JavaScript

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

* JavaScript is used to create client-side dynamic pages.
* JavaScript is an object-based scripting language which is lightweight and cross-platform.
* JavaScript is not a compiled language, but it is a translated language. The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.

## Advantages of JavaScript

The merits of using JavaScript are −

* **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
* **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something.
* **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
* **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

## JavaScript Development Tools

One of major strengths of JavaScript is that it does not require expensive development tools. You can start with a simple text editor such as Notepad. Since it is an interpreted language inside the context of a web browser, you don't even need to buy a compiler.

To make our life simpler, various vendors have come up with very nice JavaScript editing tools. Some of them are listed here −

* **Microsoft FrontPage** − Microsoft has developed a popular HTML editor called FrontPage. FrontPage also provides web developers with a number of JavaScript tools to assist in the creation of interactive websites.
* **Macromedia Dreamweaver MX** − Macromedia Dreamweaver MX is a very popular HTML and JavaScript editor in the professional web development crowd. It provides several handy prebuilt JavaScript components, integrates well with databases, and conforms to new standards such as XHTML and XML.
* **Macromedia HomeSite 5** − HomeSite 5 is a well-liked HTML and JavaScript editor from Macromedia that can be used to manage personal websites effectively.

# JavaScript - Syntax

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.

You can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags.

The script tag takes two important attributes −

* **Language** − This attribute specifies what scripting language you are using. Typically, its value will be javascript.
* **Type** − This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

**<html>**

**<head>**

**<script** type="text/javascript"**>**

function msgDemo()

{

 alert("Hello World");

}

**</script>**

**</head>**

**<body>**

**<p>**Welcome to JavaScript**</p>**

**<form>**

**<input** type="button" value="click" onclick="msgDemo()"**/>**

**</form>**

**</body>**

**</html>**

# External JavaScript file

We can create external JavaScript file and embed it in many html page.

An external JavaScript file must be saved by .js extension.

Let’s create an external JavaScript file that prints Hello World in a alert dialog box.

**message.js**

function msgDemo()

{

 alert("Hello World");

}

**<html>**

**<head>**

**<script** type="text/javascript" src="message.js"**></script>**

**</head>**

**<body>**

**<p>**Welcome to JavaScript**</p>**

**<form>**

**<input** type="button" value="click" onclick="msgDemo()"**/>**

**</form>**

**</body>**

**</html>**

Types of JavaScript Comments

There are two types of comments in JavaScript.

1. Single-line Comment
2. Multi-line Comment

## JavaScript Single line Comment

It is represented by double forward slashes (//). It can be used before and after the statement.

## JavaScript Multi line Comment

It can be used to add single as well as multi line comments. So, it is more convenient.

It is represented by forward slash with asterisk then asterisk with forward slash. For example:

/\* your code here  \*/

Semicolons are Optional

Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements are placed on a separate line. For example, the following code could be written without semicolons.

<script language = "javascript" type = "text/javascript">

<!--

var1 = 10

var2 = 20

//-->

</script>

But when formatted in a single line as follows, you must use semicolons −

<script language = "javascript" type = "text/javascript">

<!--

var1 = 10; var2 = 20;

//-->

</script>

## Case Sensitivity

JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.

**NOTE** − Care should be taken while writing variable and function names in JavaScript.

# JavaScript - Placement in HTML File

There is a flexibility given to include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTML file are as follows −

* Script in <head>...</head> section.
* Script in <body>...</body> section.
* Script in <body>...</body> and <head>...</head> sections.
* Script in an external file and then include in <head>...</head> section.

# Variables and Datatypes in JavaScript

JavaScript supports 3 primitive datatypes:

* **Numbers**  5, 6.5, 789
* **String**  “Hello ”
* **Boolean**  true, false.

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

1. Name must start with a letter (a to z or A to Z), underscore( \_ ), or dollar( $ ) sign.
2. After first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.
4. Reserved words (like JavaScript keywords) cannot be used as names

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.

