Back-end Web Development

February 4th, 2020

Dorset College Dublin

JavaScript

What we will cover today....

- **Refresher**
  - Primitive data types
  - JS specifics
  - Declaring a variable
  - Data type conversion
  - Operators
- **var, let, const**
- **Arrays**
  - Creating an array
  - Accessing values
  - Changing values

- **Control Flow**
  - Logical operators
  - Falsy values
  - Equality
  - Relational Expressions
  - Variable Hoisting
  - not defined, undefined, null

Refresher

JavaScript

object-oriented computer programming language commonly used to create **interactive effects** within web browsers.

- JS code can be **referenced** in HTML or **embedded** directly into it
- JS is **dynamically typed**(loosely typed) which means data-types are bound to **values** not variables *eg. var x = 5;*
- JS is **case-sensitive** so name and Name are 2 different variables
- Comments are written in:
  // Single Line
  /* Multiline */

Primitive Data Types

Numeric Data Type

This data type holds numerical values such as **numbers** and **decimals**.

The value can be **positive** OR **negative**.

String Data Type

This data type consists of **letters** and other **characters**.

The value should be enclosed within a pair of quotes. They can be either **single** or **double quotes**.

Boolean Data Type

This data type holds only 2 values, **true** or **false**.

Think of it in terms of a light switch, is it **ON** or **OFF**.

JavaScript Specifics

- JS **functionality** provided by a set of given objects **built-in** as part of the core of JS language:
  - Math
  - String
  - Number
  - Boolean
  - Date
  - Array

- JS doesn't support classes, uses **Objects** instead
- Objects
  - Have **properties** (characteristics) and **methods**
  - Use the **dot (member) operator** to access object's properties and methods

eg. document.write("Hello");

Declaring a Variable

Unassigned Variable

- Before you can use a variable you need to assign it by giving it a name:

```
var width;
var firstName, lastName;
var isTrue;
var greeting;
```

- **var** is the variable keyword
- You have to **declare** the variable name (also known as an **identifier**)

Assigned Variable

- Once you declare a variable, you can specify what information you want that variable to store:

```
var width = 5;
var isTrue = true;
var greeting = “hello”;
```

- The variable name should describe the **kind of data** it holds
- The equals sign is an **assignment operator**
- The value is **undefined** until you assign a value to the variable

Data Type Conversion

Since JavaScript is **dynamically** typed, we do not have to declare the date-type when declaring variables. Which means we can change the data type of the variable.

```
Eg. var myNumber = 6;
    myNumber = “Six”;
```

The **concatenation operator (+)** will automatically convert numbers to strings.

To convert strings to numbers you must use **parseInt()** and **parseFloat()** methods otherwise you will get a **NaN (Not a Number)** value.

The **String** object has **defined methods** such as length,toUpperCase(), indexOf(), slice, substring(), replace(), toLowerCase(), concat() and more...

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	3 + 11	14
-	Subtraction	9 - 4	5
*	Multiplication	3 * 4	12
/	Division	21 / 7	3
%	Modulus (*remainder after division*)	21 % 8	5
++	Increment	a = 5; ++a	(a *equals*) 6
--	Decrement	a = 5; --a	(a *equals*) 4

Assignment Operators

Operator	Example	Equivalent Arithmetic Operators	Resulting x
=	x=5	x=5	5
+=	x+=5	x=x+5	15
-=	x-=5	x=x-5	5
*=	x*=5	x=x*5	50
/=	x/=5	x=x/5	2
%=	x%=5	x=x%5	0

Comparison Operators for logical statements

Operator	Description	Example	Result
==	Is equal to (value only)	x==8	false
		x==10	true
===	Both value and type are equal	x===10	true
		x==="10"	false
!=	Is not equal	x!=5	true
!==	Both value and type are not equal	x!=="10"	true
		x!==10	false
>	Is greater than	x>5	true
>=	Is greater than or equal to	x>=10	true
<	Is less than	x<5	false
<=	Is less than or equal to	x<=10	true

Logical Operators

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true

var, let and const

---

var

The var statement declares a variable which holds an **undefined(empty)** value until assigned. It defines a global variable regardless of the block scope.

let

Declares a **block scope local** variable.

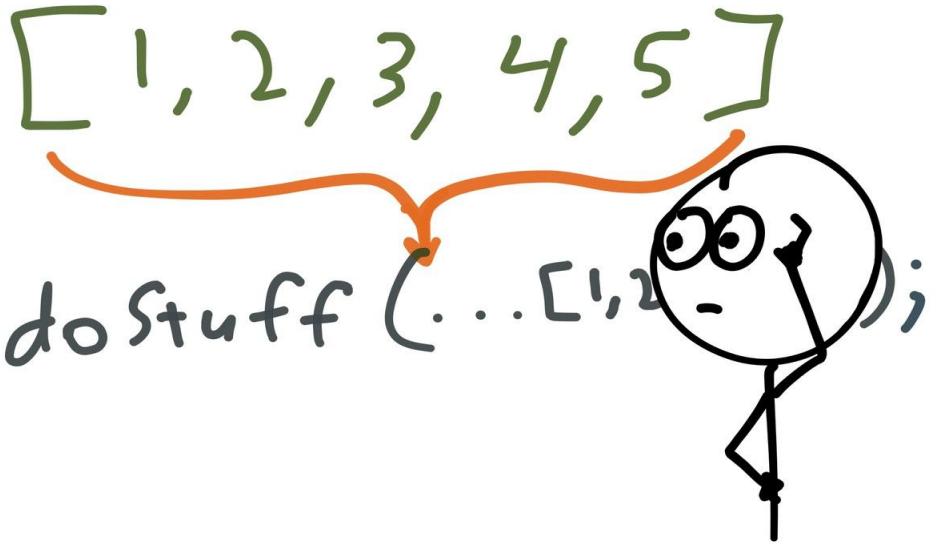
const

Block scoped similar to let however the **value cannot be changed** or redeclared.

