LAL BAHADUR SHASTRI TRAINING INSTITUTE

**A-1272 First FLOOR, MAYUR VIHAR PHASE III, DELHI 110096**

**Java Script**

What is JavaScript

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

Features of JavaScript

There are following features of JavaScript:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

## JS Implementation

Javascript example is easy to code. JavaScript provides 3 places to put the JavaScript code: within body tag, within head tag and external JavaScript file.  
  
The **script** tag specifies that we are using JavaScript.

The **text/javascript** is the content type that provides information to the browser about the data.

The **document.write()** function is used to display dynamic content through JavaScript.

3 Places to put JavaScript code

1. Between the head tag of html
2. Between the body tag of html
3. In .js file (external javaScript)

1) JavaScript Example : code between the head tag

Let’s see the same example of displaying alert dialog box of JavaScript that is contained inside the head tag.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

document.write("Hello from Lbsti Baba");

</script>

</head>

<body>

<h1>Hello</h1>

</body>

</html>

2) JavaScript Example : code between the body tag

In this example, we have displayed the dynamic content using JavaScript in body tag. 

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

</head>

<body>

<h1>Hello</h1>

<script>

document.write("Hello from Lbsti Baba");

</script>

</body>

</html>

3) External JavaScript file

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

It provides **code re usability** because single JavaScript file can be used in several html pages.

An external JavaScript file must be saved by .js extension. It is recommended to embed all JavaScript files into a single file. It increases the speed of the webpage.

Let's create an external JavaScript  test.js file that prints Hey Hello Again.

// test.js file

document.write("Hey Hello Again");

Let's include the JavaScript file into html page. It calls the JavaScript function.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<!-- External JavaScript -->

<script src="js/test.js"></script>

</head>

<body>

<h1>Hello</h1>

</body>

</html>

## HTML Tags in Javascript

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

document.write("Hello World <br><br>");

document.write("<i><b>Hello from Lbsti Baba<b></i>");

</script>

</head>

<body>

<h1>Lbsti Baba</h1>

</body>

</html>

**JavaScript Comment**

The **JavaScript comments** are meaningful way to deliver message. It is used to add information about the code, warnings or suggestions so that end user can easily interpret the code.

The JavaScript comment is ignored by the JavaScript engine i.e. embedded in the browser.

Advantages of JavaScript comments

There are mainly two advantages of JavaScript comments.

1. **To make code easy to understand** It can be used to elaborate the code so that end user can easily understand the code.
2. **To avoid the unnecessary code** It can also be used to avoid the code being executed. Sometimes, we add the code to perform some action. But after sometime, there may be need to disable the code. In such case, it is better to use comments.

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.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

//document.write("Hello World <br><br>");

document.write("<i><b>Hello from Lbsti Baba<b></i>");

</script>

</head>

<body>

<h1>LBSTI </h1>

</body>

</html>

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:

1. /\* your code here  \*/

It can be used before, after and middle of the statement.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

/\*document.write("Hello World <br><br>");

document.write("<i><b>Hello from Lbsti Baba<b></i>");\*/

document.write("Hello");

</script>

</head>

<body>

<h1>Lbsti Baba</h1>

</body>

</html>

## JS Variables

Variables are Containers for Storing data of value

JavaScript Variables can be declared in 4 ways:

* Automatically
* Using var
* Using let
* Using const

Var Keyword:

The var keyword was used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

The var keyword should only be used in code written for older browsers.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

var x = "Hello";

var y = "World";

document.write(x + y);

</script>

</head>

<body>

</body>

</html>

## JS Variables ( Let & Const )

### Let Variable:

Variables defined with let cannot be **Redeclared**

Variables defined with let must be **Declared** before use

Variables defined with let have **Block Scope**

##### **Example:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

let z = "Hello World";

document.write(z);

</script>

</head>

<body>

</body>

</html>

### Const Variable:

Variables defined with const cannot be **Redeclared**

Variables defined with const cannot be **Reassigned**

Variables defined with const have **Block Scope**

##### **Example:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

const second = "Hello World";

document.write(second);

</script>

</head>

<body>

</body>

</html>

## JS Data Types

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

1. var a=40;//holding number
2. var b="Rahul";//holding string

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| String | represents sequence of characters e.g. "hello" |
| Number | represents numeric values e.g. 100 |
| Boolean | represents boolean value either false or true |
| Undefined | represents undefined value |
| Null | represents null i.e. no value at all |

JavaScript non-primitive data types

The non-primitive data types are as follows:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| Object | represents instance through which we can access members |
| Array | represents group of similar values |
| RegExp | represents regular expression |

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

/\* String Data Type \*/

var a = "Hello World";

document.write(a);

document.write("<br>");

document.write(typeof a);

document.write("<br><br>");

/\* Number Data Type \*/

var b = 25;

document.write(b);

document.write("<br>");

document.write(typeof b);

document.write("<br><br>");

/\* Boolean Data Type \*/

var c = true;

document.write(c);

document.write("<br>");

document.write(typeof c);

document.write("<br><br>");

/\* Array Data Type \*/

var d= ["HTML","CSS","JS"];

document.write(d);

document.write("<br>");

document.write(typeof d);

document.write("<br><br>");

/\* Object Data Type \*/

var x= {first:"Jane",last:"Doe"};

document.write(x);

document.write("<br>");

document.write(typeof x);

document.write("<br><br>");

/\* Null Data Type \*/

var y = null;

document.write(y);

document.write("<br>");

document.write(typeof y);

document.write("<br><br>");

/\* undefined Data Type \*/

var z;

document.write(z);

document.write("<br>");

document.write(typeof z);

document.write("<br><br>");

</script>

</head>

<body>

</body>

</html>

## JS Arithmetic 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 |

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

/\* Addition Arithmetic Operators \*/

var a = 10;

var b = 10;

var c = a + b;

document.write(c);

document.write("<br><br>");

/\* Subtraction Arithmetic Operators \*/

var m = 10;

var n = 5;

var o = m - n;

document.write(o);

document.write("<br><br>");

/\* Multiplication Arithmetic Operators \*/

var p = 10;

var q = 10;

var r = a \* b;

document.write(r);

document.write("<br><br>");

/\* Exponentiation Arithmetic Operators \*/

var d = 10;

var e = 3;

var f = d \*\* e;

document.write(f);

document.write("<br><br>");

/\* Division Arithmetic Operators \*/

var s = 14;

var t = 2;

var w = s / t;

document.write(w);

document.write("<br><br>");

/\* Modulus(Remainder) Arithmetic Operators \*/

var x = 14;

var y = 2;

var z = a % b;

document.write(z);

document.write("<br><br>");

/\* Increment Arithmetic Operators \*/

var i = 10;

var j = 3;

document.write(i + j);

document.write("<br>");

i++;

document.write(i + j);

document.write("<br><br>");

/\* Decrement Arithmetic Operators \*/

var g = 10;

var h = 3;

document.write(g + h);

document.write("<br>");

g--;

document.write(g + h);

</script>

</head>

<body>

</body>

</html>

## JS Assignment Operators

JavaScript **assignment operator** is **equal (=)** which assigns the value of the right-hand operand to its left-hand operand. That is if a = b assigns the value of b to a.

The simple assignment operator is used to assign a value to a variable. The assignment operation evaluates the assigned value. Chaining the assignment operator is possible in order to assign a single value to multiple variables.

### Syntax

data=value

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

/\* Equalto Assignment Operators \*/

var m = 10;

var n = 3;

o = m + n;

document.write(o);

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

/\* Addition Assignment Operators \*/

var a = 10;

var b = 3;

a += b;

document.write(a);

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

/\* Subtraction Assignment Operators \*/

var m = 10;

var n = 3;

m -= n;

document.write(m);

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

/\* Multiplication Assignment Operators \*/

var p = 10;

var q = 3;

p \*= q;

document.write(p);

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

/\* Exponentiation Assignment Operators \*/

var d = 10;

var e = 3;

d \*\*= e;

document.write(d);

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

/\* Division Assignment Operators \*/

var s = 10;

var t = 3;

s /= t;

document.write(s);

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

/\* Modulus(Remainder) Assignment Operators \*/

var x = 10;

var y = 3;

x %= y;

document.write(x);

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

</script>

</head>

<body>

</body>

</html>

## JS with Google Chrome Console

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

var x = 50;

console.log(x);

console.log([1,2,3]);

console.table([1, 2, 3]);

console.error("Something went wrong.");

console.time("Test");

console.warn("This is just warning");

console.timeEnd("Test");

//console.clear();

</script>

</head>

<body>

<h1 id="main">Lbsti Baba</h1>

</body>

</html>

## JS Comparison Operators

avaScript **Comparison operators** are mainly used to perform the logical operations that determine the equality or difference between the values. **Comparison operators** are used in logical expressions to determine their equality or differences in variables or values.

JavaScript**Comparison Operators list:** There are so many comparison operators as shown in the table with the description.

| **OPERATOR NAME** | **USAGE** | **OPERATION** |
| --- | --- | --- |
| Equality Operator | a==b | Compares the equality of two operators |
| Inequality Operator | a!=b | Compares inequality of two operators |
| Strict Equality Operator | a===b | Compares both value and type of the operand |
| Strict Inequality Operator | a!==b | Compares inequality with type |
| Greater than Operator | a>b | Checks if the left operator is greater than the right operator |
| Greater than or equal Operator | a>=b | Checks if the left operator is greater than or equal to the right operator |
| Less than Operator | a<b | Checks if the left operator is smaller than the right operator |
| Less than or equal Operator | a<=b | Checks if the left operator is smaller than or equal to the right operator |

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

/\* Equal to Comparison Operators \*/

var a = 10;

var b = 20;

document.write( a == b);

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

/\* Equal value and equal type Comparison Operators \*/

var c = 10;

var d= "10";

document.write(a === b);

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

/\* Not Equal Comparison Operators \*/

var e = 2;

var f = 3;

document.write(e != f);

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

/\* Not Equal value or not equal type Comparison Operators \*/

var g = 2;

var h = '2';

document.write(g !== h);

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

/\* Greater Than Comparison Operators \*/

var i = 2;

var j = 4;

document.write(i > j);

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

/\* Less Than Comparison Operators \*/

var m = 10;

var n = 20;

document.write(m > n);

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

/\* Greater Than or Equal To Comparison Operators \*/

var p = 10;

var q = 20;

document.write(p >= q);

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

/\* Less Than or Equal To Comparison Operators \*/

var x= 20;

var y = 20;

document.write(x <= y);

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

</script>

</head>

<body>

</body>

</html>

## JS If Statement

JavaScript If-else

The **JavaScript if-else statement** is used *to execute the code whether condition is true or false*. There are three forms of if statement in JavaScript.

1. If Statement
2. If else statement
3. if else if statement

JavaScript If statement

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

if(expression){

//content to be evaluated

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

/\* Greater Than If Condition \*/

var a = 100;

var b = 20;

if(a > b) {

document.write("A is Greater");

}

document.write("<br><br>");

/\* Equal to If Condition \*/

var m = 100;

var n = 100;

if (m == n) {

document.write("A is Greater");

}

document.write("<br><br>");

/\* Equal value and equal type If Condition \*/

var x = 100;

var y = "100";

if (x === y) {

document.write("Lbsti Baba");

}

document.write("<br><br>");

</script>

</head>

<body>

</body>

</html>

## JS Logical Operators

### ****Logical Operators in JavaScript****

The following are the logical operators supported by JavaScript:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| && | Logical And | ( a< 5 && b>2) |
| || | Logical Or | (a<5 || b>2) |
| ! | Not | (a!=5) |

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

/\* Logical And Operator \*/

var age = 18;

if (age >= 18 && age <= 21) {

document.write("Yes you are eligible.");

}

document.write("<br><br>");

/\* Logical OR Operator \*/

var a = 10;

var b = 15;

if (a >= 8 || b <= 15) {

document.write("Yes you are eligible.");

}

document.write("<br><br>");

/\* Logical Not Operator \*/

var x = 30;

console.log(!x >= 12);

</script>

</head>

<body>

</body>

</html>

## JS If Else Statement

### JavaScript If...else Statement

It evaluates the content whether condition is true or false. The syntax of JavaScript if-else statement is given below.

if(expression){

//content to be evaluated if condition is true

}

else{

//content to be evaluated if condition is false

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

/\* Greater Than If Else Condition \*/

var a = 15;

if (a > 30) {

document.write("A is Greater");

}else {

document.write("A is smaller");

}

document.write("<br><br>");

/\* Equal to If Else Condition \*/

var x = 100;

if (x == 100) {

document.write("X is Same");

}else {

document.write("X is not smaller");

}

</script>

</head>

<body>

</body>

</html>

## JS If Else If Statement

### JavaScript If...else if statement

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

if(expression1){

//content to be evaluated if expression1 is true

}

else if(expression2){

//content to be evaluated if expression2 is true

}

else if(expression3){

//content to be evaluated if expression3 is true

}

else{

//content to be evaluated if no expression is true

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script>

/\* If Else IF Condition \*/

var per = prompt("Enter your Percentage : ");

if (per >= 80 && per <= 100) {

document.write("You are in Merit.");

}else if (per >= 60 && per < 80) {

document.write("You are in Ist Division.");

}else if (per >= 45 && per < 60) {

document.write("You are in IIst Division.");

}else if (per >= 33 && per < 45) {

document.write("You are in IIIst Division.");

}else if (per < 33) {

document.write("You are Fail.");

}else {

document.write("Please Enter Valid Percentage.");

}

</script>

</head>

<body>

</body>

</html>

## JS Conditional Ternary Operator

### ****Ternary Operator:****

The “Question mark” or “conditional” operator in JavaScript is a ternary operator that has three operands. It is the simplified operator of if/else.

### ****Examples:****

Input: let result = (10 > 0) ? true : false;  
Output: true

Input: let message = (20 > 15) ? "Yes" : "No";  
Output: Yes

### ****Syntax:****

condition ? value if true : value if false

* **condition:** Expression to be evaluated which returns a boolean value.
* **value if true:** Value to be executed if the condition results in a true state.
* **value if false:** Value to be executed if the condition results in a false state.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = 100;

var b;

b = "Value is " + (a > 10 ? "True" : "False");

document.write(b);

</script>

</head>

<body>

</body>

</html>

## JS Switch Case

JavaScript Switch Statement

The **JavaScript switch statement** is used *to execute one code from multiple expressions*. It is just like else if statement that we have learned in previous page. But it is convenient than *if..else..if* because it can be used with numbers, characters etc.

The signature of JavaScript switch statement is given below.

switch(expression)

{

case x:

// code block

break;

case y:

// code block

break;

default:

// code block

}

This is how it works:

1. The switch expression is evaluated once.
2. The value of the expression is compared with the values of each case.
3. If there is a match, the associated block of code is executed.
4. If there is no match, the default code block is executed.

The break Keyword

When JavaScript reaches a break keyword, it breaks out of the switch block.

This will stop the execution inside the switch block.

It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway.

**Note:**If you omit the break statement, the next case will be executed even if the evaluation does not match the case.

The default Keyword

The default keyword specifies the code to run if there is no case match:

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var day = 1;

switch (day) {

case 0:

document.write("Today is Monday");

break;

case 1:

document.write("Today is Tuesday");

break;

case 2:

document.write("Today is Wednesday");

break;

case 3:

document.write("Today is Thursday");

break;

case 4:

document.write("Today is Friday");

break;

case 5:

document.write("Today is Saturday");

break;

case 6:

document.write("Today is Sunday");

break;

default:

document.write("Enter the valid Week Day");

}

</script>

</head>

<body>

</body>

</html>

### JavaScript alert()

The **alert()** method in JavaScript is used to display a virtual alert box. It is mostly used to give a warning message to the users. It displays an alert dialog box that consists of some specified message (which is optional) and an OK button. When the dialog box pops up, we have to click "OK" to proceed.

The alert dialog box takes the focus and forces the user to read the specified message. So, we should avoid overusing this method because it stops the user from accessing the other parts of the webpage until the box is closed.

### Syntax

alert(message)

### Values

**message:** It is an optional string that specifies the text to display in the alert box. It consists of the information that we want to show to the users.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = 40;

var b = 20;

if(a > b){

alert(b + a);

}else{

alert("Value of B : " + b);

}

</script>

</head>

<body>

</body>

</html>

## JS Confirm Box

JavaScript confirm method invokes a function that asks the user for a confirmation dialogue on a particular action. The confirm () method uses a window object to invoke a dialogue with a question and two option buttons, **OK** and **Cancel**. If the user selects the OK option, it will continue to the function execution; selecting the Cancel option will abort the block code's execution.

It returns **true** if the user selects the OK option; otherwise, it returns **false**.

### Syntax:

confirm("Select an Option!");

### Parameters:

It takes a "message" value in string format to display in the confirmation dialogue you want to show the user.

### Return value:

The confirm method returns a Boolean output, either true or false, if the OK is selected.

A boolean indicating whether OK (true) or Cancel (false) was selected. If a browser ignores in-page dialogues, then the returned value is always false.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

/\* Confirm Box\*/

var a = confirm ("Do you like our Website?");

if (a) {

alert("Thanks");

} else {

alert("Sorry");

}

</script>

</head>

<body>

</body>

</html>

### JavaScript prompt() dialog box

The **prompt()** method in JavaScript is used to display a prompt box that prompts the user for the input. It is generally used to take the input from the user before entering the page. It can be written without using the **window** prefix. When the prompt box pops up, we have to click "OK" or "Cancel" to proceed.

The box is displayed using the **prompt()** method, which takes two arguments: The first argument is the label which displays in the text box, and the second argument is the default string, which displays in the textbox. The prompt box consists of two buttons, **OK** and **Cancel**. It returns null or the string entered by the user. When the user clicks "OK," the box returns the input value. Otherwise, it returns null on clicking "Cancel".

The prompt box takes the focus and forces the user to read the specified message. So, it should avoid overusing this method because it stops the user from accessing the other parts of the webpage until the box is closed.

### Syntax

prompt(message, default)

### Values

The parameter values of this function are defined as follows.

**message:** It is an optional parameter. It is the text displays to the user. We can omit this value if we don't require to show anything in the prompt.

**default:** It is also an optional parameter. It is a string that contains the default value displayed in the textbox.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = prompt("What is your Name ?");

document.write(a);

</script>

</head>

<body>

</body>

</html>

## JS Functions

JavaScript Functions

**JavaScript functions** are used to perform operations. We can call JavaScript function many times to reuse the code.

Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

1. **Code reusability**: We can call a function several times so it save coding.
2. **Less coding**: It makes our program compact. We don’t need to write many lines of code each time to perform a common task.

JavaScript Function Syntax

The syntax of declaring function is given below.

function functionName([arg1, arg2, ...argN])

{

//code to be executed

}

JavaScript Functions can have 0 or more arguments.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

function hello() {

document.write("Hello Everybody");

}

function lbsti() {

document.write("Lbsti Baba");

}

hello();

document.write("</br>");

lbsti();

document.write("</br>");

hello();

document.write("</br>");

hello();

</script>

</head>

<body>

</body>

</html>

## JS Functions with Parameters

The Javascript **Function Parameters** are the names that are defined in the function definition and real values passed to the function in the function definition are known as arguments.

Syntax**:**

function Name(paramet1, paramet2, paramet3,...)

{

// Statements

}

Parameter Rules:

* There is no need to specify the data type for parameters in JavaScript function definitions.
* It does not perform type-checking based on the passed-in JavaScript functions.
* It does not check the number of received arguments.

Parameters:

* **Name:** It is used to specify the name of the function.
* **Arguments:** It is provided in the argument field of the function.

Default Parameter:

The default parameters are used to initialize the named parameters with default values in case, when no value or undefined is passed.

Syntax:

function Name(paramet1 = value1, paramet2 = value2 .. .)

{

// statements

}

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

/\* Functions With Parameters\*/

function hello(fname= "Lbsti",lname= "Baba") {

document.write("Hello" + fname + " " + lname + "<br>");

}

hello("Ram","Singh");

hello("Salman", "Khan");

</script>

</head>

<body>

</body>

</html>

## JS Functions with Return Value

The return statement stops the execution of a function and returns a value.

### Syntax

return value;

### Parameters

|  |
| --- |
|  |
| Parameter | Description |
| value | Optional. The value to be returned. If omitted, it returns undefined |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

function fullname(fname = "Lbsti", lname = "Baba") {

var a = fname + " " + lname;

return a;

}

var fn = fullname("Ram","Singh");

document.write(fn);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

function sum(math,eng,sc){

var s = math + eng + sc;

return s;

}

function percentage(tt){

var per = tt/300 \* 100;

document.write(per);

}

var total = sum(80,80,80);

percentage(total);

</script>

</head>

<body>

</body>

</html>

## JS Global & Local Variable

### Global Variable:

These are variables that are defined in global scope i.e. outside of functions. These variables have global scope, so they can be accessed by any function directly. In the case of global scope variables, the keyword they are declared with does not matter they all act the same. A variable declared without a keyword is also considered global even though it is declared in the function.

#### **Example:**

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = "Lbsti Baba";

function hello() {

document.write(a + "<br>");

}

hello();

document.write(a);

</script>

</head>

<body>

</body>

</html>

### Local Variable:

When you use JavaScript, local variables are variables that are defined within functions. They have local scope, which means that they can only be used within the functions that define them. Accessing them outside the function will throw an error

#### **Example:**

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

function hello(){

var a = "Lbsti Baba";

document.write(a + "<br>");

}

hello();

document.write(a);

</script>

</head>

<body>

</body>

</html>

## JS Events

The change in the state of an object is known as an **Event**. In html, there are various events which represents that some activity is performed by the user or by the browser. When javascript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called **Event Handling**. Thus, js handles the HTML events via **Event Handlers**.

**For example**, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

Some of the HTML events and their event handlers are:

### Mouse events:

|  |  |  |
| --- | --- | --- |
| **Event Performed** | **Event Handler** | **Description** |
| click | onclick | When mouse click on an element |
| mouseover | onmouseover | When the cursor of the mouse comes over the element |
| mouseout | onmouseout | When the cursor of the mouse leaves an element |
| mousedown | onmousedown | When the mouse button is pressed over the element |
| mouseup | onmouseup | When the mouse button is released over the element |
| mousemove | onmousemove | When the mouse movement takes place. |

### Keyboard events:

|  |  |  |
| --- | --- | --- |
| **Event Performed** | **Event Handler** | **Description** |
| Keydown & Keyup | onkeydown & onkeyup | When the user press and then release the key |

### Form events:

|  |  |  |
| --- | --- | --- |
| **Event Performed** | **Event Handler** | **Description** |
| focus | onfocus | When the user focuses on an element |
| submit | onsubmit | When the user submits the form |
| blur | onblur | When the focus is away from a form element |
| change | onchange | When the user modifies or changes the value of a form element |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

function hello() {

alert("Hello Everybody");

}

</script>

</head>

<body onresize="hello()">

<button onclick ="hello()">Click Me</button>

<p ondblclick="hello()">Double Click on Me</p>

<p onmouseenter="hello()">On Mouse Enter</p>

<p onmouseout="hello()">On Mouse Out</p>

</body>

</html>

## JS While Loop

The while loop loops through a block of code as long as a specified condition is true.

### Syntax

while (condition)

{

// code block to be executed

}

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

/\*Increase While Loop Condition \*/

var a = 1;

while (a <= 10) {

document.write(a + ") Hello Lbsti Baba <br>");

a = a + 1;

}

/\* Decrease While Loop Condition \*/

var x = 10;

while (x >= 1) {

document.write(x + ") Hello Lbsti Baba <br>");

x = x - 1;

}

</script>

</head>

<body>

</body>

</html>

## JS Do While Loop

A **do… while loop** in JavaScript is a control statement in which the code is allowed to execute continuously based on a given boolean condition. It is like a repeating if statement.

The do…while loop can be used to execute a specific block of code at least once.

### Syntax:

do {

// Statements

}

while(conditions)

The main difference between do…while and while loop is that it is guaranteed that do…while loop will run at least once. Whereas, the while loop will not run even once if the given condition is not satisfied.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = 1;

do{

document.write(a + " Hello Lbsti Baba<br>");

a++;

}while(a <= 10)

</script>

</head>

<body>

</body>

</html>

## JS For Loop

The for statement creates a loop with 3 optional expressions:

for (expression 1; expression 2; expression 3)

{

// code block to be executed

}

**Expression 1** is executed (one time) before the execution of the code block.  
**Expression 2** defines the condition for executing the code block.  
**Expression 3** is executed (every time) after the code block has been executed.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

for(var x = 1; x <= 10; x++){

document.write("Hello Lbsti Baba <br>");

}

</script>

</head>

<body>

</body>

</html>

## JS Break & Continue Statement

### Continue statement:

The continue statement “jumps over” one iteration in the loop. It breaks iteration in the loop and continues executing the next iteration in the loop.

### syntex:

continue labelname;

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

for(var a = 1; a <= 10; a++){

if(a == 3){

document.write("Hey : " + a + "<br>");

continue;

}

document.write("Number : " + a + "<br>");

}

</script>

</head>

<body>

</body>

</html>

### Break Statement:

The break statement is used to jump out of a loop. It can be used to “jump out” of a switch() statement. It breaks the loop and continues executing the code after the loop.

### Syntex:

break labelname;

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

for(var a = 1; a <= 10; a++){

if(a == 3){

document.write("Hey : " + a + "<br>");

break;

}

document.write("Number : " + a + "<br>");

}

</script>

</head>

<body>

</body>

</html>

## JS Even & Odd with Loops

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

for(var a = 1; a <= 10; a++){

if(a % 2 == 0){

document.write(a + "<br>");

}

}

</script>

</head>

<body>

</body>

</html>

## JS Nested Loop

### The nested for loop in JavaScript

A simple for loop executes a specified number of times depending on the initialization value and the terminating condition. A nested for loop on the other hand, resides one or more for loop inside an outer for loop.

In a nested loop the statement inside the for loop body is again a for loop. This causes The inside for loop to execute all the way through , for each iteration of the outer for loop.

for(let i = 0 ; i < limit; i++)

{

for(let j = 0 ; j < limit; j++)

{

// statement

}

// statement for outer loop

}

The inside loop in this example runs **limit** number of times for every iteration of the outer loop. So, in total, the loop runs **limit x** limit number of times.

The initialization value, terminating condition as well as updating of the loop variables for both loops are independent of one another.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

for(var a = 1; a <= 5; a++){

for(var b = 1; b <= a ; b++){

document.write(a + " ");

}

document.write("<br>");

}

</script>

</head>

<body>

</body>

</html>

## JavaScript Nested Loop - II

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

for (var a = 1; a <= 5; a++) {

for (var b = 1; b <= a; b++) {

document.write(b + " ");

}

document.write("<br>");

}

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

for (var a = 1; a <= 5; a++) {

for (var b = 1; b <= a; b++) {

document.write(a + " ");

}

document.write("<br>");

}

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

for (var a = 5; a >= 1; a--) {

for (var b = 1; b <= a; b++) {

document.write(a + " ");

}

document.write("<br>");

}

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

for (var a = 5; a >= 1; a--) {

for (var b = a; b >= 1; b--) {

document.write(b + " ");

}

document.write("<br>");

}

</script>

</head>

<body>

</body>

</html>

## JS Arrays

**JavaScript Array** is a single variable that is used to store elements of different data types. JavaScript arrays are zero-indexed. The Javascript Arrays are not associative in nature.

Arrays are used when we have a list of items. An array allows you to store several values with the same name and access them by using their index number.

Declaration of an Array

There are basically two ways to declare an array.

1. JavaScript array literal

The syntax of creating array using array literal is given below:

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

2. JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

var arrayname=new Array();

Here, **new keyword** is used to create instance of array.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ary = [10, 20, 30, 40, 50]

document.write("<ul>");

for (a = 0; a <= 4; a++) {

document.write("<li>" + ary[a] + "</li>");

}

document.write("</ul>");

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ary = [10, 20, 30, 40, 50]

var sum = 0;

document.write("<ul>");

for (a = 0; a <= 4; a++) {

document.write("<li>" + ary[a] + "</li>");

sum = sum + ary[a];

}

document.write("</ul>");

document.write("Total Sum:" + sum);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ary = [10, "Harry","Sarah", true, null];

document.write("<ul>");

for (a = 0; a < 5; a++) {

document.write("<li>" + ary[a] + "</li>");

}

document.write("</ul>");

</script>

</head>

<body>

</body>

</html>

## JS Create Arrays Method - II

The syntax of creating array directly is given below:

var arrayname=new Array();

Here, **new keyword** is used to create instance of array.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ary = new Array();

ary[0] = 10;

ary[1] = 'Harry';

ary[2] = true;

document.write("<ul>");

for(var a = 0; a < 3; a++){

document.write("<li>" + ary[a] + "</li>");

}

document.write("</ul>");

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ary = new Array(3);

for(var g = 0; g < 3; g++){

ary[g] = prompt("Enter the Value : ");

}

document.write("<ul>");

for(var a = 0; a < 3; a++){

document.write("<li>" + ary[a] + "</li>");

}

document.write("</ul>");

</script>

</head>

<body>

</body>

</html>

## JS Multidimensional Arrays

Multidimensional arrays are not directly provided in JavaScript. If we want to use anything which acts as a multidimensional array then we need to create a multidimensional array by using another one-dimensional array. So multidimensional arrays in JavaScript is known as arrays inside another array. We need to put some arrays inside an array, then the total thing is working like a multidimensional array. The array, in which the other arrays are going to insert, that array is use as the multidimensional array in our code. To define a multidimensional array its exactly the same as defining a normal one-dimensional array.

