1. **The "use strict" Directive :**

You can use strict mode in all your programs. It helps you to write cleaner code, like preventing you from using undeclared variables.

"use strict" is just a string expression. Old browsers will not throw an error if they don't understand it.

"use strict";

1. **charAt() :**

The charAt() method returns the character at a specified index (position) in a string

var str = "HELLO WORLD";  
str.charAt(0);

## Strings Over Multiple Lines

ES5 allows string literals over multiple lines if escaped with a backslash

"Hello \  
Dolly!";

A safer way to break up a string literal, is to use string addition:

"Hello " +  
"Dolly!";

## Reserved Words as Property Names :

## ES5 allows reserved words as property names:

## var obj = {name: "John", new: "yes"};

## document.getElementById("demo").innerHTML = obj.new;

## String trim()

## The trim() method removes whitespace from both sides of a string.

## var str = "       Hello World!        "; alert(str.trim());

## Array.isArray()

## The isArray() method checks whether an object is an array.

## o/p => yes

## Array forEach()

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

let txt = "";

fruits.forEach(displayFruit);

function displayFruit(value){

    console.log(value);

    txt = txt + value + "<br>";

}

## Array map()

## This example multiplies each array value by 2:

var numbers1 = [45, 4, 9, 16, 25];

/\* Map \*/

var numbers1 = [45, 4, 9, 16, 25];

var numbers2 = numbers1.map(getModifiedNumber);

function getModifiedNumber(value){

    return value \* 2;

}

document.getElementById("p5").innerHTML = numbers2;

## Array reduce()

This example finds the sum of all numbers in an array:

## Array reduceRight()

## var numbers1 = [45, 4, 9, 16, 25]; var sum = numbers1.reduceRight(myFunction); function myFunction(total, value) {   return total + value; }

## Array every()

## This example checks if all values are over 18:

/\* Reduce \*/

var sum = numbers1.reduce(getSumOfNumbers);

function getSumOfNumbers(total,value){

    return total + value;

}

document.getElementById("p5").innerHTML = sum;

## Array some()

This example checks if some values are over 18:

var anyoneOver18 = numbers1.some(isAnyoneOver18);

function isAnyoneOver18(value){

    return value > 18;

}

document.getElementById("p5").innerHTML = anyoneOver18;

## Array indexOf()

## Search an array for an element value and returns its position.

/\* indexOf() \*/

var index = fruits.indexOf("Apple");

document.getElementById("p5").innerHTML = index;

## Array lastIndexOf()

lastIndexOf() is the same as indexOf(), but searches from the end of the array

var lastIndex = fruits.lastIndexOf("Apple");

document.getElementById("p5").innerHTML = lastIndex;

## JSON.parse()

## Imagine you received this text string from a web server:

/\* parse the json \*/

var empStr =  '{"name":"John", "age":30, "city":"New York"}';

var empJson = JSON.parse(empStr);

console.log(empStr);

console.log(empJson);

## JSON.stringify()

When sending data to a web server, the data has to be a string.

## Date.now()

## Date.now() returns the number of milliseconds since zero date (January 1. 1970 00:00:00 UTC).

## var timInMSs = Date.now();

## note:

## Date.now() returns the same as getTime() performed on a Date object.

## Date toISOString()

## The toISOString() method converts a Date object to a string, using the ISO standard format:

## const d = new Date(); document.getElementById("demo").innerHTML = d.toISOString();

## Date toJSON()

toJSON() converts a Date object into a string, formatted as a JSON date.

JSON dates have the same format as the ISO-8601 standard: YYYY-MM-DDTHH:mm:ss.sssZ:

## d = new Date(); document.getElementById("demo").innerHTML = d.toJSON();

## Property Getters and Setters

ES5 lets you define object methods with a syntax that looks like getting or setting a property.

This example creates a **getter** for a property called getFirstname:

And setter for a property call setFirstname

var person = {

    firstname :'johan',

    lastname :'siddle',

    get getFirstname() {

        return this.firstname + " " + this.lastname;

    },

    set getFirstname(name){

        this.firstname = name;

    }

}

console.log('firstname name is: ' + person.getFirstname);

person.getFirstname = 'Peter';

console.log('After set firstname name is: ' + person.getFirstname);

## Object.defineProperty()

Object.defineProperty() is a new Object method in ES5.

It lets you define an object property and/or change a property's value and/or metadata.

var person = {

    firstName: "John",

    lastName : "Doe",

    language : "NO",

};

Object.defineProperty(person,'language',{

    value :'EN',

    enumerable : true,

    configurable : true,

    writable : true

});

console.log(person);

/\*If enumeration property false then we cant access the value\*/

Object.defineProperty(person, "language", {

    value: "EN",

    writable : true,

    enumerable : false,

    configurable : true

  });

var txt2 = "";

for (var x in person) {

  txt2 += person[x] + "<br>";

}

document.getElementById("p5").innerHTML = txt2;

This example creates a setter and a getter to secure upper case updates of language:

/\* define property with getter and setter \*/

var person3 = {

    firstName: "John",

    lastName : "Doe",

    language : "NO",

};

Object.defineProperty(person3,'language',{

    get language(){

        return this.language;

    },

    set language(value){

        this.language = value;

    }

});

console.log(person3.language);

person3.language = 'IN';

console.log(person3.language);

## Function Bind() :

With the bind() method, an object can borrow a method from another object.

This example creates 2 objects (person and member).

The member object borrows the fullname method from the person object: