Arrays in JavaScript

In JavaScript, array is a single variable that is used to store different elements. It is often used when we want to store list of elements and access them by a single variable. Unlike most languages where array is a reference to the multiple variable, in JavaScript array is a single variable that stores multiple elements.

**Declaration of an Array**

There are basically two ways to declare an array.

1. var myArray = [];
2. var myArray = new Array();

**Initialization of an Array**

Var myArray = [22, 45, 74, 12, 10,];

Var myArray = new Array (22, 45, 74, 12, 10,] ;

**An array in JavaScript can hold different elements**we can store Numbers, Strings and Boolean etc. in a single array.

Var myArray = [22,’Honda’ 45, true,’Apple’, null, 74, 12, 10,];

A sample JavaScript program Example.

var nums = [5, 1, 41,78,25,45,5]

console.log (nums.sort () +'<br>');

var mixedArray =[2,'Minhaj',null,{name:'Hungary',capital:'Budapest'},true];

console.log (mixedArray);

var val=nums.sort(function(x,y)// Sorting Array in Ascending Order

{

return x-y;

});

function below30 (num)// It will return number below 30

{

return num>30;

}

var belownum = nums.find(below30);

document.write (belownum+'<br>');

console.log (belownum);

The Date Object in JavaScript

The Date [object](https://www.digitalocean.com/community/tutorials/understanding-objects-in-javascript) is a built-in object in JavaScript that stores the date and time. It provides a number of built-in methods for formatting and managing that data.

By default, a new Date instance without arguments provided creates an object corresponding to the current date and time. This will be created according to the current computer's system settings.

**[Creation](https://javascript.info/date" \l "creation)**

To create a new Date object call new Date () with one of the following arguments:

new Date ()

Without arguments – create a Date object for the current date and time:

let now = new Date ();

new Date (datestring)

If there is a single argument, and it’s a string, then it is parsed with the Date.parse algorithm (see below).

let date = new Date ("2017-01-26");

**[Access date components](https://javascript.info/date" \l "access-date-components)**

There are many methods to access the year, month and so on from the Date object.

[getFullYear ()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getFullYear)

Get the year (4 digits)

[getMonth ()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getMonth)

Get the month, from 0 to 11.

[getDate ()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getDate)

Get the day of month, from 1 to 31, the name of the method does look a little bit strange.

[getHours ()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getHours), [getMinutes ()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getMinutes), [getSeconds ()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getSeconds), [getMilliseconds ()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getMilliseconds)

[getDay()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getDay)

Get the day of week, from 0 (Sunday) to 6 (Saturday). The first day is always Sunday, in some countries that’s not so, but can’t be changed.

[getTime ()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getTime)

Returns the timestamp for the date – a number of milliseconds passed from the January 1st of 1970 UTC+0.

**[Setting date components](https://javascript.info/date" \l "setting-date-components)**

The following methods allow to set date/time components:

* [setFullYear(year [, month, date])](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/setFullYear)
* [setMonth(month [, date])](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/setMonth)
* [setDate(date)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/setDate)
* [setHours(hour [, min, sec, ms])](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/setHours)
* [setMinutes(min [, sec, ms])](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/setMinutes)
* [setSeconds(sec [, ms])](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/setSeconds)
* [setMilliseconds(ms)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/setMilliseconds)
* [setTime(milliseconds)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/setTime)

A sample JavaScript program Example.

var today=new Date();

console.log (today);

var time = today.getTime();

console.log (time);

const dob = new Date('September 10 2015');

var Birthday = dob;

console.log (Birthday);

JavaScript If Else Statements

**If Statement**

The most fundamental of the conditional statements is the if statement

JavaScript If statements are an example of conditional statements. With If statements, you can tell the browser to execute a piece of code only if a given condition is true.

An if statement is written with the if keyword, followed by a condition in parentheses, with the code to be executed in between curly brackets. In short, it can be written as if () {}.

**Else Statement**

With if statements, we only execute code when a statement evaluates to true, but often we will want something else to happen if the condition fails.

The else statement is written after the if statement, and it has no condition in parentheses

**Syntax for a basic if...else  statement.**

**if** (condition) { // code that will execute if condition is true}

**else** { // code that will execute if condition is false}

A sample JavaScript program Example.

var color = 'Green'

var color2 = 'Yellow'

if( color == = color2)

{

console.log('Color is Same ');

}

else{

console.log('Color is Different ');

}

**Comparison Operator**

**===**

Is identical (is equal to and is of the same type)

JavaScript object deep comparison. Comparing x === y, where x and y are values, return true or false. Comparing x === y, where x and y are objects, returns true if x and y refer to the same object. Otherwise, returns false even if the objects appear identical.

var id = 100;

If( id === 100) // Here === will compare content and type

{

console.log (‘Correct’);

}

else {

console.log (‘IN Correct’);

}