# Javascript Output

## Javascript display possibilities.

* InnerHTML : Writing into an HTML element using InnerHTML.
  + - Ex: document.getElementById(‘demo’).InnerHTML = ‘Text’;
    - This will display the output in a HTML element ( Div or P tag)
* document.write : Writing into an HTML element using document.write.
  + - Ex: document.write(‘Text’).
    - This will display the out in the web browser.
* Alert : Writing into an alert box by using alert method.
  + - Ex: alert(‘text’);
    - This will display on page load before DOM content will loaded.
* Console.log : Writing in to the content in the browser by using console.log
  + - Ex: console.log(‘text’);
    - This will display the content in the console.
    - Right click -> Inspect-> console.

# Javascript statements

* Ex:
  + Let x, y, z;
  + Let x = 10;
  + Let y = 10;
  + document.getElementById(‘demo’).InnerHTML = ‘Text’;

# Javascript commenting

Ex:

Single comment use //

Multi comment use /\* xxxxxxxxxxxxxxxxxxxxxxxxxx \*/

# Javascript variables

We can define variables in 3 types

* let
* var
* const

Ex 1:

var x = 10;

var y = 20;

var z = x + y;

document.write(‘The sum is’ + z);

Ex 2:

let x = 10;

let y = 20;

let z = x + y;

document.write(‘The sum is’ + z);

Ex 3:

const x = 10;

const y = 20;

const z = x + y;

document.write(‘The sum is’ + z);

## Difference between var vs let vs const.

By using var, we can re-declare the variable.

By using var, we can re-assign a value to the variable.

var fName = ‘Welcome’;

xxxxxx

xxxxxx

xxxxxx

var fName = ‘Welcome India’;

By using let, we can’t re-declare the variable.

By using let, we can re-assign the variable.

let fName = ‘Welcome’;

xxxxxx

xxxxxx

xxxxxx

let fName = ‘Welcome India’; 

fName = ‘Welcome India’; 

By using const, we can’t re-declare the variable.

By using const, we can’t re-assign the variable.

const fName = ‘Welcome’;

xxxxxx

xxxxxx

xxxxxx

const fName = ‘Welcome India’; 

fName = ‘Welcome India’; 

# Javascript Arithmetic Operators.

## Addition +

Ex: let a , b;

a=10; b=20;

z = a+b;

Subtraction -

Ex: let a , b;

a=10; b=20;

z = a - b;

Multiplication (\*) -

Ex: let a , b;

a=10; b=20;

z = a \* b;

Division (/) - (will give quotient)

Ex: let a , b;

a=20; b=10;

z = a/ b; - Output is 2

Module (%) - (will give reminder)

Ex: let a , b;

a=20; b=10;

z = a % b; - Output is 0.

Increment ( ++) -

Ex: let a = 2

a++; - Output is 3.

Decrement ( --) -

Ex: let a = 2

a--; - Output is 1.

# Javascript Logical Operators.

## Greaterthan ( > )

Ex: let a = 2;

let b = 6;

if(b > a)

{

return true;

}

## Lessthan ( < )

Ex: let a = 2;

let b = 6;

if(a < b)

{

return true;

}

## Greaterthanorequalto ( >= )

Ex: let a = 2;

let b = 2;

if(b >= a)

{

return true;

}

## Lessthanorequalto ( <= )

Ex: let a = 2;

let b = 2;

if(b <= a)

{

return true;

}

# Javascript assignment Operators.

|  |  |  |
| --- | --- | --- |
| Operator | Usage | Actual functionality. |
| = | x = y | Assign y value to x. |
| += | x += y | x = x +y;  let x = 5;  let y = 10;  x = x + y;  x += y  output : 15; |
| -= | x -= y | x = x - y; |
| \*= | x \*= y | x = x \* y; |
| /= | x /= y | x = x / y; |
| %= | x %= y | x = x % y; |

# Javascript datatypes

* String
* Number
* Boolean
* BigInt
* Undefined
* Null
* Object
  + Object
  + Array.

String:

Ex: let fName = ‘Emplicare’;

fName = “New emplicar”;

Number:

Ex:

var a = 10;

var a = 10.50;

Note: Integers can only accurate upto 15 digits.

Boolean:

Ex:

let is\_active = true;

let is\_active = false;

BigInt.

Let bigint\_var = BigInt(1234567890123456789012345);

Note: A bigint variable can hold more than 15 digits.