### One-Dimensional array:

var arr = []; // Empty 1D array

var arr1 = ["A", "B", "C", "D"] // 1D array contains some alphabets

var arr1 = [1, 2, 3, 4, 5] // 1D array contains some digits

### Multidimensional-Dimensional array:

### Method 1:

1st, need to define some 1D array

var arr1 = ["ABC", 24, 18000];

var arr2 = ["EFG", 30, 30000];

var arr3 = ["IJK", 28, 41000];

var arr4 = ["EFG", 31, 28000];

var arr5 = ["EFG", 29, 35000];

// "salary" defines like a 1D array but it already contains some 1D array

var salary = [arr1, arr2, arr3, arr4, arr5];

### 

1st, need to define some 1D array var arr1 = ["ABC", 24, 18000]; var arr2 = ["EFG", 30, 30000]; var arr3 = ["IJK", 28, 41000]; var arr4 = ["EFG", 31, 28000]; var arr5 = ["EFG", 29, 35000]; // "salary" defines like a 1D array but it already contains some 1D array var salary = [arr1, arr2, arr3, arr4, arr5];

Here arr1, arr2, …arr5 are some 1D arrays that are inside salary array.

### Method 2:

var salary = [

["ABC", 24, 18000],

["EFG", 30, 30000],

["IJK", 28, 41000],

["EFG", 31, 28000],

];

Here, salary array works like a multidimensional array. This notations are known as array literals.

### ****Accessing the element of salary array:****

* To access the array element we need a simple index-based notation

// This notation access the salary of "ABC" person which is 18000,  
// [0] selects 1st row, and [2] selects the 3rd element  
// of that 1st row which is 18000  
salary[0][2];

// Similarly,  
salary[3][2]; // Selects 28000

\*\*This notation is used for both **Method 1** and **Method 2**.

* For many iteration, we need to use loop to access the elements,

// This loop is for outer array

for (var i = 0, l1 = salary.length; i < l1; i++)

{

// This loop is for inner-arrays

for (var j = 0, l2 = salary[i].length; j < l2; j++) {

// Accessing each elements of inner-array

documents.write( salary[i][j] );

}

}

### Adding elements in Multidimensional Array:

Adding elements in multi-dimensional arrays can be achieved in two ways in **inner array** or **outer array**. The inner array can be done in two different ways.

### Adding elements to inner array:

* + We can use simple square bracket notation to add elements in multidimensional array.

salary[3][3] = "India";

// It adds "India" at the 4th index of 4th sub-array,  
// If we print the entire 4th sub-array, document.write(salary[3]);  
// the output will be :  ["EFG", 31, 28000, "India"]  
// indexing starts from 0

* We can use **push()** method to add elements in the array.

salary[3].push("India", "Mumbai");

// It add "India" at the 4th index and "Mumbai" at

// 5th index of 4th sub-array

// If we print the entire 4th sub-array,

// document.write(salary[3]);

// The output will be : ["EFG", 31, 28000, "India", "Mumbai"]

// Indexing starts from 0

* **Adding elements to outer array:**

                       It is much similar to previous methods.

salary.push(["MNO", 29, 33300]);

// This row added after the last row in the "salary" array

### Removing elements in Multidimensional Array:

We can use **pop()** methods to remove elements from inner-arrays, and also use **pop()** method for removing a entire inner array.

// Remove last element from 4th sub-array

// That is 28000 indexing starts from 0

salary[3].pop();

// Removes last sub-array

// That is "["EFG", 31, 28000]"

salary.pop();

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ary = [

["Harry",18,"Male","B.Com"],

["Sunny",19,"Male","BCA"],

["Sarah",18,"Male","BCA"],

["Tom",17,"Male","B.A."],

["Mac",17,"Male","B.A."]

];

document.write("<table border='1px' cellspacing='0'>");

for(var a = 0; a < 5; a++){

document.write("<tr>");

for(var b=0; b < 4; b++){

document.write("<td>" + ary[a][b] + "</td>");

}

document.write("</tr>");

}

document.write("</table>")

</script>

</head>

<body>

</body>

</html>

### With Array Length

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ary = [

["Harry",18,"Male","B.Com"],

["Sunny",19,"Male","BCA"],

["Sarah",18,"Male","BCA"],

["Tom",17,"Male","B.A."],

["Mac",17,"Male","B.A."]

];

document.write(ary.length);

document.write("<table border='1px' cellspacing='0'>");

for(var a = 0; a < ary.length; a++){

document.write("<tr>");

for(var b=0; b < ary[a].length; b++){

document.write("<td>" + ary[a][b] + "</td>");

}

document.write("</tr>");

}

document.write("</table>")

</script>

</head>

<body>

</body>

</html>

## JS Modify & Delete Array

Modify Array Element

Array elements are accessed using their **index number**:

Array **indexes** start with 0:

[0] is the first array element  
[1] is the second  
[2] is the third ...

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Harry",18,"Male","BCA"];

document.write(a + "<br>");

a[0] = "Sunny";

document.write(a + "<br>");

a[1] = 20;

document.write(a + "<br>");

</script>

</head>

<body>

</body>

</html>

Delete Array Element

Array elements can be deleted using the JavaScript operator delete.

Using delete leaves undefined holes in the array.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Harry",18,"Male","BCA"];

document.write(a + "<br>");

a[0] = "Sunny";

document.write(a + "<br>");

delete a[1];

document.write(a + "<br>");

</script>

</head>

<body>

</body>

</html>

## JS Array Sort & Reverse

Sorting an Array

The sort() sorts the elements of an array.

The sort() overwrites the original array.

The sort() sorts the elements as strings in alphabetical and ascending order.

Syntax

array.sort()

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay","Aman","Rehman","Karan"];

document.write(a + "<br><br>");

a.sort();

document.write(a);

</script>

</head>

<body>

</body>

</html>

Reversing an Array

The reverse() method reverses the order of the elements in an array.

The reverse() method overwrites the original array.

Syntax

array.reverse()

Return Value

|  |
| --- |
| The array after it has been reversed. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var b = ["Sanjay", "Aman", "Rehman", "Karan"];

document.write(b + "<br><br>");

b.reverse();

document.write(b);

</script>

</head>

<body>

</body>

</html>

## JS Array Pop & Push

Array Pop()

The pop() method removes (pops) **the last element** of an array.

The pop() method changes the original array.

The pop() method returns the removed element.

Syntax

array.pop()???????

Parameters

None

Return Value

|  |  |
| --- | --- |
| Type | Description |
| A variable | The removed item. A string, a number, an array, or any other type allowed in an array. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay", "Aman", "Rehman", "Karan"];

document.write(a + "<br><br>");

a.pop();

document.write(a);

</script>

</head>

<body>

</body>

</html>

Array push()

The push() method adds new items **to the end** of an array.

The push() method changes the length of the array.

The push() method returns the new length.

Syntax

array.push(item1, item2, ..., itemX)

Parameters

|  |  |
| --- | --- |
| Parameters | Description |
| *item1* *item2* .. *itemX* | The item(s) to add to the array. Minimum one item is required. |

Return Value

|  |  |
| --- | --- |
| Type | Description |
| A number | The new length of the array. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay", "Aman", "Rehman"];

a.push("Rahul");

document.write(a + "<br><br>");

a.push("Salman");

document.write(a + "<br><br>");

</script>

</head>

<body>

</body>

</html>

## JS Array Concat & Join

Array concat()

The JavaScript**Array concat()**Method is used to merge two or more arrays together. This method does not alter the original arrays passed as arguments but instead, returns a new Array.

Syntax:

let newArray1 = oldArray.concat()

let newArray2 = oldArray.concat(value0)

let newArray3 = oldArray.concat(value0,value1)

.......

.......

let newArray = oldArray.concat(value1 , [ value2, [ ...,[ valueN]]])

Parameters**:**

The parameters of this method are the arrays or the values that need to be added to the given array. The number of arguments to this method depends upon the number of arrays or values to be merged.

Return value:

This method returns a newly created array that is created after merging all the arrays passed to the method as arguments.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay","Aman","Rehman"];

var b = a.concat("rahul","Karan");

document.write(b);

</script>

</head>

<body>

</body>

</html>

Concat Two arrays

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay","Aman","Rehman"];

var b = ["Rahul","Karan"];

var c = a.concat(b);

document.write(c);

</script>

</head>

<body>

</body>

</html>

Concat multiple arrays

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay","Aman","Rehman"];

var b = ["Rahul","Karan"];

var d = ["Neha","Mahima"];

var c = a.concat(b,d);

document.write(c);

</script>

</head>

<body>

</body>

</html>

Array join()

The join() method returns an array as a string.

The join() method does not change the original array.

Any separator can be specified. The default is comma (,).

Syntax

array.join(separator)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *separator* | Optional. The separator to be used. Default is a comma. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Aman","Rehman","Karan"];

var b = ["Rahul", "Sanjay"];

var c = a.concat(b);

document.write(c + "<br><br>");

var d = c.join(" - ");

document.write(d);

</script>

</head>

<body>

</body>

</html>

## JS Array Slice & Splice

Array slice()

The slice() method returns selected elements in an array, as a new array.

The slice() method selects from a given *start*, up to a (not inclusive) given *end*.

The slice() method does not change the original array.

Syntax

array.slice(start, end)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *start* | Optional. Start position. Default is 0. Negative numbers select from the end of the array. |
| *end* | Optional. End position. Default is last element. Negative numbers select from the end of the array. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay", "Aman", "Rehman","Rahul", "Karan"];

document.write(a + "<br><br>");

var b = a.slice(1 , 4);

document.write(b);

</script>

</head>

<body>

</body>

</html>

Array splice()

The splice() method adds and/or removes array elements.

The splice() method overwrites the original array.

Syntax

array.splice(index, howmany, item1, ....., itemX)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *index* | Required. The position to add/remove items. Negative value defines the position from the end of the array. |
| *howmany* | Optional. Number of items to be removed. |
| *item1*, ..., *itemX* | Optional. New elements(s) to be added. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay", "Aman", "Rehman", "Rahul"];

document.write(a + "<br><br>");

a.splice(2,0,"Neha","Karan");

document.write(a);

</script>

</head>

<body>

</body>

</html>

## JS isArray

The isArray() method returns true if an object is an array, otherwise false.

Array.isArray()

Array.isArray() is a static property of the JavaScript Array object.

You can only use it as Array.isArray().

Using x.isArray(), where x is an array will return undefined.

Syntax

Array.isArray(obj)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *obj* | Required. An object (or any data type) to be tested. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| A boolean | true if the object is an array, otherwise false. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay", "Aman", "Rehman"];

document.write(a + "<br><br>");

var b = Array.isArray(a);

document.write(b);

</script>

</head>

<body>

</body>

</html>

isArray() with if condition

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay", "Aman", "Rehman"];

if(Array.isArray(a)){

document.write("This is an Array");

}else{

document.write("This is not an Array");

}

</script>

</head>

<body>

</body>

</html>

## JS Array index

### Array indexOf()

The JavaScript **Array indexOf()**Method is used to find the index of the first occurrence of the search element provided as the argument to the method. This method always compares the search element to the element present in the array using strict equality. Therefore, when the search element is NaN then it returns -1 because NaN values are never compared as equal.

### Syntax:

array.indexOf(element, start)

**Parameters:** This method accepts two parameters as mentioned above and described below:

* **element:** This parameter holds the element whose index will be returned.
* **start:** This parameter is optional and it holds the starting point of the array, where to begin the search the default value is 0.

**Return value:** This method returns the index of the first occurrence of the element. If the element cannot be found in the array, then this method returns -1.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay", "Aman", "Rehman", "Aman", "Rahul"];

document.write(a + "<br><br>");

var b = a.indexOf("Aman",2);

document.write(b);

</script>

</head>

<body>

</body>

</html>

### Array lastIndexOf()

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay", "Aman", "Rehman", "Aman", "Rahul"];

document.write(a + "<br><br>");

var c = a.lastIndexOf("Aman");

document.write(c + "<br><br>");

</script>

</head>

<body>

</body>

</html>

## JS Array Includes

The includes() method returns true if an array contains a specified value.

The includes() method returns false if the value is not found.

The includes() method is case sensitive.

Syntax

array.includes(element, start)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *element* | Required. The value to search for. |
| *start* | Optional. Start position. Default is 0. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| A boolean | true if the value is found, otherwise false. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Sanjay", "Aman", "Rehman", "Rahul"];

document.write(a + "<br><br>");

var b = a.includes("Neha");

document.write(b + "<br><br>");

var c = a.includes("Aman");

document.write(c + "<br><br>");

var d = a.includes("aman");

document.write(d + "<br><br>");

</script>

</head>

<body>

</body>

</html>

## JS Array Some & Every

### Array some()

The Javascript **arr.some()**method checks whether at least one of the elements of the array satisfies the condition checked by the argument method.

### Syntax

arr.some(callback(element,index,array),thisArg)

**Parameters:**This method accepts five parameters as mentioned above and described below:

* **callback:**This parameter holds the function to be called for each element of the array.
  + **element:** The parameter holds the value of the elements being processed currently.
  + **index:** This parameter is optional, it holds the index of the currentValue element in the array starting from 0.
  + **array:** This parameter is optional, it holds the complete array on which Array.every is called.
* **thisArg:** This parameter is optional, it holds the context to be passed as this is to be used while executing the callback function. If the context is passed, it will be used like this for each invocation of the callback function, otherwise undefined is used as default.

**Return value:**This method returns true even if one of the elements of the array satisfies the condition(and does not check the remaining values) implemented by the argument method. If no element of the array satisfies the condition then it returns false.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ages = [10, 13, 18, 2];

document.write(ages + "<br>");

var b = ages.some(checkAdult);

document.write(b + "<br>");

</script>

</head>

<body>

</body>

</html>

### Array every()

The Javascript **Array.every()** method considers all the elements of an array and then further checks whether all the elements of the array satisfy the given condition (passed by in user) or not that is provided by a method passed to it as the argument.

### Syntax:

// Arrow function

every((element) => { /\* … \*/ })

every((element, index) => { /\* … \*/ })

every((element, index, array) => { /\* … \*/ })

// Callback function

every(callbackFn)

every(callbackFn, thisArg)

// Inline callback function

every(function (element) { /\* … \*/ })

every(function (element, index) { /\* … \*/ })

every(function (element, index, array) { /\* … \*/ })

every(function (element, index, array) { /\* … \*/ }, thisArg)

**Parameters:** This method accepts five parameters as mentioned above and described below:

* **callback:** This parameter holds the function to be called for each element of the array.
* **element:**The parameter holds the value of the elements being processed currently.
* **index:** This parameter is optional, it holds the index of the current value element in the array starting from 0.
* **array:**This parameter is optional, it holds the complete array on which Array.every is called.
* **thisArg:** This parameter is optional, it holds the context to be passed as this is to be used while executing the callback function. If the context is passed, it will be used like this for each invocation of the callback function, otherwise undefined is used as default.

**Return value:**This method returns a Boolean value true if all the elements of the array follow the condition implemented by the argument method. If any one of the elements of the array does not satisfy the argument method, then this method returns false.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ages = [10, 19, 18, 20];

document.write(ages + "<br>");

var b = ages.every(checkAdult);

document.write(b + "<br>");

</script>

</head>

<body>

</body>

</html>

## JS Array find & findIndex

### Array find() Method

The Javascript **arr.find()** method in Javascript is used to get the value of the first element in the array that satisfies the provided condition. It checks all the elements of the array and whichever the first element satisfies the condition is going to print. This function will not work function having the empty array elements and also does not change the original array.

### Syntax:

array.find(function(currentValue, index, arr),thisValue);

**Parameters:** This method accepts 5 parameters as mentioned above and described below:

* **function:** It is the function of the array that works on each element.
* **currentValue:** This parameter holds the current element.
* **index:** It is an optional parameter that holds the index of the current element.
* **arr:** It is an optional parameter that holds the array object to which the current element belongs to.
* **thisValue:** This parameter is optional. If a value is to be passed to the function to be used as its “this” value else the value “**undefined**” will be passed as its “**this**” value.

**Return value:** It returns the array element value if any of the elements in the array satisfy the condition, otherwise it returns undefined.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ages = [10, 23, 19, 20];

document.write(ages + "<br>");

var b = ages.find(checkAdult);

document.write(b + "<br>");

function checkAdult(age) {

return age >= 18;

}

</script>

</head>

<body>

</body>

</html>

### Array findIndex() Method

The Javascript **Array.findIndex()** method is used to return the first index of the element in a given array that satisfies the provided testing function (passed in by the user while calling). Otherwise, if no data is found then the value of -1 is returned.

* It does not execute the method once it finds an element satisfying the testing method.
* It does not change the original array.

### Syntax:

array.findIndex(function(currentValue, index, arr), thisValue)

**Parameters:** This method accepts five parameters as mentioned above and described below:

* **function:** It is the function of the array that works on each element.
* **currentValue:** This parameter holds the current element.
* **index:** It is an optional parameter that holds the index of the current element.
* **arr:** It is an optional parameter that holds the array object to which the current element belongs.
* **thisValue:** This parameter is optional if a value is to be passed to the function to be used as its “this” value else the value “undefined” will be passed as its “this” value.

**Return value:**It returns the array element index if any of the elements in the array pass the test, otherwise, it returns -1.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ages = [10, 12, 23, 20];

document.write(ages + "<br><br>");

var b = ages.findIndex(checkAdult);

document.write(b + "<br>");

function checkAdult(age) {

return age >= 18;

}

</script>

</head>

<body>

</body>

</html>

## JS Array Filter

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ages = [10, 12, 19, 20];

document.write(ages + "<br><br>");

var b = ages.filter(checkAdult);

document.write(b + "<br>");

function checkAdult(age) {

return age >= 18;

}

</script>

</head>

<body>

</body>

</html>

## JS Array Methods

### Array toString() Method

The JavaScript **Array toString()** Method returns the string representation of the array elements

### Syntax:

arr.toString()

**Parameters:**This method does not accept any parameter.

**Return value:**

* The method returns the string representation of the array elements.
* If the array is empty, then it returns an empty string.
* The original array is not changed by this method

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Rahul","Karan","Aman","Neha"];

a.toString();

document.write(a + "<br><br>");

</script>

</head>

<body>

</body>

</html>

### Array valueOf() Method

The JavaScript **Array valueOf()**method in JavaScript is used to return the array. It is a default method of the Array Object. This method returns all the items in the same array. It will not change the original content of the array. It does not contain any parameter values.

### Syntax:

array.valueOf()

**Parameters:** This method does not accept any parameter.

**Return Value:**It returns an Array.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Rahul", "Karan", "Aman", "Neha"];

document.write(a.valueOf() + "<br><br>");

</script>

</head>

<body>

</body>

</html>

### Array fill() Method

The fill() method fills specified elements in an array with a value.

The fill() method overwrites the original array.

Start and end position can be specified. If not, all elements will be filled.

### Syntax

array.fill(value, start, end)

### Parameters

|  |
| --- |
|  |
| Parameter | Description |
| value | Required. The value to fill in. |
| start | Optional. The start index (position). Default is 0. |
| end | Optional. The stop index (position). Default is array length. |

### Return Value

|  |
| --- |
|  |
| Type | Description |
| Array | The filled array. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Rahul", "Karan", "Aman", "Neha"];

a.fill("Ram");

document.write(a);

</script>

</head>

<body>

</body>

</html>

## JS forEach Loop

The forEach() method calls a function for each element in an array.

The forEach() method is not executed for empty elements.

Syntax

array.forEach(function(currentValue, index, arr), thisValue)

Parameters

|  |  |
| --- | --- |
| *function()* | Required. A function to run for each array element. |
| *currentValue* | Required. The value of the current element. |
| *index* | Optional. The index of the current element. |
| *arr* | Optional. The array of the current element. |
| *thisValue* | Optional. Default undefined. A value passed to the function as its this value. |

Return Value

|  |
| --- |
| undefined |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Rahul","Karan","Aman","Neha"];

a.forEach(function(value,index) {

document.write(index + " : " + value + "<br>");

})

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = ["Rahul","Karan","Aman","Neha"];

a.forEach(loop);

function loop(value, index){

document.write(index + " : " + value + "<br>");

}

</script>

</head>

<body>

</body>

</html>

## JS Objects

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

### Creating Objects in JavaScript

### There are 3 ways to create objects.

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

### 1) JavaScript Object by object literal

The syntax of creating object using object literal is given below:

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

### 2) By creating instance of Object

The syntax of creating object directly is given below:

var objectname=new Object();

Here, **new keyword** is used to create object.

### 3) 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.

The **this keyword** refers to the current object.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = {

fname : LBSTI,

lname : Institute',

age : 25,

email: 'hello@lbsti.com',

}

console.log(a);

document.write(a.fname + "<br>");

document.write(a.email);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = {

fname : 'Lbsti',

lname : 'Baba',

age : 25,

email: 'hello@lbstibaba.net',

favMovies: ['Dhoom', 'Sholay', 'Hum']

}

console.log(a);

document.write(a.favMovies[0]);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = {

fname : 'Lbsti',

lname : 'Baba',

age : 25,

email: 'hello@lbstibaba.net',

favMovies: ['Dhoom', 'Sholay', 'Hum'],

salary : function(){

return 25000;

},

fullname : function(){

return this.fname + " " + this.lname;

}

}

document.write(a.salary() + "<br>");

document.write(a.fullname() + "<br>");

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = {

fname : 'Lbsti',

lname : 'Baba',

age : 25,

email: 'hello@lbstibaba.net',

favMovies: ['Dhoom', 'Sholay', 'Hum'],

living : {

'city' : 'Chandigarh',

'country' : 'India'

},

salary : function(){

return 25000;

},

fullname : function(){

return this.fname + " " + this.lname;

}

}

document.write(a.fname + "<br>");

document.write(a.lname + "<br>");

document.write(a.age + "<br>");

document.write(a.email + "<br>");

document.write(a.favMovies + "<br>");

document.write(a.favMovies[0] + "<br>");

document.write(a.favMovies[1] + "<br>");

document.write(a.favMovies[2] + "<br>");

document.write(a.living.city + "<br>");

document.write(a.salary() + "<br>");

document.write(a.fullname() + "<br>");

</script>

</head>

<body>

</body>

</html>

## JS Objects - II

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var person = new Object();

person.firstname = 'Ram';

person.lastname = 'Kumar';

person.age = 25;

document.write(person.firstname + "<br>");

document.write(person.lastname + "<br>");

document.write(person['age'] + "<br>");

</script>

</head>

<body>

</body>

</html>

## JS Array of Objects

The **Array** object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

### Syntax

Use the following syntax to create an **Array** object −

var fruits = new Array( "apple", "orange", "mango" );

The **Array** parameter is a list of strings or integers. When you specify a single numeric parameter with the Array constructor, you specify the initial length of the array. The maximum length allowed for an array is 4,294,967,295.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var student = [

{name : 'Ram', age : 15},

{name : 'Karan', age : 13},

{name : 'Rahul', age : 14},

];

console.log(student);

for(var a= 0;a < student.length;a++){

document.write(student[a].name + " " + student[a].age + "<br>");

}

</script>

</head>

<body>

</body>

</html>

## JS Const Variable with Array & Objects

CONST ARRAYS

The code above has an array variable called numbers holding three values. Even though the numbers array is a const you’re able to update or change the variable. For example, you can add another number to the numbers array by using the push method. Methods are actions you perform on the array or object.  
  
const numbers = [1,2,3];  
numbers.push(4);  
console.log(numbers) // Outpusts [1,2,3,4];

CONST OBJECTS

The modifying principle applies to an object for example.  
  
const user = {  
  name: "Gary",  
}  
  
user.age = 29  
console.log(user) // {name: "Gary", age: 29

The code above creates a user object with a name property then it assigns a new age property to object. One thing to remember const does not stop array and objects from being modified it only stops the variable itself from being reassigned or being overwritten for example.  
  
const user = {  
  name: "Gary",  
}  
  
person = { name: "Bob" } // Uncaught TypeError: Assignment to constant variable.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

const a = {

name : "Ram",

age : 25

};

a.name = "Lbsti Baba";

a.age = 55;

console.log(a);

</script>

</head>

<body>

</body>

</html>

## JS For in Loop

For-in loop in JavaScript is used to iterate over the properties of an object. It can be a great debugging tool if we want to show the contents of an object. The for-in loop iterates only over those keys of an object which have their enumerable property set to “true”. The key values in an object have four attributes (value, writable, enumerable, and configurable). Enumerable when set to “true” means that we can iterate over that property.

Syntax

for (key in object) {

// code block to be executed

}

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var obj = {

firstName : "Lbsti",

lastName : "Baba",

Age : 25,

email : "hello@lbstibaba.net"

};

for(var key in obj){

document.write(key + " : " + obj[key] + "<br>");

}

</script>

</head>

<body>

</body>

</html>

## JS Map Method

map() creates a new array from calling a function for every array element.

map() does not execute the function for empty elements.

map() does not change the original array.

Syntax

class = "table table-striped table-bordered"

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *function()* | Required. A function to be run for each array element. |
| *currentValue* | Required. The value of the current element. |
| *index* | Optional. The index of the current element. |
| *arr* | Optional. The array of the current element. |
| *thisValue* | Optional. Default value undefined. A value passed to the function to be used as its this value. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| An array | The results of a function for each array element. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ary = [11,4,9,16];

var b = ary.map(test);

document.write(b);

function test(x){

return x \* 10;

}

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var ary = [

{fname : "Lbsti" , lname : "Baba"},

{fname : "Rahul" , lname : "Kumar"},

{fname : "Karan" , lname : "Sharma"},

];

var b = ary.map(test);

document.write(b);

function test(x){

return x.fname + " " + x.lname;

}

</script>

</head>

<body>

</body>

</html>

## JS String Methods

String Methods:

| **Instance Methods** | **Description** |
| --- | --- |
| at() | Find the character at the specified index. |
| anchor() | Creates an anchor element that is used as a hypertext target. |
| charAt() | Returns that character at the given index of the string. |
| charCodeAt() | Returns a Unicode character set code unit of the character present at the index in the string. |
| codePointAt() | Return a non-negative integer value i.e, the code point value of the specified element. |
| concat() | Join two or more strings together in JavaScript. |
| endsWith() | Whether the given string ends with the characters of the specified string or not. |
| includes() | Returns true if the string contains the characters, otherwise, it returns false. |
| indexOf() | Finds the index of the first occurrence of the argument string in the given string. |
| lastIndexOf() | Finds the index of the last occurrence of the argument string in the given string. |
| localeCompare() | Compare any two elements and returns a positive number |
| match() | Search a string for a match against any regular expression. |
| matchAll() | Return all the iterators matching the reference string against a regular expression. |
| normalize() | Return a Unicode normalization form of a given input string. |
| padEnd() | Pad a string with another string until it reaches the given length from rightend. |
| padStart() | Pad a string with another string until it reaches the given length from leftend. |
| repeat() | Build a new string containing a specified number of copies of the string. |
| replace() | Replace a part of the given string with some another string or a regular expression |
| replaceAll() | Returns a new string after replacing all the matches of a string with a specified string/regex. |
| search() | Search for a match in between regular expressions and a |
| slice() | Return a part or slice of the given input string. |
| split() | Separate given string into substrings using a specified separator provided in the argument. |
| startsWith() | Check whether the given string starts with the characters of the specified string or not. |
| substr() | Returns the specified number of characters from the specified index from the given string. |
| substring() | Return the part of the given string from the start index to the end index. |
| toLocaleUpperCase() | Returns the calling string value converted to a uppercase letter. |
| toUpperCase() | Converts the entire string to uppercase. |
| toString() | Return the given string itself. |
| trim() | Remove the white spaces from both ends of the given string. |
| trimEnd() | Remove white space from the end of a string. |
| trimStart() | Remove white space from the start of a string. |
| valueOf() | Return the value of the given string. |
| string[Symbol.iterator]() | This method is used to make String iterable. [@@iterator]() returns an iterator object which iterates over all code points of the String. |
| toLowerCase() | Converts the entire string to lowercase. |
| toLocaleLowerCase() | Returns the calling string value converted to a lowercase letter. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var str = " JavaScript is a Great is Language";

var a = str.length; /\* Length Method \*/

var b = str.toLowerCase(); /\* LowerCase Method \*/

var c = str.toUpperCase(); /\* UpperCase Method \*/

var d = str.includes("ipt"); /\* Include Method \*/

var e = str.startsWith("Java"); /\* StartWith Method \*/

var f = str.endsWith("age"); /\* EndWith Method \*/

var g = str.search("is"); /\* Search Method \*/

var h = str.match(/is/g); /\* Match Method \*/

var i = str.indexOf("is"); /\* IndexOf Method \*/

var j = str.lastIndexOf("is"); /\* LastIndexOf Method \*/

var k = str.replace("JavaScript","PHP"); /\* Replace Method \*/

var l = str.trim(); /\* Trim Method \*/

document.write(a + "<br>");

document.write(b + "<br>");

document.write(c + "<br>");

document.write(d + "<br>");

document.write(e + "<br>");

document.write(f + "<br>");

document.write(g + "<br>");

document.write(h + "<br>");

document.write(i + "<br>");

document.write(j + "<br>");

document.write(k + "<br>");

alert(l);

</script>

</head>

<body>

</body>

</html>

## JS String Methods - II

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var str = "JavaScript is a Great is Language";

var str2 = "Hello";

var str3 = 50;

var a = str.charAt(3); /\* CharAt Method \*/

document.write(a + "<br>");

var b = str.charCodeAt(1); /\* CharCodeAt Method \*/

document.write(b + "<br>");

var c = str.fromCharCode(65); /\* FromCharCode Method \*/

document.write(c + "<br>");

var d = str.concat(str2); /\* Concat Method \*/

document.write(d + "<br>");

var e = str.split(" "); /\* Split Method \*/

document.write(e + "<br>");

var f = str.repeat(2); /\* Repeat Method \*/

document.write(f + "<br>");

var g = str.slice(3,10); /\* Slice Method \*/

document.write(g + "<br>");

var h = str.substr(2,5); /\* Substr Method \*/

document.write(h + "<br>");

var i = str.substring(3,5); /\* Substring Method \*/

document.write(i + "<br>");

var j = str3.toString(); /\* Tostring Method \*/

document.write(j + 20 + "<br>");

var k = str.valueOf(); /\* Valueof Method \*/

document.write(k + "<br>");

</script>

</head>

<body>

</body>

</html>

## JS Number Methods

JS Number Methods

These **number methods** can be used on all JavaScript numbers:

|  |  |
| --- | --- |
| **Method** | **Description** |
| toString() | Returns a number as a string |
| toExponential() | Returns a number written in exponential notation |
| toFixed() | Returns a number written with a number of decimals |
| toPrecision() | Returns a number written with a specified length |
| ValueOf() | Returns a number as a number |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var a = "99";

var num = Number(a);

document.write(num + 10 +"<br>");

var b = true;

var num = Number(b);

document.write(num +"<br>");

/\* ParesInt Method \*/

var c = "10.00";

var num = parseInt(c);

document.write(num + "<br>");

/\* ParesFloat Method \*/

var m = "10.33";

var num = parseFloat(m);

document.write(num + "<br>");

/\* IsFinite Method \*/

var n = 100;

var num = Number.isFinite(n);

document.write(num + "<br>");

/\* IsInteger Method \*/

var x = 10.50;

var num = Number.isInteger(x);

document.write(num + "<br>");

/\* ToFixed Method \*/

var y = 5.56789;

var num = y.toFixed(2);

document.write(num + "<br>");

/\* ToPrecision Method \*/

var z = 5.56789;

var num = z.toPrecision(3);

document.write(num + "<br>");

</script>

</head>

<body>

</body>

</html>

## JS Math Methods

JavaScript Math

The **JavaScript math** object provides several constants and methods to perform mathematical operation. Unlike date object, it doesn't have constructors.

