Module 3 – Vanilla JavaScript: Types, Functions and Variables

# Lab Objectives

Become familiar with the declaration and use of variables

Create functions that utilise variables

Understand when to use const and let

Variable Types

JavaScript has very few main types of variable, these are Numbers, Strings, Booleans, Objects, Null and Undefined. We will now define each and show their examples.

In JavaScript we don’t have to be as specific as other languages when we are defining variables as JavaScript automatically decides what type each variable is. In the past the var keyword has been the industry standard for defining variables, however more recently a new version of JavaScript known as es6 was released and changed this to let.

In this Lab we will use a setup similar to the previous Lab, creating a basic web page linked to a separate JavaScript file as shown below, the main difference here is that when we want to be able to view a value we will use console.log rather than window.alert. For this lab all changes we make will be in the JavaScript file.

variables.html

<html>

<body>

<p>First Paragraph</p>

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

<p>Second Paragraph</p>

</body>

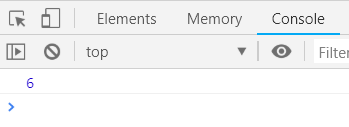
</html>

variables.js

console.log('Hello, World!');

To view the output of console.log in most browsers we must open the developer tools, this is commonly done using the f12 key or can be found in the menu, before navigating to the console tab where the result of your JavaScript should be found.

We will now start defining variables we can use in our code. The most simple type to use is a number, so we can simply type let numberOne = 1; after we have done this we can now use the number 1 anywhere in our code by typing numberOne. We can do basic maths with these variables as shown below:



let numberOne = 1;

let numberTwo = 2;

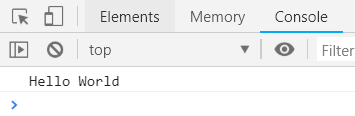
let numberThree = 3;

console.log( numberOne + numberTwo + numberThree );

Task: Try doing some other basic calculations using the other mathematical operators, Subtraction is done with -, Multiplication is done with \*, and Division is done with /.

We will now define some strings, these are done in a very similar fashion to numbers however must have single quotation marks around them, for example let helloWorldString = ‘Hello, World!’;

An example of string usage is shown below:



let stringOne = 'Hello';

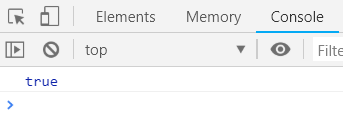
let stringTwo = 'World';

console.log(stringOne + ' ' + stringTwo);

Task: Try constructing a third string from stringOne and stringTwo and printing this out rather than all 3 strings in the console log.

We will now look at Booleans, Booleans can only have 2 values, true or false, when we define these we simply define them as true or false. For example let exampleBool = true;

An example is shown below:



let exampleBoolean = true;

console.log(exampleBoolean);

Objects are the most complex type in JavaScript. They are constructed using key-value pairs and can contain many other variables within them. The process to define a variable to shown below:

let exampleCarObject = {

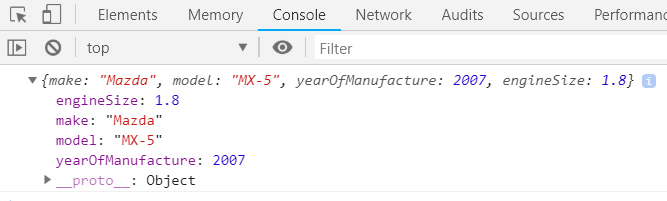
make: 'Mazda',

model: 'MX-5',

yearOfManufacture: 2007,

engineSize: 1.8

};



Each object is wrapped in curly braces, and has a value key on the left hand side of each value, there is then a colon which separates the key from the value, and on the right hand side is the value itself. This value can be of any type.

We can access the individual properties within an object by using the object name, followed by a full stop, then the name of the property you wish to access. An example of this is show below.

let exampleCarObject = {

make: 'Mazda',

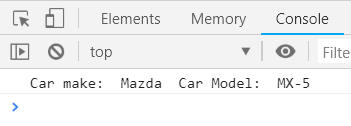
model: 'MX-5',

yearOfManufacture: 2007,

engineSize: 1.8

};

console.log('Car make: ', exampleCarObject.make, ' Car Model: ', exampleCarObject.model);



Task: Create an Object which describes a dog, and print these out in a series of correctly formatted console log statements

Functions

Functions can be passed, returned and stored in the same way as variables, they typically take variables as an input and manipulate these values in some way.

An example declaration of a function which takes two numbers, adds them together and returns the result is shown below:

let numberFive = 5;

let numberSeven = 7;

let result;

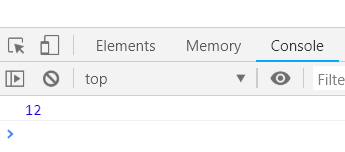
function add(numOne, numTwo) {

return numOne + numTwo;

}

result = add(numberFive, numberSeven);

console.log(result);



**Task:** Make and use functions which can subtract, multiply and divide two numbers

A function does not have to take any arguments, and does not have to have a return type as it will simply return nothing. It can simply change values without returning a new one as shown below:

let exampleNumber = 5;

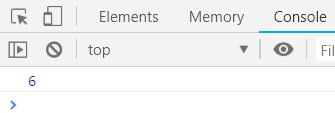
function addOneToExampleNumber() {

exampleNumber = exampleNumber + 1;

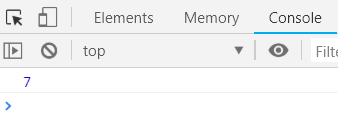
}

addOneToExampleNumber();

console.log(exampleNumber);



There is a newer and more concise way of writing functions, it is not necessary however can reduce the amount of required code. They are known as **fat arrow functions**. An example that adds two numbers together the same way as the example above is shown below:



let numberOne = 2;

let numberTwo = 5;

let add = (numOne, numTwo) => numOne + numTwo;

console.log(add(numberOne, numberTwo));

These functions are defined the same way as variables, where after the equals sign any required arguments are placed in brackets, followed by **=>** and then the code you wish the function to execute, curly braces are not required, however can be used around this.

**Task:** create a fat arrow function which does not take any arguments, or return a value, simply edits a variable as done above in a standard function.

You can also define variables in JavaScript using the **const** keyword which does not allow the variable to be changed, using a line such as **const numberOne = 1;**

**Task:** Think of a reason why you may need to use this feature