## Operators in JavaScript

1. Operators is use to perform some operation between operand.

a + b 🡪a,b is operand

+ 🡪 operator

1. Operators are used to manipulate data and make decision
2. Types of Operators -
   1. **Arithmetic Operators**
   2. **Assignment Operators**
   3. **Logical Operators**
   4. **Comparison Operators**
   5. **Ternary Operators - ?**
   6. **Type Operators – typeof**
   7. **Bitwise Operators** - This operator works on Binary Number

**Arithmetic Operators -**

* Addition +
* Subtraction -
* Multiplication \*
* Division /
* Modulus %
* Increment ++
* Decrement --
* Exponent \*\*

**Assignment Operators -**

* Assign =
* Add & assign +=
* Subtract and Assign -=
* Multiply and assign \*=
* Divide and assign /=
* Modulo and assign %=

**Comparison Operators -return Boolean expression (true or false)**

* 1. equal to == check only value
  2. Strict equal to === **check value as well as datatype**
  3. Greater than >
  4. Greater than or equal to >=
  5. Less than <
  6. Less than or equal to <=
  7. Not equal to !=
  8. **Strictly not equal to !== check value as well as datatype**

**Logical Operator - This operators are used to combine multiple condition**

**OR Operator ( || ) -**

|  |  |  |  |
| --- | --- | --- | --- |
| **i.** true | || | false - true | |
| **ii.** false | || | true - true | |
| **iii.** true | || | true - true | |
| **iv.** false | || | false - false | |
| **AND Operators ( && ) -** | | | |
| i. true | && | true | - true |
| ii. true | && | false | - false |
| iii. false | && | true | - false |
| iv. false | && | false | - false |
|  | | | |

**Functions in JavaScript**

* A function in JavaScript is a reusable block of code that performs a specific task which only run when its call.
* It allows you to create logic, avoid repetition and improve code readability and reusability.

**How to define and call function –**

A basic function definition involves the *function keyword*, a *name*, a list of *parameters* in *parentheses* and the code *executed in curly braces.*

*// function definition*

## function functionName ()

## {

**//function body**

**}**

//function call

**functionName ()**

**let firstFunction = functionName()**

**Types of Functions –**

1. **Non – Parameterized Function:**
   1. The functions with no parameters ie. the parenthesis after the function name is empty.
   2. Syntax:

function <function\_name> ()

{

#Function Body

}

1. **Parameterized Function:**
   1. The function with parameters defined at the time of function declaration.
   2. Syntax:

function <function\_name> (parameters)

{

#Function Body

}

1. **Function Expression:**
2. In function expression, function is assigned to a variable
3. Function expression never hoisted
4. **Arrow Function:**

* Arrow function is one of the features introduced in the ES6 version of JavaScript.
* The main advantage of using arrow function is it’s **shorter syntax** and **require less code.**

Syntax:

1. **Arrow function with No Parameter**

let x = () => console.log(“Hello World”);

1. **Arrow function with One Parameter**

let x = name =>console.log(“Hello”, name);

1. **Multi – Line Arrow Function**

let add = (a, b) => {

res = a + b;

console.log(“Addition is:”, res);

}

1. **Anonymous Function**
   1. These functions do not have a name
   2. This type of function passed as an argument to another function.

**Example –**

console.log('1');

console.log('2');

setTimeout(()=>{

console.log('3');

},1000)

console.log('4');

console.log('5');

**Template Literals**

Template literals provide an easy way to interpolate variables and expressions into strings.

The method is called string interpolation.

**Syntax: `**${expression}**`**

**return keyword –**

**return keyword is used for 2 purpose -**

* 1. **return keyword is use to return a value to the caller function.**
  2. **To terminate the execution of the function**

## Control Flow Statement -

1. Control flow statement control the code of execution
2. Type of Control Flow –

Conditional statement

* 1. if block
  2. else block
  3. If else block

Loops -

* + 1. for loop
    2. While loop
    3. do-while loop
    4. for in loop
    5. for of loop

## conditional statement

## A conditional statement is used to perform different actions based on different conditions. It helps your code "decide" what to do depending on a situation.

## If Statement

## The if statement checks a condition. If the condition is true, the code inside the block runs.

## Syntax:

## if (condition) {

## // code to run if condition is true

## }

## 

## If-else Statement

## Use else to run a block of code if the if condition is false.

## Syntax:

## if (condition) {

## // code to run if condition is true

## }

## else{

## //code

## }

## If-else if-else Statement

## Use else if to check multiple conditions in sequence.

## Synatx:

## if (condition1) {

## // code if condition1 is true

## } else if (condition2) {

## // code if condition2 is true

## } else {

## // code if none of the above conditions are true

## }

## Ternary Operator –

## ****ternary operator**** also called the conditional operator in JavaScript

## ****Syntax –****

## condition ? expTrue : expFalse

## *Example –*

## Age Check for Voting eligible or not

## let age = 20

## (age >= 18) : console.log(“Eligible”) : console.log(“Not Eligible”)

## Loops in JavaScript –

1. In Programming, a loop is a control flow statement that allows you to execute a block of code repeatedly as long as a certain condition is true.
2. Loops are fundamentals for automating repetitive tasks and iterating over the collection data (array).
3. While performing calculations or tasks if the condition is false then the loop automatically terminates.
4. **Key Concept of loops –**
   1. Variable declaration
   2. Condition Check (Comparison Operator)
   3. Increment or decrement (++ or --)
5. There are three types of loop -
   * + 1. **for loop**

In the for loop the first condition is checked and then code is executed.

Syntax -

for(variable; condition; inc++/dec--){

  //code execute

}

1. **while loop**

In the while loop the first condition is checked and then

code is executed.

Syntax -

Variable declare

while (**condition**) {

**//code execute**

     inc++/dec--

}

1. **do…..while loop**

In the do…..while loop the first code is executed and then

condition is check that do while print the statement at least

once.

Syntax -

variable declare

do{

//code execute

  inc++/dec--

}while(**condition**)

1. **for…..in Loop -**

* This loop is used to iterate properties (key) of an object.
* This loop is only used for Object not for Array.

Syntax -

for(let key in Object){

//code of block

}

1. **for…..of Loop -**

* This loop is used to iterate array, string , maps etc.
* This method provides direct access to values not key.
* This method never recommend for Iterate Object

Syntax -

* 1. for(let **value** of iterable){

//code of block

}