JavaScript - Basics

```
<script>
if(age > 18){
    alert("Adult");
} else{
    alert("Teenager");
}
</script>
```

Siddharth Singha Roy

Winter 2019

JavaScript is a light weight object oriented cross platform programming language.

It is used mainly in the client side of the web.
However thanks to NodeJS it is used now used both on the client and server side.

JavaScript can be added to a web browser in 2 different ways

1. One of them is using inline scripts inside HTML
2. Using an external JavaScript File

Inline Script

```
<script>
….JS Lines
</script>
```

All JavaScript line ends with a semicolon

Running JavaScript Files on the Terminal

1. Open Terminal or Command Prompt.
2. Set Path to where File is Located (using cd).
3. Type **"node New.js"** and Click Enter

External Script

1. Create a new file
2. <script type="file.js"></script

Variables and DataTypes (ES5 Version)

In JavaScript variables can be declared using the **<u>var</u>** keyword.

In JavaScript we have 5 different primitive datatypes

1. Number
2. Boolean
3. String
4. Undefined
5. Null

A Number in JavaScript is always a **floating point.**

We can declare string in JavaScript using both single and double quotes.

Undefined is when we declare a variable but never assign any value
For example : var job;

JavaScript uses **dynamic typing** which means that unlike other languages we do not need to specify the variable type…..it will be automatically detected.

Variable Naming Convention : **Camel Case**
Example : isAdult, isMarried, fullName, fullAddressOrName

Naming Rules

1. Cannot start with a number
2. Can start with $ or _
3. Only allowed symbols are $ and _
4. Cannot be a JavaScript keyword

Comments

Single Line Comments : //

Multi-Line Comments : /*   */

JavaScript Variable Mutation and Type Coercion

TYPE COERCION

JavaScript will automatically convert variables into certain types based on the requirements.

```
var age = 18
console.log(“Age : “ + age);
```

Here age will automatically converted into a string datatype. Not only number, but also boolean and undefined can be converted into string.

VARIABLE MUTATION

This simply means that we can assign value of any variables to whatever datatype we want.

Instead of logging all the values, we can use alert(“String…..”)

One of the ways to get input from the user is using

```
var lastName = prompt(“What is your name”);
```

Basic Operators

JavaScript supports all the basic operators that are supported by most of the languages.

One of the most useful operator in JavaScript is the **typeof** operator that specifies which type of datatype the variable belongs to.

```
var x, y;
x = y = 5*2;
```

Both the variables will get the value of 10. The reason being the assignment operator has associativity from right to left. So first y get 10 then x gets y.

We can also console.log more than one item using

```
console.log(x, y);
```

JavaScript also supports shorthand notation like

1. i++
2. i*=2;
3. i—;
4. ++i;

Decision operations

Like most other programming languages, we have **if/else-if/else** in JavaScript.

Logical Operators

1. ===
2. !==
3. >=
4. <=
5. >
6. <

Boolean Operators

1. && (and)
2. || (or)
3. ! (not)

Ternary operators

```
Condition ? True_result : False_result;
```

Switch

```
switch(variable) {
        case val:
                break;
        default:
                break;
}
```

What switch does is it will evaluate val and variable for equality and then if same will proceed with the case val.

So to convert a if/else into a switch case, just use

```
switch(true) {
        case age<13:
                break;
        default:        this is the else condition
                break;
}
```

Truth and False Conditions

False

1. undefined
2. null
3. ‘’
4. 0
5. NaN

```
var height;
if(height) {
   console.log("Height is defined");
} else {
   console.log("Height is not defined");
}
```

This will evaluate to false. However if we set height = 5 it evaluates to true. But if it is 0 then it evaluates to false.

Equality Operators

=== does strict equality comparison (Better to use) [Similarly we have !==]

== does type coercion if required while comparing [Similarly we have !=]

FUNCTIONS

Functions help us to follow the DRY (Do not Repeat Yourself) principle.

```
Function funcname ( arg1, arg2, arg3, …. ) {
        // Do something
        return variableName
}
```

A function can also have no return values.

Function declaration

```
Function funcname ( arg1, arg2, arg3, …. ) {
        // Do something
        return variableName
}
```

Function Expression

```
Var funcname = function(arg1, arg2, arg3….) {
        // Do something
        return variableName
}
```

JavaScript expressions expect us to return some values while JavaScript declaration or statements do not expect us to return some value. A statement will always produce a result.

Arrays

Declaration

var arr = new Array(5,6,2,8);

OR

var arr = [5,6,2,8];

Array values are mutable.

In array we can can have different datatypes in the same array.

Arrays can hold arrays inside them (nested array) and objects as well.

Methods on arrays

1. arr.length Gives the length of the array
2. arr.push(5) Adds an element at the end of the array
3. arr.unshift(4) Adds an element at the beginning of an array
4. arr.pop() Removes from the end
5. arr.shift() Removes from the beginning
6. arr.indexOf(searchItem) Returns the index of the searchItem in arr or returns -1

Objects

It is used to hold a collection of variables inside it.

Objects have a key and value, the key will be used to access the value.

Objects can hold different datatypes in them like objects (nested objects) and even arrays.

Accessing elements in an object can be done in two ways :

1. Dot notation => object_name.key_name
2. Index notation => object_name['key_name']

Objects inside the string are also mutable.

user.username = "sid"

user['password'] = "pwd"

We can even add new key-value pair using the above two notations

user.new_key = new_val

We can even have functions inside our objects (called methods) which always needs to be function expressions ie. it always needs to return a value.

Now sometimes it may happen that we need to access the value of the object inside the object.

We can use the this keyword which will refer to the current object.

Now instead of separately specifying a new key-value relation we can directly do that in the method using this keyword.

Loops and Iterations

For Loop

```
for(var i = 0; i < 10 ; i++)
{
}
```

While Loop

```
var i = 0;
while(i<10)
{
	i++;
}
```

Continue : Skip the current iteration

We also have forEach and map method. Both of them take a callback function as input. forEach will simply iterate over the array whereas map will also iterate over the array and return a new array. So the callback function of a map must always return something

```
arr.forEach(function(value, index, array) {
	console.log(value+1);
});

arr.map(function(value, index, array) {
	return value.length;
});
```

array will store arr