<script type = "text/javascript">

var num;

var name;

</script>

You can also declare multiple variables with the same **var** keyword as follows −

<script type = "text/javascript">

var num , name;

</script>

<script type = "text/javascript">

var name = "Geu";

var num;

num = 25.90;

</script>

JavaScript is **untyped** language. This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

## JavaScript Variable Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

* **Global Variables** − A global variable has global scope which means it can be defined anywhere in your JavaScript code.
* **Local Variables** − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

JavaScript Reserved Words

A list of all the reserved words in JavaScript are given in the following table. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

|  |  |  |  |
| --- | --- | --- | --- |
| **abstract** | **else** | **instanceof** | **switch** |
| **boolean** | **enum** | **int** | **synchronized** |
| **break** | **export** | **interface** | **this** |
| **byte** | **extends** | **long** | **throw** |
| **case** | **false** | **native** | **throws** |
| **catch** | **final** | **new** | **transient** |
| **char** | **finally** | **null** | **true** |
| **class** | **float** | **package** | **try** |
| **const** | **for** | **private** | **typeof** |
| **continue** | **function** | **protected** | **var** |
| **debugger** | **goto** | **public** | **void** |
| **default** | **if** | **return** | **volatile** |
| **delete** | **implements** | **short** | **while** |
| **do** | **import** | **static** | **with** |
| **double** | **in** | **super** |  |

## Value = undefined

In computer programs, variables are often declared without a value. The value can be something that has to be calculated, or something that will be provided later, like user input.

A variable declared without a value will have the value undefined.

The variable carName will have the value undefined after the execution of this statement:

# JavaScript - Operators

JavaScript supports the following types of operators.

* Arithmetic Operators
* Comparison Operators
* Logical Operators
* Assignment Operators
* Conditional (or ternary) Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Addition | 10+20 = 30 |
| - | Subtraction | 20-10 = 10 |
| \* | Multiplication | 10\*20 = 200 |
| / | Division | 20/10 = 2 |
| % | Modulus (Remainder) | 20%10 = 0 |
| ++ | Increment | var a=10; a++; Now a = 11 |
| -- | Decrement | var a=10; a--; Now a = 9 |

JavaScript Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | Is equal to | 10==20 = false |
| === | Identical (equal and of same type) | 10==20 = false |
| != | Not equal to | 10!=20 = true |
| !== | Not Identical | 20!==20 = false |
| > | Greater than | 20>10 = true |
| >= | Greater than or equal to | 20>=10 = true |
| < | Less than | 20<10 = false |
| <= | Less than or equal to | 20<=10 = false |

JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| && | Logical AND | (10==20 && 20==33) = false |
| || | Logical OR | (10==20 || 20==33) = false |
| ! | Logical Not | !(10==20) = true |

JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Assign | 10+10 = 20 |
| += | Add and assign | var a=10; a+=20; Now a = 30 |
| -= | Subtract and assign | var a=20; a-=10; Now a = 10 |
| \*= | Multiply and assign | var a=10; a\*=20; Now a = 200 |
| /= | Divide and assign | var a=10; a/=2; Now a = 5 |
| %= | Modulus and assign | var a=10; a%=2; Now a = 0 |

JavaScript Type Operators

|  |  |
| --- | --- |
| **Operator** | **Description** |
| typeof | Returns the type of a variable |
| instanceof | Returns true if an object is an instance of an object type |

## Adding Strings and Numbers

Adding two numbers, will return the sum, but adding a number and a string will return a string:

### Example

var x = 5 + 5;  
var y = "5" + 5;  
var z = "Hello" + 5;

The result of x, y, and z will be:

10  
55  
Hello5

## JavaScript String Operators

The + operator can also be used to add (concatenate) strings.

### Example

var txt1 = "John";  
var txt2 = "Doe";  
var txt3 = txt1 + " " + txt2;

The result of txt3 will be:

John Doe

### Conditional Operator (? :)

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

|  |  |
| --- | --- |
| **Sr.No.** | **Operator and Description** |
| 1 | **? : (Conditional )**  If Condition is true? Then value X : Otherwise value Y |

# JavaScript - if...else Statement

JavaScript supports the following forms of **if..else** statement

* if statement
* if...else statement
* if...else if... statement.

### Syntax

The syntax for a basic if statement is as follows −

if (expression)

{

Statement(s) to be executed if expression is true

}

## if...else statement

The **'if...else'** statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

### Syntax

if (expression)

{

Statement(s) to be executed if expression is true

}

else

{

Statement(s) to be executed if expression is false

}

## if...else if... statement

The **if...else if...** statement is an advanced form of **if…else** that allows JavaScript to make a correct decision out of several conditions.

### Syntax

The syntax of an if-else-if statement is as follows −

if (expression 1)

{

Statement(s) to be executed if expression 1 is true

} else if (expression 2) {

Statement(s) to be executed if expression 2 is true

} else if (expression 3) {

Statement(s) to be executed if expression 3 is true

} else {

Statement(s) to be executed if no expression is true

}

# JavaScript - Switch Case

switch (expression) {

case condition 1: statement(s)

break;

case condition 2: statement(s)

break;

...

case condition n: statement(s)

break;

default: statement(s)

}

# JavaScript - While Loops

### Syntax

The syntax of **while loop** in JavaScript is as follows −

while (expression)

{

Statement(s) to be executed if expression is true

}

## The do...while Loop

### Syntax

The syntax for **do-while** loop in JavaScript is as follows −

do {

Statement(s) to be executed;

} while (expression);

# JavaScript - For Loop

### Syntax

The syntax of **for** loop is JavaScript is as follows −

for (initialization; test condition; iteration statement) {

Statement(s) to be executed if test condition is true

}

# JavaScript *for...in* loop

The **for...in** loop is used to loop through an object's properties.

Syntax

The syntax of ‘for..in’ loop is −

for (variablename in object) {

statement or block to execute

}

# JavaScript - Loop Control

## The break Statement

## The continue Statement

# JavaScript – Functions

## Function Definition

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

### Syntax

The basic syntax is shown here.

**<script type = "text/javascript">**

**<!--**

**function functionname(parameter-list) {**

**statements**

**}**

**//-->**

**</script>**

<script type = "text/javascript">

<!--

function sayHello() {

alert("Hello there");

}

//-->

</script>

## Function Parameters

**<html>**

**<head>**

**<script type = "text/javascript">**

**function sayHello(name, age)**

**{**

**document.write (name + " is " + age + " years old.");**

**}**

**</script>**

**</head>**

**<body>**

**<p>Click the following button to call the function</p>**

**<form>**

**<input type = "button" onclick = "sayHello('xyz', 20)" value = "Say Hello">**

**</form>**

**</body>**

**</html>**

## The return Statement

A JavaScript function can have an optional **return** statement. This is required if you want to return a value from a function. This statement should be the last statement in a function.

**<html>**

**<head>**

**<script type = "text/javascript">**

**function sum(n1, n2)**

**{**

**var s;**

**s = n1 + n2;**

**return s;**

**}**

**function secondFunction()**

**{**

**var result;**

**result = sum(50, 60);**

**document.write (result );**

**}**

**</script>**

**</head>**

**<body>**

**<p>Click the following button to call the function</p>**

**<form>**

**<input type = "button" onclick = "secondFunction()" value = "Call Function">**

**</form>**

**</body>**

**</html>**

# JavaScript - HTML DOM Methods

# (Document Object Model )

HTML DOM methods are **actions** you can perform (on HTML Elements).

HTML DOM properties are **values** (of HTML Elements) that you can set or change.

## The DOM Programming Interface

The HTML DOM can be accessed with JavaScript (and with other programming languages).

In the DOM, all HTML elements are defined as **objects**.

The programming interface is the properties and methods of each object.

A **property** is a value that you can get or set (like changing the content of an HTML element).

A **method** is an action you can do (like add or deleting an HTML element).

## Example

The following example changes the content (the innerHTML) of the <p> element with id="demo":

<html>  
<body>  
<p id="demo"></p>  
<script>  
document.getElementById("demo").innerHTML = "Hello World!";  
</script>  
</body>  
</html>

In the example above, getElementById is a **method**, while innerHTML is a **property**.

The getElementById Method

The most common way to access an HTML element is to use the id of the element.

