JAVASCRIPT

Accessing HTML element in Javascript:

document.getElementById(Id)

Changing the HTML element:

innerHTML

document.getElementbyId(Id).innerHTML or document.write()

window.alert() --- shows alert box to display data

The window object is the global scope object, that means that variables, properties, and methods by default belong to the window object.

So we can write the above command by skipping window like alert()

Console.log() for debugging;

Window.print() for printing the page

Fixed values are called Literals. Variable values are called Variables.

Javascript is case sensitive.

Comment. - // or /\* \*/

Variables declaration –

var – global scope, can be redeclared anywhere

let – scope is within the block, can be declared only once inside a block

const – block scope, cant be reassigned or redeclared

use const when declaring – an array, regexp, object or function

typeof – returns the data type of a variable

this()  -- owner of the function

new – creates a new object

slice ( start, end ) – gives part of the string between start and end

if the parameter is negative it starts from the end of the string.

Substring(start,end) – similar to slice but cannot except negative parameters.

Substr(start,end) – 1st parameter is the starting position but 2nd parameter specifies the length of the string to be extracted.

Replace() – replaces the first matched string

i – ignore case. g – replaces all the matched string

padStart(x,y) and padEnd() – padding x number of spaces with y

split () converts string to array

indexOf(), and lastIndexOf() -returns the position of the searched string

string.match() – returns array of matched strings within the main string

ex. - text.match(/ain/g)

includes() – returns true or false based on whether the matched string is available or not.

startsWith() , endsWith() – returns true or false

NaN – Not a Number

Infinity

toFixed(). And toPrecision()

valueOf() , toParseInt(), toParseFloat()

MIN\_VALUE, MAX\_VALUE – gives the minimum and maximum available number in Javascript

Array – pop() and push() -remove and add elements to/from the end of an array.

Shift() and unshift() – removes / adds element at the beginning of an array.

Sort() , resverse() – Array functions

Comparison - == and === ( equal and equal type )

Conditions:

If (condition) {stmt;}

If(condition) { stmt;} else { stmt;}

if(condition) { stmt; } else if (condition) { stmt; } else { stmt; }

switch(condition) {

case x:

break;

case y:

break;

case z:

break;

default:

}

For ( i=0; i<5; i++) { }

For ( variable in object) { code }

For ( variable of iterable ) { code }

While (condition) { loop code }

Do { code } while ( condtion);

 unary + operator can be used to convert a variable to a number

let y = "5";      // y is a string  
let x = + y;      // x is a number

try {

} catch(err) {}

Throw err;

Use strict; // mode helps to write cleaner code. All the variables needs to be declared in this mode.

=>(arrow function) helps to write a shorter function syntax.

JSON – JavaScript Object Notation

Debugging –

Console.log()

Debugger;

Objects :

3 ways to create them

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

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

const person = new Object();  
person.firstName = "John";  
person.lastName = "Doe";  
person.age = 50;  
person.eyeColor = "blue";

Object Properties:

Object.property or Object[“property”]

Add a new property

person.nationality = “Indian”;

delete a property

delete person.age; or delete person[“age”];

Nested:

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

access - myObj.cars.car2; or myObj.cars["car2"]; or myObj["cars"]["car2"];

Nested objects and array

const myObj = {  
  name: "John",  
  age: 30,  
  cars: [  
    {name:"Ford", "models":["Fiesta", "Focus", "Mustang"]},  
    {name:"BMW", "models":["320", "X3", "X5"]},  
    {name:"Fiat", "models":["500", "Panda"]}  
  ]  
}

Object method:

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

Access - name = person.fullName();

Converts object to an array :

const person = {  
  name: "John",  
  age: 30,  
  city: "New York"  
};  
  
const myArray = Object.values(person);

Convert to a string : let myString = JSON.stringify(person);

const person = {  
  name: "John",  
  today: new Date()  
};  
  
let myString = JSON.stringify(person);  
document.getElementById("demo").innerHTML = myString;

It will not stringify a function.

It can stringify an array.

const arr = ["John", "Peter", "Sally", "Jane"];

document.getElementById("demo").innerHTML = JSON.stringify(arr);

get() and set() for getting and setting property in an object

Object DefineProperty:

const obj = {counter : 0};

// Define Setters and Getters

Object.defineProperty(obj, "reset", {

get : function () {this.counter = 0;}

});

Object.defineProperty(obj, "increment", {

get : function () {this.counter++;}

});

Object.defineProperty(obj, "decrement", {

get : function () {this.counter--;}

});

Object.defineProperty(obj, "add", {

set : function (value) {this.counter += value;}

});

Object.defineProperty(obj, "subtract", {

set : function (value) {this.counter -= value;}

});

Cosntructor:

function Person(first, last, age, eyecolor) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eyecolor;  
}  
  
const myFather = new Person("John", "Doe", 50, "blue");  
const myMother = new Person("Sally", "Rally", 48, "green");

Prototype property to the object constructor:

Person.prototype.nationality = "English";

Person.prototype.name = function() {  
  return this.firstName + " " + this.lastName;  
};

Map() and Set()

Functions:

Argument object:

x = findMax(1, 123, 500, 115, 44, 88);  
  
function findMax() {  
  let max = -Infinity;  
  for (let i = 0; i < arguments.length; i++) {  
    if (arguments[i] > max) {  
      max = arguments[i];  
    }  
  }  
  return max;  
}

call() method:

const person = {  
  fullName: function() {  
    return this.firstName + " " + this.lastName;  
  }  
}  
const person1 = {  
  firstName:"John",  
  lastName: "Doe"  
}  
const person2 = {  
  firstName:"Mary",  
  lastName: "Doe"  
}  
  
// This will return "John Doe":  
person.fullName.call(person1);

apply() method is similar to call ()

person.fullName.apply(person1);

difference :

call() – argument is taken separately person.fullName.call(person1, "Oslo", "Norway");

apply() – argument is taken as an array

person.fullName.apply(person1, ["Oslo", "Norway"]);

Class:

Class myClass {

Constructor(name, car){

This.name = name;

This.car = car;

}

}

Let myCar = new myClass(“Accord”,”Honda”);

Inheritance : child class extends properties from parent class

Callbacks : function passed as an argument to another function

function myDisplayer(some) {  
  document.getElementById("demo").innerHTML = some;  
}  
  
function myCalculator(num1, num2, myCallback) {  
  let sum = num1 + num2;  
  myCallback(sum);  
}  
  
myCalculator(5, 5, myDisplayer);

Callback after a timeout – setTimeout(myFunction, 3000);

HTML DOM:

Document.getElementById(“Id”);

Document.getElementsByTagName("p");

document.getElementsByClassName("intro");

document.querySelectorAll("p.intro");

document.getElementById("p2").style.color = "blue";

JSON:

JSON.parse(text);

JSON.stringify(text);