Node Js and JavaScript

[Declarations](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Grammar_and_types#declarations):

**Var:**

Ex: var test1 = 12,

test2= 14,

test3 = 16

function foo(){

console.log(test1, test2, test3);

}

foo();

12 14 16

Ex:

console.log(test);

var test = 12; undefined

Ex:

var test = 12;

var test = 100;

console.log(test);--------100

Ex:

var test = 12;

function foo(){

var test = 100;

console.log(test);

}

foo();

console.log(test);

**Const:**

Cannot be reassigned.

It has Block Scope

It can be assigned to the variable on the declaration line.

It’s a Primitive value.

The property of a const object can be changed but it cannot be changed to a reference to the new object

The values inside the const array can be changed, it can add new items to const arrays but it cannot reference a new array.

Re-declaring of a const variable inside different block scopes is allowed.

Cannot be Hoisted.

Creates only read-only references to value.

Ex1:

const x = 12;

x = 13;

x += 1;-------------Uncaught TypeError: Assignment to constant variable.

Ex2:

const x = 22;

{

const x = 90;

console.log(x);

{

const x = 77;

console.log(x);

}

{

const x = 45;

console.log(x);

}

}

console.log(x);-----------------90,77,45,22

Ex3:

Variables must be Assigned:

It describes the const variable and assigned it after declaration.

const x;

x = 12;----------Uncaught SyntaxError: Missing initializer in const declaration

--------------

Ex4:

x = 3;

console.log(x);

const x; ----------Uncaught SyntaxError: Missing initializer in const declaration

Ex5:

// Changing the content of array is

// possible in cost array

const arr1 = ["pankaj", "sumit", "chandan", "ajay"];

console.log(arr1.toString());

arr1[2] = "Narayan"; // possible

console.log(arr1.toString());-----------------pankaj,sumit,chandan,ajay /pankaj,sumit,Narayan,ajay

const person = {

first\_name: "Pankaj",

last\_name: "Singh",

Age: 20,

About: "Web Developer and Competitive Programmer"

};

console.log(person);

// It is possible

person.first\_name = "Aryan";

person.last\_name = "Yadav";

person.Age = 22;

person.About = "Commerce undergraduate";

console.log(person);

// it is not possible

// const person={

// "first\_name":"Aryan",

// "last\_name":"Yadav",

// "Age":22,

// "About":"Commerce undergraduate"

// }

**Let:**

let variable\_name = value

1. Block Scope:

Ex:

{

let num = 10;

// calling the function inside block

console.log(num)

}

// Calling a function outside

// block throws an Error

console.log(num)-----------------10

Uncaught ReferenceError: num is not defined

Ex2:

2. Global Scope

let num = 10;

console.log(num);

function fun() {

console.log(num);

}

fun();

Ans:

10

10

3. Function Scope

Ex:

function fun() {

let num = 10;

console.log(num);

}

fun(); // Calling the function

console.log(num);

Asn:10

"ReferenceError: num is not defined

1. Redeclaring Variables in different blocks

Ex:

let x = 77;

{

let x = 23;

console.log(x);

}

console.log(x); ----23

77

2. Redeclaring Variables in the same blocks:

let x = 77;

{

let x = 23; // legal

console.log(x);

}

let x = 67; // illegal

console.log(x);

Ans: Uncaught SyntaxError: Identifier 'x' has already been declared

Does not support Hoisting

x = 12;

console.log(x);

let x;-------Uncaught ReferenceError: Cannot access 'x' before initialization

\*\*\*\*The behavior of moving the declarations on top of the script is known as hoisting.

**Functions:**

A JavaScript function is executed when “something” invokes it (calls it).

function functionName(Parameter1, Parameter2, ...)

{

// Function body

}

Arrow Function: efficient method create function

let function\_name = (argument1, argument2 ,..) => expression

const a = ["Hydrogen", "Helium", "Lithium", "Beryllium"];

const a2 = a.map(function (s) {

return s.length;

});

console.log("Normal way ", a2); // [8, 6, 7, 9]

const a3 = a.map((s) => s.length);

console.log("Using Arrow Function ", a3); // [8, 6, 7, 9]

Types Of Functions in Javascript:

1. Named function:

function add(a, b) {

return a + b;

}

console.log(add(5, 4));

2. Anonymous function:

let add = function (a, b) {

return a + b;

}

console.log(add(5, 4));

3. Nested Functions:

function msg(firstName) {

function hey() {

console.log("Hey " + firstName);

}

return hey();

}

msg("Ravi");

call(): It calls and returns a method with the owner object being the argument

// function that returns product of two numbers

function product(a, b) {

return a \* b;

}

// Calling product() function

let result = product.call(this, 20, 5);

console.log(result);

2. JavaScript Methods:

object = {

methodName: function() {

// Content

}

};

object.methodName()

Ex:

let employee = {

empname: "Rahul",

department: "sales",

details: function () {

return this.empname +

" works with Department " +

this.department;

}

};

console.log(employee.details());

Function:

1) a block of code designed to perform a particular task.

