JavaScript

1) Comments

2) identifiers

3) Variables

4) Operators

5) Data types

6) Conditional Statements

7) Looping Statements

Comments

JavaScript comments can be used to explain JavaScript code, and to make it more readable, it is used to prevent code execution

1. Single Line Comment

Single Line Comment starts with //

Any code after the // till the end of the line will be ignored by JavaScript

2. Multiline Comment

Multi-line comments start with /\* and end with \*/.

Any code written between /\* and \*/ will be ignored by JavaScript

Ex: /\*The code below will change  
the heading with id = "demo"  
and the paragraph with id = "text"  
in my web page: \*/

Identifiers

An **identifier** is the name used to identify **variables**, **functions**, **arrays**, **objects**, **classes**, or any user-defined element in JavaScript.

Must begin with a **letter (A–Z or a–z)**, underscore (\_) , or **dollar sign ($)**.

Cannot begin with a **digit (0–9)**.

Can contain **letters**, **digits**, **underscores**, and **dollar signs**.

Cannot be a **JavaScript reserved keyword** (like let, class, if, etc.).

Identifiers are **case-sensitive** (name and Name are different)

Variables

Variables are The Containers for Storing Data

JavaScript Variables can be declared in 3 ways:

* Using var
* Using let
* Using const

**1) Declaring a variable using var**

**--------------------------------------------**

In JavaScript, the var keyword is used to declare variables

i) Variables declared with var are function scoped

Example:

function test () {

if (true) {

var x = 5;

}

console.log(x); // 5 (accessible outside the if block)

}

ii) Can be re-declared and updated

We can declare the same variable multiple times with var in the same scope

var a=10

var a=20 // No Error

iii) Hoisting

Var declarations are hoisted to the top of the scope , only declaration is hoisted not initialization.

Ex: console.log(b); // undefined (not ReferenceError)

var b = 15;

**2) Declaring the variables by using let**

**----------------------------------------------------**

i) The let keyword was introduced in [ES6 (2015)](https://www.w3schools.com/js/js_es6.asp)

ii) Variables declared with let have Block Scope

Ex: Variables declared inside a { } block cannot be accessed from outside the block:

{  
  let x = 2;

}  
// x can NOT be used here

iii) Variables declared with let must be Declared before use

Variables defined with let are also hoisted to the top of the block, but not initialized.

Using a let variable before it is declared will result in a ReferenceError

Ex: carName = "Thar";  
 let carName = "KIA";

iv)Variables declared with let cannot be Redeclared in the same scope

Ex: let x = "John Doe";  
 let x = 0;

**3) Declaring the variables by using const**

**---------------------------------------------------------**

i) The const keyword was introduced in [ES6 (2015)](https://www.w3schools.com/js/js_es6.asp)

ii) Variables defined with const cannot be Redeclared, Redeclaring a variable with const, in another scope, or in another block, is allowed

const pi=3.142

{

const pi=3.1

}

iii)Variables defined with const cannot be Reassigned

Ex: const PI = 3.141592653589793;  
 PI = 3.14;

iv) Variables defined with const have Block Scope

Operators

There are different types of JavaScript operators:

* Arithmetic Operators
* Assignment Operators
* Comparison Operators
* String Operators
* Logical Operators
* Bitwise Operators
* Ternary Operators
* Type Operators

**Arithmetic Operators** are used to perform mathematical operations

(+, -, \*, /, %, \*\*, ++, --)

**Assignment operators** assign values to JavaScript variables.

(=, +=, -=, \*=, /=, %=)

**Comparison operator** is used to compare two or more variables

(==, ===, >, <, <=, >=, ?)

**Logical operator** (&&, ||,!)

**Type Operators**

**typeof:**

The typeof operator returns the type of a variable or an expression:

Example

typeof ""             // Returns "string"  
typeof "John"         // Returns "string"  
typeof "John Doe"     // Returns "string"

**Instanceof:** Returns true if an object is an instance of an object type

Datatypes

JavaScript has 8 Datatypes

String  
Number  
Bigint  
Boolean  
Undefined  
Null  
Symbol  
Object

Note: A JavaScript variable can hold any type of data.