Undefined :

Without defining the variable try to execute the variable.

Ex:

document.write(fName);

typeOf(fName) -> Undefined.

fName variable not defined.

Null:

Null value means nothing, mean it should not contain either ‘0’ or ‘blank’.

let department = NULL;

Object :

A object contains one or more values with key & value pair.

Let person= {fName:’Emplicar’, ‘lName’ : ‘Software solutions’};

How to get value from the object.

person. fName : Output -> Emplicar.

Array.

A array contains one or more values with index based or key based.

Index based is the default array

Key based is the associate array.

Default array:

Ex:

let student = [‘Ben’, ‘Can’, ‘Ram’];

How to get the value from the default array.

Student[2] : Output -> Ram.

Associative array:

Ex:

let users = [‘internal’=>’Staff’, ‘external’=>’Students’];

How to get the value from the associative array.

users [‘internal’] = Staff.

# Javascript functions.

What is the function?

To perform / execute a single task is called function.

Functions are two types

1. Void
2. Non-void

Void functions should not return any value.

Ex:

function displayMessage(msg)

{

document.write(msg);

}

displayMessage(‘My new message’);

Ex: Non-void.

function addingtwoNumbers(a , b)

{

return a + b;

}

let result = addingtwoNumbers(25,10);

displayMessage(result);

function addingtwoNumberswithtrycatch(a , b)

{

try{

return a + b;

}catch(err){

document.write(err.message());

}

}

addingtwoNumberswithtrycatch (25,10);

# Objects.

How to define a object.

let employees = {};

How to add values to an object.

employees.fName = ‘Emplicar’;

employees.lName = ‘Software Solutions’;

employees.cources = [‘PHP’, ‘Java’];

employees.fee = 10;

How to get the value from an object.

document.write(employes.lName);

# Javascript events.

* Onchange
  + - * Whenever user change the value of the DOM element, this event will be triggered

Ex:

<input type="text" id ="salary\_user" value="" onchange="pushsalary();">

* + - Onclick
      * Whenever user click on the DOM element, this event will be triggered.

Ex:

<button onclick="loaddynamic();">Click me</button>

* + - Onmouseout

Whenever user remove the mouse pointer from the DOM element, this event will be triggered.

<input type="text" id ="user\_name" value="" onmouseout="displayname();">

* + - Onmouseover

Whenever user move the mouse pointer on to the DOM element, this event will be triggered.

<input type="text" id ="user\_grade" value="" onmouseover="displayGrade();">

* + - Onkeydown

Whenever user put the key down from the keyboard, this event will be triggered.

<input type="text" id ="user\_grade" value="" onkeydown="keydownmethod()">

* + - Onkeyup

Whenever user release the key from the keyboard, this event will be trigged.

<input type="text" id ="user\_grade" value="" onkeyup="keydownmethod()">

* + - Onload

When the page loads , this event will be triggered.

<Body onload="loadDynamicContent()">

# String methods.

* String length: To find the string length.

Ex:

let business = “Software”;

business.length();

* String slice: To split the string from start position to end position. To extract string from main string for given start position & end position.

let business = “Software Solutions”

business.slice(7,10);

* String replace: This method will replace a specified value with another value in the string.

let business = “Software Solutions”;

business.replace(“Software” , “Hardware”);

* String toUpperCase: This method will modify all the string values into upper cases

let business = “Software”

business.toUppercase();

// Output : ( SOFTWARE).

* String toLowerCase: This method will modify all the string values into lower cases

let business = “SOFTWARE”

business.toUppercase();

Output ( software).

* String concat : Concatenating one or more string values in to a single string.

let student = “ABC”;

let student1 = “XYZ”

student.concat(“ “, student1)

// Output student student1.

* String trim : This method will remove the white spaces before and after of the string.

let student = “ Hello “;

student.trim();

// Output -> Hello.

* String rtrim : This method will remove the white spaces after of the string.

let student = “ Hello “;

student.trim();

// Output -> “ Hello.”;

* String ltrim : This method will remove the white spaces before of the string.

let student = “ Hello “;

student.trim();

// Output -> “Hello. ”;

* String pad: This method will replace the given string or symbol infront of the main string.

let student = “John”;

Student.pad(15,”$”)

// output $$$$$$$$$$$John

* String padstart: This method similar to String pad.
* String padend: This method will replace the given string or symbol after the main string.

let student = “John”;

Student.padend(15,”$”)

// output John$$$$$$$$$$$

* String CharAt: To find the value of the character based on the index from the main string.

let student = “Ben”

student.charAt(0) – Output is “B”.