2)Syntax of Function is:

function functionName(parameters) {

// Content

}

3)A function can pass the data that is operated and may return the data

4 )Data passed to a function is explicit.

5)A function lives on its own.

6)The () Operator is used to Invoke the Function

Method:

1) an object property that has a function value.

2)object = {

methodName: function() {

// Content

}

};

object.methodName()

3)The method operates the data contained in a Class.

4)A method implicitly passes the object on which it was called.

5)A method is a function associated with an object property.

6)We can access object method by the following the syntax:

objectName.methodName()

=======================

**Arrays:**

// Declaration of an empty array

// using Array constructor

let names = new Array();

console.log(names);

// Creating and Initializing an array with values

let courses = new Array("HTML", "CSS", "Javascript", "React");

console.log(courses);

// Initializing Array while declaring

let arr = new Array(3);

arr[0] = 10;

arr[1] = 20;

arr[2] = 30;

console.log(arr);

Output

[]

[ 'HTML', 'CSS', 'Javascript', 'React' ]

[ 10, 20, 30 ]

1. Accessing Elements of an Array

// Creating an Array and Initializing with Values

let courses = ["HTML", "CSS", "Javascript", "React"];

// Accessing Array Elements

console.log(courses[0]);

5. Adding Elements to the Array

// Add Element to the end of Array

courses.push("Node.js");

// Add Element to the beginning

courses.unshift("Web Development");

// Removes and returns the last element

let lastElement = courses.pop();

console.log("After Removed the last elements: " + courses);

// Removes and returns the first element

let firstElement = courses.shift();

console.log("After Removed the First elements: " + courses);

// Removes 2 elements starting from index 1-----CSS,Javascript,React

courses.splice(1, 2);

console.log("After Removed 2 elements starting from index 1: " + courses);------CSS

7. Array Length:

let len = courses.length;

Increase and Decrease the Array Length:

// Creating an Array and Initializing with Values

let courses = ["HTML", "CSS", "Javascript", "React", "Node.js"];

// Increase the array length to 7

courses.length = 7;

console.log("Array After Increase the Length: ", courses);

// Decrease the array length to 2

courses.length = 2;

console.log("Array After Decrease the Length: ", courses)

Iterating Through Array Elements:

// Iterating through for loop

for (let i = 0; i < courses.length; i++) {

console.log(courses[i])

}

// Iterating through forEach loop

courses.forEach(function myfunc(elements) {

console.log(elements);

});

// Creating an Array and Initializing with Values

let courses = ["HTML", "CSS", "JavaScript", "React"];

let otherCourses = ["Node.js", "Expess.js"];

// Concatenate both arrays

let concateArray = courses.concat(otherCourses);

console.log("Concatenated Array: ", concateArray);

// Creating an Array and Initializing with Values

let courses = ["HTML", "CSS", "JavaScript", "React"];

// Convert array ot String

console.log(courses.toString());

console.log("Using Array.isArray() method: ", Array.isArray(courses))

console.log("Using instanceof method: ", courses instanceof Array)

Join():

console.log(courses.join('|'));-------->HTML|CSS|JavaScript|React

delete:

delete object

// or

delete object.property

// or

delete object['property']

ex:

let emp = {

firstName: "Raj",

lastName: "Kumar",

salary: 40000

}

console.log(delete emp.salary);

console.log(emp);

Output

true

{ firstName: 'Raj', lastName: 'Kumar' }

some(): check any condition satified

function isGreaterThan5(element, index, array) {

return element > 5;

}

// Driver code

// Original array

let array = [2, 5, 8, 1, 4];

// Checking for condition in array

let value = array.some(isGreaterThan5);

console.log(value);

Date:

let date = new Date();

let year = date.getFullYear();

let month = date.getMonth(); // Note: Month is zero-based (0 for January, 11 for December)

let day = date.getDate();

let hours = date.getHours();

let minutes = date.getMinutes();

let seconds = date.getSeconds();