Web Advanced: Javascript

Karla Polo
karla@newschool.edu

Let’s Understand the Basics!

SYNTAX, DATA TYPES, OPERATORS, CONDITIONS, LOOPS, FUNCTIONS

JAVASCRIPT SYNTAX

- ➔ Comments
- ➔ Expressions
- ➔ Statements
- ➔ Blocks

COMMENTS

```
// I should comment everything - it's a good practice

var myVariable; //I can comment pretty much anywhere.

/*      Let's think in plane English what do I want to do:
            • Step 1 - Describe what do you want to do
            • Step 2 - Describe what do you want to do
            • Step 3 - Describe what do you want to do
            • Step 4 - Describe what do you want to do
*/
```

EXPRESSIONS

// An expression returns a value and can be written wherever a value is expected

```
x = 7  // assigns value to a variable

3 + 4 // resolves to a value

true / false  // evaluates true or false, involving logical operators

this  // primary expressions. Basic keywords and general expressions in JavaScript.

“Hello” + “World”  // strings. Evaluates to a character string
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operators

STATEMENTS

// Statements are composed of: Values, Operators, Expressions, Keywords, and Comments.

```
let answer = 42;  // let is block scoped
alert ("Hello" + answer);

var greeting =  "Good"     + "    " + "Morning";  // var is function scoped
console.log (greeting);
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let

BLOCKS

// A block is used to group statements. The block is delimited by a pair of curly brackets and may optionally be labeled

```
{
        let answer       =      42;
        alert ( "Hello"   +      answer );
        let greeting      =      "Good" +  "    "  + "Morning";
        console.log (greeting);

}
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/block

DATA TYPES

- ➔ Number
- ➔ String
- ➔ Symbols
- ➔ Booleans
- ➔ Undefined/null
- ➔ Arrays
- ➔ Functions
- ➔ Objects

NUMBERS

These are all number expressions:

42

3.1415

3e8 // 3 x 10^8

4*(12+6)/3

NaN

Infinity / -Infinity

STRINGS

// The String global object is a constructor for strings or a sequence of characters.

“Hello World”

“Hello 42 and other #”

“{Who} / [When]”

UNDEFINED / NULL

```
// is this defined?

document.write(varName);


// what is this value?
var   nullVariable = null;
```

// **not define** means a variable hasn't been declared

// **undefined** means a variable has been declared but has not yet been assigned a value.

// **null** is an assignment value. It can be assigned to a variable as a representation of no value. Null is an object.

BOOLEANS

Every value/expression in JS has a Boolean value: true or false.

Boolean(expression) 40 > 39 //true

"A" > "B" // false

"a" > "A" // true (lowercase has a higher value)

Most values always are TRUE except a few:.

False values: "" 0 NaN false null undefined

TYPE COERCION

Javascript auto-converts the value from one type to another (such as string to number, object to boolean, and so on) as needed to complete an expression:

```
14    +     ""     //     "14"
42    +     "0"    //     "420"
"42"  -     7      //     35
"42"  *     7      //     294
null  ||    "34"   //     34
```

OPERATORS

- ➔ Arithmetic
- ➔ Comparison
- ➔ Logical
- ➔ Assignment
- ➔ Conditional

ARITHMETIC

Addition (+)

Subtraction (-)

Division (/)

Multiplication (*)

Reminder (%)

Exponentiation (**)

Increment (++)

Decrement (--)

COMPARISON

```
5     ==   6          // false
5     !=   6          // true
"1"   ==   1          // true
"1"   ===  1          // false
1     ==   true       // true
1     ===  true       // false
```

== vs ===

For "a **==** b" to evaluate to true a and b need to be the same value.

In the case of "a **===** b" a and b must be the same value and also the same type for it to evaluate to true.

LOGICAL

Logical operators are typically used with `Boolean` (logical) values. When they are, they return a Boolean value. However, the `&&` and `||` operators actually return the value of one of the specified operands, so if these operators are used with non-Boolean values, they will return a non-Boolean value.

AND (&&)
OR (||)
NOT (!)

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Logical_Operators

ASSIGNMENT

// An assignment operator assigns a value to its left operand based on the value of its right operand.

```
var x = 2;
var y = 3;

console.log(x);

console.log(x = y + 1);

console.log(x = x * y);
```

VARIABLES

Storing a String

```
let     myString    =    "This is the end.";
let     myString    =    new String("This is the end.");        // not used much
```

Storing a Number

```
let     myNum       =    12;
let     myNum       =    new    Number(12);   // not used much
```

ways to define

```
let     myNum =     12;     //      preferred
```

```
var     myNum       =    12;     // traditional and loose
```

```
const   myNum       =    12;     // cannot change it
```

http://2ality.com/2015/02/es6-scoping.html

SCOPE

Scope is how a variable is available to the entire program:

➔ Global scope:

Any variables or functions declared outside of a function will be available to all JavaScript code on the page, whether that code is inside a function or otherwise

➔ Functional/Local scope:

Variables and functions declared inside a function are visible only inside that function—no code outside the function can access them.

Local variables also have a lifetime - they die when the function finishes executing.

CONDITIONS

- ➔ if...else...
- ➔ switch

IF … THEN … ELSE ...

```
let   age = 23;

if (age == 18) {
    console.log("Sorry, you shouldn't be here.");
}

if    (age      <     18)  {
    alert("Sorry, you shouldn't be here.");
}     else{
    console.log("Please proceed.");
}
```

SWITCH

```
let num  = Math.floor(Math.random() * 10);

switch (num) {

    case (4):
        console.log("You rolled a four");  break;


    case (5):
        console.log("You rolled a five");  break;


    case (6):
        console.log("You rolled a six");  break;


    default:
        console.log("You rolled a number less than four");  break;


}
```

15 Minutes!!!

ARRAYS

- ➔ Defining - literal, constructor
- ➔ Common Operations
- ➔ Statements
- ➔ Blocks

DEFINING AN ARRAY

Arrays are special types of objects.

```
const myArray = [ ];

const myArray = new Array();  //not used much
```

DEFINING AN ARRAY WITH VALUES

```
//      prepopulating

const myArray = ["blue", "red", "green"];

//      adding
myArray[0]       =  "pink";
myArray[3]       =  "purple";
myArray[5]       =  null;
myArray[6]       =  4;
```

PROPERTIES & METHODS

```
//   property

console.log(myArray.length);

//   modifiers

myArray.pop();          // updates array

myArray.push(item); // updates array

myArray.concat(second_array); // new array

myArray.join(joiner); // new string

myArray.slice(2,4);   // new array starting at index 2 and ending at   index 3

myArray.splice(2,1,"brown");

myArray.includes("brown");
```

LOOPS

- ➔ while
- ➔ do...while
- ➔ for
- ➔ for...in

WHILE LOOPS

Repeat a block of code until a condition remains true:

```
let maxTime = 7;

while (maxTime< 10){

        console.log("Keep working. It's still only " + maxTime);  maxTime++;


}


let maxTime = 10;

while (maxTime--){

        console.log("Keep working. It's still only " + maxTime);

}
```

DO...WHILE LOOPS

Run a block of code at least once and then until a condition remains true:

```
let maxTime = 7;

do {

        console.log("Keep working. It's still only " + maxTime);  maxTime++;


} while (maxTime < 10);
```

FOR LOOPS

Keeps all loop-related vars in one place:

```
for (var maxTime = 7; maxTime < 10; maxTime++) {
        console.log ("Keep working. It's still only "+maxTime);

        var myArray = ["blue", "red", "green"];
            for (var i = 0; i < myArray.length; i++) {
                console.log("The selected color: " + myArray [i] );
        }
}
```

FOR...OF LOOPS

New in ES6 for looping over arrays:

```
let myArray =        ["blue", "red", "green"];

        for (const value of myArray) {

        console.log("The selected color: "+value);

}
```

* **ES6** refers to version 6 of the ECMA Script programming language. ECMA Script is the standardized name for JavaScript, and version 6 is the next version after version 5, which was released in 2011.
ECMAScript, or **ES6**, was published in June 2015. It was subsequently renamed to ECMAScript 2015.

FUNCTIONS

Functions encapsulate a block of code that does a specific task to make it reusable.

BUILT-IN FUNCTIONS

```
//      typical built-in functions

myString.charAt(1); //returns   a string
parseInt(12.34);    //returns   integer
Math.random();      //returns   a floating number
[1,2,3,4].map();    //returns   array
```

NEW FUNCTIONS

```
// a new basic function - pretty useless
function    randomNumber()     {
 console.log('I am returning', Math.random());
}


// a new basic function - better function
convertToCelsius(deg_fah) {
        let converted_deg = (deg_fah-32) * 5/9;
        console.log('The converted temperature is', converted_deg);

}
```

CALLING FUNCTIONS

```
//      this   returns the actual reference, not  the function evaluation

randomNumber;

//      this   executes the function

randomNumber();
```

FUNCTION PARAMETERS / ARGUMENTS

- ➔ **Parameters: variables needed by the function itself to run. These are set and then destroyed once complete**
- ➔ **Arguments: the vars or values sent to the function when called.**

```
function convertToCelsius(deg_fah) {

    let converted_deg = (deg_fah-32) * 5/9;
    return(converted_deg);

}

console.log( convertToCelsius(32) );
```

EXAMPLES

```
// a new basic function - better

function convertToCelsius(deg_fah) {
    let converted_deg = (deg_fah-32) * 5/9;  return(converted_deg);


}

let converted_deg;

for ( let i = -148;  i <= 212; i = i+10) {

          converted_deg = convertToCelsius(i);


        console.log('The converted temperature of',i,' is',  converted_deg);

}
```

Assignment: