**Ways of writing Java Script**

There are three ways of writting JavaScript depending on the platform :

* Inline Scripting
* Internal Scripting
* External Scripting

**Internal Scripting:**

When JavaScript code are written within the HTML file itself, it is called internal scripting.

Internal scripting, is done with the help of HTML tag : <script> </script>

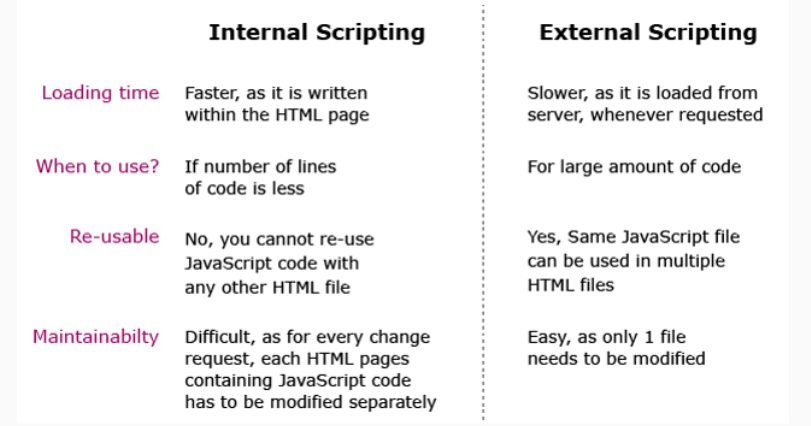
This tag can be placed either in the head tag or body tag within the HTML file.

**External Scripting**

If java script is written in external file and linked in HTML file then it is called external scripting

To include the external JavaScript file, the script tag is used with attribute 'src' in head of html as shown in the below-given code-snippet-

**Inline vs External scripting comparison**



**Ideftifiers**

Name of any variable. Rules for identifiers are: -

It can contain number, alphabet, underscore or $.

First character of identifier can be alphabet, underscore or $.

Identifiers are case sensitive.

**Identifier type: -**

Identifier can be classified in below three type-

* let
* const
* var

**let**

An identifier declared using ‘let’ keyword has a block scope i.e., it is available only within the block in which it is defined.

The value assigned to the identifier can be done either at the time of declaration or later in the code and can also be altered further.

let var\_name=value

let var\_name

Example:

let name="William";

console.log("Welcome to JS course, Mr."+name);

let name = "Goth"; /\* This will throw an error because the identifier  'name' has been already declared and we are redeclaring the variable, which is not allowed using the 'let' keyword. \*/

console.log("Welcome to JS course, Mr."+name);

**cont:**

If any variable is declared with cont then value of that variable can’t be changed again.

The identifier that hold data and data does not vary is called 'Constant' and to declare a constant, 'const' keyword is used, followed by an identifier. The value is initialized during the declaration itself and cannot be altered later.

const pi = 3.14;

console.log("The value of Pi is: "+pi);

 /\* This will throw an error because the assignment to a const needs to be done at the time of declaration

 and it cannot be re-initialized. \*/

 pi = 3.141592;

 console.log("The value of Pi is: "+pi);

**var**

The identifiers declared to hold data that vary are called 'Variables' and to declare a variable with 'var' keyword .

It is used for declaring any variable.

var variable\_name ----- Declaring variable

var variable\_name= value ------- Declaring and assigning value

Note:

As a best practice, use the 'var' keyword for variable declarations for function scope or global scope in the program.

|  |  |  |  |
| --- | --- | --- | --- |
| **Keyword** | **Scope** | **Declaration** | **Assignment** |
| let | Block | Redeclaration not allowed | Re-assigning allowed |
| const | Block | Redeclaration not allowed | Re-assigning not allowed |
| var | Function | Redeclaration allowed | Re-assigning allowed |

**Operator in javascript**

We can classify JavaScript in below type-

* Arithmetic Operators
* Assignment Operators
* Comparison Operators
* Logical Operators
* Conditional Operators
* Type Operators

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Arithmetic Operators | |  |  | | --- | --- | | **Operator** | **Description** | | + | Addition | | - | Subtraction | | \* | Multiplication | | \*\* | Exponentiation ([ES2016](https://www.w3schools.com/js/js_2016.asp)) | | / | Division | | % | Modulus (Division Remainder) | | ++ | Increment | | -- | Decrement | |
| Assignment operator | |  |  |  | | --- | --- | --- | | **Operator** | **Example** | **Same As** | | = | x = y | x = y | | += | x += y | x = x + y | | -= | x -= y | x = x - y | | \*= | x \*= y | x = x \* y | | /= | x /= y | x = x / y | | %= | x %= y | x = x % y | | \*\*= | x \*\*= y | x = x \*\* y | |
| Shift operator | |  |  |  | | --- | --- | --- | | **Operator** | **Example** | **Same As** | | <<= | x <<= y | x = x << y | | >>= | x >>= y | x = x >> y | | >>>= | x >>>= y | x = x >>> y | |
| Bitwise operator | |  |  |  | | --- | --- | --- | | **Operator** | **Example** | **Same As** | | &= | x &= y | x = x & y | | ^= | x ^= y | x = x ^ y | | |= | x |= y | x = x | y | |
| Logical operator | |  |  |  | | --- | --- | --- | | &&= | x &&= y | x = x && (x = y) | | ||= | x ||= y | x = x || (x = y) | | ??= | x ??= y | x = x ?? (x = y) | |
|  |  |