* String split : To convert the string into array based on the delimited provided.

let text = “Welcome to Javascript Classes”;

text.split(“ “).

// out put - > 3 index array.

* String indexOf : To find the first occurrence index position of the search string from the main string.

let text = “Welcome to Javascript to Classes”;

text.indexOf(“to”);

// Output -> 8

text.indexOf(“to”, 12); After the 12th position, it will search next first occurrence index position.

// Output -> 21.

* String Search : To find the first occurrence index position of the search string from the main string.

let text = “Welcome to Javascript to Classes”;

text.search(“to”);

// Output -> 8

Search will always return the first occurrence, because in the search function we can’t pass 2 parameter like indexOf.

# Array methods

* array.length : To find the length of the array.

Ex: let x = [10,20,30];

let xlen = x.length;

document.write(xlen) : output -> 3

* array.toString : Convert array to string.

Ex: let x = [8,9,5,2,4];

let xstr = x.toString();

document.write(xstr);

* array.push : Adding one or more values to the end of the array.

EX:

let fruits = [‘Banana’ , ‘Orange’];

fruits.push(‘Apple’);

output is -> [‘Banana’ , ‘Orange’ , ‘Apple’];

* array.pop: Remove one element at the end of the array.

Ex:

let fruits = [‘Banana’ , ‘Orange’ , ‘Apple’];

let fruits\_new\_array = fruits.pop();

document.write(fruits\_new\_array);

Output : Banana, Orange.

* array.shift: Adding one ore more values to the beginning of the array.

Ex:

let fruits = [‘Banana’, ‘Orange’];

let fruits\_new\_array = fruits.shift(‘Apple’);

document.write(fruits\_new\_array);

Output : Apple, Banana, Orange.

* array.unshift: Removing one value from the beginning of the array.

let fruits = [‘Banana’, ‘Orange’];

let fruits\_new\_array = fruits.unshift();

document.write(fruits\_new\_array);

Output : Orange.

* array.concat : Merging one ore more arrays into a single array.

EX:

let languages = [‘Java’, ‘PHP’];

let fruits = [‘Apple’, ‘Orange’];

languages.concat(fruits);

document.write(languages);

Java, PHP , Apple, Orange.

* srray.splice : Adding one ore more elements to the array in the specified position.

Ex:

Let xy = [100,200,300,400];

Xy.splice(3,1,’Kiwi’,’Apple’);

No of elements to add to an array

How many elements want to remove

Element Position

Output : 100,200,300,’Kiwi’,’Apple’.

Ex 2:

Let xy = [100,200,300,400];

Xy.splice(3,1);

How many elements want to remove

Element Position

Let xy = [100,200,300];

* array.slice : Remove element from the array, extract specific positioned element into a new array.
* Starting element exclude & End element include

Ex:

let x = [10,20,30,40,50,60,70,80,90];

let y = x.slice(4,7);

Output : 50,60,70.

* Array Soring :

By default ar,rays will be sort based on the alphabeta patten.

Ex:

let fruits = [‘Banana’ , ‘Orange’ , ‘Apple’];

fruits.sort();

Apple, Banana , Orange.

If array is a numeric values, then you have to do the callback function.

let x = [10,20,30,40, 50,60,70, 80,90];

x.sort(function(x,y) { return x-y });

* array.Reverse: This method will reverse all the array values from back to front.

let x = [10,20,30,40,50,60,70,80,90];

x.reverse();

Output : 90,8,70,60,50,40,30,20,10.

# Javascript loops

array.forEach : This loop will pass each value to the call back function as a parameter.

Ex:

const numbers = [455, 45, 95, 165, 255];

numbers.forEach(vasudev\_fn);

function vasudev\_fn(value) {

document.write(value);

document.write('<br />');

}

array.map : This method will receive each value of the array and performing task on each of the value and returns new array.

Ex:

const numbers = [455, 45, 95, 165, 255];

let newArray = numbers.map(vasudev\_fn);

function vasudev\_fn(value) {

return value \* 2;

}

array.filter: This method will receive each of the value and pass the test or condition and returns a new array.