JavaScript Math Methods

Let's see the list of JavaScript Math methods with description.

|  |  |
| --- | --- |
| **Methods** | **Description** |
| abs() | It returns the absolute value of the given number. |
| acos() | It returns the arccosine of the given number in radians. |
| asin() | It returns the arcsine of the given number in radians. |
| atan() | It returns the arc-tangent of the given number in radians. |
| cbrt() | It returns the cube root of the given number. |
| ceil() | It returns a smallest integer value, greater than or equal to the given number. |
| cos() | It returns the cosine of the given number. |
| cosh() | It returns the hyperbolic cosine of the given number. |
| exp() | It returns the exponential form of the given number. |
| floor() | It returns largest integer value, lower than or equal to the given number. |
| hypot() | It returns square root of sum of the squares of given numbers. |
| log() | It returns natural logarithm of a number. |
| max() | It returns maximum value of the given numbers. |
| min() | It returns minimum value of the given numbers. |
| pow() | It returns value of base to the power of exponent. |
| random() | It returns random number between 0 (inclusive) and 1 (exclusive). |
| round() | It returns closest integer value of the given number. |
| sign() | It returns the sign of the given number |
| sin() | It returns the sine of the given number. |
| sinh() | It returns the hyperbolic sine of the given number. |
| sqrt() | It returns the square root of the given number |
| tan() | It returns the tangent of the given number. |
| tanh() | It returns the hyperbolic tangent of the given number. |
| trunc() | It returns an integer part of the given number. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

/\* Ceil Method Example \*/

var a = Math.ceil(1.2);

document.write(a + "<br>");

/\* Floor Method \*/

var b = Math.floor(5.10);

document.write(b + "<br>");

/\* Round Method \*/

var c = Math.round(2.60);

document.write(c + "<br>");

/\* Trunc Method \*/

var d = Math.trunc(8.19);

document.write(d + "<br>");

/\* Max Method \*/

var i = Math.max(8, 10 ,2 ,50 ,25);

document.write(i + "<br>");

/\* Min Method \*/

var j = Math.min(8, 10, 2, 50, 25);

document.write(j + "<br>");

/\* Sqrt Method \*/

var k = Math.sqrt(64);

document.write(k + "<br>");

/\* Cbrt Method \*/

var l = Math.cbrt(125);

document.write(l + "<br>");

/\*Pow Method \*/

var p = Math.pow(2,3);

document.write(p + "<br>");

/\*Random Method \*/

var x = Math.floor(Math.random() \* 10) + 1;

document.write(x + "<br>");

/\*Abs Method \*/

var y = Math.abs(5.25);

document.write(y + "<br>");

/\*Pi Method \*/

var z = Math.PI;

document.write(z + "<br>");

</script>

</head>

<body>

</body>

</html>

## JS Date Methods

The new Date() Constructor

In JavaScript, date objects are created with new Date().

new Date() returns a date object with the current date and time.

Get the Current Time

const date = new Date();

Date Get Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| getFullYear() | Get **year** as a four digit number (yyyy) |
| getMonth() | Get **month** as a number (0-11) |
| getDate() | Get **day** as a number (1-31) |
| getDay() | Get **weekday** as a number (0-6) |
| getHours() | Get **hour** (0-23) |
| getMinutes() | Get **minute** (0-59) |
| getSeconds() | Get **second** (0-59) |
| getMilliseconds() | Get **millisecond** (0-999) |
| getTime() | Get **time** (milliseconds since January 1, 1970) |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

var now = new Date();

/\* ToDateString Method Example \*/

document.write(now.toDateString() + "<br><br>");

/\* GetDate Method \*/

document.write(now.getDate() + "<br><br>");

/\* GetFullYear Method \*/

document.write(now.getFullYear() + "<br><br>");

/\* GetMonth Method \*/

document.write(now.getMonth() + "<br><br>");

/\* GetDay Method \*/

document.write(now.getDay() + "<br><br>");

/\* GetHours Method \*/

document.write(now.getHours() + "<br><br>");

/\* GetMinutes Method \*/

document.write(now.getMinutes() + "<br><br>");

/\* GetSeconds Method \*/

document.write(now.getSeconds() + "<br><br>");

/\* GetMilliseconds Method \*/

document.write(now.getMilliseconds() + "<br><br>");

/\*setDate Method \*/

now.setDate(20);

/\*setFullYear Method \*/

now.setFullYear(2020);

/\*setFullYear Method \*/

now.setMonth(4);

/\*setHours Method \*/

now.setHours(14);

document.write(now.getDate() + "/" + now.getMonth() + "/" + now.getFullYear());

</script>

</head>

<body>

</body>

</html>

## JS DOM Introduction

### The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:  
  
With the object model, JavaScript gets all the power it needs to create dynamic HTML:

* JavaScript can change all the HTML elements in the page
* JavaScript can change all the HTML attributes in the page
* JavaScript can change all the CSS styles in the page
* JavaScript can remove existing HTML elements and attributes
* JavaScript can add new HTML elements and attributes
* JavaScript can react to all existing HTML events in the page
* JavaScript can create new HTML events in the page

### What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

* Core DOM - standard model for all document types
* XML DOM - standard model for XML documents
* HTML DOM - standard model for HTML documents

### What is the HTML DOM?

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

* The HTML elements as **objects**
* The **properties** of all HTML elements
* The **methods** to access all HTML elements
* The **events** for all HTML elements

## JS DOM Targeting Methods

### JS DOM Targeting Methods

Most often, you want to manipulate HTML elements. To manipulate element you have to find the elements first.  
Javascript provides us with various methods to find an element within the document.

| **Methods** | **Description** |
| --- | --- |
| document.getElementById() | Select the unique element with given id. In case there are 2 same ID then it selects the first element. |
| document.getElementsByClassName() | Select collection elements with given classname |
| document.getElementsByTagName() | Select collection elements with given tagname |

### DOM getElementsById() Method

The **getElementById()** method returns the elements that have given an ID which is passed to the function. This function is a widely used HTML DOM method in web designing to change the value of any particular element or get a particular element. If the passed ID to the function does not exist then it returns null. The element is required to have a unique id, in order to get access to that specific element quickly, & also that particular id should only be used once in the entire document.

**Syntax:**

document.getElementById( element\_ID )

**Parameter:** This function accepts single parameter element\_ID which is used to hold the ID of the element.

**Return Value:** It returns the object of the given ID. If no element exists with the given ID then it returns null.

### DOM getElementsByClassName() Method

The **getElementsByClassName()** method in Javascript returns an object containing all the elements with the specified class names in the document as objects. Each element in the returned object can be accessed by its index. The index value will start with 0. This method can be called upon by any individual element to search for its descendant elements with the specified class names.

**Syntax:**

document.getElementsByClassName(classnames);

**Parameters:** This is a required method that takes only one parameter, which is a string containing space-separated class names of the elements that are to be searched for. For searching with multiple class names, it must be separated with space.

### DOM getElementsByTagName() Method

The HTML**DOM getElementsByTagName()**method in HTML returns the collection of all the elements in the document with the given tag name. To extract any info just iterate through all the elements using the length property.

**Syntax:**

var elements = document.getElementsByTagName(name);

var elements = document.getElementsByTagName(name);

**Parameters:**

* **elements** is a collection of all the found elements in the order they appear with the given tag name.
* **name** is a string representing the name of the elements. The special string “\*” represents all elements.

### html file

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<link rel="stylesheet" href="css/main.css">

<script src="js/dom-main.js"></script>

</head>

<body>

<div id="wrapper">

<div id="header">

<h1>Lbsti Baba</h1>

</div>

<div id="menu">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="content">

<h2>Sub Heading</h2>

<img src="images/flower.jpg" width="200px" class="content-image" alt="">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt, veniam eius architecto ullam cupiditate quam aspernatur quis facilis tempora vel! Aspernatur, consequatur, laborum, explicabo consequuntur minima beatae perferendis impedit accusantium ex animi odit quisquam sint pariatur minus amet ullam reprehenderit rerum inventore sed officiis voluptas? Dolore, perferendis, minus eum debitis vero ipsam tempora voluptate nam ut autem itaque provident consequatur nobis quia libero! Magni, molestiae, laborum architecto natus inventore facere quis pariatur quia quos quasi quo dicta dolores. Deleniti, facere, fugit sed minus ducimus ut modi voluptatum eaque praesentium saepe aperiam nam quidem laboriosam assumenda voluptate vitae inventore et quibusdam!</p>

<ul class="list">

<li>Lorem ipsum dolor sit amet. </li>

<li>Modi nihil in animi necessitatibus.</li>

<li>Consectetur adipisicing elit.</li>

<li>Lorem ipsum dolor sit amet.</li>

<li>Modi nihil in animi dolore natus.</li>

</ul>

<p class="list">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, quam.</p>

</div>

<div id="sidebar">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="footer">

lbstibaba@copyright 2018.

</div>

</div>

</body>

</html>

### main.css

@import "color.css";

\*{

box-sizing: border-box;

}

body{

font: 18px/24px arial;

}

#wrapper{

border:1px solid black;

width: 1000px;

background: white;

margin: 0 auto;

}

#top-bar{

background: MEDIUMPURPLE;

}

#top-bar ul{

margin: 0;

padding: 0;

list-style: none;

}

#top-bar ul li{

display: inline-block;

margin: 5px;

}

#top-bar a{

color: #fff;

font-size: 20px;

}

#top-bar a:hover{

color: crimson;

}

#header{

min-height: 100px;

font-family: arial;

border-bottom: 1px solid black;

}

#header h1{

padding:30px 0 0 20px;

margin: 0;

}

#menu{

border-bottom: 1px solid black;

}

#menu ul{

margin: 0;

padding: 0;

}

#menu li{

display:inline-block;

}

#menu li a{

display:block;

padding: 10px 13px;

text-decoration:none;

}

#menu li a:hover{

}

#content{

width:795px;

min-height:500px;

padding: 15px;

float:left;

box-sizing:border-box;

position:relative;

}

#content h2{

font-family: arial;

}

.content-image{

float:left;margin:0 15px 10px 0;

}

#sidebar{

width:200px;

min-height:500px;

float:right;

border-left: 1px solid black;

}

#sidebar a{

text-decoration:none;

color:black;

}

#footer{

padding: 5px 10px;

text-align:right;

font-size: 14px;

clear:both;

border-top: 1px solid black;

}

@import "mediatest.css" screen and (max-width:1020px);

@media screen and (max-width:810px){

#wrapper{

width:500px;

}

#content{

width:65%;

}

#sidebar{

width:34%

}

}

@media screen and (max-width:510px){

#wrapper{

width:auto;

}

#content{

width:100%;

float:none;

}

#sidebar{

width:100%;

float:none;

}

#header h1{

padding:30px 0 0 0;

text-align:center;

}

}

### dom-main.js

var element;

//element = document.all;

//element = document.head;

//element = document.title;

//element = document.body;

//element = document.links;

//element = document.links[0];

//element = document.images;

//element = document.forms;

//element = document.doctype;

//element = document.URL;

//element = document.domain;

//element = document.baseURI;

//element = document.getElementById("header");

//element = document.getElementsByClassName("list");

//element = document.getElementsByClassName("list")[0]

//element = document.getElementsByTagName("ul");

element = document.getElementsByTagName("ul")[2]

console.log(element);

## JS DOM Get & Set Value Methods

Getting Element's Attribute Value

The getAttribute() method is used to get the current value of a attribute on the element.

If the specified attribute does not exist on the element, it will return null.

Setting Attributes on Elements

The setAttribute() method is used to set an attribute on the specified element.

If the attribute already exists on the element, the value is updated; otherwise a new attribute is added with the specified name and value.

html file

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<link rel="stylesheet" href="css/main.css">

</head>

<body>

<div id="wrapper">

<div id="header" class="abc" style="border:10px solid yellow;">

<h1>Lbsti Baba</h1>

</div>

<div id="menu">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="content">

<h2>Sub Heading</h2>

<img src="images/flower.jpg" width="200px" class="content-image" alt="">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt, veniam eius architecto ullam cupiditate quam aspernatur quis facilis tempora vel! Aspernatur, consequatur, laborum, explicabo consequuntur minima beatae perferendis impedit accusantium ex animi odit quisquam sint pariatur minus amet ullam reprehenderit rerum inventore sed officiis voluptas? Dolore, perferendis, minus eum debitis vero ipsam tempora voluptate nam ut autem itaque provident consequatur nobis quia libero! Magni, molestiae, laborum architecto natus inventore facere quis pariatur quia quos quasi quo dicta dolores. Deleniti, facere, fugit sed minus ducimus ut modi voluptatum eaque praesentium saepe aperiam nam quidem laboriosam assumenda voluptate vitae inventore et quibusdam!</p>

<ul class="list">

<li>Lorem ipsum dolor sit amet. </li>

<li>Modi nihil in animi necessitatibus.</li>

<li>Consectetur adipisicing elit.</li>

<li>Lorem ipsum dolor sit amet.</li>

<li>Modi nihil in animi dolore natus.</li>

</ul>

<p class="list">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, quam.</p>

</div>

<div id="sidebar">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="footer">

lbstibaba@copyright 2018.

</div>

</div>

<script src="js/dom-main.js"></script>

</body>

</html>

main.css

@import "color.css";

\*{

box-sizing: border-box;

}

body{

font: 18px/24px arial;

}

#wrapper{

border:1px solid black;

width: 1000px;

background: white;

margin: 0 auto;

}

#top-bar{

background: MEDIUMPURPLE;

}

#top-bar ul{

margin: 0;

padding: 0;

list-style: none;

}

#top-bar ul li{

display: inline-block;

margin: 5px;

}

#top-bar a{

color: #fff;

font-size: 20px;

}

#top-bar a:hover{

color: crimson;

}

#header{

min-height: 100px;

font-family: arial;

border-bottom: 1px solid black;

}

#header h1{

padding:30px 0 0 20px;

margin: 0;

}

#menu{

border-bottom: 1px solid black;

}

#menu ul{

margin: 0;

padding: 0;

}

#menu li{

display:inline-block;

}

#menu li a{

display:block;

padding: 10px 13px;

text-decoration:none;

}

#menu li a:hover{

}

#content{

width:795px;

min-height:500px;

padding: 15px;

float:left;

box-sizing:border-box;

position:relative;

}

#content h2{

font-family: arial;

}

.content-image{

float:left;margin:0 15px 10px 0;

}

#sidebar{

width:200px;

min-height:500px;

float:right;

border-left: 1px solid black;

}

#sidebar a{

text-decoration:none;

color:black;

}

#footer{

padding: 5px 10px;

text-align:right;

font-size: 14px;

clear:both;

border-top: 1px solid black;

}

@import "mediatest.css" screen and (max-width:1020px);

@media screen and (max-width:810px){

#wrapper{

width:500px;

}

#content{

width:65%;

}

#sidebar{

width:34%

}

}

@media screen and (max-width:510px){

#wrapper{

width:auto;

}

#content{

width:100%;

float:none;

}

#sidebar{

width:100%;

float:none;

}

#header h1{

padding:30px 0 0 0;

text-align:center;

}

}

dom-main.js

var element;

//element = document.getElementById("header").innerText;

//element = document.getElementById("content").innerText;

//element = document.getElementById("content").innerHTML;

//element = document.getElementById("header").innerHTML;

//element = document.getElementById("header").getAttribute("class");

//element = document.getElementById("header").getAttribute("style");

//element = document.getElementById("header").getAttribute("onClick");

//element = document.getElementById("header").getAttributeNode("onClick");

//element = document.getElementById("header").getAttributeNode("style");

//element = document.getElementById("header").getAttributeNode("style").value;

//element = document.getElementById("header").attributes;

//element = document.getElementById("header").attributes[1];

//element = document.getElementById("header").attributes[2].name;

/\* DOM Set Method \*/

//document.getElementById("header").innerHTML = "<h1>WOW</h1>";

//element = document.getElementById("header").setAttribute("style","border:10px dotted yellow");

//element = document.getElementById("header").attributes[1].value = "xyz";

element = document.getElementById("header").removeAttribute("style");

element = document.getElementById("header").removeAttribute("class");

console.log(element);

## JS DOM querySelectors

JS DOM querySelectors

The querySelector() method returns the first child element that matches a specified *CSS selector(s)* of an element.

**Note:** The querySelector() method only returns the first element that matches the specified selectors. To return all the matches, use the querySelectorAll() method instead.

html file

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<link rel="stylesheet" href="css/main.css">

</head>

<body>

<div id="wrapper">

<div id="header" class="abc">

<h1>Lbsti Baba</h1>

<h1>Lbsti Babaaaaaaaa</h1>

</div>

<div id="menu">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="content">

<h2>Sub Heading</h2>

<img src="images/flower.jpg" width="200px" class="content-image" alt="">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt, veniam eius architecto ullam cupiditate quam aspernatur quis facilis tempora vel! Aspernatur, consequatur, laborum, explicabo consequuntur minima beatae perferendis impedit accusantium ex animi odit quisquam sint pariatur minus amet ullam reprehenderit rerum inventore sed officiis voluptas? Dolore, perferendis, minus eum debitis vero ipsam tempora voluptate nam ut autem itaque provident consequatur nobis quia libero! Magni, molestiae, laborum architecto natus inventore facere quis pariatur quia quos quasi quo dicta dolores. Deleniti, facere, fugit sed minus ducimus ut modi voluptatum eaque praesentium saepe aperiam nam quidem laboriosam assumenda voluptate vitae inventore et quibusdam!</p>

<ul class="list">

<li>Lorem ipsum dolor sit amet. </li>

<li>Modi nihil in animi necessitatibus.</li>

<li>Consectetur adipisicing elit.</li>

<li>Lorem ipsum dolor sit amet.</li>

<li>Modi nihil in animi dolore natus.</li>

</ul>

<p class="list">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, quam.</p>

</div>

<div id="sidebar">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="footer">

lbstibaba@copyright 2018.

</div>

</div>

<script src="js/dom-main.js"></script>

</body>

</html>

main.css

@import "color.css";

\*{

box-sizing: border-box;

}

body{

font: 18px/24px arial;

}

#wrapper{

border:1px solid black;

width: 1000px;

background: white;

margin: 0 auto;

}

#top-bar{

background: MEDIUMPURPLE;

}

#top-bar ul{

margin: 0;

padding: 0;

list-style: none;

}

#top-bar ul li{

display: inline-block;

margin: 5px;

}

#top-bar a{

color: #fff;

font-size: 20px;

}

#top-bar a:hover{

color: crimson;

}

#header{

min-height: 100px;

font-family: arial;

border-bottom: 1px solid black;

}

#header h1{

padding:30px 0 0 20px;

margin: 0;

}

#menu{

border-bottom: 1px solid black;

}

#menu ul{

margin: 0;

padding: 0;

}

#menu li{

display:inline-block;

}

#menu li a{

display:block;

padding: 10px 13px;

text-decoration:none;

}

#menu li a:hover{

}

#content{

width:795px;

min-height:500px;

padding: 15px;

float:left;

box-sizing:border-box;

position:relative;

}

#content h2{

font-family: arial;

}

.content-image{

float:left;margin:0 15px 10px 0;

}

#sidebar{

width:200px;

min-height:500px;

float:right;

border-left: 1px solid black;

}

#sidebar a{

text-decoration:none;

color:black;

}

#footer{

padding: 5px 10px;

text-align:right;

font-size: 14px;

clear:both;

border-top: 1px solid black;

}

@import "mediatest.css" screen and (max-width:1020px);

@media screen and (max-width:810px){

#wrapper{

width:500px;

}

#content{

width:65%;

}

#sidebar{

width:34%

}

}

@media screen and (max-width:510px){

#wrapper{

width:auto;

}

#content{

width:100%;

float:none;

}

#sidebar{

width:100%;

float:none;

}

#header h1{

padding:30px 0 0 0;

text-align:center;

}

}

dom-main.js

var element;

//element = document.querySelector("#header").innerHTML = "<h1>WOW</h1>";

//element = document.querySelector("#header").getAttribute ("class");

//element = document.querySelectorAll(".list");

element = document.querySelectorAll(".list")[1].innerHTML;

element = document.querySelectorAll("ul");

element = document.querySelectorAll("ul")[1].innerHTML;

element = document.querySelectorAll("#header h1");

element = document.querySelectorAll("#header h1")[1].innerHTML;

console.log(element);

## JS DOM CSS Styling Methods

html file

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<link rel="stylesheet" href="css/main.css">

</head>

<body>

<div id="wrapper">

<div id="header" class="abc">

<h1>Lbsti Baba</h1>

</div>

<div id="menu">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="content">

<h2>Sub Heading</h2>

<img src="images/flower.jpg" width="200px" class="content-image" alt="">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt, veniam eius architecto ullam cupiditate quam aspernatur quis facilis tempora vel! Aspernatur, consequatur, laborum, explicabo consequuntur minima beatae perferendis impedit accusantium ex animi odit quisquam sint pariatur minus amet ullam reprehenderit rerum inventore sed officiis voluptas? Dolore, perferendis, minus eum debitis vero ipsam tempora voluptate nam ut autem itaque provident consequatur nobis quia libero! Magni, molestiae, laborum architecto natus inventore facere quis pariatur quia quos quasi quo dicta dolores. Deleniti, facere, fugit sed minus ducimus ut modi voluptatum eaque praesentium saepe aperiam nam quidem laboriosam assumenda voluptate vitae inventore et quibusdam!</p>

<ul class="list">

<li>Lorem ipsum dolor sit amet. </li>

<li>Modi nihil in animi necessitatibus.</li>

<li>Consectetur adipisicing elit.</li>

<li>Lorem ipsum dolor sit amet.</li>

<li>Modi nihil in animi dolore natus.</li>

</ul>

<p class="list">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, quam.</p>

</div>

<div id="sidebar">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="footer">

lbstibaba@copyright 2018.

</div>

</div>

<script src="js/dom-main.js"></script>

</body>

</html>

main.css

@import "color.css";

\*{

box-sizing: border-box;

}

body{

font: 18px/24px arial;

}

#wrapper{

border:1px solid black;

width: 1000px;

background: white;

margin: 0 auto;

}

#top-bar{

background: MEDIUMPURPLE;

}

#top-bar ul{

margin: 0;

padding: 0;

list-style: none;

}

#top-bar ul li{

display: inline-block;

margin: 5px;

}

#top-bar a{

color: #fff;

font-size: 20px;

}

#top-bar a:hover{

color: crimson;

}

#header{

min-height: 100px;

font-family: arial;

border-bottom: 1px solid black;

}

#header h1{

padding:30px 0 0 20px;

margin: 0;

}

#menu{

border-bottom: 1px solid black;

}

#menu ul{

margin: 0;

padding: 0;

}

#menu li{

display:inline-block;

}

#menu li a{

display:block;

padding: 10px 13px;

text-decoration:none;

}

#menu li a:hover{

}

#content{

width:795px;

min-height:500px;

padding: 15px;

float:left;

box-sizing:border-box;

position:relative;

}

#content h2{

font-family: arial;

}

.content-image{

float:left;margin:0 15px 10px 0;

}

#sidebar{

width:200px;

min-height:500px;

float:right;

border-left: 1px solid black;

}

#sidebar a{

text-decoration:none;

color:black;

}

#footer{

padding: 5px 10px;

text-align:right;

font-size: 14px;

clear:both;

border-top: 1px solid black;

}

@import "mediatest.css" screen and (max-width:1020px);

@media screen and (max-width:810px){

#wrapper{

width:500px;

}

#content{

width:65%;

}

#sidebar{

width:34%

}

}

@media screen and (max-width:510px){

#wrapper{

width:auto;

}

#content{

width:100%;

float:none;

}

#sidebar{

width:100%;

float:none;

}

#header h1{

padding:30px 0 0 0;

text-align:center;

}

}

main-dom.js

var element;

//element = document.querySelector("#header").style.border;

//element = document.querySelector("#header").style.color;

//document.querySelector("#header").style.backgroundColor = "tan";

//document.querySelector("#header").style.color = "blue";

//document.querySelector("#header").className = "abc xyz";

//element = document.querySelector("#header").className;

//document.querySelector("#header").classList = "abc xyz";

//element = document.querySelector("#header").classList;

//document.querySelector("#header").classList.add("xyz","efg");

//element = document.querySelector("#header").classList;

//document.querySelector("#header").classList.remove("xyz");

element = document.querySelector("#header").classList;

//console.log(element);

## JS addEventListener Method

JS  addEventListener Method

The **addEventListener()** method is used to attach an event handler to a particular element. It does not override the existing event handlers. Events are said to be an essential part of the JavaScript. A web page responds according to the event that occurred. Events can be user-generated or generated by API's. An event listener is a JavaScript's procedure that waits for the occurrence of an event.

The addEventListener() method is an inbuilt function of . We can add multiple event handlers to a particular element without overwriting the existing event handlers.

Syntax:

element.addEventListener(event, function, useCapture);

Although it has three parameters, the parameters ***event*** and ***function*** are widely used. The third parameter is optional to define. The values of this function are defined as follows.

Parameter Values

**event:** It is a required parameter. It can be defined as a string that specifies the event's name.  
  
**function:** It is also a required parameter. It is a JavaScript function which responds to the event occur.  
  
**useCapture:** It is an optional parameter. It is a Boolean type value that specifies whether the event is executed in the bubbling or capturing phase. Its possible values are **true** and **false**. When it is set to true, the event handler executes in the capturing phase. When it is set to false, the handler executes in the bubbling phase. Its default value is **false**.

html file for add event listener and remove event listener

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<link rel="stylesheet" href="css/main.css">

</head>

<body>

<div id="wrapper">

<div id="header" class="abc">

<h1>Lbsti Baba</h1>

</div>

<div id="menu">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="content">

<h2>Sub Heading</h2>

<img src="images/flower.jpg" width="200px" class="content-image" alt="">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt, veniam eius architecto ullam cupiditate quam aspernatur quis facilis tempora vel! Aspernatur, consequatur, laborum, explicabo consequuntur minima beatae perferendis impedit accusantium ex animi odit quisquam sint pariatur minus amet ullam reprehenderit rerum inventore sed officiis voluptas? Dolore, perferendis, minus eum debitis vero ipsam tempora voluptate nam ut autem itaque provident consequatur nobis quia libero! Magni, molestiae, laborum architecto natus inventore facere quis pariatur quia quos quasi quo dicta dolores. Deleniti, facere, fugit sed minus ducimus ut modi voluptatum eaque praesentium saepe aperiam nam quidem laboriosam assumenda voluptate vitae inventore et quibusdam!</p>

<ul class="list">

<li>Lorem ipsum dolor sit amet. </li>

<li>Modi nihil in animi necessitatibus.</li>

<li>Consectetur adipisicing elit.</li>

<li>Lorem ipsum dolor sit amet.</li>

<li>Modi nihil in animi dolore natus.</li>

</ul>

<p class="list">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, quam.</p>

</div>

<div id="sidebar">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="footer">

lbstibaba@copyright 2018.

</div>

</div>

<script src="js/dom-main.js"></script>

</body>

</html>

main.css for add event listener and remove event listener

@import "color.css";

\*{

box-sizing: border-box;

}

body{

font: 18px/24px arial;

}

#wrapper{

border:1px solid black;

width: 1000px;

background: white;

margin: 0 auto;

}

#top-bar{

background: MEDIUMPURPLE;

}

#top-bar ul{

margin: 0;

padding: 0;

list-style: none;

}

#top-bar ul li{

display: inline-block;

margin: 5px;

}

#top-bar a{

color: #fff;

font-size: 20px;

}

#top-bar a:hover{

color: crimson;

}

#header{

min-height: 100px;

font-family: arial;

border-bottom: 1px solid black;

}

#header h1{

padding:30px 0 0 20px;

margin: 0;

}

#menu{

border-bottom: 1px solid black;

}

#menu ul{

margin: 0;

padding: 0;

}

#menu li{

display:inline-block;

}

#menu li a{

display:block;

padding: 10px 13px;

text-decoration:none;

}

#menu li a:hover{

}

#content{

width:795px;

min-height:500px;

padding: 15px;

float:left;

box-sizing:border-box;

position:relative;

}

#content h2{

font-family: arial;

}

.content-image{

float:left;margin:0 15px 10px 0;

}

#sidebar{

width:200px;

min-height:500px;

float:right;

border-left: 1px solid black;

}

#sidebar a{

text-decoration:none;

color:black;

}

#footer{

padding: 5px 10px;

text-align:right;

font-size: 14px;

clear:both;

border-top: 1px solid black;

}

@import "mediatest.css" screen and (max-width:1020px);

@media screen and (max-width:810px){

#wrapper{

width:500px;

}

#content{

width:65%;

}

#sidebar{

width:34%

}

}

@media screen and (max-width:510px){

#wrapper{

width:auto;

}

#content{

width:100%;

float:none;

}

#sidebar{

width:100%;

float:none;

}

#header h1{

padding:30px 0 0 0;

text-align:center;

}

}

dom-main.js for add event listner and remove event listener

// onclick event

//document.getElementById("header").onclick = abc;

// onmouseenter event

//document.getElementById("header").onmouseenter = abc;

// with addEventListener

/\* document.getElementById("header").addEventListener("mouseenter",abc);

document.getElementById("header").addEventListener("click", function(){

document.getElementById("header").style.border = "10px solid red";

}); \*/

/\* document.getElementById("header").addEventListener("click", abc);

document.getElementById("header").addEventListener("click", function() {

this.style.border = "10px solid red";

}); \*/

// with removeEventListener

//document.getElementById("header").removeEventListener('mouseleave',abc);

//document.getElementById("header").addEventListener("click",xyz);

function abc(){

document.getElementById("header").style.background = "green";

}

function xyz() {

document.getElementById("header").removeEventListener('mouseleave',abc);

}

/\* ====================================================== \*/

html for usecapture

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<style>

#outer{

width: 500px;

height: 500px;

margin: 0 auto;

background: LIGHTSALMON;

}

#inner{

width: 300px;

height: 300px;

margin: 100px auto 0;

background: MEDIUMORCHID;

}

</style>

</head>

<body>

<div id="outer">

<h2>Outer</h2>

<div id="inner">

<h2>Inner</h2>

</div>

</div>

<script src="js/usecapture.js"></script>

</body>

</html>

usecapture.js

document.querySelector("#inner").addEventListener('click',function(){

alert('Inner Div');

},false);

document.querySelector("#outer").addEventListener('click',function(){

alert('Outer Div');

},false);

## JS classList Methods

### JS classList Methods

The classList property is used for representing the value of the class elements which is a **DOMTokenList** object. It is a read-only property but we can modify its value by manipulating the classes used in the program. The JavaScript classList property consists of following methods through which we can perform different operations on the class elements:

* **add():** The add() method is used for adding one or more classes to the element.
* **remove():** The remove() method is used for removing one or more classes from the number of classes present in the element.
* **toggle():** The toggle() method is used for toggling the specified class names of an element. It means on one click the specified class gets added and on another click the class gets removed. It is known as the toggle property of an element.
* **replace():** The replace() method is used for replacing an existing class with a new class.
* **contains():** The contains() method of the JavaScript classList property is used for returning the Boolean value as an output. If the class is present, the value is returned as true otherwise false is returned.
* **item():** The item() method is used for displaying the name of the classes at the particular index. Thus, it returns the class name.

### html file

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<link rel="stylesheet" href="css/main.css">

</head>

<body>

<div id="wrapper">

<div id="header" class="abc">

<h1>Lbsti Baba</h1>

</div>

<div id="menu">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="content">

<h2>Sub Heading</h2>

<img src="images/flower.jpg" width="200px" class="content-image" alt="">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt, veniam eius architecto ullam cupiditate quam aspernatur quis facilis tempora vel! Aspernatur, consequatur, laborum, explicabo consequuntur minima beatae perferendis impedit accusantium ex animi odit quisquam sint pariatur minus amet ullam reprehenderit rerum inventore sed officiis voluptas? Dolore, perferendis, minus eum debitis vero ipsam tempora voluptate nam ut autem itaque provident consequatur nobis quia libero! Magni, molestiae, laborum architecto natus inventore facere quis pariatur quia quos quasi quo dicta dolores. Deleniti, facere, fugit sed minus ducimus ut modi voluptatum eaque praesentium saepe aperiam nam quidem laboriosam assumenda voluptate vitae inventore et quibusdam!</p>

<ul class="list">

<li>Lorem ipsum dolor sit amet. </li>

<li>Modi nihil in animi necessitatibus.</li>

<li>Consectetur adipisicing elit.</li>

<li>Lorem ipsum dolor sit amet.</li>

<li>Modi nihil in animi dolore natus.</li>

</ul>

<p class="list">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, quam.</p>

</div>

<div id="sidebar">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="footer">

lbstibaba@copyright 2018.

</div>

</div>

<script src="js/dom-main.js"></script>

</body>

</html>

### main.css

@import "color.css";

\*{

box-sizing: border-box;

}

body{

font: 18px/24px arial;

}

#wrapper{

border:1px solid black;

width: 1000px;

background: white;

margin: 0 auto;

}

#top-bar{

background: MEDIUMPURPLE;

}

#top-bar ul{

margin: 0;

padding: 0;

list-style: none;

}

#top-bar ul li{

display: inline-block;

margin: 5px;

}

#top-bar a{

color: #fff;

font-size: 20px;

}

#top-bar a:hover{

color: crimson;

}

#header{

min-height: 100px;

font-family: arial;

border-bottom: 1px solid black;

}

#header h1{

padding:30px 0 0 20px;

margin: 0;

}

#menu{

border-bottom: 1px solid black;

}

#menu ul{

margin: 0;

padding: 0;

}

#menu li{

display:inline-block;

}

#menu li a{

display:block;

padding: 10px 13px;

text-decoration:none;

}

#menu li a:hover{

}

#content{

width:795px;

min-height:500px;

padding: 15px;

float:left;

box-sizing:border-box;

position:relative;

}

#content h2{

font-family: arial;

}

.content-image{

float:left;margin:0 15px 10px 0;

}

#sidebar{

width:200px;

min-height:500px;

float:right;

border-left: 1px solid black;

}

#sidebar a{

text-decoration:none;

color:black;

}

#footer{

padding: 5px 10px;

text-align:right;

font-size: 14px;

clear:both;

border-top: 1px solid black;

}

@import "mediatest.css" screen and (max-width:1020px);

@media screen and (max-width:810px){

#wrapper{

width:500px;

}

#content{

width:65%;

}

#sidebar{

width:34%

}

}

@media screen and (max-width:510px){

#wrapper{

width:auto;

}

#content{

width:100%;

float:none;

}

#sidebar{

width:100%;

float:none;

}

#header h1{

padding:30px 0 0 0;

text-align:center;

}

}

### < dom-main.js

document.getElementById("header").addEventListener("click", abc);

function abc() {

//document.getElementById("header").classList.add("xyz","efg");

//document.getElementById("header").classList.remove("xyz");

//var a = document.getElementById("header").classList.length;

//document.getElementById("header").classList.toggle("xyz");

//var a = document.getElementById("header").classList;

//var a = document.getElementById("header").classList.item(0);

var a = document.getElementById("header").classList.contains("first");

console.log(a);

}

## JS parent Method

### parentNode:

The parentNode property returns the parent node of an element or node.

The parentNode property is read-only.

### parentElement:

The parentElement property returns the parent element of the specified element.

The difference between parentElement and parentNode, is that parentElement returns null if the parent node is not an element node:

### html file

<!DOCTYPE html>

<html id="main">

<head>

<title>DOM Navigation</title>

</head>

<style>

#outer{

width: 550px;

height: 300px;

padding:10px 10px;

margin: 0 auto;

background: lightsalmon;

}

#inner{

width: 500px;

height: 200px;

padding:10px 10px;

margin:0 auto ;

background: mediumorchid;

}

#inner div{

display: inline-block;

background: #fff;

width: 95px;

height: 50px;

line-height: 50px;

text-align: center;

}

</style>

<body>

<div id="outer">

<h2>Outer</h2>

<div id="inner">

<h2>Inner</h2>

<div>A</div>

<div>B</div>

<div id="child-c">C</div>

<div>D</div>

<div>E</div>

</div>

</div>

<script src="js/dom-nav.js"></script>

</body>

</html>

### dom-nav.js

//var a = document.getElementById("inner").parentElement;

//var a = document.getElementById("outer").parentElement;

//var a = document.body.parentElement;

//var a = document.getElementById("inner").parentElement.style.background = "red";

/\* document.getElementById("child-c").parentElement.style.background = "red";

var a = document.getElementById("child-c").parentElement; \*/

document.getElementById("child-c").parentElement.style.background = "red";

var a = document.getElementById("main").parentNode;

console.log(a);

## JS Children Methods

### JS Children Methods

The children property returns a collection of an element's child elements.

The children property returns an HTMLCollection object.

### childNodes:

The childNodes property returns a collection (list) of an elements's child nodes.

The childNodes property returns a NodeList object.

The childNodes property is read-only.

childNodes[0] is the same as firstChild.

### html file

<!DOCTYPE html>

<html id="main">

<head>

<title>Basic Layout</title>

</head>

<style>

#outer {

width: 550px;

height: 300px;

padding: 10px 10px;

margin: 0 auto;

background: lightsalmon;

}

#inner {

width: 500px;

height: 200px;

padding: 10px 10px;

margin: 0 auto;

background: mediumorchid;

}

#inner div {

display: inline-block;

background: #fff;

width: 95px;

height: 50px;

line-height: 50px;

text-align: center;

}

</style>

<body>

<div id="outer">

<h2>Outer</h2>

<div id="inner">

<h2 id="child-head">Inner</h2>

<div>A</div>

<div>B</div>

<div id="child-C">C</div>

<div>D</div>

<div id="child-E">E</div>

</div>

</div>

<script src="js/dom-nav.js"></script>

</body>

</html>

### dom-nav.js

//var a = document.getElementById("outer").children;

//var a = document.getElementById("inner").children;

//var a = document.getElementById("inner").children[1];

/\* document.getElementById("inner").children[1].style.background = "red";

var a = document.getElementById("inner").children[1]; \*/

/\* document.getElementById("inner").children[0].style.background = "red";

var a = document.getElementById("inner").children[0].innerHTML; \*/

//var a = document.getElementById("inner").childNodes;

//var a = document.getElementById("inner").childNodes[0].innerHTML;

document.getElementById("inner").childNodes[3].style.background = "red";

var a = document.getElementById("inner").childNodes[3];

console.log(a);

## JS firstChild & lastChild Method

### firstChild:

The firstChild property returns the first child node of a node.

The firstChild property returns a node object.

The firstChild property is read-only.

The firstChild property is the same as childNodes[0].

### lastChild:

The lastChild property returns the last child node of a node.

The lastChild property returns returns a node object.

The lastChild property is read-only.

### html file

<!DOCTYPE html>

<html id="main">

<head>

<title>DOM Navigation</title>

</head>

<style>

#outer{

width: 550px;

height: 300px;

padding:10px 10px;

margin: 0 auto;

background: lightsalmon;

}

#inner{

width: 500px;

height: 200px;

padding:10px 10px;

margin:0 auto ;

background: mediumorchid;

}

#inner div{

display: inline-block;

background: #fff;

width: 95px;

height: 50px;

line-height: 50px;

text-align: center;

}

</style>

<body>

<div id="outer">

<h2>Outer</h2>

<div id="inner">

<h2>Inner</h2>

<div>A</div>

<div>B</div>

<div id="child-c">C</div>

<div>D</div>

<div>E</div>

</div>

</div>

<script src="js/dom-nav.js"></script>

</body>

</html>

### dom-nav.js

//var a = document.getElementById("inner").firstElementChild;

//var a = document.getElementById("inner").firstElementChild.innerHTML;

/\* document.getElementById("inner").firstElementChild.style.background = "red";

var a = document.getElementById("inner").firstElementChild.innerHTML; \*/

/\* document.getElementById("outer").lastElementChild.style.background = "red";

var a = document.getElementById("outer").lastElementChild; \*/

/\* document.getElementById("inner").lastElementChild.style.background = "red";

var a = document.getElementById("inner").lastElementChild; \*/

//var a = document.getElementById("inner").firstChild;

//var a = document.getElementById("inner").lastChild;

//var a = document.getElementById("child-c").firstChild;

var a = document.getElementById("child-c").lastChild;

console.log(a);

## JS nextSibling & prevSibling Method

nextSibling:  
The nextSibling property returns the next node on the same tree level.

The nextSibling returnes a node object.

The nextSibling property is read-only.

previousSibling:  
The previousSibling property returns the previous node on the same tree level.

The previousSibling property returns a node object.

The previousSibling property is read-only.

html file

<!DOCTYPE html>

<html id="main">

<head>

<title>Basic Layout</title>

</head>

<style>

#outer {

width: 550px;

height: 300px;

padding: 10px 10px;

margin: 0 auto;

background: lightsalmon;

}

#inner {

width: 500px;

height: 200px;

padding: 10px 10px;

margin: 0 auto;

background: mediumorchid;

}

#inner div {

display: inline-block;

background: #fff;

width: 95px;

height: 50px;

line-height: 50px;

text-align: center;

}

</style>

<body>

<div id="outer">

<h2>Outer</h2>

<div id="inner">

<h2 id="child-head">Inner</h2>

<div>A</div>

<div>B</div>

<div id="child-C">C</div>

<div>D</div>

<div id="child-E">E</div>

</div>

</div>

<script src="js/dom-nav.js"></script>

</body>

</html>

dom-nav.js

//var a = document.getElementById("child-C").nextElementSibling;

//var a = document.getElementById("child-C").previousElementSibling;

//var a = document.getElementById("child-C").previousElementSibling.innerHTML ;

/\* document.getElementById("child-C").previousElementSibling.style.background = "red";

var a = document.getElementById("child-C").previousElementSibling.innerHTML; \*/

/\* document.getElementById("child-C").nextElementSibling.style.background = "red";

var a = document.getElementById("child-C").nextElementSibling; \*/

//var a = document.getElementById("child-E").nextElementSibling;

//var a = document.getElementById("child-head").previousElementSibling;

//var a = document.getElementById("child-C").previousSibling;

var a = document.getElementById("child-C").nextSibling;

console.log(a);

## JS create & TextNode

createElement()

The createElement() method creates an element node.

Syntax

document.createElement()

Parameters

|  |  |
| --- | --- |
| Parameter | Description |
| *type* | Required. The type of element to create. |

Return Value

|  |  |
| --- | --- |
| Type | Description |
| Node | The created element node. |

createTextNode()

The createTextNode() method creates a text node.

Syntax

document.createTextNode(text)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *text* | Required. The text for the node. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Node | The created text node. |

createComment()

The createComment() method creates a comment and returns the comment node.

Syntax

document.createComment(text)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *text* | Optional. The comment text. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Node | The created comment node. |

html file

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

<style>

h1{

text-align: center;

color:#ff0000;

}

</style>

</head>

<body>

<h1>Lbsti Baba : DOM Create Methods</h1>

<script src="js/dom-create.js"></script>

</body>

</html>

dom-create.js

//var newElement = document.createElement("p");

var newElement = document.createElement("h2");

console.log(newElement);

var newText = document.createTextNode("This is just text");

console.log(newText);

/\* Dom Create Comment\*/

var newComment = document.createComment("this is comment");

console.log(newComment);

## JS appendChild & insertBefore

JS AppendChild

The appendChild() method appends a node (element) as the last child of an element.

Syntax

element.appendChild(node)

or

node.appendChild(node)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *node* | Required. The node to append. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Node | The appended node. |

JS InsertBefore

The insertBefore() method inserts a child node before an existing child.

Syntax

element.insertBefore(new, existing)

or

node.insertBefore(new, existing)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *new* | Required. The node (element) to insert. |
| *existing* | Required. The node (element) to insert before. If null, it will be inserted at the end. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Node | The inserted node. |

html file

<!DOCTYPE html>

<html>

<head>

<title>DOM append and insertBefore</title>

<style>

h1{

text-align: center;

color:#ff0000;

}

#test{

background: #ffff00;

width: 800px;

height: 200px;

padding:10px 10px;

margin: 0 auto;

}

</style>

</head>

<body>

<h1>Lbsti Baba : DOM Create Methods</h1>

<div id="test">

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Consectetur aperiam eos vel consequatur. Delectus voluptas dolorem id exercitationem, ad ipsam consectetur hic ullam provident! Adipisci exercitationem ipsam rerum sunt doloremque magni soluta, delectus maiores, sapiente quasi labore praesentium accusamus earum cum nam saepe qui? Accusamus provident quo perferendis sint sed.</p>

<h3>lbsti baba</h3>

</div>

<script src="js/dom-create.js"></script>

</body>

</html>

dom-create.js

/\* Dom Create \*/

//var newElement = document.createElement("p");

var newElement = document.createElement("h2");

console.log(newElement);

var newText = document.createTextNode("This is just text");

console.log(newText);

/\* JavaScript AppendChild\*/

newElement.appendChild(newText);

//document.getElementById("test").appendChild(newElement);

/\* JavaScript InsertBefore \*/

var target = document.getElementById("test");

target.insertBefore(newElement,target.childNodes[0])

## JS insert

### insertAdjacentElement()

The insertAdjacentElement() method inserts a an element into a specified position.

### Legal positions:

|  |  |
| --- | --- |
| **Value** | **Description** |
| afterbegin | After the beginning of the element (first child) |
| afterend | After the element |
| beforebegin | Before the element |
| beforeend | Before the end of the element (last child) |

### Syntax

element.insertAdjacentElement(position, element)

or

node.insertAdjacentElement(position, element)

### Parameters

|  |  |
| --- | --- |
| Parameter | Description |
| position | Required. A position relative to the element: afterbegin afterend beforebegin beforeend |
| element | The element to insert. |

### insertAdjacentHTML()

The insertAdjacentHTML() method inserts HTML code into a specified position.

### Legal positions:

|  |  |
| --- | --- |
| **Value** | **Description** |
| afterbegin | After the beginning of the element (first child) |
| afterend | After the element |
| beforebegin | Before the element |
| beforeend | Before the end of the element (last child) |

### Syntax

element.insertAdjacentHTML(position, html)

or

node.insertAdjacentHTML(position, html)

### Parameters

|  |
| --- |
|  |
| Parameter | Description |
| position | Required. A position relative to the element: afterbegin afterend beforebegin beforeend |
| html | The HTML to insert. |

### html file

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

<style>

#test{

background: #ffff00;

width: 800px;

height: 200px;

padding: 10px 10px;

margin: 0 auto;

}

</style>

</head>

<body>

<div id="test">

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Consectetur aperiam eos vel consequatur. Delectus voluptas

dolorem id exercitationem, ad ipsam consectetur hic ullam provident! Adipisci exercitationem ipsam rerum sunt

doloremque magni soluta, delectus maiores, sapiente quasi labore praesentium accusamus earum cum nam saepe qui?

Accusamus provident quo perferendis sint sed.</p>

</div>

<script src="js/dom-create.js"></script>

</body>

</html>

### dom-create.js

// insertAdjacentElement Method

var newElement = document.createElement("h2");

var newText = document.createTextNode("This is just element");

newElement.appendChild(newText);

var target = document.getElementById("test");

target.insertAdjacentElement("afterbegin",newElement);

// insertAdjacentHTML Method

var newElement = "<h2>This is just Html</h2>";

var target = document.getElementById("test");

target.insertAdjacentHTML("afterend",newElement);

// insertAdjacentText Method

var newText = "<h2>This is just Text</h2>";

var target = document.getElementById("test");

target.insertAdjacentHTML("beforeend",newText);

## JS replaceChild & removeChild

### replaceChild()

The replaceChild() method replaces a child node with a new node.

### Syntax

node.replaceChild(newnode, oldnode)

## **Parameters**

|  |
| --- |
|  |
| Parameter | Description |
| newnode | Required. The node to insert. |
| oldnode | Required. The node to remove. |

## **Return Value**

|  |
| --- |
|  |
| Type | Description |
| Node | The replaced node. |

### removeChild()

The removeChild() method removes an element's child.

### Syntax

element.removeChild(node)

or

node.removeChild(node)

### Parameters

|  |
| --- |
|  |
| Parameter | Description |
| node | Required. The node (element) to remove. |

### Return Value

|  |
| --- |
|  |
| Type | Description |
| Node | The removed node (element). null if the child does not exist. |

### html file

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

</head>

<body>

<ul id= "list">

<li>orange</li>

<li>Apple</li>

<li>Grapes</li>

<li>Banana</li>

</ul>

<script src="js/dom-create.js"></script>

</body>

</html>

### dom-create.js

/\*JavaScript ReplaceChild\*/

/\*var newElement = document.createElement("li");

var newText = document.createTextNode("WOW new Text");

newElement.appendChild(newText);

var target = document.getElementById("list");

var oldElement = target.children[0];\*/

//console.log(oldElement);

//target.replaceChild(newElement,oldElement);

/\*JavaScript RemoveChild\*/

var target = document.getElementById("list");

var oldElement = target.children[1];

## JS cloneNode

JS cloneNode

The cloneNode() method creates a copy of a node, and returns the clone.

The cloneNode() method clones all attributes and their values.

Syntax

node.cloneNode(deep)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *deep* | Optional. false - Default. Clone only the node and its attributes. true - Clone the node, its attributes, and its descendants. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Node | The cloned node. |

html file

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

<style>

#test{

background: #ffff00;

width: 800px;

height: 200px;

padding: 10px 10px;

margin: 0 auto;

}

</style>

</head>

<body>

<ul id= "list1">

<li class="abc">orange</li>

<li>Apple</li>

<li>Grapes</li>

<li>Banana</li>

</ul>

<ul id= "list2">

<li>Carrot</li>

<li>Reddish</li>

</ul>

<div id="test"></div>

<script src="js/dom-create.js"></script>

</body>

</html>

dom-create.js

/\* JavaScript CloneNode\*/

var target = document.getElementById("list1").children[0];

var copyElement = target.cloneNode(true);

console.log(copyElement);

document.getElementById("list2").appendChild(copyElement);

document.getElementById("test").appendChild(copyElement);

## JS Contains

JS Contains

The contains() method returns true if a node is a descendant of a node.

The contains() method returns false if not.

Syntax

node.contains(node)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *node* | Required. The node that may be a descendant of the node. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Boolean | true - The node is a descendant false - The node is NOT a descendant |

html file

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

<style>

h2{ margin: 0 0 10px; }

#test{

background: #ffff00;

width: 800px;

min-height: 250px;

padding: 10px 10px;

margin: 0 auto;

}

</style>

</head>

<body>

<div id="test">

<h2>Lbsti Baba : JavaScript Contains Method</h2>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo corporis adipisci alias cupiditate officiis consequatur, fuga, amet quos numquam perferendis saepe labore dolorem reiciendis reprehenderit facilis repudiandae praesentium quis delectus voluptates, repellendus recusandae. Itaque consequuntur, corrupti quasi illum iusto perferendis autem blanditiis magni eum, repellendus architecto ratione ipsum molestiae laboriosam.

</p>

<div>

<p id="abc">

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Voluptatum magni vel inventore illum facere tenetur eveniet quam ex nemo eum. Dolorum suscipit corrupti asperiores dicta inventore debitis facilis amet totam neque reprehenderit minima ratione corporis quasi aperiam iusto consectetur nostrum enim optio, quas rerum doloribus nesciunt dolor deleniti. Ea, minima.

</p>

</div>

</div>

<script src="js/dom-create.js"></script>

</body>

</html>

dom-create.js

/\* JavaScript Contains Method\*/

var parentElement = document.getElementById("test");

var target = document.getElementById("abc");

var find = parentElement.contains(target);

console.log(find);

## JS has

### JS hasAttribute(

The hasAttribute() method returns true if the attribute exists, otherwise false.

### Syntax

element.hasAttribute(name)

### Parameters

|  |
| --- |
|  |
| Parameter | Description |
| name | Required. The name of the attribute. |

### Return Value

|  |
| --- |
|  |
| Type | Description |
| Boolean | true if the element has the attribute, otherwise false. |

### hasChildNodes(

The hasChildNodes() method returns true if the specified node has any child nodes, otherwise false.

The hasChildNodes() method is read-only.

### Syntax

element.hasChildNodes()

### Parameters

|  |
| --- |
| NONE |

### Return Value

|  |
| --- |
|  |
| Type | Description |
| Boolean | true if the element has child nodes, otherwise false. |

## **html file**

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

<style>

h2{ margin: 0 0 10px; }

#test{

background: #ffff00;

width: 800px;

min-height: 250px;

padding: 10px 10px;

margin: 0 auto;

}

</style>

</head>

<body>

<div id="test" class="abc">

<h2>Lbsti Baba : JavaScript hasAttribute Method</h2>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo corporis adipisci alias cupiditate officiis consequatur, fuga, amet quos numquam perferendis saepe labore dolorem reiciendis reprehenderit facilis repudiandae praesentium quis delectus voluptates, repellendus recusandae. Itaque consequuntur, corrupti quasi illum iusto perferendis autem blanditiis magni eum, repellendus architecto ratione ipsum molestiae laboriosam.

</p>

<h2>Lbsti Baba : JavaScript hasChildNodes Method</h2>

</div>

<script src="js/dom-create.js"></script>

</body>

</html>

## **dom-create.js**

/\* JavaScript hasAttribute\*/

var target = document.getElementById("test");

var find = target.hasAttribute("class");

console.log(find);

/\* JavaScript hasChildNodes\*/

var target = document.getElementById("test");

var find = target.hasChildNodes();

console.log(find);

[Previous](https://www.yahubaba.com/javascript/js-contains)[Next](https://www.yahubaba.com/javascript/js-isequalnode)

## JS isEqualNode

### JS isEqualNode

The isEqualNode() returns true if two elements (or nodes) are equal.

### Syntax

element.isEqualNode(node)

or

node.isEqualNode(node)

### Parameters

|  |
| --- |
|  |
| Parameter | Description |
| node | Required. The node to compare. |

## **Return Value**

|  |
| --- |
|  |
| Type | Description |
| Boolean | true if the nodes are equal, otherwise false. |

### html file

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

</head>

<body>

<ul id="list-1">

<li class="abc">orange</li>

<li>Banana</li>

<li>Apple</li>

<li>Grapes</li>

</ul>

<ul id="list-2">

<li>Guava</li>

<li class="abc">orange</li>

<li>PineApple</li>

</ul>

<script src="js/dom-create.js"></script>

</body>

</html>

dom-create.js

var target1 = document.getElementById("list-1").firstElementChild;

var target2 = document.getElementById("list-2").children[1];

var equal = target1.isEqualNode(target2);

console.log(equal);

## JS Form Events

### html file

<!DOCTYPE html>

<html lang="en">

<head>

<title>Form Events</title>

</head>

<body>

<form action="">

<label for="">Name</label><input type="text" onfocus="focusFunction(this)" onblur="blurFunction(this)" oninput="inputFunction(this)"><br><br>

<label for="">Class</label><input type="text" id="" onfocus="focusFunction(this)" onblur="blurFunction(this)"><br><br>

<select id="" onfocus="focusFunction(this)" onblur="blurFunction(this)">

<option value="">India</option>

<option value="">Pakistan</option>

<option value="">Bangladesh</option>

<option value="">Sri Lanka</option>

<option value="">Nepal</option>

</select>

<br>

</form>

<div id="test" style="border:1px solid red"></div>

<script>

/\* JavaScript Focus Event \*/

function focusFunction(element){

element.style.background = "lime";

}

/\* JavaScript Blur Event \*/

function blurFunction(element){

element.style.background = "";

}

/\* JavaScript Input Event \*/

function inputFunction(element){

var x = element.value;

document.getElementById("test").innerHTML = x;

}

</script>

</body>

</html>

## JS Form Events - II

### html file

<!DOCTYPE html>

<html lang="en">

<head>

<title>Form Events</title>

</head>

<body>

<form action="" onsubmit="submitFunction()">

<label for="">Name</label><input type="text" id="fname" onchange="onchangeFunction(this)" onselect="selectFunction()" oninvalid="alert("Please fill the First Name.")" required><br><br>

<label for="">Class</label><input type="text" id="" ><br><br>

<select id="" onchange="onchangeFunction(this)">

<option value="India">India</option>

<option value="Pakistan">Pakistan</option>

<option value="Bangladesh">Bangladesh</option>

<option value="Sri Lanka">Sri Lanka</option>

<option value="Nepal">Nepal</option>

</select>

<input type="submit">

</form>

<div id="test" style="border:1px solid red;margin-top:20px;"></div>

<script>

function focusFunction(element){

element.style.background = "lime";

}

/\* JavaScript Blur Event \*/

function blurFunction(element){

element.style.background = "";

}

/\* JavaScript Input Event \*/

function inputFunction(element){

var x = element.value;

document.getElementById("test").innerHTML = x;

}

/\* JavaScript Change Event \*/

function onchangeFunction(element){

var x = element.value;

document.getElementById("test").innerHTML = x;

}

/\* JavaScript Select Event \*/

function selectFunction(){

console.log("You selected some text.");

}

/\* JavaScript Submit Event \*/

/\*function submitFunction(){

alert("You submitted a form.");

}\*/

function submitFunction(){

var x = document.getElementById("fname").value;

alert("Hello " + x);

}

</script>

</body>

</html>

## JS Interval

setInterval()

The setInterval() method calls a function at specified intervals (in milliseconds).

The setInterval() method continues calling the function until clearInterval() is called, or the window is closed.

1 second = 1000 milliseconds.

Syntax

setInterval(function, milliseconds, param1, param2, ...)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *function* | Required. The function to execute |
| *milliseconds* | Required. The execution interval. If the value is less than 10, 10 is used |
| *param1, param2, ...* | Optional. Additional parameters to pass to the *function* Not supported in IE9 and earlier. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| A number | The id of the timer. Use this id with clearInterval() to cancel the timer. |

clearInterval()

The clearInterval() method clears a timer set with the setInterval() method.

Syntax

clearInterval(intervalId)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *intervalId* | Required. The interval id returned from setInterval(). |

Return Value

|  |
| --- |
| NONE |

html file

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript Animation</title>

<style>

#test{

width:150px;

height:150px;

background: red;

}

</style>

</head>

<body>

<div id="test"></div>

<script src="js/animation.js"></script>

</body>

</html>

 animation.js

var a = 0;

var id = setInterval(Anim,500);

function Anim(){

a = a + 10;

if(a == 200){

clearInterval(id);

}else{

var target = document.getElementById("test");

//target.style.marginLeft = a + 'px';

target.style.width = a + 'px';

}

}

## JS Timeout

setTimeout()

The setTimeout() method calls a function after a number of milliseconds. 1 second = 1000 milliseconds.

Syntax

setTimeout(function, milliseconds, param1, param2, ...)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *function* | Required. The function to execute. |
| *milliseconds* | Optional. Number of milliseconds to wait before executing. Default value is 0. |
| *param1, param2, ...* | Optional. Parameters to pass to the *function.* Not supported in IE9 and earlier. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| A number | The id of the timer. Use this id with clearTimeout(id) to cancel the timer. |

clearTimeout()

The clearTimeout() method clears a timer set with the setTimeout() method.

Syntax

clearTimeout(id\_of\_settimeout)

Parameters

|  |  |
| --- | --- |
| Parameter | Description |
| *timeout id* | Required. The id returned by the setTimeout() method. |

Return Value

|  |
| --- |
| NONE |

html file

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript Animation</title>

<style>

#test{

width:150px;

height:150px;

background: red;

}

</style>

</head>

<body>

<div id="test"></div>

<button onclick="stopAnimation()">Stop Animation</button>

<script src="js/animation.js"></script>

</body>

</html>

animation.js

/\* JavaScript setTimeout\*/

/\*var id = setTimeout(Anim,5000);

function Anim(){

var target = document.getElementById("test");

target.style.width = "500px";

//console.log("Hello");

}\*/

var id = setTimeout(function(){

var target = document.getElementById("test");

target.style.width = "500px";

},5000);

/\* JavaScript clearTimeout\*/

function stopAnimation(){

clearTimeout(id);

}

## JS BOM Introduction

Browser Object Model

The Browser Object Model (BOM) is used to interact with the browser. The default object of browser is window means you can call all the functions of window by specifying window or directly. For example: window.alert("hello javatpoint"); is same as: alert("hello javatpoint"); You can use a lot of properties (other objects) defined underneath the window object like document, history, screen, navigator, location, innerHeight, innerWidth.

The Window Object

The window object is supported by all browsers. It represents the browser's window.

All global JavaScript objects, functions, and variables automatically become members of the window object.

Global variables are properties of the window object.

Global functions are methods of the window object.

Even the document object (of the HTML DOM) is a property of the window object:

window.document.getElementById("header");

is the same as:

document.getElementById("header");

## JS Window Height & Width

### JS Window Height & Width

Two properties can be used to determine the size of the browser window.

Both properties return the sizes in pixels:

* window.innerHeight - the inner height of the browser window (in pixels)
* window.innerWidth - the inner width of the browser window (in pixels)
* window.outerWidth -   returns the outer width of a window, including all interface elements (like toolbars/scrollbars).
* window.outerHeight- returns the outer height of a window, including all interface elements (like toolbars/scrollbars).

These properties are read-only.

var w = window.outerWidth;

var h = window.outerHeight;

var w = window.innerWidth;

var h = window.innerHeight;

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript</title>

</head>

<body onresize="resizeFunction()">

<script>

function resizeFunction(){

console.clear();

var iHeight = window.innerHeight;

console.log('Inner Height :' + iHeight);

var oHeight = window.outerHeight;

console.log('Outer Height :' + oHeight);

}

function resizeFunction(){

console.clear();

var iWidth = window.innerWidth;

console.log('Inner Width :' + iWidth);

var oWidth = window.outerWidth;

console.log('Outer Width :' + oWidth);

}

</script>

</body>

</html>

## JS Window Open & Close

### Javascript Window Open() & Window Close() Method

The Javascript **Window.open()**method is used to open the web pages into a new window or a new tab. It depends on the browser settings and the values that are assigned to the parameter.

**Syntax:**

window.open(URL, name, specs, replace)

**Parameters:** This method accepts four parameters as mentioned above and described below:

* **URL:** It is an optional parameter. It is used to set the URL of web pages that need to open. If URL is not set then window.open() method open a blank window.
* **name:** It is an optional parameter and is used to set the window name.
* **specs:** It is an optional parameter used to separate the item using a comma.
* **replace:** It is an optional parameter and used to specify the URL URL creates a new entry or replaces the current entry in the history list. This parameter returns a boolean value. If this parameter returns true then URL replaces the current document in the history list and if returns false then URL creates a new entry in the history list.

**Return Value:** This method creates a new window.

**Window.close():** This method is used to close the window which is opened by the window.open() method.

**Syntax:**

window.close()

**Parameters:** This method does not contain any parameters.

**Return Value:** This method does not return any value.

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript</title>

</head>

<body>

<button onclick="openWindow()">Open Window</button>

<button onclick="closeWindow()">Close Window</button>

<script>

/\* JavaScript Open Close Window Method \*/

var myWindow;

function openWindow(){

//window.open("http://www.lbstibaba.net","lbstibaba");

//window.open("http://www.lbstibaba.net","\_blank");

//window.open("http://www.lbstibaba.net","\_parent");

myWindow = window.open("http://www.lbstibaba.net","","width=500px,height=200px,left=100px,top=200px");

}

function closeWindow(){

myWindow = window.close();

}

</script>

</body>

</html>

## JS Window move

JS Window moveBy()

The moveBy() method moves a window a number of pixels relative to its current coordinates.

Syntax

window.moveBy(x, y)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *x* | Required. A positive or negative number. The number of pixels to move the window horizontally. |
| *y* | Required. A positive or negative number. The number of pixels to move the window vertically. |

Return Value

|  |
| --- |
| NONE |

JS Window moveTo()

The moveTo() method moves a window to the specified coordinates.

Syntax

window.moveTo(x, y)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *x* | Required. A positive or negative number. The horizontal coordinate to move to. |
| *y* | Required. A positive or negative number. The vertical coordinate to move to. |

Return Value

|  |
| --- |
| NONE |

<!DOCTYPE html>

<html lang="en">

<head>

<title>Document</title>

</head>

<body>

<button onclick="openWindow()">Open Window</button>

<button onclick="moveWindow()">Move Window</button>

<script>

/\* JavaScript MoveBy And MoveTo Window Method \*/

var myWindow;

function openWindow(){

myWindow = window.open("","","width=500px,height=200px,left=100px,top=100px");

myWindow.document.write("<p>This is my Window.</p>")

}

/\*function moveWindow(){

myWindow.moveTo(200,200);

myWindow.focus();

}\*/

function moveWindow(){

myWindow.moveBy(200,200);

myWindow.focus();

}

</script>

</body>

</html>

## JS resize

JS resizeBy()

The resizeBy() method resizes a window by a specified amount.

Syntax

resizeBy(width, height)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *width* | Required. A positive or a negative number. The number of pixels to resize the width by. |
| *height* | Required. A positive or a negative number. The number of pixels to resize the height by. |

Return Value

|  |
| --- |
| NONE |

Window resizeTo()

The resizeTo() method resizes a window to a new width and height.

Syntax

window.resizeTo(width, height)

Parameters

|  |  |
| --- | --- |
| Parameter | Description |
| *width* | Required. The new window width, in pixels |
| *height* | Required. The new window height, in pixels |

Return Value

|  |
| --- |
| NONE |

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript</title>

</head>

<body>

<button onclick="openWindow()">Open Window</button>

<button onclick="resizeWindow()">Resize Window</button>

<script>

/\* JavaScript ResizeBy And ResizeTo Window Method \*/

var myWindow;

function openWindow(){

myWindow = window.open("","","width=500px,height=200px,left=30px,top=30px");

myWindow.document.write("<p>This is my Window.</p>")

}

function resizeWindow(){

myWindow.resizeTo(400,400);

}

/\*function resizeWindow(){

myWindow.resizeBy(400,400);

}\*/

</script>

</body>

</html>

## JS scroll

JS scrollBy()

The scrollBy() method scrolls the document by the specified number of pixels.

Syntax

window.scrollBy(x, y)

or just:

scrollBy(x, y)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *x* | Required. Number of pixels to scroll (horizontally). Positive values scroll to the right, negative values to the left. |
| *y* | Required. Number ofpixels to scroll (vertically). Positive values scroll down, negative values scroll up. |

Return Value

|  |
| --- |
| NONE |

JS scrollTo()

The scrollTo() method scrolls the document to specified coordinates.

Syntax

window.scrollTo(x, y)

or just:

scrollTo(x, y)

Parameters

|  |  |
| --- | --- |
| Parameter | Description |
| *x* | Required. The coordinate to scroll to (horizontally), in pixels. |
| *y* | Required. The coordinate to scroll to (vertically), in pixels. |

Return Value

|  |
| --- |
| NONE |

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript</title>

</head>

<body>

<h1>JavaScript : scrollTo & scrollBy</h1>

<div>

<p style="width:1000px;border:1px solid red;">Lorem, ipsum dolor sit amet consectetur adipisicing, elit. Eius, obcaecati eum distinctio? Itaque fuga a adipisci obcaecati perferendis consectetur mollitia nam reiciendis numquam, voluptatem. Sint ipsam vitae incidunt placeat. Veritatis tempore earum sequi exercitationem iure rem placeat, dolores voluptatum vero explicabo eum, fuga distinctio. Amet temporibus aliquid non quam corrupti! Dignissimos maiores dolorum dicta earum accusamus consectetur hic nesciunt officia aspernatur ex necessitatibus non iste est voluptates ratione unde perspiciatis placeat magni quia voluptatibus, magnam harum ipsam natus debitis, atque! Error eos provident quis repudiandae eum eaque doloremque sint animi ea. Eum aperiam ratione consectetur, dolor excepturi numquam et sed expedita doloribus, delectus. Repudiandae adipisci, repellendus, odit nesciunt voluptates nam hic repellat eaque eveniet obcaecati. In, aut blanditiis saepe reprehenderit quibusdam, repellendus necessitatibus pariatur odit fuga odio aliquam? Reiciendis quae, temporibus eaque libero unde aut vitae assumenda, officiis sint molestiae magnam fugiat earum iure expedita minus repellat nemo quo blanditiis facere error ullam ut similique cum eveniet. Labore omnis debitis minima libero deleniti assumenda odio beatae quas porro, incidunt, cupiditate, facere commodi. Eveniet quas repellat enim aliquid, nam, dicta rem ipsam delectus quae ut nisi, aut. Quod ratione sunt corporis harum enim, ut dolores molestias voluptatem inventore soluta. Repudiandae odit distinctio rerum fuga dignissimos, mollitia beatae commodi aliquid ex ipsam magni voluptatibus ratione sunt eligendi, saepe velit nulla maxime doloremque atque cum consequuntur deserunt alias? Numquam quod tempora est, ducimus ab voluptatibus, aspernatur asperiores fuga quisquam rem sapiente nulla officia accusantium laborum placeat perspiciatis corporis sequi, quis nam facere. Quis tempore unde accusantium placeat accusamus laboriosam hic commodi id optio numquam asperiores, necessitatibus vitae sit ad amet dicta, expedita aspernatur veniam eius reprehenderit perferendis vero, quae! Sint, quidem incidunt dolores distinctio voluptatibus animi, perspiciatis, consequuntur cum quae quas, magni vero. Esse quidem labore adipisci ducimus sint excepturi quam molestias, autem doloribus magnam nobis corrupti dignissimos assumenda at ratione, soluta minus debitis magni deleniti ipsa. Autem natus vel, quas aperiam corrupti, delectus. Fuga itaque reprehenderit non aliquid consequuntur ipsa reiciendis, quam repellat, molestiae? Id, sint. Asperiores, cupiditate consectetur, veniam tenetur voluptate itaque doloribus voluptates porro sequi quas ex optio similique distinctio eum molestiae rerum quis facere explicabo eos, qui, sint quasi culpa molestias. Veniam commodi ipsum eum velit consequatur fuga quasi aliquam tenetur numquam hic quisquam et totam atque est debitis officiis eos accusantium odit minima deleniti, sunt molestiae consectetur itaque! Ipsum totam corrupti, voluptas deleniti facilis provident doloribus dolore autem atque tempore alias, harum ex omnis quisquam nobis veritatis earum eligendi, similique corporis dignissimos. Reiciendis earum vitae eligendi autem. Necessitatibus laudantium vero dicta dolorem debitis ratione. Deleniti quam quisquam totam, enim placeat fuga, possimus ratione architecto magnam ducimus, quis excepturi nesciunt molestiae magni sapiente non, cumque est repellendus nam rerum? Vero sequi nisi odit voluptate consequatur cupiditate vel praesentium adipisci harum, nesciunt placeat voluptates! Velit cupiditate numquam sit nulla accusamus nesciunt nemo voluptatem est omnis consequuntur nobis quas aut sed a, quos natus ducimus consectetur deleniti. Neque obcaecati adipisci ratione velit placeat consequatur perferendis delectus quidem cum alias similique minus magni quo quos libero dolores est doloremque dignissimos iste, nobis fuga veniam, quia. Excepturi, animi, recusandae sapiente delectus sint similique enim inventore culpa ducimus et ea optio quas corrupti natus voluptatibus tempore voluptates neque, placeat, ad aliquam! Laudantium aspernatur minus placeat. Facere maxime modi eligendi doloremque est deleniti placeat optio consequatur incidunt, reiciendis voluptatem distinctio cum suscipit, maiores officiis quod, voluptas accusantium vitae non provident aut consectetur in. Reiciendis, dicta. Ea vero earum illo, debitis saepe, iusto expedita officiis corporis fuga. Vitae velit nemo, eligendi delectus provident minima temporibus reiciendis quae vero sed, et, consequatur doloremque odio voluptates exercitationem commodi maiores maxime sequi ipsum at! Maiores, totam id fugit harum quos facere, consequuntur earum a ex doloremque sed, delectus illum ipsa at, consequatur inventore nostrum est. Mollitia blanditiis incidunt autem eligendi accusantium recusandae a magni veritatis repellendus dolores, doloremque hic. Minima nihil enim esse molestias eos cupiditate praesentium inventore reprehenderit sunt, aut deleniti natus laboriosam recusandae iusto exercitationem magni eligendi tempore ab doloribus, deserunt at, nostrum amet vitae. Quas totam a dolorum nesciunt vero delectus libero velit ex voluptatum praesentium repellat possimus officia tenetur facilis, maxime doloribus, nostrum unde? Corporis quas cumque quis ab, nesciunt sed, neque officia nam quidem? Laboriosam doloremque aliquid, corrupti eaque tenetur, repellendus cupiditate rerum distinctio corporis a sit officia? Aut explicabo nostrum laudantium officiis, incidunt iusto! Beatae excepturi dolores cumque est quidem maxime itaque quas, delectus tempore, expedita, dicta recusandae, sequi. Earum molestias dolore iste soluta, vel! Itaque nisi iure, explicabo voluptatem dolore vitae fugit animi vero earum. Neque, excepturi, autem rerum molestiae, laborum adipisci perferendis tempora nisi, deleniti dolorum atque natus labore odit. Tenetur exercitationem suscipit nemo reiciendis, rem facere, laborum quasi laudantium explicabo blanditiis quia doloribus eos voluptate? Vitae voluptatum accusamus voluptatem vel praesentium dolorem, cum, odio ratione laboriosam, asperiores necessitatibus quas ut minus quos maiores omnis corrupti quis, nam ad molestiae assumenda. Nobis nemo exercitationem, veniam quia fugiat quo voluptatem suscipit ad tempora inventore, incidunt porro atque, voluptatum at provident molestias perferendis alias impedit. Repellendus dolor earum iusto officia voluptatem quod facere incidunt minima, explicabo, amet hic, neque veritatis laborum cum distinctio atque eius pariatur et aut, doloremque quos quidem odio laboriosam tempora vel. Aperiam sequi corrupti ab aliquam assumenda officia, earum accusamus ratione corporis cumque, sint veniam error alias eum voluptatem tempore obcaecati sapiente veritatis dolore autem nemo facere molestiae. Minus animi pariatur ab sint eius dicta ad quas, minima sapiente incidunt sequi quae quisquam impedit et praesentium id alias? Distinctio sequi, aperiam animi eveniet, voluptate aut in est maiores nobis, id optio ullam possimus consectetur rerum cumque et, tempore itaque. Corporis nesciunt delectus ullam provident fugit quo. Consequuntur cumque recusandae, reprehenderit voluptate officia inventore tempora numquam ducimus velit, laboriosam eius tempore a dolorem neque, fugiat veritatis delectus sapiente libero. Nesciunt saepe accusantium, natus maxime fuga dolorum, perspiciatis, minus et tempore ipsum cum provident, doloribus ut amet? Iure distinctio, minus velit, doloremque repellendus accusamus iste aut eius dolorem sapiente expedita asperiores. Id voluptatibus quos minima sit ab? Molestias cupiditate modi dignissimos voluptate odit, assumenda minus amet sed quas dicta provident at architecto, ipsa distinctio aliquid eveniet! Laudantium vero voluptatibus, impedit, reiciendis non, amet porro atque architecto deleniti doloribus vel et ut laboriosam facere vitae tempore perferendis eos rerum eum. Ex veniam accusamus voluptates ipsa et odio quam, modi, labore explicabo minus totam ipsum! Aperiam, libero, mollitia ipsam a ullam at necessitatibus minus laborum! Doloremque autem beatae vero quasi porro a natus repellendus laborum nostrum hic cupiditate iste quos saepe fuga quas quo inventore nam, voluptas accusamus corrupti aliquam. Quod, perspiciatis. Magni voluptas a in eos officia facilis ipsum minus ea enim rem doloribus officiis consequatur reiciendis consectetur, velit cum nemo eligendi dolorem ratione temporibus corrupti inventore id exercitationem cumque? Dignissimos earum minima repudiandae eveniet aliquid, tempore rerum eaque laborum officia, repellat asperiores natus atque vitae maxime quaerat neque. Facere quo consequatur quaerat, tempore perferendis consectetur laudantium accusantium et iusto, cumque amet eveniet perspiciatis error dolorum harum ipsum voluptates illo rem necessitatibus maiores sit? Omnis a eveniet cumque ad perspiciatis, fugit, necessitatibus harum illo sit quae voluptatem, amet natus laudantium dolorem quaerat placeat corporis quam minus eligendi unde et assumenda rerum. Quae itaque nulla ipsam consequuntur obcaecati debitis eveniet recusandae. In eaque harum temporibus totam veniam culpa necessitatibus labore soluta rerum exercitationem! Nihil ducimus minima enim voluptates tenetur, quasi doloribus debitis vel laborum aut eos, eius fugiat facere earum, ipsum optio voluptate quis numquam, nostrum doloremque officiis. Pariatur quia amet impedit, possimus necessitatibus id voluptate. Quo aut, adipisci ut dolor quae earum rem aperiam dignissimos asperiores necessitatibus consequatur ea quaerat eaque alias labore animi quis pariatur, enim, repellat iusto voluptatem. Eaque sed atque deleniti laboriosam quaerat nam illo, quis autem sunt officiis alias inventore expedita dolores omnis minus quisquam perspiciatis? Cum laudantium cupiditate distinctio ducimus, officiis exercitationem, impedit odit optio nihil, minima ad officia mollitia, illum corrupti aspernatur quod odio veniam aperiam eum harum pariatur facilis culpa eos. Quod, inventore? Odio consectetur, autem veniam repellat temporibus? Aliquid debitis cum provident distinctio, repudiandae maiores fugiat eum et, quam, odit quaerat! Eveniet quod quisquam repellat, pariatur repellendus consequuntur culpa, doloremque distinctio facere assumenda impedit velit, ipsum possimus. Facere placeat corporis magnam fugiat harum, sunt officiis quasi obcaecati adipisci, explicabo molestias similique itaque in voluptates sint quia minus aspernatur ullam tempore dignissimos natus officia repellendus repudiandae. Inventore consequatur velit maxime laudantium rem ut, ab corrupti voluptas ipsum! Sunt enim vel ex quasi et minima cumque aliquid quisquam ab? Id, suscipit, doloremque optio hic consectetur perspiciatis nemo assumenda, rem vitae tempore recusandae nesciunt saepe, alias unde. Hic sint beatae consectetur architecto nulla vitae incidunt ullam quas ex fugiat officia sunt, repellendus harum voluptate ratione culpa vero omnis alias corrupti eveniet, velit voluptas accusamus eum dolores, doloribus! Harum fugiat quae cum distinctio aut qui eum tempora debitis obcaecati adipisci eos corrupti id voluptatem nihil numquam sed, at temporibus iusto a ex. Reprehenderit consequatur quam molestias ad. Quia dolor sint dicta optio nostrum fugiat, officiis molestias nobis officia laborum dolores cumque temporibus rerum iste? Vel corrupti, eius ducimus.</p>

</div>

<button onclick="scrollWindow()" style="position:fixed;left:50px;bottom:50px;background:red;color:white">Scroll Window</button>

<script>

function scrollWindow(){

window.scrollBy(0,20);

}

/\*function scrollWindow(){

window.scrollBy(100,0);

}

/\*function scrollWindow(){

window.scrollTo(0,0);

}\*/

</script>

</body>

</html>

## JS Location Object

The Window Location Object

The **location object** contains information about the current URL.

The **location object** is a property of the **window object**.

The **location object** is accessed with:

window.location or just location

Location Object Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| hash | Sets or returns the anchor part (#) of a URL |
| host | Sets or returns the hostname and port number of a URL |
| hostname | Sets or returns the hostname of a URL |
| href | Sets or returns the entire URL |
| origin | Returns the protocol, hostname and port number of a URL |
| pathname | Sets or returns the path name of a URL |
| port | Sets or returns the port number of a URL |
| protocol | Sets or returns the protocol of a URL |
| search | Sets or returns the querystring part of a URL |

Location Object Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| assign() | Loads a new document |
| reload() | Reloads the current document |
| replace() | Replaces the current document with a new one |

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript</title>

</head>

<body>

<h1>JavaScript : Location Object</h1>

<button onclick="newFunction()">Click</button>

<script>

/\* JavaScript Location Object\*/

//location.href = "http://www.lbstibaba.net";

//function newFunction(){

// location.href = "http://www.lbstibaba.net";

//}

/\* JavaScript Reload Function\*/

/\*function newFunction(){

location.reload();

}\*/

/\* JavaScript Assign Function\*/

/\*function newFunction(){

location.assign("https://www.google.com");

}\*/

/\* JavaScript Replace Function\*/

function newFunction(){

location.replace("https://www.google.com");

}

</script>

</body>

</html>

## JS History Object

JS History Object

The **history object** contains the URLs visited by the user (in the browser window).

The **history object** is a property of the **window object**.

The **history object** is accessed with:

window.history or just history:

History Object Properties and Methods

|  |  |
| --- | --- |
| **Property/Method** | **Description** |
| back() | Loads the previous URL (page) in the history list |
| forward() | Loads the next URL (page) in the history list |
| go() | Loads a specific URL (page) from the history list |
| length | Returns the number of URLs (pages) in the history list |

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript</title>

</head>

<body>

<h1>JavaScript: History Object</h1>

<!-- <button onclick="backFunction()">Back</button>

<button onclick="forwardFunction()">Forword</button> -->

<button onclick="goFunction()">Go Button</button>

<script>

/\* JavaScript Back Function\*/

function backFunction(){ history.back(); }

/\* JavaScript Forward Function\*/

function forwardFunction(){ history.forward(); }

/\* JavaScript Go Function\*/

function goFunction(){ history.go(1); }

</script>

</body>

</html>

## JS pageYOffset & pageXOffset

Window pageXOffset

The pageXOffset property returns the pixels a document has scrolled from the upper left corner of the window.

The pageXOffset property is equal to the scrollX property.

The pageXOffset property is read-only.

Syntax

window.pageXOffset

or just:

pageXOffset

Return Value

|  |
| --- |
|  |
| Type | Description |
| A number | The number of pixels the document has scrolled from the upper left corner of the window. |

Window pageYOffset

The pageYOffset property returns the pixels a document has scrolled from the upper left corner of the window.  
The pageYOffset property is equal to the scrollY property.  
The pageYOffset property is read-only.

Syntax

window.pageYOffset

or just:

pageYOffset

Return Value

|  |
| --- |
|  |
| Type | Description |
| A number | The number of pixels the document has scrolled from the upper left corner of the window. |

## **JS offsetTop & offsetLeft**

### JS OffsetTop

The offsetTop property returns the top position (in pixels) relative to the parent.

The returned value includes:

* the top position, and margin of the element
* the top padding, scrollbar and border of the parent

The offsetTop property is read-only.

### Syntax

element.offsetTop

### Return Value

|  |
| --- |
|  |
| Type | Description |
| Number | The top position of the element, in pixels. |

### JS OffsetLeft

The offsetLeft property returns the left position (in pixels) relative to the parent.

The returned value includes:

* the left position, and margin of the element
* the left padding, scrollbar and border of the parent

The offsetLeft property is read-only.

### Syntax

element.offsetLeft

### Return Value

|  |
| --- |
|  |
| Type | Description |
| Number | The left position of the element, in pixels. |

## **JS scrollTop & scrollLeft**

JS ScrollTop

The scrollTop property sets or returns the number of pixels an element's content is scrolled vertically.

Syntax

Return the scrollTop property:

element.scrollTop

Set the scrollTop property:

element.scrollTop = pixels

Property Values

|  |
| --- |
|  |
| Value | Description |
| *pixels* | The number of pixels the element's content is scrolled vertically.  If the number is negative, the number is set to 0. If the element cannot be scrolled, the number is set to 0. If the number is greater than maximum allowed, the number is set to the maximum. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Number | The number of pixels the element's content is scrolled vertically. |

JS ScrollLeft

The scrollLeft property sets or returns the number of pixels an element's content is scrolled horizontally.

Syntax

Return the scrollLeft property:

element.scrollLeft

Set the scrollLeft property:

element.scrollLeft = pixels

Property Values

|  |
| --- |
|  |
| Value | Description |
| *pixels* | The number of pixels the element's content is scrolled horizontally.  If the number is negative, the number is set to 0. If the element cannot be scrolled, the number is set to 0. If the number is greater than maximum allowed, the number is set to the maximum. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Number | The number of pixels the element's content is scrolled horizontally. |

[Previous](https://www.yahubaba.com/javascript/js-offsettop-offsetleft)[Next](https://www.yahubaba.com/javascript/js-scrollwidth-scrollheight)

## **JS offsetWidth & offsetHeight**

JS OffsetWidth

The offsetWidth property returns the viewable width of an element (in pixels) including padding, border and scrollbar, but not the margin.

The offsetWidth property is read-only.

Syntax

element.offsetWidth

Return Value

|  |
| --- |
|  |
| Type | Description |
| Number | The viewable width of an element (in pixels) including padding, border and scrollbar. |

JS OffsetHeight

The offsetHeight property returns the viewable height of an element (in pixels), including padding, border and scrollbar, but not the margin.