In the example above the getElementById method used id="demo" to find the element.

The innerHTML Property

The easiest way to get the content of an element is by using the innerHTML property.

The innerHTML property is useful for getting or replacing the content of HTML elements.

The innerHTML property can be used to get or change any HTML element, including <html> and <body>.

**<!DOCTYPE html>**

**<html>**

**<body>**

**<h2>JavaScript Functions</h2>**

**<p>This example calls a function to convert from Fahrenheit to Celsius:</p>**

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

**<script>**

**function toCelsius(f)**

**{**

**return (5/9) \* (f-32);**

**}**

**document.getElementById("demo").innerHTML = toCelsius(77);**

**</script>**

**</body>**

**</html>**

<!doctype html>

**<html>**

**<head>**

**<script>**

**function** add(){

**var** a,b,c;

a=Number(document.getElementById("first").value);

b=Number(document.getElementById("second").value);

c= a + b;

document.getElementById("answer").value= c;

}

**</script>**

**</head>**

**<body>**

Enter the First number : **<input** id="first"**>**

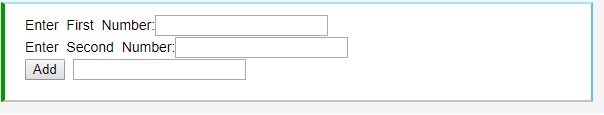
Enter the Second number: **<input** id="second"**>**

**<button** onclick="add()"**>**Add**</button>**

**<input** id="answer"**>**

**</body>**

**</html>**



### Code Explanation

* **no=Number(document.getElementById("first").value);**  
  This code is used for receive first input value form input field which have id first.
* **no=Number(document.getElementById("second").value);**  
  This code is used for receive first input value form input field which have id second.
* **document.getElementById("answer").value= fact;**  
  This code is used for receive calculated value of factorial and display in input field which have id answer
* **<button onclick="add()">Add</button>**  
  This code is used for call add function when button clicked.

<!doctype html>

**<html>**

**<head>**

**<script>**

**function** armstr()

{

**var** arm=0,a,b,c,d,num;

num=Number(document.getElementById("no\_input").value);

temp=num;

**while**(temp>0)

{

a=temp%10;

temp=parseInt(temp/10); // convert float into Integer

arm=arm+a\*a\*a;

}

**if**(arm==num)

{

alert("Armstrong number");

}

**else**

{

alert("Not Armstrong number");

}

}

**</script>**

**</head>**

**<body>**

Enter any Number: **<input** id="no\_input"**>**

**<button** onclick="armstr()"**>**Check**</button></br></br>**

**</body>**

**</html>**

**Javascript Example1**

**<!doctype html>**

**<html>**

**<head>**

**<script>**

**function add(){**

**var a,b,c;**

**a=Number(document.getElementById("first").value);**

**b=Number(document.getElementById("second").value);**

**c= a + b;**

**document.getElementById("answer").value= c;**

**}**

**</script>**

**</head>**

**<body>**

**Enter the First number : <input id="first">**

**Enter the Second number: <input id="second">**

**<button onclick="add()">Add</button>**

**<input id="answer">**

**</body>**

**</html>**

**Java Script Example2**

**<!DOCTYPE html>**

**<html>**

**<body>**

**<script>**

**function myFunction()**

**{**

**var name=prompt("Please enter your name");**

**var desig=prompt("Please enter your Designation");**

**var salary=prompt("Please enter your Salary");**

**if (name != null&& desig !=null && salary !=null)**

**{**

**document.getElementById("mname").innerHTML=name;**

**document.getElementById("mdesig").innerHTML=desig;**

**document.getElementById("msalary").innerHTML=salary;**

**}**

**}**

**</script>**

**<p>Click the Button</p>**

**<button onclick="myFunction()">Check</button>**

**<table border="1">**

**<tr>**

**<th>Name</th><th>Designation</th><th>Salary</th>**

**</tr>**

**<tr>**

**<td id="mname">null</td>**

**<td id="mdesig">null</td>**

**<td id="msalary">null</td>**

**</tr>**

**</table>**

**</body>**

**</html>**