Ex:

const numbers = [455, 45, 95, 165, 255];

let newArray = numbers.filter(vasudev\_fn);

function vasudev\_fn(value) {

return value >100;

}

array.reduce : The reduce() method runs a function on each array element to produce (reduce it to) a single value.

const numbers = [45, 4, 9, 16, 25];

let sum = numbers.reduce(fn\_reduce);

let total = 100;

function fn\_reduce(total, value, index, array) {

return total + value;

}

array.every : This method will check all the values are pass the test or condition and it will return true or false.

const numbers = [45, 4, 9, 16, 25];

let allOver18 = numbers.every(myFunction);

function myFunction(value, index, array) {

return value > 1;

}

# Javascript If / else condition.

If else condition:

Ex:

let x = 40;

if(x > 20)

{

// Statements.

}

else{

// Statements.

}

If else-if condition.

let x = 45;

if(x >= 10 && x < 20)

{

// x value between 10 and 20.

}

Else if (x >= 20 && x < 30) {

// x value between 20 and 30.

}

Else{

// statements.

}

# Javascript switch condition.

Ex:

let userole = ‘admin’;

switch(userole)

{

case : ‘admin’

// statements

break;

case : ‘user

// statements

break;

default:

// statements.

Break;

}

# Javascript for in Loop.

This loop specifically designed for objects, for In loops through the properties of the object.

Ex:

const person = {fname:"John", lname:"Doe", age:25};

let text = "";

for (let x\_key in person) {

document.write(person. x\_key);

document.write(‘<br />’);

}

}

# Javascript try catch.

Try catch feature will be used to capture the run-time exceptions in the Javascript. Ex: divided by any value with ‘0’;

function trycatch\_fn(){

try{

let x = 20;

let y = 3;

let z = y / x;

let x = 0;

let y = 3;

let z = y / x;

}

catch(error\_object){

document.write(error\_object.message)

// same message writting into logs.

}

}

# Javascript class & object.

Class : Classes are a template for creating objects

Object: Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values.

Objects of a class are created using the new keyword.

Example:

class Car {

constructor(name, year) {

this.name = name;

this.year = year;

}

age() {

const date = new Date();

return date.getFullYear() - this.year;

}

}

const myCar = new Car("Ford", 2016);

const myCar2 = new Car("Maruthi", 2015);

document.getElementById("dynamicDemo").innerHTML =

"My car is " + myCar.age() + " years old.";

# Javascript Objects:

A JavaScript object is a collection of **named values like key value pair**

const person = {  
  firstName:"John",  
  lastName:"Doe",  
  age:50,

eyeColor:"blue"  
}  
  
const x = person;  
x.age = 10;      // Will change both x.age and person.age

## **Javascript object properties:**

Properties are the values associated with a JavaScript object.

*objectName.property*// person.age

*objectName*["*property*"]   // person["age"]

## **Object loop**

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

## **Delete a property from an object**.

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

## **Nested object**.

myObj = {  
   name:"John",  
  age:30,  
   cars: {  
    car1:"Ford",  
     car2:"BMW",  
    car3:"Fiat"

},

features:{

milage : 40,

color : red

}

}

myObj.cars.car2;

## **Javascript object methods.**

## .

const **person** = {  
  firstName: "John",  
   lastName: "Doe",  
   id: 5566,  
   fullName: function() {  
     return **this**.firstName + " " + **this**.lastName;  
   }  
};

let fullname = person.fullName();

## **Display Javascript object properties.**

const person = {  
 name: "John",  
 age: 30,  
 city: "New York"  
};  
  
document.getElementById("demo").innerHTML =  
person.name + "," + person.age + "," + person.city;

## **JavaScript Accessors (Getters and Setters)**

const person = {  
  firstName: "John",  
  lastName: "Doe",  
  language: "en",

get lang() {  
    return this.language;  
  }  
};  
  
// Display data from the object using a getter:  
document.getElementById("demo").innerHTML = person.lang;

const person = {  
  firstName: "John",  
  lastName: "Doe",  
  language: "",  
  set lang(lang) {  
    this.language = lang;  
  }  
};  
  
// Set an object property using a setter:  
person.lang = "en";  
  
// Display data from the object:  
document.getElementById("demo").innerHTML = person.language;

## **JavaScript function definition**

function functionName(parameters) {  
  // code to be executed  
}

## **JavaScript function parameters**

## function myFunction(x, y) {    if (y === undefined) {     y = 2;   } }

function myFunction(x, y = 10) {  
  return x + y;  
}  
myFunction(5);