The offsetHeight property id read-only.

Syntax

element.offsetHeight

Return Value

|  |
| --- |
|  |
| Type | Description |
| Number | The viewable height of the element (in pixels) including padding, border and scrollbar. |

[Previous](https://www.yahubaba.com/javascript/js-scrollwidth-scrollheight)[Next](https://www.yahubaba.com/javascript/js-clientwidth-clientheight)

## **JS clientWidth & clientHeight**

JS ClientWidth

The clientWidth property returns the viewable width of an element in pixels, including padding, but not the border, scrollbar or margin.

The clientWidth property is read-only.

Syntax

element.clientWidth

Return Value

|  |
| --- |
|  |
| Type | Description |
| Number | The viewable width of an element (in pixels) including padding. |

JS ClientHeight

The clientHeight property returns the viewable height of an element in pixels, including padding, but not the border, scrollbar or margin.

The clientHeight property is read-only.

Syntax

element.clientHeight

Return Value

|  |  |
| --- | --- |
| Type | Description |
| Number | The viewable height of the element (in pixels) including padding |

[Previous](https://www.yahubaba.com/javascript/js-offsetwidth-offsetheight)[Next](https://www.yahubaba.com/javascript/js-clientx-clienty)

## **JS ClientX & ClientY**

JS ClientX

The clientX property returns the **horizontal client coordinate** of the mouse pointer when a mouse event occurs.

The clientX property is read-only.

The **client area** is the current window.

Syntax

event.clientX

|  |  |
| --- | --- |
| **Return Value:** | A Number The horiz wontal window pixel coordinate of the mouse pointer |

JS ClientY

The clientY property returns the **vertical client coordinate** of the mouse pointer when a mouse event occurs.

The clientY property is read-only.

The **client area** is the current window.

Syntax

event.clientY

|  |  |
| --- | --- |
| **Return Value:** | A Number The vertical window pixel coordinate of the mouse pointer |

[Previous](https://www.yahubaba.com/javascript/js-clientwidth-clientheight)[Next](https://www.yahubaba.com/javascript/js-pagex-pagey)

## **JS pageX & pageY**

JS PageX

The pageX property returns the **document relative X coordinate** of the mouse pointer when a mouse event occurs.

The pageX property is read-only.

The **document** is the web page.

Syntax

event.pageX

|  |  |
| --- | --- |
| **Return Value:** | A Number. The X coordinate of the mouse pointer, in pixels. |

JS PageY

The pageY property returns the **document relative coordinate** the mouse pointer when a mouse event occurs.

The pageY property is read-only.

The **document** is the web page.

Syntax

event.pageY

|  |  |
| --- | --- |
| **Return Value:** | A Number. The Y (vertical) pixel coordinate of the mouse pointer. |

[Previous](https://www.yahubaba.com/javascript/js-clientx-clienty)[Next](https://www.yahubaba.com/javascript/js-screenx-screeny)

## **JS screenX & screenY**

JS PageX

The pageX property returns the **document relative X coordinate** of the mouse pointer when a mouse event occurs.

The pageX property is read-only.

The **document** is the web page.

Syntax

event.pageX

|  |  |
| --- | --- |
| **Return Value:** | A Number. The X coordinate of the mouse pointer, in pixels. |

JS PageY

The pageY property returns the **document relative coordinate** the mouse pointer when a mouse event occurs.

The pageY property is read-only.

The **document** is the web page.

Syntax

event.pageY

|  |  |
| --- | --- |
| **Return Value:** | A Number. The Y (vertical) pixel coordinate of the mouse pointer. |

[**Previous**](https://www.yahubaba.com/javascript/js-clientx-clienty)[**Next**](https://www.yahubaba.com/javascript/js-screenx-screeny)

## **JS pageX & pageY**

JS PageX

The pageX property returns the **document relative X coordinate** of the mouse pointer when a mouse event occurs.

The pageX property is read-only.

The **document** is the web page.

Syntax

event.pageX

|  |  |
| --- | --- |
| **Return Value:** | A Number. The X coordinate of the mouse pointer, in pixels. |

JS PageY

The pageY property returns the **document relative coordinate** the mouse pointer when a mouse event occurs.

The pageY property is read-only.

The **document** is the web page.

Syntax

event.pageY

|  |  |
| --- | --- |
| **Return Value:** | A Number. The Y (vertical) pixel coordinate of the mouse pointer. |

[**Previous**](https://www.yahubaba.com/javascript/js-clientx-clienty)[**Next**](https://www.yahubaba.com/javascript/js-screenx-screeny)

## **JS CtrlKey AltKey ShiftKey MetaKey**

### MouseEvent ctrlKey Property

The **mouseEvent ctrlKey** property is used to define whether the ctrl key is pressed or not. It is a boolean value. When the ctrl key is pressed then on click of the mouse buttons it returns true and if it is not pressed then it returns false.

**Syntax:**

event.ctrlKey

**Return Value:** It returns a boolean value indicating whether the ctrl key is pressed or not.

* **true:**it indicates the ctrl key is pressed.
* **false:**it indicates the ctrl key is not pressed.

### MouseEvent AltKey Property

The **mouseEvent altKey** property is used to define whether the alt key is pressed or not. It is a boolean value. When the alt key is pressed then on click of the mouse buttons it returns true and if it is not pressed then it returns false.

**Syntax:**

event.altKey

**Return Value:** It returns a boolean value indicating whether the alt key is pressed or not.

* **true:**it indicates the alt key is pressed
* **false:**it indicates the alt key is not pressed

### mouseEvent shiftKey Property

The **mouseEvent shiftKey** property is used to define whether the shift key is pressed or not. It is a boolean value. When the shift key is pressed then on click of the mouse left button, it returns true and if the shift key is not pressed then it returns false.

**Syntax:**

event.shiftKey

**Return Value:** It returns a boolean value indicating whether the shift key is pressed or not.

* **true:**it indicates the shift key is pressed.
* **false:**it indicates the shift key is not pressed.

### mouseEvent MetaKey Property

The metaKey property returns true if the **META key is pressed** when a mouse event is triggered, otherwise it returns false.

The metaKey property is read-only.

### Syntax

event.metaKey

### Return Value:

A Boolean.  
true if the META key was pressed when a mouse event occured, otherwise false.

[**Previous**](https://www.yahubaba.com/javascript/js-offsetx-offsety)[**Next**](https://www.yahubaba.com/javascript/js-button)

## **JS Button**

### MouseEvent button property

The MouseEvent button property is used to define the left or right-click events. When the mouse button is clicked then it returns an integer value which describes the left, right, or middle mouse button.   
  
**Syntax:**

event.button

**Return Value:** This event returns an integer value on mouse click events are:

* **0:** It indicates the left mouse button.
* **1:** It indicates the middle mouse button.
* **2:** It indicates the right mouse button.

**The onmousedown event:** This event occurs when a user presses a mouse button over an element.

[Previous](https://www.yahubaba.com/javascript/js-ctrlkey-altkey-shiftkey-metakey)[Next](https://www.yahubaba.com/javascript/js-key-property)

## **JS Key Property**

### JS Key Property

The key property returns **the key that was pressed** when the event occured.

The key property is read-only.

The return value can be:

* A single character ("a", "W", "4", "+", "$")
* Multiple characters ("F1", "Enter", "HOME", "CAPS LOCK")

### Syntax

event.key

### Return Value:

A String  
The key that was pressed:  
A single character ("A", "a", "4", "+", "$")  
Multiple characters ("F1", "Enter", "HOME", "CAPS LOCK")

## **JS Key Property**

### JS Key Property

The key property returns **the key that was pressed** when the event occured.

The key property is read-only.

The return value can be:

* A single character ("a", "W", "4", "+", "$")
* Multiple characters ("F1", "Enter", "HOME", "CAPS LOCK")

### Syntax

event.key

### Return Value:

A String  
The key that was pressed:  
A single character ("A", "a", "4", "+", "$")  
Multiple characters ("F1", "Enter", "HOME", "CAPS LOCK")

## **JS DOM Introduction**

### The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:  
  
With the object model, JavaScript gets all the power it needs to create dynamic HTML:

* JavaScript can change all the HTML elements in the page
* JavaScript can change all the HTML attributes in the page
* JavaScript can change all the CSS styles in the page
* JavaScript can remove existing HTML elements and attributes
* JavaScript can add new HTML elements and attributes
* JavaScript can react to all existing HTML events in the page
* JavaScript can create new HTML events in the page

### What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

* Core DOM - standard model for all document types
* XML DOM - standard model for XML documents
* HTML DOM - standard model for HTML documents

### What is the HTML DOM?

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

* The HTML elements as **objects**
* The **properties** of all HTML elements
* The **methods** to access all HTML elements
* The **events** for all HTML elements

[Previous](https://www.yahubaba.com/javascript/js-date-methods)[Next](https://www.yahubaba.com/javascript/js-dom-targeting-methods)

## **JS DOM Targeting Methods**

### JS DOM Targeting Methods

Most often, you want to manipulate HTML elements. To manipulate element you have to find the elements first.  
Javascript provides us with various methods to find an element within the document.

| **Methods** | **Description** |
| --- | --- |
| document.getElementById() | Select the unique element with given id. In case there are 2 same ID then it selects the first element. |
| document.getElementsByClassName() | Select collection elements with given classname |
| document.getElementsByTagName() | Select collection elements with given tagname |

### DOM getElementsById() Method

The **getElementById()** method returns the elements that have given an ID which is passed to the function. This function is a widely used HTML DOM method in web designing to change the value of any particular element or get a particular element. If the passed ID to the function does not exist then it returns null. The element is required to have a unique id, in order to get access to that specific element quickly, & also that particular id should only be used once in the entire document.

**Syntax:**

document.getElementById( element\_ID )

**Parameter:** This function accepts single parameter element\_ID which is used to hold the ID of the element.

**Return Value:** It returns the object of the given ID. If no element exists with the given ID then it returns null.

### DOM getElementsByClassName() Method

The **getElementsByClassName()** method in Javascript returns an object containing all the elements with the specified class names in the document as objects. Each element in the returned object can be accessed by its index. The index value will start with 0. This method can be called upon by any individual element to search for its descendant elements with the specified class names.

**Syntax:**

document.getElementsByClassName(classnames);

**Parameters:** This is a required method that takes only one parameter, which is a string containing space-separated class names of the elements that are to be searched for. For searching with multiple class names, it must be separated with space.

### DOM getElementsByTagName() Method

The HTML**DOM getElementsByTagName()**method in HTML returns the collection of all the elements in the document with the given tag name. To extract any info just iterate through all the elements using the length property.

**Syntax:**

var elements = document.getElementsByTagName(name);

var elements = document.getElementsByTagName(name);

**Parameters:**

* **elements** is a collection of all the found elements in the order they appear with the given tag name.
* **name** is a string representing the name of the elements. The special string “\*” represents all elements.

### html file

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<link rel="stylesheet" href="css/main.css">

<script src="js/dom-main.js"></script>

</head>

<body>

<div id="wrapper">

<div id="header">

<h1>Lbsti Baba</h1>

</div>

<div id="menu">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="content">

<h2>Sub Heading</h2>

<img src="images/flower.jpg" width="200px" class="content-image" alt="">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt, veniam eius architecto ullam cupiditate quam aspernatur quis facilis tempora vel! Aspernatur, consequatur, laborum, explicabo consequuntur minima beatae perferendis impedit accusantium ex animi odit quisquam sint pariatur minus amet ullam reprehenderit rerum inventore sed officiis voluptas? Dolore, perferendis, minus eum debitis vero ipsam tempora voluptate nam ut autem itaque provident consequatur nobis quia libero! Magni, molestiae, laborum architecto natus inventore facere quis pariatur quia quos quasi quo dicta dolores. Deleniti, facere, fugit sed minus ducimus ut modi voluptatum eaque praesentium saepe aperiam nam quidem laboriosam assumenda voluptate vitae inventore et quibusdam!</p>

<ul class="list">

<li>Lorem ipsum dolor sit amet. </li>

<li>Modi nihil in animi necessitatibus.</li>

<li>Consectetur adipisicing elit.</li>

<li>Lorem ipsum dolor sit amet.</li>

<li>Modi nihil in animi dolore natus.</li>

</ul>

<p class="list">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, quam.</p>

</div>

<div id="sidebar">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="footer">

lbstibaba@copyright 2018.

</div>

</div>

</body>

</html>

### main.css

@import "color.css";

\*{

box-sizing: border-box;

}

body{

font: 18px/24px arial;

}

#wrapper{

border:1px solid black;

width: 1000px;

background: white;

margin: 0 auto;

}

#top-bar{

background: MEDIUMPURPLE;

}

#top-bar ul{

margin: 0;

padding: 0;

list-style: none;

}

#top-bar ul li{

display: inline-block;

margin: 5px;

}

#top-bar a{

color: #fff;

font-size: 20px;

}

#top-bar a:hover{

color: crimson;

}

#header{

min-height: 100px;

font-family: arial;

border-bottom: 1px solid black;

}

#header h1{

padding:30px 0 0 20px;

margin: 0;

}

#menu{

border-bottom: 1px solid black;

}

#menu ul{

margin: 0;

padding: 0;

}

#menu li{

display:inline-block;

}

#menu li a{

display:block;

padding: 10px 13px;

text-decoration:none;

}

#menu li a:hover{

}

#content{

width:795px;

min-height:500px;

padding: 15px;

float:left;

box-sizing:border-box;

position:relative;

}

#content h2{

font-family: arial;

}

.content-image{

float:left;margin:0 15px 10px 0;

}

#sidebar{

width:200px;

min-height:500px;

float:right;

border-left: 1px solid black;

}

#sidebar a{

text-decoration:none;

color:black;

}

#footer{

padding: 5px 10px;

text-align:right;

font-size: 14px;

clear:both;

border-top: 1px solid black;

}

@import "mediatest.css" screen and (max-width:1020px);

@media screen and (max-width:810px){

#wrapper{

width:500px;

}

#content{

width:65%;

}

#sidebar{

width:34%

}

}

@media screen and (max-width:510px){

#wrapper{

width:auto;

}

#content{

width:100%;

float:none;

}

#sidebar{

width:100%;

float:none;

}

#header h1{

padding:30px 0 0 0;

text-align:center;

}

}

### dom-main.js

var element;

//element = document.all;

//element = document.head;

//element = document.title;

//element = document.body;

//element = document.links;

//element = document.links[0];

//element = document.images;

//element = document.forms;

//element = document.doctype;

//element = document.URL;

//element = document.domain;

//element = document.baseURI;

//element = document.getElementById("header");

//element = document.getElementsByClassName("list");

//element = document.getElementsByClassName("list")[0]

//element = document.getElementsByTagName("ul");

element = document.getElementsByTagName("ul")[2]

console.log(element);

## **JS DOM Get & Set Value Methods**

Getting Element's Attribute Value

The getAttribute() method is used to get the current value of a attribute on the element.

If the specified attribute does not exist on the element, it will return null.

Setting Attributes on Elements

The setAttribute() method is used to set an attribute on the specified element.

If the attribute already exists on the element, the value is updated; otherwise a new attribute is added with the specified name and value.

html file

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<link rel="stylesheet" href="css/main.css">

</head>

<body>

<div id="wrapper">

<div id="header" class="abc" style="border:10px solid yellow;">

<h1>Lbsti Baba</h1>

</div>

<div id="menu">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="content">

<h2>Sub Heading</h2>

<img src="images/flower.jpg" width="200px" class="content-image" alt="">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt, veniam eius architecto ullam cupiditate quam aspernatur quis facilis tempora vel! Aspernatur, consequatur, laborum, explicabo consequuntur minima beatae perferendis impedit accusantium ex animi odit quisquam sint pariatur minus amet ullam reprehenderit rerum inventore sed officiis voluptas? Dolore, perferendis, minus eum debitis vero ipsam tempora voluptate nam ut autem itaque provident consequatur nobis quia libero! Magni, molestiae, laborum architecto natus inventore facere quis pariatur quia quos quasi quo dicta dolores. Deleniti, facere, fugit sed minus ducimus ut modi voluptatum eaque praesentium saepe aperiam nam quidem laboriosam assumenda voluptate vitae inventore et quibusdam!</p>

<ul class="list">

<li>Lorem ipsum dolor sit amet. </li>

<li>Modi nihil in animi necessitatibus.</li>

<li>Consectetur adipisicing elit.</li>

<li>Lorem ipsum dolor sit amet.</li>

<li>Modi nihil in animi dolore natus.</li>

</ul>

<p class="list">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, quam.</p>

</div>

<div id="sidebar">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="footer">

lbstibaba@copyright 2018.

</div>

</div>

<script src="js/dom-main.js"></script>

</body>

</html>

main.css

@import "color.css";

\*{

box-sizing: border-box;

}

body{

font: 18px/24px arial;

}

#wrapper{

border:1px solid black;

width: 1000px;

background: white;

margin: 0 auto;

}

#top-bar{

background: MEDIUMPURPLE;

}

#top-bar ul{

margin: 0;

padding: 0;

list-style: none;

}

#top-bar ul li{

display: inline-block;

margin: 5px;

}

#top-bar a{

color: #fff;

font-size: 20px;

}

#top-bar a:hover{

color: crimson;

}

#header{

min-height: 100px;

font-family: arial;

border-bottom: 1px solid black;

}

#header h1{

padding:30px 0 0 20px;

margin: 0;

}

#menu{

border-bottom: 1px solid black;

}

#menu ul{

margin: 0;

padding: 0;

}

#menu li{

display:inline-block;

}

#menu li a{

display:block;

padding: 10px 13px;

text-decoration:none;

}

#menu li a:hover{

}

#content{

width:795px;

min-height:500px;

padding: 15px;

float:left;

box-sizing:border-box;

position:relative;

}

#content h2{

font-family: arial;

}

.content-image{

float:left;margin:0 15px 10px 0;

}

#sidebar{

width:200px;

min-height:500px;

float:right;

border-left: 1px solid black;

}

#sidebar a{

text-decoration:none;

color:black;

}

#footer{

padding: 5px 10px;

text-align:right;

font-size: 14px;

clear:both;

border-top: 1px solid black;

}

@import "mediatest.css" screen and (max-width:1020px);

@media screen and (max-width:810px){

#wrapper{

width:500px;

}

#content{

width:65%;

}

#sidebar{

width:34%

}

}

@media screen and (max-width:510px){

#wrapper{

width:auto;

}

#content{

width:100%;

float:none;

}

#sidebar{

width:100%;

float:none;

}

#header h1{

padding:30px 0 0 0;

text-align:center;

}

}

dom-main.js

var element;

//element = document.getElementById("header").innerText;

//element = document.getElementById("content").innerText;

//element = document.getElementById("content").innerHTML;

//element = document.getElementById("header").innerHTML;

//element = document.getElementById("header").getAttribute("class");

//element = document.getElementById("header").getAttribute("style");

//element = document.getElementById("header").getAttribute("onClick");

//element = document.getElementById("header").getAttributeNode("onClick");

//element = document.getElementById("header").getAttributeNode("style");

//element = document.getElementById("header").getAttributeNode("style").value;

//element = document.getElementById("header").attributes;

//element = document.getElementById("header").attributes[1];

//element = document.getElementById("header").attributes[2].name;

/\* DOM Set Method \*/

//document.getElementById("header").innerHTML = "<h1>WOW</h1>";

//element = document.getElementById("header").setAttribute("style","border:10px dotted yellow");

//element = document.getElementById("header").attributes[1].value = "xyz";

element = document.getElementById("header").removeAttribute("style");

element = document.getElementById("header").removeAttribute("class");

console.log(element);

## **JS DOM querySelectors**

JS DOM querySelectors

The querySelector() method returns the first child element that matches a specified *CSS selector(s)* of an element.

**Note:** The querySelector() method only returns the first element that matches the specified selectors. To return all the matches, use the querySelectorAll() method instead.

html file

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<link rel="stylesheet" href="css/main.css">

</head>

<body>

<div id="wrapper">

<div id="header" class="abc">

<h1>Lbsti Baba</h1>

<h1>Lbsti Babaaaaaaaa</h1>

</div>

<div id="menu">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="content">

<h2>Sub Heading</h2>

<img src="images/flower.jpg" width="200px" class="content-image" alt="">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt, veniam eius architecto ullam cupiditate quam aspernatur quis facilis tempora vel! Aspernatur, consequatur, laborum, explicabo consequuntur minima beatae perferendis impedit accusantium ex animi odit quisquam sint pariatur minus amet ullam reprehenderit rerum inventore sed officiis voluptas? Dolore, perferendis, minus eum debitis vero ipsam tempora voluptate nam ut autem itaque provident consequatur nobis quia libero! Magni, molestiae, laborum architecto natus inventore facere quis pariatur quia quos quasi quo dicta dolores. Deleniti, facere, fugit sed minus ducimus ut modi voluptatum eaque praesentium saepe aperiam nam quidem laboriosam assumenda voluptate vitae inventore et quibusdam!</p>

<ul class="list">

<li>Lorem ipsum dolor sit amet. </li>

<li>Modi nihil in animi necessitatibus.</li>

<li>Consectetur adipisicing elit.</li>

<li>Lorem ipsum dolor sit amet.</li>

<li>Modi nihil in animi dolore natus.</li>

</ul>

<p class="list">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, quam.</p>

</div>

<div id="sidebar">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="footer">

lbstibaba@copyright 2018.

</div>

</div>

<script src="js/dom-main.js"></script>

</body>

</html>

main.css

@import "color.css";

\*{

box-sizing: border-box;

}

body{

font: 18px/24px arial;

}

#wrapper{

border:1px solid black;

width: 1000px;

background: white;

margin: 0 auto;

}

#top-bar{

background: MEDIUMPURPLE;

}

#top-bar ul{

margin: 0;

padding: 0;

list-style: none;

}

#top-bar ul li{

display: inline-block;

margin: 5px;

}

#top-bar a{

color: #fff;

font-size: 20px;

}

#top-bar a:hover{

color: crimson;

}

#header{

min-height: 100px;

font-family: arial;

border-bottom: 1px solid black;

}

#header h1{

padding:30px 0 0 20px;

margin: 0;

}

#menu{

border-bottom: 1px solid black;

}

#menu ul{

margin: 0;

padding: 0;

}

#menu li{

display:inline-block;

}

#menu li a{

display:block;

padding: 10px 13px;

text-decoration:none;

}

#menu li a:hover{

}

#content{

width:795px;

min-height:500px;

padding: 15px;

float:left;

box-sizing:border-box;

position:relative;

}

#content h2{

font-family: arial;

}

.content-image{

float:left;margin:0 15px 10px 0;

}

#sidebar{

width:200px;

min-height:500px;

float:right;

border-left: 1px solid black;

}

#sidebar a{

text-decoration:none;

color:black;

}

#footer{

padding: 5px 10px;

text-align:right;

font-size: 14px;

clear:both;

border-top: 1px solid black;

}

@import "mediatest.css" screen and (max-width:1020px);

@media screen and (max-width:810px){

#wrapper{

width:500px;

}

#content{

width:65%;

}

#sidebar{

width:34%

}

}

@media screen and (max-width:510px){

#wrapper{

width:auto;

}

#content{

width:100%;

float:none;

}

#sidebar{

width:100%;

float:none;

}

#header h1{

padding:30px 0 0 0;

text-align:center;

}

}

dom-main.js

var element;

//element = document.querySelector("#header").innerHTML = "<h1>WOW</h1>";

//element = document.querySelector("#header").getAttribute ("class");

//element = document.querySelectorAll(".list");

element = document.querySelectorAll(".list")[1].innerHTML;

element = document.querySelectorAll("ul");

element = document.querySelectorAll("ul")[1].innerHTML;

element = document.querySelectorAll("#header h1");

element = document.querySelectorAll("#header h1")[1].innerHTML;

console.log(element);

[**Previous**](https://www.yahubaba.com/javascript/js-dom-get-set-value-methods)[**Next**](https://www.yahubaba.com/javascript/js-dom-css-styling-methods)

## **JS DOM CSS Styling Methods**

html file

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<link rel="stylesheet" href="css/main.css">

</head>

<body>

<div id="wrapper">

<div id="header" class="abc">

<h1>Lbsti Baba</h1>

</div>

<div id="menu">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="content">

<h2>Sub Heading</h2>

<img src="images/flower.jpg" width="200px" class="content-image" alt="">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt, veniam eius architecto ullam cupiditate quam aspernatur quis facilis tempora vel! Aspernatur, consequatur, laborum, explicabo consequuntur minima beatae perferendis impedit accusantium ex animi odit quisquam sint pariatur minus amet ullam reprehenderit rerum inventore sed officiis voluptas? Dolore, perferendis, minus eum debitis vero ipsam tempora voluptate nam ut autem itaque provident consequatur nobis quia libero! Magni, molestiae, laborum architecto natus inventore facere quis pariatur quia quos quasi quo dicta dolores. Deleniti, facere, fugit sed minus ducimus ut modi voluptatum eaque praesentium saepe aperiam nam quidem laboriosam assumenda voluptate vitae inventore et quibusdam!</p>

<ul class="list">

<li>Lorem ipsum dolor sit amet. </li>

<li>Modi nihil in animi necessitatibus.</li>

<li>Consectetur adipisicing elit.</li>

<li>Lorem ipsum dolor sit amet.</li>

<li>Modi nihil in animi dolore natus.</li>

</ul>

<p class="list">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, quam.</p>

</div>

<div id="sidebar">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="footer">

lbstibaba@copyright 2018.

</div>

</div>

<script src="js/dom-main.js"></script>

</body>

</html>

main.css

@import "color.css";

\*{

box-sizing: border-box;

}

body{

font: 18px/24px arial;

}

#wrapper{

border:1px solid black;

width: 1000px;

background: white;

margin: 0 auto;

}

#top-bar{

background: MEDIUMPURPLE;

}

#top-bar ul{

margin: 0;

padding: 0;

list-style: none;

}

#top-bar ul li{

display: inline-block;

margin: 5px;

}

#top-bar a{

color: #fff;

font-size: 20px;

}

#top-bar a:hover{

color: crimson;

}

#header{

min-height: 100px;

font-family: arial;

border-bottom: 1px solid black;

}

#header h1{

padding:30px 0 0 20px;

margin: 0;

}

#menu{

border-bottom: 1px solid black;

}

#menu ul{

margin: 0;

padding: 0;

}

#menu li{

display:inline-block;

}

#menu li a{

display:block;

padding: 10px 13px;

text-decoration:none;

}

#menu li a:hover{

}

#content{

width:795px;

min-height:500px;

padding: 15px;

float:left;

box-sizing:border-box;

position:relative;

}

#content h2{

font-family: arial;

}

.content-image{

float:left;margin:0 15px 10px 0;

}

#sidebar{

width:200px;

min-height:500px;

float:right;

border-left: 1px solid black;

}

#sidebar a{

text-decoration:none;

color:black;

}

#footer{

padding: 5px 10px;

text-align:right;

font-size: 14px;

clear:both;

border-top: 1px solid black;

}

@import "mediatest.css" screen and (max-width:1020px);

@media screen and (max-width:810px){

#wrapper{

width:500px;

}

#content{

width:65%;

}

#sidebar{

width:34%

}

}

@media screen and (max-width:510px){

#wrapper{

width:auto;

}

#content{

width:100%;

float:none;

}

#sidebar{

width:100%;

float:none;

}

#header h1{

padding:30px 0 0 0;

text-align:center;

}

}

main-dom.js

var element;

//element = document.querySelector("#header").style.border;

//element = document.querySelector("#header").style.color;

//document.querySelector("#header").style.backgroundColor = "tan";

//document.querySelector("#header").style.color = "blue";

//document.querySelector("#header").className = "abc xyz";

//element = document.querySelector("#header").className;

//document.querySelector("#header").classList = "abc xyz";

//element = document.querySelector("#header").classList;

//document.querySelector("#header").classList.add("xyz","efg");

//element = document.querySelector("#header").classList;

//document.querySelector("#header").classList.remove("xyz");

element = document.querySelector("#header").classList;

//console.log(element);

[Previous](https://www.yahubaba.com/javascript/js-dom-queryselectors)[Next](https://www.yahubaba.com/javascript/js-addeventlistener-method)

## **JS addEventListener Method**

JS  addEventListener Method

The **addEventListener()** method is used to attach an event handler to a particular element. It does not override the existing event handlers. Events are said to be an essential part of the JavaScript. A web page responds according to the event that occurred. Events can be user-generated or generated by API's. An event listener is a JavaScript's procedure that waits for the occurrence of an event.

The addEventListener() method is an inbuilt function of . We can add multiple event handlers to a particular element without overwriting the existing event handlers.

Syntax:

element.addEventListener(event, function, useCapture);

Although it has three parameters, the parameters ***event*** and ***function*** are widely used. The third parameter is optional to define. The values of this function are defined as follows.