Arrays

An Array is a special type of variable that can store **a list** of values.

You should consider using arrays when working with **a list** or a set of related values.



Creating an Array

```
1  var colors;
2  colors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'];
3
4  console.log(colors);
```

- You can create an array and give it a name just like you would with any other variable (using the **var** keyword)
- The values assigned to an array lay inside the **square brackets** and each value is separated by a **comma**
- You can store **different data types** in an array such as ***strings***, ***numbers*** or ***booleans***
- This technique for creating an array is known as an **array literal**
- You can also place each array value on a **new line**

Accessing Values in Arrays

```
colors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'];

console.log(colors);
console.log(colors[0][1][2][3][4][5][6]);
```

- Array values can be accessed as part of a numbered list. **The list starts from zero** (not one)
- Each item in an array is automatically given a number called an **index**
- Remember that **index values start from 0** if you want to access the first item!
- To access a specific item, the **name of the array** must be specified with the array index as shown above
- Each array has a **length property** which holds the number of all items of that array

Accessing and Changing values in an Array

You can change the value of an array by accessing it's index and **adding a value** as shown on the right.

There are 2 colours specified with the third one being **empty**, we access that item's index which is **[2]** and add a string value.

You can see the value then printed in the terminal.

```
1  var colors;
2  colors = ['red', 'orange', ''];
3
4  colors[2] = 'yellow';
5
6  console.log(colors);
```

Problems >_ gitpod /workspace/fewd_arrays ×

```
gitpod /workspace/fewd_arrays $ node app.js
[ 'red', 'orange', 'yellow' ]
gitpod /workspace/fewd_arrays $
```

Control Flow

The **order** in which the computer executes statements in a script.

If/Else statements make binary decisions which execute the code based on **conditions**.

- All conditions are evaluated to be **truthy** or **falsy**
- We can use an **else if** to add more conditional statements to if/else statements
- **Switch statements** can be used to achieve the same result as if/else statements

Continued...

- The **ternary operator (?)** and a **colon (:)** allow us to refactor simple if/else statements
- **Comparison operators**, including <, >, <=, and >= can compare two variables or values.
- After two values are compared, the conditional statement evaluates to **true** or **false**.

Control Flow - Logical Operators

- The **&&** logical operator checks if **both sides** of a condition are truthy
- The **||** logical operator checks if **either side** is truthy
- The logical operator **!==** checks if the two sides are **not equal**.
- An **exclamation mark (!)** switches the **truthiness / falsiness** of the value of a variable.
- One **equals symbol (=)** is used to **assign a value** to a variable.
- **Three equals symbols (===)** are used to check if **two variables** are equal to each other.

Falsy Values - Inherent false boolean values

- false
- 0 and -0
- “” and ‘’ (empty strings)
- null
- undefined
- NaN (Not a Number)
- **document.all**(something you will rarely encounter)

Demonstration

Equality

We will be using **strict** `===` and `!==` operators to check equality rather than looser `==` and `!=`

Why?

Relational Expressions

- There are different definitions of **sameness** for the **==** and **===** operators.
- The === operator is known as a **strict equality operator** (*identity operator*) which checks if two operands are **“identical”**.
- The == operator is known as the **equality operator** which checks if two operands are **“equal”** using less strict definition of sameness. It allows **type conversions**.
- The != and !== operators test for the exact opposite of the == and === operators.

Variable Hoisting

Hoisting is JS **default behavior** of moving declarations to the **top**.

- A **variable (var)** can be used **before** it has been declared
- The variable declaration is said to have been **hoisted**.
- **var** variable declarations are hoisted to the **top** of the current script.

What value will be logged to the console?

```
1    console.log(x);
2
3    var x = 5;
```

```
1   console.log(x);
2
3   var x = 5;
```

Problems >_ /workspace/bewd_js_2 × Open Ports

```
gitpod /workspace/bewd_js_2 $ node variable_hoisting.js
undefined
```

Variable Hoisting

If you do not **assign values** to variables, hoisting can cause the code to be more difficult to **extend**, **maintain** and **understand**.

Always declare variables before use.

```
var x = 5;

console.log(x);
```

Variable Hoisting let and const

When using let and const variables they must be declared before use.

```
1   console.log(x,y);
2
3   let x = 5;
4   const y = 4;
5
```

Problems >_ gitpod /workspace/bewd_js_2 × Open Ports

```
gitpod /workspace/bewd_js_2 $ node variable_hoisting.js
/workspace/bewd_js_2/variable_hoisting.js:1
console.log(x,y);
            ^

ReferenceError: x is not defined
```

not defined, undefined and null

- **not defined** variables don't exist
- **undefined** variables exist but are not assigned and hold no value
- **null** variables exist and have null assigned

```
1  // unefined value
2  let x;
3  console.log(x);
4  // null value
5  let y = null;
6  console.log(y);
7  // not defined value
8  console.log(z);
9
```

Lets Code!

If you don't have a Codecademy account yet, sign up and do the following lab.

https://www.codecademy.com/learn/introduction-to-javascript