Javascript – II

Function Calls:

Alert ()

The alert () method displays an alert box with a specified message an OK button. It is often used if you want to make sure information comes through to the user. Alert function does not return a value, but confirm does.

```
var userConfirmed = confirm('Are you sure?');
console.log(userConfirmed)
```

If we select yes, in the console it would show `true` else, `false`

Confirm ()

The confirm () method displays a dialog box with a specified message, along with an OK and a CANCEL button. It is often used if you want the user to verify or accept something. This returns true if the use clicked “OK”, and false otherwise.

```
var userInput = prompt('What is your name?')
console.log(userInput)
```

A Boolean, indicating whether “OK” or “CANCEL” was clicked in the dialog box:

- True – the user clicked “OK”
- False – the user clicked “CANCEL” (or the “x” (close)) button in the top right corner that is available in all major browsers

If we give our input, then it would show the input that we have given in the console, else it shows **null**.

Prompt ()

The prompt () method displays a dialog box that prompts the visitor for input. It is often used if you want the user to input a value before entering a page.

It returns the input value if the user clicks “OK”. If the user clicks “CANCEL” the method returns **null**.

If the user clicks OK without entering any text, an empty string is returned.

Arithmetic Operators:

Exponentiation (**):

The **exponentiation** operator (**) raises the first operand to the power of the second operand.

Example:

```
-4 ** 2 // This is an incorrect expression
-(4 ** 2) gives -16, this is a correct expression
2 ** 5 gives 32
3 ** 3 gives 27
3 ** 2.5 gives 15.588457268119896
10 ** -2 gives 0.01
2 ** 3 ** 2 gives 512
NaN ** 2 gives NaN
```

Increment (++):

The increment operator increments (adds one to) its operand and returns a value.

- If used postfix with operator after operand (for example, x++), then it increments and returns the value before incrementing.
- If used prefix with operator before operand (for example, ++x), then it increments and returns the value after incrementing.

Example:

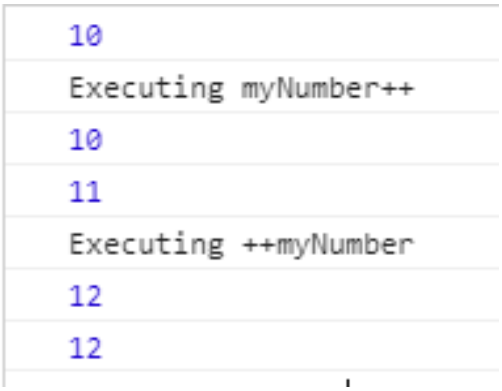
```
// Postfix
var a = 2;
b = a++; // b = 2, a = 3

// Prefix
var x = 5;
y = ++x; // x = 6, y = 6
```

In the script.js we can do it as

```
var myNumber = 10
console.log(myNumber)
console.log('Executing myNumber++')
console.log('myNumber++')    // 10
console.log('myNumber')      // 11
console.log('Executing ++myNumber')
console.log(++myNumber)      // 12
console.log(myNumber)        // 12
```

Ouput:



Decrement (-):

The decrement operator decrements (subtracts one from) its operand and returns a value.

- If used prefix, with operator after operand (for example x-), then it decrements and returns the value before decrementing.
- If used prefix, with operator before operand (for example -x), then it decrements and returns the value after decrementing.

Example:

```
// Prefix
var x = 2;
y = --x; gives x = 1, y = 1

// Postfix
var x = 3;
y = x--; gives y = 3, x = 2
```

Assignment Operators:

```
var abc = 10 + 2 +11
abc += 10 // abc = abc + 10
```

First it will come to the assignment operator. abc will be replace to 23 and then it would add 10 to it. The final value would be 33.

Assignment (Arithmetic) Operators

JavaScript supports the following assignment operators –

Sr.No.	Operator & Description
1	= (Simple Assignment) Assigns values from the right side operand to the left side operand Ex: C = A + B will assign the value of A + B into C
2	+= (Add and Assignment) It adds the right operand to the left operand and assigns the result to the left operand. Ex: C += A is equivalent to C = C + A
3	−= (Subtract and Assignment) It subtracts the right operand from the left operand and assigns the result to the left operand. Ex: C -= A is equivalent to C = C - A
4	*= (Multiply and Assignment) It multiplies the right operand with the left operand and assigns the result to the left operand. Ex: C *= A is equivalent to C = C * A
5	/= (Divide and Assignment) It divides the left operand with the right operand and assigns the result to the left operand. Ex: C /= A is equivalent to C = C / A
6	%= (Modules and Assignment) It takes modulus using two operands and assigns the result to the left operand.

