Introduction to JavaScript

Some may argue that JavaScript (JS) is one of the most important programming languages out there. It is certainly the language of the Internet. You need to use JS for anything dynamic in your web browser. Closing a popup by clicking a button? Logging into an app? Displaying content endlessly on your Facebook feed? All of this is achieved with the magic of JavaScript.

Are JavaScript and Java the same?

No, they are two completely different computer languages. Only their names are similar.

What do I need to run JavaScript?

JavaScript support is built right into all the major web browsers, including Internet Explorer, Firefox and Safari. Provided that the visitors to your site are using web browsers that support JavaScript (most do) and have JavaScript enabled (it is by default), then your JavaScript will run when they visit the page.

Can I use HTML instead of JavaScript?

No. HTML and JavaScript are two completely different things. HTML is a markup language designed for defining static web page content. JavaScript is a programming language designed for performing dynamic tasks.
Sometimes the distinction is confusing because JavaScript code can go in the same file as HTML.

Does the JavaScript go in the same file as HTML?

It can, but your scripts will be more easily reused on multiple pages of your site if you place them in separate files. (Using a .JS extension helps identify them as Javascript.) You then just link the JavaScript to your HTML by inserting a `<script>` tag. The same JavaScript can then be added to several pages just by adding the appropriate tag into each of the pages to set up the link.

If you are interested in the history of JavaScript, it's intentions and where the language comes from, here is a guide to it's origin.

In the next section we will review some of the core fundamentals of JavaScript, before we really start playing around with writing code.

Learning objectives for this session:

- Understand variables
- Use logical operators

Important resources

The JavaScript bible for every developer is the Mozilla Developer Network (MDN), the guys who originally invented JavaScript actually work there. It is extremely rich in detail but sometimes a bit hard to understand, at least for beginners. MDN

The W3school is equally rich in resources and sometimes a bit easier to understand. (W3school). W3school

ECMAScript

ECMAScript (ES) is a scripting language specification. A set of rules how a language should be scripted. JavaScript is an implementation of these rules. You will come across different standards of ECMAScript, the newest being ES7. This means that there are different versions of JavaScript out there, following these different standards. They are always backwards compatible, meaning that JavaScript following ES7 standards will work with JavaScript written in ES5 standards. However browsers using an old implementation of ES, like ES5 will not necessarily understand JavaScript written in ES6.

Since there is still a large segment of browsers that can only understand ES5 we will rely on ES5 standards for the majority of this course. However we will introduce you to the newer ES7/ ES6 standards when relevant.

Running JavaScript code

Browser Console

To test JavaScript code you can rely on your console in the browser. How to open the console in Chrome and Firefox. Once you the developer tools, to open the console just click the console tab, now you can just start by typing JavaScript code in there, or copy and paste it.

More on the Chrome and Firefox console.

Repl.it

We can also use repl.it to test your JavaScript snippet. You can read and run the code side by side.

NodeJS

While the console is handy to test out some JavaScript, it isn't really useful when you are writing large functions. Also, as soon as you reload the page everything that was saved in the console will be gone. It is therefore a lot better to write larger JavaScript functions in your editor and run them with Node.js. All you need to do is type:

```
node file.js # file.js (being the file in which the JS code is saved)
```

Commenting in JavaScript

Comments are important to help you and other readers of your code to understand it. When writing code you may have some complex logic that is confusing, this is the perfect opportunity to include some comments in the code that will explain what is going on. Not only will this help you remember it later on, but if someone else views your code, they will also be able to understand the code (hopefully)!

Another great thing about comments is the ability for comments to remove bits of code from execution when you are debugging your scripts. There are two types of comments in JavaScript: single line comments and multi-line comments.

```
// introduce a one line comment

/*
allows you to comment
multiple lines
*/
```

Single Line Comments:

Single line comments start with //. Any text between // and the end of the line will be ignored by JavaScript (will not be executed).

Multi-line Comments

Multi-line comments start with `/*` and end with `*/`. Any text between `/*` and `*/` will be ignored by JavaScript.

Variables

Variables are containers for storing data values.

```
var x = 3;
var y = 5;
var z = x + y; // z = 8
```

Variables can be updated:

```
var a = 9;
a = 12; // a is now 12 not 9
```

You can place data into these containers and then refer to the data simply by naming the container or variable.

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the keywords: var, let or const.

You can also declare multiple variables with the same var keyword as follows –

Storing a value in a variable is called variable initialization. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

For instance, you might create a variable named money and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

Note – Use the var keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.

JavaScript is untyped language. This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

Naming Conventions

- When naming variables and functions, be as descriptive as possible so that another person reading your code can understand what the variable is used for. `abc` is not a good name for a variable, no one will understand what stands for. `firstName` is lot more descriptive.
- We do this to enable code reviews to focus on more important issues than arguing over syntax and naming standards.
- It can also provide meaningful data to be used in project handovers which require submission of program source code and all relevant documentation.
- When naming your variables use camel case (`iAmCamelCaseBecauseEveryNewWordStartsWithACapitalLetter`) to combine several words.
- By convention the first letter of a variable is always a lower case letter.
- Variable names can contain letters, digits, underscore and dollar signs as well.
- They can start with a letter, _ or $ but not with a number.
- Names are case sensitive.
- Certain reserved words cannot be used (JavaScript, document etc.)

Data Types

In JavaScript there are 7 different data types:

- `undefined` - has no value assigned
- `null` - has exactly one value the value `null`
- Booleans - has a value of either `true` or `false`
- number - any number e.g. 55, -99.87
- string - any string e.g. “I’m a bit of text”
- symbol - a new type, we don’t need to worry about right now
- objects - a collection of properties, that are assembled with `{ }` brackets

To determine what kind of value a data type is you can use the `typeof` operator

```
typeof 99; // number

typeof true; // boolean

typeof {}; // object
```

Note that if you type:

```
typeof [1, 2, 3]; // object
```

Arrays are also objects, to determine the type of an array:

```
Array.isArray([1, 2, 3]); // true
```

`NaN` which stands for Not a Number is returned if you try to do mathematical calculations which don't make sense. Like multiplying a string with a number. To determine if a variable is `NaN` you can use `isNaN(variable)`

Operators

Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that x = 3, y = 6;

`!` (NOT) returns `true` for `false` values and `false` for `true` values;

```
!(x = y); // true
```

`&&` (AND) value on the right is only evaluated if the value on the left is `true`

```
x < 4 && y > 2; // true
```

`||` (OR) value on the right is only evaluated if the value on the left is `false`

```
x == 5 || y == 5; // false
```

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

`==` (equality) and `!=` (inequality) check if the items on either side match.

```
1 == 1; //true
1 == '1'; // true
1 == true; // true
```

`===` checks if they strictly match

```
1 === 1; // true
1 === '1'; // false
1 === true; // false
```

`<` (less than) `>` (greater than) `<=` (less than or equal) `>=` (greater than or equal)

```
4 < 5; // true
4 <= 5; // true
5 > 5; // false
5 >= 5; // true
```

Conditional Operators

```
condition ? value1 : value2;
```

if the condition is `true` `value1` will be evaluated, if the condition is `false` `value2` will be evaluated.

Exercise

Given that `x = 8` and `y = 3`. Answer the following:

Make a g.js file, copy the code below into the file, look at the operators and consider if these statements are true or false, to the right of each statement give your evaluation i.e. // => true;

```
x == y
x && y < 10
x || y =< 8
y === '3'
!(x == 5)
```