: 35};
```

Another way of writing the above that is more readable is:

```
var person = {
    firstname: "John",
    lastname: "Doe",
    age: 35
};
```

To access the properties from an object within our JavaScript functions we would reference the object followed by the property name within that object — for example:

```
Document.write(person.firstname);
```

Objects and Methods in JavaScript?

A method is simply a function within an object. Taking the above example we can create a method within the object that will write the full name of the object person.

```
var person = {
  firstname: "John",
  lastname: "Doe",
  age: 35,
  fullName: function(){
    document.write(this.firstname + " " + this.lastname);
  }
};
```

When creating the method in the object we do not need to write the name of the function after the `function()` keyword this is because we have already named the function within the object at the beginning i.e. `fullName`.

The keyword `this` refers to the name of the object i.e. this is the same as saying `person`. — because the method is within the object JavaScript understands which object `this` is referring to.

To call the method we can simply write the object followed by the method — example:

```
person.fullName()
```

Objects allows us to record different information/data using properties and this is all located within one place. However, we can call on these properties & methods as many times as we want throughout our website using a simple short code. Therefore, if we would want to change a data of an object, we simply change it in the Object once and it will automatically apply the changes everywhere that makes reference to the object. This makes the code flexible and easy to update/change.

The preferred way of writing the above method is to use the `return` keyword within the method rather than `document.write` keyword. This will return the value but will not show in the browser if the method was called upon. Instead we would use the `document.write` function to call the method — see syntax below:

```
var person = {
  firstname: "John",
  lastname: "Doe",
  age: 35,
  fullName: function(){
    return this.firstname + " " + this.lastname;
  }
};

document.write(person.fullName());
```