**Data Types:**

Java script have below 8 data types-

1. String

2. Number

3. Bigint

4. Boolean

5. Undefined

6. Null

7. Symbol

8. Object

**The Object Datatype**

The object data type can contain:

1. An object

2. An array

3. A date

**Function in java script**

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function if defined with function keyword and is executed when "something" invokes it (calls it).

function name(parameter1, parameter2, parameter3) {

// code to be executed

}

**Function Invocation**

The code inside the function will execute when "something" invokes (calls) the function:

* When an event occurs (when a user clicks a button)
* When it is invoked (called) from JavaScript code
* Automatically (self invoked)

**Java Script objects:**

A javaScript object is an entity having state and behavior.

JavaScript object is a non-primitive data-type that allows you to store multiple collections of data.

**Creating Objects in JavaScript**

We can create java script object in below three ways-

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

**JavaScript Object by object literal**

In this way , java script objects are declared in form of key:value pairs.

object\_name={property1:value1,property2:value2.....propertyN:valueN}

**By creating instance of Object**

In this method object are created using the new keyword.

var objectname=new Object();

Example:

<html>

<head>

</head>

<body>

    <h3>details of employee object is:</h3>

    <script>

        var emp=new Object();

        emp.id=101;

        emp.name="Ravi Malik";

        emp.salary=50000;

        document.write(emp.id+" "+emp.name+" "+emp.salary);

        </script>

</body>

</html>

**By using an Object constructor**

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

this keyword points to current object instance.

<html>

<head>

</head>

<body>

    <h3>details of employee object is:</h3>

    <script>

        // Create function

        function emp(id,name,salary){

        // aggign value to emp object using this keyword

        this.id=id;

        this.name=name;

        this.salary=salary;

        }

        e=new emp(103,"Vimal Jaiswal",30000);

        document.write(e.id+" "+e.name+" "+e.salary);

    </script>

</body>

</html>

**JavaScript Events**

HTML events are "things" that happen to HTML elements.

An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

* An HTML web page has finished loading
* An HTML input field was changed
* An HTML button was clicked

<element event=event\_handler>

Example:

<!DOCTYPE html>

<html>

<body>

#Button which will diplay the time on click

<button onclick="document.getElementById('demo').innerHTML=Date()">Click on button for time</button>

<p id="demo"></p>

</body>

</html>

**Common HTML event:**

We have below common html events that are frequently used.

|  |  |
| --- | --- |
| **Event** | **Description** |
| onchange | An HTML element has been changed |
| onclick | The user clicks an HTML element |
| onmouseover | The user moves the mouse over an HTML element |
| onmouseout | The user moves the mouse away from an HTML element |
| onkeydown | The user pushes a keyboard key |
| onload | The browser has finished loading the page |

We can check all events on below link-

<https://www.w3schools.com/jsref/dom_obj_event.asp>

**JavaScript Array**

JavaScript array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

* By array literal
* By creating instance of Array directly (using new keyword)
* By using an Array constructor (using new keyword)

**JavaScript array by literal**

var arrayname=[value1,value2.....valueN];

**JavaScript Array directly (new keyword)**

The syntax of creating array directly is given below:

var arr\_var= new Array()

In JavaScript we can create array without specifying the size of array and can initialize the value later.

<script>

var i;

var emp = new Array();

emp[0] = "Arun";

emp[1] = "Varun";

emp[2] = "John";

for (i=0;i<emp.length;i++){

document.write(emp[i] + "<br>");

}

</script>

**By using an Array constructor (using new keyword)**

Here, We need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

var emp=new Array(val1, val2 , ... valn);