**Numbers:** All JavaScript numbers are stored as decimal numbers (floating point).  
let length = 16;  
let weight = 7.5;  
  
**Strings:** Anything which is written in single or double quotes will be considered as string  
let color = "Yellow";  
let lastName = ‘Johnson’;

**Booleans:** any variables which contains value true or false such type of data is comes under boolean type  
let x = true;  
let y = false;

**BigInt datatype:** Any value which is greater than 2 to the power of 53-1 comes under BigInt datatype

**Undefined**: In JavaScript, a variable without a value, has the value undefined. The type is also undefined.

Example

let car;   // Value is undefined, type is undefined

**Symbol:** Any data which is created by using constructor symbol () comes under symbol datatype

**Object:**  
JavaScript objects are written with curly braces {}.

Object properties are written as name:value pairs, separated by commas.

Example

const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

**Array object:**

JavaScript arrays are written with square brackets.

Array items are separated by commas.

The following code declares (creates) an array called cars, containing three items (car names):

Example

const cars = ["Saab", "Volvo", "BMW"];

**Date object:**  
 const date = new Date("2022-03-25");

Conditional Statements

**The if Statement**

if statement is used to specify a block of JavaScript code to be executed if a condition is true.

Syntax

if (*condition*) {  
  //*block of code to be executed if the condition is true*}

**The else Statement**

else statement is used to specify a block of code to be executed if the condition is false.

syntax

if (*condition*) {  
  //*block of code to be executed if the condition is true*} else {  
  //*block of code to be executed if the condition is false*}

**The else if Statement**

else if statement is used to specify a new condition if the first condition is false.

Syntax

if (*condition1*) {  
  //*block of code to be executed if condition1 is true*} else if (*condition2*) {  
  //*block of code to be executed if the condition1 is false and condition2 is true*  
} else {  
  //*block of code to be executed if the condition1 is false and condition2 is false*}

**Switch Statement**

The switch statement is used to perform different actions based on different conditions.

switch(*expression*) {  
  case *x*:  
*// code block*    break;  
  case *y*:  
*// code block*    break;  
  default:  
    // *code block*  
}

This is how it works:

* The switch expression is evaluated once.
* The value of the expression is compared with the values of each case.
* If there is a match, the associated block of code is executed.
* If there is no match, the default code block is executed.

When JavaScript reaches a break keyword, it breaks out of the switch block. This will stop the execution inside the switch block.

**Note:**If you omit the break statement, the next case will be executed even if the evaluation does not match the case.

The default keyword specifies the code to run if there is no case match

Looping Statements

 If you want to run the same code over and over again, each time with a different value.

JavaScript supports different kinds of loops:

* **for** - loops through a block of code a number of times
* **for/in** - loops through the properties of an object
* **for/of** - loops through the values of an iterable object
* **while** - loops through a block of code while a specified condition is true
* **do/while** - also loops through a block of code while a specified condition is true

**The For Loop**

For loop is used to execute the code specified number of times

for (*expression 1*;*expression 2*;*expression 3*) {  
  // *code block to be executed*  
}

**The For In Loop**

The JavaScript for in statement loops through the properties of an Object:

Syntax

for (key in object) {  
  // *code block to be executed*  
}

Example:

const person = {fname:"John", lname:"Doe", age:25};  
  
let text = "";  
for (let x in person) {  
  text += person[x];  
}

**Array.forEach()**

The forEach() method calls a function (a callback function) once for each array element.

Example

const numbers = [45, 4, 9, 16, 25];  
  
let txt = "";  
numbers.forEach(myFunction);  
  
function myFunction(value, index, array) {  
  txt += value;  
}

o/p : 45 4 9 16 15

**The For Of Loop**

The JavaScript for of statement loops through the values of an iterable object.

Syntax

for (variable of iterable) {  
  // *code block to be executed*  
}

Example:

const cars = ["BMW", "Volvo", "Mini"];  
  
let text = "";  
for (let x of cars) {  
  text += x;  
}

o/p: BMW Volvo Mini

**The While Loop**

The while loop loops through a block of code as long as a specified condition is true.

Syntax

while (*condition*) {  
*// code block to be executed*  
}

Example:

while (i < 10) {  
  text += "The number is " + i;  
  i++;  
}

**The Do While Loop**

Do While loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

do {  
*// code block to be executed*}  
while (*condition*);

Example:

do {  
  text += "The number is " + i;  
  i++;  
}  
while (i < 10);