	Ex: C %= A is equivalent to C = C % A
7	**= (Exponentiation and Assignment) It raises the value of a variable to the power of the right operand. Ex: B **= A is equivalent to B = B**A

Note − Same logic applies to Bitwise operators so they will become like <<=, >>=, >>=, &=, |= and ^=.

Comparison Operators:

JavaScript supports the following comparison operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No.	Operator & Description
1	= = (Equal) Checks if the value of two operands are equal or not, if yes, then the condition becomes true. Ex: (A == B) is not true.
2	!= (Not Equal) Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. Ex: (A != B) is true.
3	> (Greater than) Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. Ex: (A > B) is not true.
4	< (Less than) Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. Ex: (A < B) is true.
5	>= (Greater than or Equal to)

	Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: (A >= B) is not true.
6	<= (Less than or Equal to) Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: (A <= B) is true.

7	= = = (Strictly Equality)
	Checks whether its two operands are equal, return a Boolean result. Unlike the equality operator, the strict equality operator always considers operands of different types to be different. It checks if the two operands are of same data type or not.
	Ex: (A = = = B)
	`console.log("hello" === "hello");  // true`
	`console.log("hello" === "hola");  // false`
	`console.log(3 === 3);  // true`
	`console.log(3 === 4);  // false`
	`console.log(true === true);  // true`
	`console.log(true === false);  // false`
	`console.log(null === null);  // true`
	`console.log("3" === 3);  // false`
	`console.log(true === 1);  // false`

8	! = = (Strick Inequality)
	Checks whether its two operands are not equal, returning a Boolean result. Unlike the inequality operator, the strict inequality operator always considers operands of the different types to be different.
	`console.log("hello" !== "hello");  // false`
	`console.log("hello" !== "hola");  // true`
	`console.log(3 !== 3);  // false`
	`console.log(3 !== 4);  // true`

```
console.log(true !== true);          // false
console.log(true !== false);         // true
console.log(null !== null);          // false
console.log("3" !== 3);              // true
console.log(true !== 1);             // true
```

Truthy Value and Falsy Value:

In javascript, a truthy value is a value that is considered true when encountered in a Boolean context. All values are truthy unless they are defined as falsy.

A truthy values represent a binary true and a falsy value represent a binary false. In binary an expression can be either true or false. In binary we have **0** and **1**.

- **0** represents false
- **1** represents true

Now, if 1 && 0 ➔ true AND false → so the entire expression results to false. Therefore, the expression is returning **0**.

False → false, '', "", 0, -0, 0n, Nan, null, undefined
True → anything that is not mentioned above

Conditional Operator:

For comparison, we generally use the if else statement.

```
if (20 > 10) {
    // your computations
}
else {
}
```

```
if (condition) {
}
else if (anotherCondition) {
}
else{
}
```

NOTE: We can have multiple else if just as in python.

In the comparison operator, we have conditional (ternary) operator '?' that takes three operands: a condition followed by a question mark, then an expression to execute if the condition is truthy followed by a colon (:) and finally the expression to execute if the condition is falsy. This operator is frequently used as a shortcut for the if statement.

Using this operator, instead of using the complete if else statement we write it as

```
20 > 10 ? console.log('Greater') : console.log('Smaller')
Greater

20 > 10 ? console.log('A') : console.log('B')
A
```

Loops:

For Loop:

A for loop repeats until a specified condition evaluates to false. The javascript for loop is similar to the Java and C for loop.

```
for (var i = 0; i < 5; i++){
    console.log(i);
}
```

Output:

0
1
2
3
4
5

Loops offer a quick and easy way to do something repeatedly. It offers different ways to determine the start and end points of the loop. There are different kinds of loops, but they all essentially do the same thing: repeat a particular action, some number of times.

While Loop:

`A while` statement executes its statements as long as a specified condition evalutes to true. if the condition becomes false, statement within the loop stops executing and control passes to the statement following the loop.

The condition test occurs before statement in the loop is executed. If the condition returns true, statement is executed and the condition is tested again. if the condition returns false, execution stops and control is passed to the statement following while.

To execute multiple statements, use a block statement ({}) to group those statements.

```
var index = 10
while(index < 30){
    // your logic
    index++
}
```

Output:

10	17	24
11	18	25
12	19	26
13	20	27
14	21	28
15	22	29
16	23	

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

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