**JavaScript Array Methods**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| [concat()](https://www.javatpoint.com/javascript-array-concat-method) | It returns a new array object that contains two or more merged arrays. |
| [copywithin()](https://www.javatpoint.com/javascript-array-copywithin-method) | It copies the part of the given array with its own elements and returns the modified array. |
| [entries()](https://www.javatpoint.com/javascript-array-entries-method) | It creates an iterator object and a loop that iterates over each key/value pair. |
| [every()](https://www.javatpoint.com/javascript-array-every-method) | It determines whether all the elements of an array are satisfying the provided function conditions. |
| [flat()](https://www.javatpoint.com/javascript-array-flat-method) | It creates a new array carrying sub-array elements concatenated recursively till the specified depth. |
| [flatMap()](https://www.javatpoint.com/javascript-array-flatmap-method) | It maps all array elements via mapping function, then flattens the result into a new array. |
| [fill()](https://www.javatpoint.com/javascript-array-fill-method) | It fills elements into an array with static values. |
| [from()](https://www.javatpoint.com/javascript-array-from-method) | It creates a new array carrying the exact copy of another array element. |
| [filter()](https://www.javatpoint.com/javascript-array-filter-method) | It returns the new array containing the elements that pass the provided function conditions. |
| [find()](https://www.javatpoint.com/javascript-array-find-method) | It returns the value of the first element in the given array that satisfies the specified condition. |
| [findIndex()](https://www.javatpoint.com/javascript-array-findindex-method) | It returns the index value of the first element in the given array that satisfies the specified condition. |
| [forEach()](https://www.javatpoint.com/javascript-array-foreach-method) | It invokes the provided function once for each element of an array. |
| [includes()](https://www.javatpoint.com/javascript-array-includes-method) | It checks whether the given array contains the specified element. |
| [indexOf()](https://www.javatpoint.com/javascript-array-indexof-method) | It searches the specified element in the given array and returns the index of the first match. |
| [isArray()](https://www.javatpoint.com/javascript-array-isarray-method) | It tests if the passed value ia an array. |
| [join()](https://www.javatpoint.com/javascript-array-join-method) | It joins the elements of an array as a string. |
| [keys()](https://www.javatpoint.com/javascript-array-keys-method) | It creates an iterator object that contains only the keys of the array, then loops through these keys. |
| [lastIndexOf()](https://www.javatpoint.com/javascript-array-lastindexof-method) | It searches the specified element in the given array and returns the index of the last match. |
| [map()](https://www.javatpoint.com/javascript-array-map-method) | It calls the specified function for every array element and returns the new array |
| [of()](https://www.javatpoint.com/javascript-array-of-method) | It creates a new array from a variable number of arguments, holding any type of argument. |
| [pop()](https://www.javatpoint.com/javascript-array-pop-method) | It removes and returns the last element of an array. |
| [push()](https://www.javatpoint.com/javascript-array-push-method) | It adds one or more elements to the end of an array. |
| [reverse()](https://www.javatpoint.com/javascript-array-reverse-method) | It reverses the elements of given array. |
| [reduce(function, initial)](https://www.javatpoint.com/javascript-array-reduce-method) | It executes a provided function for each value from left to right and reduces the array to a single value. |
| [reduceRight()](https://www.javatpoint.com/javascript-array-reduceright-method) | It executes a provided function for each value from right to left and reduces the array to a single value. |
| [some()](https://www.javatpoint.com/javascript-array-some-method) | It determines if any element of the array passes the test of the implemented function. |
| [shift()](https://www.javatpoint.com/javascript-array-shift-method) | It removes and returns the first element of an array. |
| [slice()](https://www.javatpoint.com/javascript-array-slice-method) | It returns a new array containing the copy of the part of the given array. |
| [sort()](https://www.javatpoint.com/javascript-array-sort-method) | It returns the element of the given array in a sorted order. |
| [splice()](https://www.javatpoint.com/javascript-array-splice-method) | It add/remove elements to/from the given array. |
| [toLocaleString()](https://www.javatpoint.com/javascript-array-tolocalestring-method) | It returns a string containing all the elements of a specified array. |
| [toString()](https://www.javatpoint.com/javascript-array-tostring-method) | It converts the elements of a specified array into string form, without affecting the original array. |
| [unshift()](https://www.javatpoint.com/javascript-array-unshift-method) | It adds one or more elements in the beginning of the given array. |
| [values()](https://www.javatpoint.com/javascript-array-values-method) | It creates a new iterator object carrying values for each index in the array. |

######################

# Conditional Statement #

######################

**If / If -else , if -elif- else statement**

if condition {

#Some action;

} else if condition {

#Some action;

} else {

#Some action;

}

Example: Write a javascript code to display the greeting message on button click.

<!DOCTYPE html>

<html>

<body>

<button onclick="greetings()">Click for Greetings</button>

<p id="demo"></p>

<script>

    function greetings(){

        const time = new Date().getHours();

        let greeting;

        if (time < 10) {

        greeting = "Good morning";

        } else if (time < 20) {

        greeting = "Good day";

        } else {

        greeting = "Good evening";

        }

        document.getElementById("demo").innerHTML = greeting;

    }

</script>

</body>

</html>

**JavaScript Switch statement:**

Use the switch statement to select one of many code blocks to be executed.

switch(expression) {

  case x:

// code block

break;

  case y:

// code block

break;

  default:

// code block

}

Example:

Write the javascript code to display the day name on button click

<!DOCTYPE html>

<html>

<body>

<button onclick="greetings()">Click getting day name</button>

<p id="demo"></p>

<script>

    function greetings(){

        let day

        switch (new Date().getDay()) {

        case 0:

            day = "Sunday";

            break;

        case 1:

            day = "Monday";

            break;

        case 2:

            day = "Tuesday";

            break;

        case 3:

            day = "Wednesday";

            break;

        case 4:

            day = "Thursday";

            break;

        case 5:

            day = "Friday";

            break;

        case  6:

            day = "Saturday";

        }

        document.getElementById("demo").innerHTML = day;

    }

</script>

</body>

</html>