Parameter Values

**event:** It is a required parameter. It can be defined as a string that specifies the event's name.  
  
**function:** It is also a required parameter. It is a JavaScript function which responds to the event occur.  
  
**useCapture:** It is an optional parameter. It is a Boolean type value that specifies whether the event is executed in the bubbling or capturing phase. Its possible values are **true** and **false**. When it is set to true, the event handler executes in the capturing phase. When it is set to false, the handler executes in the bubbling phase. Its default value is **false**.

html file for add event listener and remove event listener

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<link rel="stylesheet" href="css/main.css">

</head>

<body>

<div id="wrapper">

<div id="header" class="abc">

<h1>Lbsti Baba</h1>

</div>

<div id="menu">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="content">

<h2>Sub Heading</h2>

<img src="images/flower.jpg" width="200px" class="content-image" alt="">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt, veniam eius architecto ullam cupiditate quam aspernatur quis facilis tempora vel! Aspernatur, consequatur, laborum, explicabo consequuntur minima beatae perferendis impedit accusantium ex animi odit quisquam sint pariatur minus amet ullam reprehenderit rerum inventore sed officiis voluptas? Dolore, perferendis, minus eum debitis vero ipsam tempora voluptate nam ut autem itaque provident consequatur nobis quia libero! Magni, molestiae, laborum architecto natus inventore facere quis pariatur quia quos quasi quo dicta dolores. Deleniti, facere, fugit sed minus ducimus ut modi voluptatum eaque praesentium saepe aperiam nam quidem laboriosam assumenda voluptate vitae inventore et quibusdam!</p>

<ul class="list">

<li>Lorem ipsum dolor sit amet. </li>

<li>Modi nihil in animi necessitatibus.</li>

<li>Consectetur adipisicing elit.</li>

<li>Lorem ipsum dolor sit amet.</li>

<li>Modi nihil in animi dolore natus.</li>

</ul>

<p class="list">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, quam.</p>

</div>

<div id="sidebar">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="footer">

lbstibaba@copyright 2018.

</div>

</div>

<script src="js/dom-main.js"></script>

</body>

</html>

main.css for add event listener and remove event listener

@import "color.css";

\*{

box-sizing: border-box;

}

body{

font: 18px/24px arial;

}

#wrapper{

border:1px solid black;

width: 1000px;

background: white;

margin: 0 auto;

}

#top-bar{

background: MEDIUMPURPLE;

}

#top-bar ul{

margin: 0;

padding: 0;

list-style: none;

}

#top-bar ul li{

display: inline-block;

margin: 5px;

}

#top-bar a{

color: #fff;

font-size: 20px;

}

#top-bar a:hover{

color: crimson;

}

#header{

min-height: 100px;

font-family: arial;

border-bottom: 1px solid black;

}

#header h1{

padding:30px 0 0 20px;

margin: 0;

}

#menu{

border-bottom: 1px solid black;

}

#menu ul{

margin: 0;

padding: 0;

}

#menu li{

display:inline-block;

}

#menu li a{

display:block;

padding: 10px 13px;

text-decoration:none;

}

#menu li a:hover{

}

#content{

width:795px;

min-height:500px;

padding: 15px;

float:left;

box-sizing:border-box;

position:relative;

}

#content h2{

font-family: arial;

}

.content-image{

float:left;margin:0 15px 10px 0;

}

#sidebar{

width:200px;

min-height:500px;

float:right;

border-left: 1px solid black;

}

#sidebar a{

text-decoration:none;

color:black;

}

#footer{

padding: 5px 10px;

text-align:right;

font-size: 14px;

clear:both;

border-top: 1px solid black;

}

@import "mediatest.css" screen and (max-width:1020px);

@media screen and (max-width:810px){

#wrapper{

width:500px;

}

#content{

width:65%;

}

#sidebar{

width:34%

}

}

@media screen and (max-width:510px){

#wrapper{

width:auto;

}

#content{

width:100%;

float:none;

}

#sidebar{

width:100%;

float:none;

}

#header h1{

padding:30px 0 0 0;

text-align:center;

}

}

dom-main.js for add event listner and remove event listener

// onclick event

//document.getElementById("header").onclick = abc;

// onmouseenter event

//document.getElementById("header").onmouseenter = abc;

// with addEventListener

/\* document.getElementById("header").addEventListener("mouseenter",abc);

document.getElementById("header").addEventListener("click", function(){

document.getElementById("header").style.border = "10px solid red";

}); \*/

/\* document.getElementById("header").addEventListener("click", abc);

document.getElementById("header").addEventListener("click", function() {

this.style.border = "10px solid red";

}); \*/

// with removeEventListener

//document.getElementById("header").removeEventListener('mouseleave',abc);

//document.getElementById("header").addEventListener("click",xyz);

function abc(){

document.getElementById("header").style.background = "green";

}

function xyz() {

document.getElementById("header").removeEventListener('mouseleave',abc);

}

/\* ====================================================== \*/

html for usecapture

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<style>

#outer{

width: 500px;

height: 500px;

margin: 0 auto;

background: LIGHTSALMON;

}

#inner{

width: 300px;

height: 300px;

margin: 100px auto 0;

background: MEDIUMORCHID;

}

</style>

</head>

<body>

<div id="outer">

<h2>Outer</h2>

<div id="inner">

<h2>Inner</h2>

</div>

</div>

<script src="js/usecapture.js"></script>

</body>

</html>

usecapture.js

document.querySelector("#inner").addEventListener('click',function(){

alert('Inner Div');

},false);

document.querySelector("#outer").addEventListener('click',function(){

alert('Outer Div');

},false);

[Previous](https://www.yahubaba.com/javascript/js-dom-css-styling-methods)[Next](https://www.yahubaba.com/javascript/js-classlist-methods)

## **JS classList Methods**

### JS classList Methods

The classList property is used for representing the value of the class elements which is a **DOMTokenList** object. It is a read-only property but we can modify its value by manipulating the classes used in the program. The JavaScript classList property consists of following methods through which we can perform different operations on the class elements:

* **add():** The add() method is used for adding one or more classes to the element.
* **remove():** The remove() method is used for removing one or more classes from the number of classes present in the element.
* **toggle():** The toggle() method is used for toggling the specified class names of an element. It means on one click the specified class gets added and on another click the class gets removed. It is known as the toggle property of an element.
* **replace():** The replace() method is used for replacing an existing class with a new class.
* **contains():** The contains() method of the JavaScript classList property is used for returning the Boolean value as an output. If the class is present, the value is returned as true otherwise false is returned.
* **item():** The item() method is used for displaying the name of the classes at the particular index. Thus, it returns the class name.

### html file

<!DOCTYPE html>

<html>

<head>

<title>Basic Layout</title>

<link rel="stylesheet" href="css/main.css">

</head>

<body>

<div id="wrapper">

<div id="header" class="abc">

<h1>Lbsti Baba</h1>

</div>

<div id="menu">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="content">

<h2>Sub Heading</h2>

<img src="images/flower.jpg" width="200px" class="content-image" alt="">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt, veniam eius architecto ullam cupiditate quam aspernatur quis facilis tempora vel! Aspernatur, consequatur, laborum, explicabo consequuntur minima beatae perferendis impedit accusantium ex animi odit quisquam sint pariatur minus amet ullam reprehenderit rerum inventore sed officiis voluptas? Dolore, perferendis, minus eum debitis vero ipsam tempora voluptate nam ut autem itaque provident consequatur nobis quia libero! Magni, molestiae, laborum architecto natus inventore facere quis pariatur quia quos quasi quo dicta dolores. Deleniti, facere, fugit sed minus ducimus ut modi voluptatum eaque praesentium saepe aperiam nam quidem laboriosam assumenda voluptate vitae inventore et quibusdam!</p>

<ul class="list">

<li>Lorem ipsum dolor sit amet. </li>

<li>Modi nihil in animi necessitatibus.</li>

<li>Consectetur adipisicing elit.</li>

<li>Lorem ipsum dolor sit amet.</li>

<li>Modi nihil in animi dolore natus.</li>

</ul>

<p class="list">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsa, quam.</p>

</div>

<div id="sidebar">

<ul>

<li><a href="">Home</a></li>

<li><a href="">About Us</a></li>

<li><a href="">Gallery</a></li>

<li><a href="">Contact Us</a></li>

</ul>

</div>

<div id="footer">

lbstibaba@copyright 2018.

</div>

</div>

<script src="js/dom-main.js"></script>

</body>

</html>

### main.css

@import "color.css";

\*{

box-sizing: border-box;

}

body{

font: 18px/24px arial;

}

#wrapper{

border:1px solid black;

width: 1000px;

background: white;

margin: 0 auto;

}

#top-bar{

background: MEDIUMPURPLE;

}

#top-bar ul{

margin: 0;

padding: 0;

list-style: none;

}

#top-bar ul li{

display: inline-block;

margin: 5px;

}

#top-bar a{

color: #fff;

font-size: 20px;

}

#top-bar a:hover{

color: crimson;

}

#header{

min-height: 100px;

font-family: arial;

border-bottom: 1px solid black;

}

#header h1{

padding:30px 0 0 20px;

margin: 0;

}

#menu{

border-bottom: 1px solid black;

}

#menu ul{

margin: 0;

padding: 0;

}

#menu li{

display:inline-block;

}

#menu li a{

display:block;

padding: 10px 13px;

text-decoration:none;

}

#menu li a:hover{

}

#content{

width:795px;

min-height:500px;

padding: 15px;

float:left;

box-sizing:border-box;

position:relative;

}

#content h2{

font-family: arial;

}

.content-image{

float:left;margin:0 15px 10px 0;

}

#sidebar{

width:200px;

min-height:500px;

float:right;

border-left: 1px solid black;

}

#sidebar a{

text-decoration:none;

color:black;

}

#footer{

padding: 5px 10px;

text-align:right;

font-size: 14px;

clear:both;

border-top: 1px solid black;

}

@import "mediatest.css" screen and (max-width:1020px);

@media screen and (max-width:810px){

#wrapper{

width:500px;

}

#content{

width:65%;

}

#sidebar{

width:34%

}

}

@media screen and (max-width:510px){

#wrapper{

width:auto;

}

#content{

width:100%;

float:none;

}

#sidebar{

width:100%;

float:none;

}

#header h1{

padding:30px 0 0 0;

text-align:center;

}

}

### < dom-main.js

document.getElementById("header").addEventListener("click", abc);

function abc() {

//document.getElementById("header").classList.add("xyz","efg");

//document.getElementById("header").classList.remove("xyz");

//var a = document.getElementById("header").classList.length;

//document.getElementById("header").classList.toggle("xyz");

//var a = document.getElementById("header").classList;

//var a = document.getElementById("header").classList.item(0);

var a = document.getElementById("header").classList.contains("first");

console.log(a);

}

[Previous](https://www.yahubaba.com/javascript/js-addeventlistener-method)[Next](https://www.yahubaba.com/javascript/js-parent-method)

## **JS parent Method**

parentNode:

The parentNode property returns the parent node of an element or node.

The parentNode property is read-only.

parentElement:

The parentElement property returns the parent element of the specified element.

The difference between parentElement and parentNode, is that parentElement returns *null* if the parent node is not an element node:

html file

<!DOCTYPE html>

<html id="main">

<head>

<title>DOM Navigation</title>

</head>

<style>

#outer{

width: 550px;

height: 300px;

padding:10px 10px;

margin: 0 auto;

background: lightsalmon;

}

#inner{

width: 500px;

height: 200px;

padding:10px 10px;

margin:0 auto ;

background: mediumorchid;

}

#inner div{

display: inline-block;

background: #fff;

width: 95px;

height: 50px;

line-height: 50px;

text-align: center;

}

</style>

<body>

<div id="outer">

<h2>Outer</h2>

<div id="inner">

<h2>Inner</h2>

<div>A</div>

<div>B</div>

<div id="child-c">C</div>

<div>D</div>

<div>E</div>

</div>

</div>

<script src="js/dom-nav.js"></script>

</body>

</html>

dom-nav.js

//var a = document.getElementById("inner").parentElement;

//var a = document.getElementById("outer").parentElement;

//var a = document.body.parentElement;

//var a = document.getElementById("inner").parentElement.style.background = "red";

/\* document.getElementById("child-c").parentElement.style.background = "red";

var a = document.getElementById("child-c").parentElement; \*/

document.getElementById("child-c").parentElement.style.background = "red";

var a = document.getElementById("main").parentNode;

console.log(a);

[**Previous**](https://www.yahubaba.com/javascript/js-classlist-methods)[**Next**](https://www.yahubaba.com/javascript/js-children-methods)

## **JS Children Methods**

### JS Children Methods

The children property returns a collection of an element's child elements.

The children property returns an HTMLCollection object.

### childNodes:

The childNodes property returns a collection (list) of an elements's child nodes.

The childNodes property returns a NodeList object.

The childNodes property is read-only.

childNodes[0] is the same as firstChild.

### html file

<!DOCTYPE html>

<html id="main">

<head>

<title>Basic Layout</title>

</head>

<style>

#outer {

width: 550px;

height: 300px;

padding: 10px 10px;

margin: 0 auto;

background: lightsalmon;

}

#inner {

width: 500px;

height: 200px;

padding: 10px 10px;

margin: 0 auto;

background: mediumorchid;

}

#inner div {

display: inline-block;

background: #fff;

width: 95px;

height: 50px;

line-height: 50px;

text-align: center;

}

</style>

<body>

<div id="outer">

<h2>Outer</h2>

<div id="inner">

<h2 id="child-head">Inner</h2>

<div>A</div>

<div>B</div>

<div id="child-C">C</div>

<div>D</div>

<div id="child-E">E</div>

</div>

</div>

<script src="js/dom-nav.js"></script>

</body>

</html>

### dom-nav.js

//var a = document.getElementById("outer").children;

//var a = document.getElementById("inner").children;

//var a = document.getElementById("inner").children[1];

/\* document.getElementById("inner").children[1].style.background = "red";

var a = document.getElementById("inner").children[1]; \*/

/\* document.getElementById("inner").children[0].style.background = "red";

var a = document.getElementById("inner").children[0].innerHTML; \*/

//var a = document.getElementById("inner").childNodes;

//var a = document.getElementById("inner").childNodes[0].innerHTML;

document.getElementById("inner").childNodes[3].style.background = "red";

var a = document.getElementById("inner").childNodes[3];

console.log(a);

## **JS firstChild & lastChild Method**

firstChild:

The firstChild property returns the first child node of a node.

The firstChild property returns a node object.

The firstChild property is read-only.

The firstChild property is the same as childNodes[0].

lastChild:

The lastChild property returns the last child node of a node.

The lastChild property returns returns a node object.

The lastChild property is read-only.

html file

<!DOCTYPE html>

<html id="main">

<head>

<title>DOM Navigation</title>

</head>

<style>

#outer{

width: 550px;

height: 300px;

padding:10px 10px;

margin: 0 auto;

background: lightsalmon;

}

#inner{

width: 500px;

height: 200px;

padding:10px 10px;

margin:0 auto ;

background: mediumorchid;

}

#inner div{

display: inline-block;

background: #fff;

width: 95px;

height: 50px;

line-height: 50px;

text-align: center;

}

</style>

<body>

<div id="outer">

<h2>Outer</h2>

<div id="inner">

<h2>Inner</h2>

<div>A</div>

<div>B</div>

<div id="child-c">C</div>

<div>D</div>

<div>E</div>

</div>

</div>

<script src="js/dom-nav.js"></script>

</body>

</html>

dom-nav.js

//var a = document.getElementById("inner").firstElementChild;

//var a = document.getElementById("inner").firstElementChild.innerHTML;

/\* document.getElementById("inner").firstElementChild.style.background = "red";

var a = document.getElementById("inner").firstElementChild.innerHTML; \*/

/\* document.getElementById("outer").lastElementChild.style.background = "red";

var a = document.getElementById("outer").lastElementChild; \*/

/\* document.getElementById("inner").lastElementChild.style.background = "red";

var a = document.getElementById("inner").lastElementChild; \*/

//var a = document.getElementById("inner").firstChild;

//var a = document.getElementById("inner").lastChild;

//var a = document.getElementById("child-c").firstChild;

var a = document.getElementById("child-c").lastChild;

console.log(a);

[Previous](https://www.yahubaba.com/javascript/js-children-methods)[Next](https://www.yahubaba.com/javascript/js-nextsibling-prevsibling-method)

## **JS nextSibling & prevSibling Method**

nextSibling:  
The nextSibling property returns the next node on the same tree level.

The nextSibling returnes a node object.

The nextSibling property is read-only.

previousSibling:  
The previousSibling property returns the previous node on the same tree level.

The previousSibling property returns a node object.

The previousSibling property is read-only.

html file

<!DOCTYPE html>

<html id="main">

<head>

<title>Basic Layout</title>

</head>

<style>

#outer {

width: 550px;

height: 300px;

padding: 10px 10px;

margin: 0 auto;

background: lightsalmon;

}

#inner {

width: 500px;

height: 200px;

padding: 10px 10px;

margin: 0 auto;

background: mediumorchid;

}

#inner div {

display: inline-block;

background: #fff;

width: 95px;

height: 50px;

line-height: 50px;

text-align: center;

}

</style>

<body>

<div id="outer">

<h2>Outer</h2>

<div id="inner">

<h2 id="child-head">Inner</h2>

<div>A</div>

<div>B</div>

<div id="child-C">C</div>

<div>D</div>

<div id="child-E">E</div>

</div>

</div>

<script src="js/dom-nav.js"></script>

</body>

</html>

dom-nav.js

//var a = document.getElementById("child-C").nextElementSibling;

//var a = document.getElementById("child-C").previousElementSibling;

//var a = document.getElementById("child-C").previousElementSibling.innerHTML ;

/\* document.getElementById("child-C").previousElementSibling.style.background = "red";

var a = document.getElementById("child-C").previousElementSibling.innerHTML; \*/

/\* document.getElementById("child-C").nextElementSibling.style.background = "red";

var a = document.getElementById("child-C").nextElementSibling; \*/

//var a = document.getElementById("child-E").nextElementSibling;

//var a = document.getElementById("child-head").previousElementSibling;

//var a = document.getElementById("child-C").previousSibling;

var a = document.getElementById("child-C").nextSibling;

console.log(a);

[Previous](https://www.yahubaba.com/javascript/js-firstchild-lastchild-method)[Next](https://www.yahubaba.com/javascript/js-create-textnode)

## **JS create & TextNode**

createElement()

The createElement() method creates an element node.

Syntax

document.createElement()

Parameters

|  |  |
| --- | --- |
| Parameter | Description |
| *type* | Required. The type of element to create. |

Return Value

|  |  |
| --- | --- |
| Type | Description |
| Node | The created element node. |

createTextNode()

The createTextNode() method creates a text node.

Syntax

document.createTextNode(text)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *text* | Required. The text for the node. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Node | The created text node. |

createComment()

The createComment() method creates a comment and returns the comment node.

Syntax

document.createComment(text)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *text* | Optional. The comment text. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Node | The created comment node. |

html file

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

<style>

h1{

text-align: center;

color:#ff0000;

}

</style>

</head>

<body>

<h1>Lbsti Baba : DOM Create Methods</h1>

<script src="js/dom-create.js"></script>

</body>

</html>

dom-create.js

//var newElement = document.createElement("p");

var newElement = document.createElement("h2");

console.log(newElement);

var newText = document.createTextNode("This is just text");

console.log(newText);

/\* Dom Create Comment\*/

var newComment = document.createComment("this is comment");

console.log(newComment);

[Previous](https://www.yahubaba.com/javascript/js-nextsibling-prevsibling-method)[Next](https://www.yahubaba.com/javascript/js-appendchild-insertbefore)

## JS appendChild & insertBefore

JS AppendChild

The appendChild() method appends a node (element) as the last child of an element.

Syntax

element.appendChild(node)

or

node.appendChild(node)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *node* | Required. The node to append. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Node | The appended node. |

JS InsertBefore

The insertBefore() method inserts a child node before an existing child.

Syntax

element.insertBefore(new, existing)

or

node.insertBefore(new, existing)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *new* | Required. The node (element) to insert. |
| *existing* | Required. The node (element) to insert before. If null, it will be inserted at the end. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Node | The inserted node. |

html file

<!DOCTYPE html>

<html>

<head>

<title>DOM append and insertBefore</title>

<style>

h1{

text-align: center;

color:#ff0000;

}

#test{

background: #ffff00;

width: 800px;

height: 200px;

padding:10px 10px;

margin: 0 auto;

}

</style>

</head>

<body>

<h1>Lbsti Baba : DOM Create Methods</h1>

<div id="test">

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Consectetur aperiam eos vel consequatur. Delectus voluptas dolorem id exercitationem, ad ipsam consectetur hic ullam provident! Adipisci exercitationem ipsam rerum sunt doloremque magni soluta, delectus maiores, sapiente quasi labore praesentium accusamus earum cum nam saepe qui? Accusamus provident quo perferendis sint sed.</p>

<h3>lbsti baba</h3>

</div>

<script src="js/dom-create.js"></script>

</body>

</html>

dom-create.js

/\* Dom Create \*/

//var newElement = document.createElement("p");

var newElement = document.createElement("h2");

console.log(newElement);

var newText = document.createTextNode("This is just text");

console.log(newText);

/\* JavaScript AppendChild\*/

newElement.appendChild(newText);

//document.getElementById("test").appendChild(newElement);

/\* JavaScript InsertBefore \*/

var target = document.getElementById("test");

target.insertBefore(newElement,target.childNodes[0])

[Previous](https://www.yahubaba.com/javascript/js-create-textnode)[Next](https://www.yahubaba.com/javascript/js-insert)

## **JS insert**

!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

<style>

#test{

background: #ffff00;

width: 800px;

height: 200px;

padding: 10px 10px;

margin: 0 auto;

}

</style>

</head>

<body>

<div id="test">

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Consectetur aperiam eos vel consequatur. Delectus voluptas

dolorem id exercitationem, ad ipsam consectetur hic ullam provident! Adipisci exercitationem ipsam rerum sunt

doloremque magni soluta, delectus maiores, sapiente quasi labore praesentium accusamus earum cum nam saepe qui?

Accusamus provident quo perferendis sint sed.</p>

</div>

<script src="js/dom-create.js"></script>

</body>

</html>

dom-create.js

// insertAdjacentElement Method

var newElement = document.createElement("h2");

var newText = document.createTextNode("This is just element");

newElement.appendChild(newText);

var target = document.getElementById("test");

target.insertAdjacentElement("afterbegin",newElement);

// insertAdjacentHTML Method

var newElement = "<h2>This is just Html</h2>";

var target = document.getElementById("test");

target.insertAdjacentHTML("afterend",newElement);

// insertAdjacentText Method

var newText = "<h2>This is just Text</h2>";

var target = document.getElementById("test");

target.insertAdjacentHTML("beforeend",newText);

[**Previous**](https://www.yahubaba.com/javascript/js-appendchild-insertbefore)[**Next**](https://www.yahubaba.com/javascript/js-replacechild-removechild)

## **JS replaceChild & removeChild**

### replaceChild()

The replaceChild() method replaces a child node with a new node.

### Syntax

node.replaceChild(newnode, oldnode)

## **Parameters**

|  |
| --- |
|  |
| Parameter | Description |
| newnode | Required. The node to insert. |
| oldnode | Required. The node to remove. |

## **Return Value**

|  |
| --- |
|  |
| Type | Description |
| Node | The replaced node. |

### removeChild()

The removeChild() method removes an element's child.

### Syntax

element.removeChild(node)

or

node.removeChild(node)

### Parameters

|  |
| --- |
|  |
| Parameter | Description |
| node | Required. The node (element) to remove. |

### Return Value

|  |
| --- |
|  |
| Type | Description |
| Node | The removed node (element). null if the child does not exist. |

### html file

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

</head>

<body>

<ul id= "list">

<li>orange</li>

<li>Apple</li>

<li>Grapes</li>

<li>Banana</li>

</ul>

<script src="js/dom-create.js"></script>

</body>

</html>

### dom-create.js

/\*JavaScript ReplaceChild\*/

/\*var newElement = document.createElement("li");

var newText = document.createTextNode("WOW new Text");

newElement.appendChild(newText);

var target = document.getElementById("list");

var oldElement = target.children[0];\*/

//console.log(oldElement);

//target.replaceChild(newElement,oldElement);

/\*JavaScript RemoveChild\*/

var target = document.getElementById("list");

var oldElement = target.children[1];

[Previous](https://www.yahubaba.com/javascript/js-insert)[Next](https://www.yahubaba.com/javascript/js-clonenode)

## **JS cloneNode**

JS cloneNode

The cloneNode() method creates a copy of a node, and returns the clone.

The cloneNode() method clones all attributes and their values.

Syntax

node.cloneNode(deep)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *deep* | Optional. false - Default. Clone only the node and its attributes. true - Clone the node, its attributes, and its descendants. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Node | The cloned node. |

html file

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

<style>

#test{

background: #ffff00;

width: 800px;

height: 200px;

padding: 10px 10px;

margin: 0 auto;

}

</style>

</head>

<body>

<ul id= "list1">

<li class="abc">orange</li>

<li>Apple</li>

<li>Grapes</li>

<li>Banana</li>

</ul>

<ul id= "list2">

<li>Carrot</li>

<li>Reddish</li>

</ul>

<div id="test"></div>

<script src="js/dom-create.js"></script>

</body>

</html>

dom-create.js

/\* JavaScript CloneNode\*/

var target = document.getElementById("list1").children[0];

var copyElement = target.cloneNode(true);

console.log(copyElement);

document.getElementById("list2").appendChild(copyElement);

document.getElementById("test").appendChild(copyElement);

[Previous](https://www.yahubaba.com/javascript/js-replacechild-removechild)[Next](https://www.yahubaba.com/javascript/js-contains)

## **JS Contains**

JS Contains

The contains() method returns true if a node is a descendant of a node.

The contains() method returns false if not.

Syntax

node.contains(node)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *node* | Required. The node that may be a descendant of the node. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| Boolean | true - The node is a descendant false - The node is NOT a descendant |

html file

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

<style>

h2{ margin: 0 0 10px; }

#test{

background: #ffff00;

width: 800px;

min-height: 250px;

padding: 10px 10px;

margin: 0 auto;

}

</style>

</head>

<body>

<div id="test">

<h2>Lbsti Baba : JavaScript Contains Method</h2>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo corporis adipisci alias cupiditate officiis consequatur, fuga, amet quos numquam perferendis saepe labore dolorem reiciendis reprehenderit facilis repudiandae praesentium quis delectus voluptates, repellendus recusandae. Itaque consequuntur, corrupti quasi illum iusto perferendis autem blanditiis magni eum, repellendus architecto ratione ipsum molestiae laboriosam.

</p>

<div>

<p id="abc">

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Voluptatum magni vel inventore illum facere tenetur eveniet quam ex nemo eum. Dolorum suscipit corrupti asperiores dicta inventore debitis facilis amet totam neque reprehenderit minima ratione corporis quasi aperiam iusto consectetur nostrum enim optio, quas rerum doloribus nesciunt dolor deleniti. Ea, minima.

</p>

</div>

</div>

<script src="js/dom-create.js"></script>

</body>

</html>

dom-create.js

/\* JavaScript Contains Method\*/

var parentElement = document.getElementById("test");

var target = document.getElementById("abc");

var find = parentElement.contains(target);

console.log(find);

[Previous](https://www.yahubaba.com/javascript/js-clonenode)[Next](https://www.yahubaba.com/javascript/js-has)

## **JS has**

### JS hasAttribute(

The hasAttribute() method returns true if the attribute exists, otherwise false.

### Syntax

element.hasAttribute(name)

### Parameters

|  |
| --- |
|  |
| Parameter | Description |
| name | Required. The name of the attribute. |

### Return Value

|  |
| --- |
|  |
| Type | Description |
| Boolean | true if the element has the attribute, otherwise false. |

### hasChildNodes(

The hasChildNodes() method returns true if the specified node has any child nodes, otherwise false.

The hasChildNodes() method is read-only.

### Syntax

element.hasChildNodes()

### Parameters

|  |
| --- |
| NONE |

### Return Value

|  |
| --- |
|  |
| Type | Description |
| Boolean | true if the element has child nodes, otherwise false. |

## **html file**

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

<style>

h2{ margin: 0 0 10px; }

#test{

background: #ffff00;

width: 800px;

min-height: 250px;

padding: 10px 10px;

margin: 0 auto;

}

</style>

</head>

<body>

<div id="test" class="abc">

<h2>Lbsti Baba : JavaScript hasAttribute Method</h2>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo corporis adipisci alias cupiditate officiis consequatur, fuga, amet quos numquam perferendis saepe labore dolorem reiciendis reprehenderit facilis repudiandae praesentium quis delectus voluptates, repellendus recusandae. Itaque consequuntur, corrupti quasi illum iusto perferendis autem blanditiis magni eum, repellendus architecto ratione ipsum molestiae laboriosam.

</p>

<h2>Lbsti Baba : JavaScript hasChildNodes Method</h2>

</div>

<script src="js/dom-create.js"></script>

</body>

</html>

## **dom-create.js**

/\* JavaScript hasAttribute\*/

var target = document.getElementById("test");

var find = target.hasAttribute("class");

console.log(find);

/\* JavaScript hasChildNodes\*/

var target = document.getElementById("test");

var find = target.hasChildNodes();

console.log(find);

[Previous](https://www.yahubaba.com/javascript/js-contains)[Next](https://www.yahubaba.com/javascript/js-isequalnode)

## **JS isEqualNode**

### JS isEqualNode

The isEqualNode() returns true if two elements (or nodes) are equal.

### Syntax

element.isEqualNode(node)

or

node.isEqualNode(node)

### Parameters

|  |
| --- |
|  |
| Parameter | Description |
| node | Required. The node to compare. |

## **Return Value**

|  |
| --- |
|  |
| Type | Description |
| Boolean | true if the nodes are equal, otherwise false. |

### html file

<!DOCTYPE html>

<html>

<head>

<title>DOM Navigation</title>

</head>

<body>

<ul id="list-1">

<li class="abc">orange</li>

<li>Banana</li>

<li>Apple</li>

<li>Grapes</li>

</ul>

<ul id="list-2">

<li>Guava</li>

<li class="abc">orange</li>

<li>PineApple</li>

</ul>

<script src="js/dom-create.js"></script>

</body>

</html>

dom-create.js

var target1 = document.getElementById("list-1").firstElementChild;

var target2 = document.getElementById("list-2").children[1];

var equal = target1.isEqualNode(target2);

console.log(equal);

[Previous](https://www.yahubaba.com/javascript/js-has)[Next](https://www.yahubaba.com/javascript/js-form-events)

## **JS Form Events**

html file

<!DOCTYPE html>

<html lang="en">

<head>

<title>Form Events</title>

</head>

<body>

<form action="">

<label for="">Name</label><input type="text" onfocus="focusFunction(this)" onblur="blurFunction(this)" oninput="inputFunction(this)"><br><br>

<label for="">Class</label><input type="text" id="" onfocus="focusFunction(this)" onblur="blurFunction(this)"><br><br>

<select id="" onfocus="focusFunction(this)" onblur="blurFunction(this)">

<option value="">India</option>

<option value="">Pakistan</option>

<option value="">Bangladesh</option>

<option value="">Sri Lanka</option>

<option value="">Nepal</option>

</select>

<br>

</form>

<div id="test" style="border:1px solid red"></div>

<script>

/\* JavaScript Focus Event \*/

function focusFunction(element){

element.style.background = "lime";

}

/\* JavaScript Blur Event \*/

function blurFunction(element){

element.style.background = "";

}

/\* JavaScript Input Event \*/

function inputFunction(element){

var x = element.value;

document.getElementById("test").innerHTML = x;

}

</script>

</body>

</html>

[Previous](https://www.yahubaba.com/javascript/js-isequalnode)[Next](https://www.yahubaba.com/javascript/js-form-events-ii)

## **JS Form Events - II**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Form Events</title>

</head>

<body>

<form action="" onsubmit="submitFunction()">

<label for="">Name</label><input type="text" id="fname" onchange="onchangeFunction(this)" onselect="selectFunction()" oninvalid="alert("Please fill the First Name.")" required><br><br>

<label for="">Class</label><input type="text" id="" ><br><br>

<select id="" onchange="onchangeFunction(this)">

<option value="India">India</option>

<option value="Pakistan">Pakistan</option>

<option value="Bangladesh">Bangladesh</option>

<option value="Sri Lanka">Sri Lanka</option>

<option value="Nepal">Nepal</option>

</select>

<input type="submit">

</form>

<div id="test" style="border:1px solid red;margin-top:20px;"></div>

<script>

function focusFunction(element){

element.style.background = "lime";

}

/\* JavaScript Blur Event \*/

function blurFunction(element){

element.style.background = "";

}

/\* JavaScript Input Event \*/

function inputFunction(element){

var x = element.value;

document.getElementById("test").innerHTML = x;

}

/\* JavaScript Change Event \*/

function onchangeFunction(element){

var x = element.value;

document.getElementById("test").innerHTML = x;

}

/\* JavaScript Select Event \*/

function selectFunction(){

console.log("You selected some text.");

}

/\* JavaScript Submit Event \*/

/\*function submitFunction(){

alert("You submitted a form.");

}\*/

function submitFunction(){

var x = document.getElementById("fname").value;

alert("Hello " + x);

}

</script>

</body>

</html>

[Previous](https://www.yahubaba.com/javascript/js-form-events)[Next](https://www.yahubaba.com/javascript/js-interval)

## **JS Interval**

setInterval()

The setInterval() method calls a function at specified intervals (in milliseconds).

The setInterval() method continues calling the function until clearInterval() is called, or the window is closed.

1 second = 1000 milliseconds.

Syntax

setInterval(function, milliseconds, param1, param2, ...)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *function* | Required. The function to execute |
| *milliseconds* | Required. The execution interval. If the value is less than 10, 10 is used |
| *param1, param2, ...* | Optional. Additional parameters to pass to the *function* Not supported in IE9 and earlier. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| A number | The id of the timer. Use this id with clearInterval() to cancel the timer. |

clearInterval()

The clearInterval() method clears a timer set with the setInterval() method.

Syntax

clearInterval(intervalId)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *intervalId* | Required. The interval id returned from setInterval(). |

Return Value

|  |
| --- |
| NONE |

html file

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript Animation</title>

<style>

#test{

width:150px;

height:150px;

background: red;

}

</style>

</head>

<body>

<div id="test"></div>

<script src="js/animation.js"></script>

</body>

</html>

 animation.js

var a = 0;

var id = setInterval(Anim,500);

function Anim(){

a = a + 10;

if(a == 200){

clearInterval(id);

}else{

var target = document.getElementById("test");

//target.style.marginLeft = a + 'px';

target.style.width = a + 'px';

}

}

[Previous](https://www.yahubaba.com/javascript/js-form-events-ii)[Next](https://www.yahubaba.com/javascript/js-timeout)

## **JS Timeout**

setTimeout()

The setTimeout() method calls a function after a number of milliseconds. 1 second = 1000 milliseconds.

Syntax

setTimeout(function, milliseconds, param1, param2, ...)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *function* | Required. The function to execute. |
| *milliseconds* | Optional. Number of milliseconds to wait before executing. Default value is 0. |
| *param1, param2, ...* | Optional. Parameters to pass to the *function.* Not supported in IE9 and earlier. |

Return Value

|  |
| --- |
|  |
| Type | Description |
| A number | The id of the timer. Use this id with clearTimeout(id) to cancel the timer. |

clearTimeout()

The clearTimeout() method clears a timer set with the setTimeout() method.

Syntax

clearTimeout(id\_of\_settimeout)

Parameters

|  |  |
| --- | --- |
| Parameter | Description |
| *timeout id* | Required. The id returned by the setTimeout() method. |

Return Value

|  |
| --- |
| NONE |

html file

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript Animation</title>

<style>

#test{

width:150px;

height:150px;

background: red;

}

</style>

</head>

<body>

<div id="test"></div>

<button onclick="stopAnimation()">Stop Animation</button>

<script src="js/animation.js"></script>

</body>

</html>

animation.js

/\* JavaScript setTimeout\*/

/\*var id = setTimeout(Anim,5000);

function Anim(){

var target = document.getElementById("test");

target.style.width = "500px";

//console.log("Hello");

}\*/

var id = setTimeout(function(){

var target = document.getElementById("test");

target.style.width = "500px";

},5000);

/\* JavaScript clearTimeout\*/

function stopAnimation(){

clearTimeout(id);

}

[Previous](https://www.yahubaba.com/javascript/js-interval)[Next](https://www.yahubaba.com/javascript/js-bom-introduction)

## **JS BOM Introduction**

Browser Object Model

The Browser Object Model (BOM) is used to interact with the browser. The default object of browser is window means you can call all the functions of window by specifying window or directly. For example: window.alert("hello javatpoint"); is same as: alert("hello javatpoint"); You can use a lot of properties (other objects) defined underneath the window object like document, history, screen, navigator, location, innerHeight, innerWidth.

The Window Object

The window object is supported by all browsers. It represents the browser's window.

All global JavaScript objects, functions, and variables automatically become members of the window object.

Global variables are properties of the window object.

Global functions are methods of the window object.

Even the document object (of the HTML DOM) is a property of the window object:

window.document.getElementById("header");

is the same as:

document.getElementById("header");

[Previous](https://www.yahubaba.com/javascript/js-timeout)[Next](https://www.yahubaba.com/javascript/js-window-height-width)

## **JS Window Height & Width**

### JS Window Height & Width

Two properties can be used to determine the size of the browser window.

Both properties return the sizes in pixels:

* window.innerHeight - the inner height of the browser window (in pixels)
* window.innerWidth - the inner width of the browser window (in pixels)
* window.outerWidth -   returns the outer width of a window, including all interface elements (like toolbars/scrollbars).
* window.outerHeight- returns the outer height of a window, including all interface elements (like toolbars/scrollbars).

These properties are read-only.

var w = window.outerWidth;

var h = window.outerHeight;

var w = window.innerWidth;

var h = window.innerHeight;

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript</title>

</head>

<body onresize="resizeFunction()">

<script>

function resizeFunction(){

console.clear();

var iHeight = window.innerHeight;

console.log('Inner Height :' + iHeight);

var oHeight = window.outerHeight;

console.log('Outer Height :' + oHeight);

}

function resizeFunction(){

console.clear();

var iWidth = window.innerWidth;

console.log('Inner Width :' + iWidth);

var oWidth = window.outerWidth;

console.log('Outer Width :' + oWidth);

}

</script>

</body>

</html>

[Previous](https://www.yahubaba.com/javascript/js-bom-introduction)[Next](https://www.yahubaba.com/javascript/js-window-open-close)

## **JS Window Open & Close**

### Javascript Window Open() & Window Close() Method

The Javascript **Window.open()**method is used to open the web pages into a new window or a new tab. It depends on the browser settings and the values that are assigned to the parameter.

**Syntax:**

window.open(URL, name, specs, replace)

**Parameters:** This method accepts four parameters as mentioned above and described below:

* **URL:** It is an optional parameter. It is used to set the URL of web pages that need to open. If URL is not set then window.open() method open a blank window.
* **name:** It is an optional parameter and is used to set the window name.
* **specs:** It is an optional parameter used to separate the item using a comma.
* **replace:** It is an optional parameter and used to specify the URL URL creates a new entry or replaces the current entry in the history list. This parameter returns a boolean value. If this parameter returns true then URL replaces the current document in the history list and if returns false then URL creates a new entry in the history list.

**Return Value:** This method creates a new window.

**Window.close():** This method is used to close the window which is opened by the window.open() method.

**Syntax:**

window.close()

**Parameters:** This method does not contain any parameters.

**Return Value:** This method does not return any value.

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript</title>

</head>

<body>

<button onclick="openWindow()">Open Window</button>

<button onclick="closeWindow()">Close Window</button>

<script>

/\* JavaScript Open Close Window Method \*/

var myWindow;

function openWindow(){

//window.open("http://www.lbstibaba.net","lbstibaba");

//window.open("http://www.lbstibaba.net","\_blank");

//window.open("http://www.lbstibaba.net","\_parent");

myWindow = window.open("http://www.lbstibaba.net","","width=500px,height=200px,left=100px,top=200px");

}

function closeWindow(){

myWindow = window.close();

}

</script>

</body>

</html>

[Previous](https://www.yahubaba.com/javascript/js-window-height-width)[Next](https://www.yahubaba.com/javascript/js-window-move)

## **JS Window move**

JS Window moveBy()

The moveBy() method moves a window a number of pixels relative to its current coordinates.

Syntax

window.moveBy(x, y)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *x* | Required. A positive or negative number. The number of pixels to move the window horizontally. |
| *y* | Required. A positive or negative number. The number of pixels to move the window vertically. |

Return Value

|  |
| --- |
| NONE |

JS Window moveTo()

The moveTo() method moves a window to the specified coordinates.

Syntax

window.moveTo(x, y)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *x* | Required. A positive or negative number. The horizontal coordinate to move to. |
| *y* | Required. A positive or negative number. The vertical coordinate to move to. |

Return Value

|  |
| --- |
| NONE |

<!DOCTYPE html>

<html lang="en">

<head>

<title>Document</title>

</head>

<body>

<button onclick="openWindow()">Open Window</button>

<button onclick="moveWindow()">Move Window</button>

<script>

/\* JavaScript MoveBy And MoveTo Window Method \*/

var myWindow;

function openWindow(){

myWindow = window.open("","","width=500px,height=200px,left=100px,top=100px");

myWindow.document.write("<p>This is my Window.</p>")

}

/\*function moveWindow(){

myWindow.moveTo(200,200);

myWindow.focus();

}\*/

function moveWindow(){

myWindow.moveBy(200,200);

myWindow.focus();

}

</script>

</body>

</html>

[**Previous**](https://www.yahubaba.com/javascript/js-window-open-close)[**Next**](https://www.yahubaba.com/javascript/js-resize)

## **JS resize**

JS resizeBy()

The resizeBy() method resizes a window by a specified amount.

Syntax

resizeBy(width, height)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *width* | Required. A positive or a negative number. The number of pixels to resize the width by. |
| *height* | Required. A positive or a negative number. The number of pixels to resize the height by. |

Return Value

|  |
| --- |
| NONE |

Window resizeTo()

The resizeTo() method resizes a window to a new width and height.

Syntax

window.resizeTo(width, height)

Parameters

|  |  |
| --- | --- |
| Parameter | Description |
| *width* | Required. The new window width, in pixels |
| *height* | Required. The new window height, in pixels |

Return Value

|  |
| --- |
| NONE |

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript</title>

</head>

<body>

<button onclick="openWindow()">Open Window</button>

<button onclick="resizeWindow()">Resize Window</button>

<script>

/\* JavaScript ResizeBy And ResizeTo Window Method \*/

var myWindow;

function openWindow(){

myWindow = window.open("","","width=500px,height=200px,left=30px,top=30px");

myWindow.document.write("<p>This is my Window.</p>")

}

function resizeWindow(){

myWindow.resizeTo(400,400);

}

/\*function resizeWindow(){

myWindow.resizeBy(400,400);

}\*/

</script>

</body>

</html>

[Previous](https://www.yahubaba.com/javascript/js-window-move)[Next](https://www.yahubaba.com/javascript/js-scroll)

## **JS scroll**

JS scrollBy()

The scrollBy() method scrolls the document by the specified number of pixels.

Syntax

window.scrollBy(x, y)

or just:

scrollBy(x, y)

Parameters

|  |
| --- |
|  |
| Parameter | Description |
| *x* | Required. Number of pixels to scroll (horizontally). Positive values scroll to the right, negative values to the left. |
| *y* | Required. Number ofpixels to scroll (vertically). Positive values scroll down, negative values scroll up. |

Return Value

|  |
| --- |
| NONE |

JS scrollTo()

The scrollTo() method scrolls the document to specified coordinates.

Syntax

window.scrollTo(x, y)

or just:

scrollTo(x, y)

Parameters

|  |  |
| --- | --- |
| Parameter | Description |
| *x* | Required. The coordinate to scroll to (horizontally), in pixels. |
| *y* | Required. The coordinate to scroll to (vertically), in pixels. |

Return Value

|  |
| --- |
| NONE |

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript</title>

</head>

<body>

<h1>JavaScript : scrollTo & scrollBy</h1>

<div>

<p style="width:1000px;border:1px solid red;">Lorem, ipsum dolor sit amet consectetur adipisicing, elit. Eius, obcaecati eum distinctio? Itaque fuga a adipisci obcaecati perferendis consectetur mollitia nam reiciendis numquam, voluptatem. Sint ipsam vitae incidunt placeat. Veritatis tempore earum sequi exercitationem iure rem placeat, dolores voluptatum vero explicabo eum, fuga distinctio. Amet temporibus aliquid non quam corrupti! Dignissimos maiores dolorum dicta earum accusamus consectetur hic nesciunt officia aspernatur ex necessitatibus non iste est voluptates ratione unde perspiciatis placeat magni quia voluptatibus, magnam harum ipsam natus debitis, atque! Error eos provident quis repudiandae eum eaque doloremque sint animi ea. Eum aperiam ratione consectetur, dolor excepturi numquam et sed expedita doloribus, delectus. Repudiandae adipisci, repellendus, odit nesciunt voluptates nam hic repellat eaque eveniet obcaecati. In, aut blanditiis saepe reprehenderit quibusdam, repellendus necessitatibus pariatur odit fuga odio aliquam? Reiciendis quae, temporibus eaque libero unde aut vitae assumenda, officiis sint molestiae magnam fugiat earum iure expedita minus repellat nemo quo blanditiis facere error ullam ut similique cum eveniet. Labore omnis debitis minima libero deleniti assumenda odio beatae quas porro, incidunt, cupiditate, facere commodi. Eveniet quas repellat enim aliquid, nam, dicta rem ipsam delectus quae ut nisi, aut. Quod ratione sunt corporis harum enim, ut dolores molestias voluptatem inventore soluta. Repudiandae odit distinctio rerum fuga dignissimos, mollitia beatae commodi aliquid ex ipsam magni voluptatibus ratione sunt eligendi, saepe velit nulla maxime doloremque atque cum consequuntur deserunt alias? Numquam quod tempora est, ducimus ab voluptatibus, aspernatur asperiores fuga quisquam rem sapiente nulla officia accusantium laborum placeat perspiciatis corporis sequi, quis nam facere. Quis tempore unde accusantium placeat accusamus laboriosam hic commodi id optio numquam asperiores, necessitatibus vitae sit ad amet dicta, expedita aspernatur veniam eius reprehenderit perferendis vero, quae! Sint, quidem incidunt dolores distinctio voluptatibus animi, perspiciatis, consequuntur cum quae quas, magni vero. Esse quidem labore adipisci ducimus sint excepturi quam molestias, autem doloribus magnam nobis corrupti dignissimos assumenda at ratione, soluta minus debitis magni deleniti ipsa. Autem natus vel, quas aperiam corrupti, delectus. Fuga itaque reprehenderit non aliquid consequuntur ipsa reiciendis, quam repellat, molestiae? Id, sint. Asperiores, cupiditate consectetur, veniam tenetur voluptate itaque doloribus voluptates porro sequi quas ex optio similique distinctio eum molestiae rerum quis facere explicabo eos, qui, sint quasi culpa molestias. Veniam commodi ipsum eum velit consequatur fuga quasi aliquam tenetur numquam hic quisquam et totam atque est debitis officiis eos accusantium odit minima deleniti, sunt molestiae consectetur itaque! Ipsum totam corrupti, voluptas deleniti facilis provident doloribus dolore autem atque tempore alias, harum ex omnis quisquam nobis veritatis earum eligendi, similique corporis dignissimos. Reiciendis earum vitae eligendi autem. Necessitatibus laudantium vero dicta dolorem debitis ratione. Deleniti quam quisquam totam, enim placeat fuga, possimus ratione architecto magnam ducimus, quis excepturi nesciunt molestiae magni sapiente non, cumque est repellendus nam rerum? Vero sequi nisi odit voluptate consequatur cupiditate vel praesentium adipisci harum, nesciunt placeat voluptates! Velit cupiditate numquam sit nulla accusamus nesciunt nemo voluptatem est omnis consequuntur nobis quas aut sed a, quos natus ducimus consectetur deleniti. Neque obcaecati adipisci ratione velit placeat consequatur perferendis delectus quidem cum alias similique minus magni quo quos libero dolores est doloremque dignissimos iste, nobis fuga veniam, quia. Excepturi, animi, recusandae sapiente delectus sint similique enim inventore culpa ducimus et ea optio quas corrupti natus voluptatibus tempore voluptates neque, placeat, ad aliquam! Laudantium aspernatur minus placeat. Facere maxime modi eligendi doloremque est deleniti placeat optio consequatur incidunt, reiciendis voluptatem distinctio cum suscipit, maiores officiis quod, voluptas accusantium vitae non provident aut consectetur in. Reiciendis, dicta. Ea vero earum illo, debitis saepe, iusto expedita officiis corporis fuga. Vitae velit nemo, eligendi delectus provident minima temporibus reiciendis quae vero sed, et, consequatur doloremque odio voluptates exercitationem commodi maiores maxime sequi ipsum at! Maiores, totam id fugit harum quos facere, consequuntur earum a ex doloremque sed, delectus illum ipsa at, consequatur inventore nostrum est. Mollitia blanditiis incidunt autem eligendi accusantium recusandae a magni veritatis repellendus dolores, doloremque hic. Minima nihil enim esse molestias eos cupiditate praesentium inventore reprehenderit sunt, aut deleniti natus laboriosam recusandae iusto exercitationem magni eligendi tempore ab doloribus, deserunt at, nostrum amet vitae. Quas totam a dolorum nesciunt vero delectus libero velit ex voluptatum praesentium repellat possimus officia tenetur facilis, maxime doloribus, nostrum unde? Corporis quas cumque quis ab, nesciunt sed, neque officia nam quidem? Laboriosam doloremque aliquid, corrupti eaque tenetur, repellendus cupiditate rerum distinctio corporis a sit officia? Aut explicabo nostrum laudantium officiis, incidunt iusto! Beatae excepturi dolores cumque est quidem maxime itaque quas, delectus tempore, expedita, dicta recusandae, sequi. Earum molestias dolore iste soluta, vel! Itaque nisi iure, explicabo voluptatem dolore vitae fugit animi vero earum. Neque, excepturi, autem rerum molestiae, laborum adipisci perferendis tempora nisi, deleniti dolorum atque natus labore odit. Tenetur exercitationem suscipit nemo reiciendis, rem facere, laborum quasi laudantium explicabo blanditiis quia doloribus eos voluptate? Vitae voluptatum accusamus voluptatem vel praesentium dolorem, cum, odio ratione laboriosam, asperiores necessitatibus quas ut minus quos maiores omnis corrupti quis, nam ad molestiae assumenda. Nobis nemo exercitationem, veniam quia fugiat quo voluptatem suscipit ad tempora inventore, incidunt porro atque, voluptatum at provident molestias perferendis alias impedit. Repellendus dolor earum iusto officia voluptatem quod facere incidunt minima, explicabo, amet hic, neque veritatis laborum cum distinctio atque eius pariatur et aut, doloremque quos quidem odio laboriosam tempora vel. Aperiam sequi corrupti ab aliquam assumenda officia, earum accusamus ratione corporis cumque, sint veniam error alias eum voluptatem tempore obcaecati sapiente veritatis dolore autem nemo facere molestiae. Minus animi pariatur ab sint eius dicta ad quas, minima sapiente incidunt sequi quae quisquam impedit et praesentium id alias? Distinctio sequi, aperiam animi eveniet, voluptate aut in est maiores nobis, id optio ullam possimus consectetur rerum cumque et, tempore itaque. Corporis nesciunt delectus ullam provident fugit quo. Consequuntur cumque recusandae, reprehenderit voluptate officia inventore tempora numquam ducimus velit, laboriosam eius tempore a dolorem neque, fugiat veritatis delectus sapiente libero. Nesciunt saepe accusantium, natus maxime fuga dolorum, perspiciatis, minus et tempore ipsum cum provident, doloribus ut amet? Iure distinctio, minus velit, doloremque repellendus accusamus iste aut eius dolorem sapiente expedita asperiores. Id voluptatibus quos minima sit ab? Molestias cupiditate modi dignissimos voluptate odit, assumenda minus amet sed quas dicta provident at architecto, ipsa distinctio aliquid eveniet! Laudantium vero voluptatibus, impedit, reiciendis non, amet porro atque architecto deleniti doloribus vel et ut laboriosam facere vitae tempore perferendis eos rerum eum. Ex veniam accusamus voluptates ipsa et odio quam, modi, labore explicabo minus totam ipsum! Aperiam, libero, mollitia ipsam a ullam at necessitatibus minus laborum! Doloremque autem beatae vero quasi porro a natus repellendus laborum nostrum hic cupiditate iste quos saepe fuga quas quo inventore nam, voluptas accusamus corrupti aliquam. Quod, perspiciatis. Magni voluptas a in eos officia facilis ipsum minus ea enim rem doloribus officiis consequatur reiciendis consectetur, velit cum nemo eligendi dolorem ratione temporibus corrupti inventore id exercitationem cumque? Dignissimos earum minima repudiandae eveniet aliquid, tempore rerum eaque laborum officia, repellat asperiores natus atque vitae maxime quaerat neque. Facere quo consequatur quaerat, tempore perferendis consectetur laudantium accusantium et iusto, cumque amet eveniet perspiciatis error dolorum harum ipsum voluptates illo rem necessitatibus maiores sit? Omnis a eveniet cumque ad perspiciatis, fugit, necessitatibus harum illo sit quae voluptatem, amet natus laudantium dolorem quaerat placeat corporis quam minus eligendi unde et assumenda rerum. Quae itaque nulla ipsam consequuntur obcaecati debitis eveniet recusandae. In eaque harum temporibus totam veniam culpa necessitatibus labore soluta rerum exercitationem! Nihil ducimus minima enim voluptates tenetur, quasi doloribus debitis vel laborum aut eos, eius fugiat facere earum, ipsum optio voluptate quis numquam, nostrum doloremque officiis. Pariatur quia amet impedit, possimus necessitatibus id voluptate. Quo aut, adipisci ut dolor quae earum rem aperiam dignissimos asperiores necessitatibus consequatur ea quaerat eaque alias labore animi quis pariatur, enim, repellat iusto voluptatem. Eaque sed atque deleniti laboriosam quaerat nam illo, quis autem sunt officiis alias inventore expedita dolores omnis minus quisquam perspiciatis? Cum laudantium cupiditate distinctio ducimus, officiis exercitationem, impedit odit optio nihil, minima ad officia mollitia, illum corrupti aspernatur quod odio veniam aperiam eum harum pariatur facilis culpa eos. Quod, inventore? Odio consectetur, autem veniam repellat temporibus? Aliquid debitis cum provident distinctio, repudiandae maiores fugiat eum et, quam, odit quaerat! Eveniet quod quisquam repellat, pariatur repellendus consequuntur culpa, doloremque distinctio facere assumenda impedit velit, ipsum possimus. Facere placeat corporis magnam fugiat harum, sunt officiis quasi obcaecati adipisci, explicabo molestias similique itaque in voluptates sint quia minus aspernatur ullam tempore dignissimos natus officia repellendus repudiandae. Inventore consequatur velit maxime laudantium rem ut, ab corrupti voluptas ipsum! Sunt enim vel ex quasi et minima cumque aliquid quisquam ab? Id, suscipit, doloremque optio hic consectetur perspiciatis nemo assumenda, rem vitae tempore recusandae nesciunt saepe, alias unde. Hic sint beatae consectetur architecto nulla vitae incidunt ullam quas ex fugiat officia sunt, repellendus harum voluptate ratione culpa vero omnis alias corrupti eveniet, velit voluptas accusamus eum dolores, doloribus! Harum fugiat quae cum distinctio aut qui eum tempora debitis obcaecati adipisci eos corrupti id voluptatem nihil numquam sed, at temporibus iusto a ex. Reprehenderit consequatur quam molestias ad. Quia dolor sint dicta optio nostrum fugiat, officiis molestias nobis officia laborum dolores cumque temporibus rerum iste? Vel corrupti, eius ducimus.</p>

</div>

<button onclick="scrollWindow()" style="position:fixed;left:50px;bottom:50px;background:red;color:white">Scroll Window</button>

<script>

function scrollWindow(){

window.scrollBy(0,20);

}

/\*function scrollWindow(){

window.scrollBy(100,0);

}

/\*function scrollWindow(){

window.scrollTo(0,0);

}\*/

</script>

</body>

</html>

[Previous](https://www.yahubaba.com/javascript/js-resize)[Next](https://www.yahubaba.com/javascript/js-location-object)

## **JS Location Object**

The Window Location Object

The **location object** contains information about the current URL.

The **location object** is a property of the **window object**.

The **location object** is accessed with:

window.location or just location

Location Object Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| hash | Sets or returns the anchor part (#) of a URL |
| host | Sets or returns the hostname and port number of a URL |
| hostname | Sets or returns the hostname of a URL |
| href | Sets or returns the entire URL |
| origin | Returns the protocol, hostname and port number of a URL |
| pathname | Sets or returns the path name of a URL |
| port | Sets or returns the port number of a URL |
| protocol | Sets or returns the protocol of a URL |
| search | Sets or returns the querystring part of a URL |

Location Object Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| assign() | Loads a new document |
| reload() | Reloads the current document |
| replace() | Replaces the current document with a new one |

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript</title>

</head>

<body>

<h1>JavaScript : Location Object</h1>

<button onclick="newFunction()">Click</button>

<script>

/\* JavaScript Location Object\*/

//location.href = "http://www.lbstibaba.net";

//function newFunction(){

// location.href = "http://www.lbstibaba.net";

//}

/\* JavaScript Reload Function\*/

/\*function newFunction(){

location.reload();

}\*/

/\* JavaScript Assign Function\*/

/\*function newFunction(){

location.assign("https://www.google.com");

}\*/

/\* JavaScript Replace Function\*/

function newFunction(){

location.replace("https://www.google.com");

}

</script>

</body>

</html>

JS History Object

The **history object** contains the URLs visited by the user (in the browser window).

The **history object** is a property of the **window object**.

The **history object** is accessed with:

window.history or just history:

History Object Properties and Methods

|  |  |
| --- | --- |
| **Property/Method** | **Description** |
| back() | Loads the previous URL (page) in the history list |
| forward() | Loads the next URL (page) in the history list |
| go() | Loads a specific URL (page) from the history list |
| length | Returns the number of URLs (pages) in the history list |

<!DOCTYPE html>

<html lang="en">

<head>

<title>JavaScript</title>

</head>

<body>

<h1>JavaScript: History Object</h1>

<!-- <button onclick="backFunction()">Back</button>

<button onclick="forwardFunction()">Forword</button> -->

<button onclick="goFunction()">Go Button</button>

<script>

/\* JavaScript Back Function\*/

function backFunction(){ history.back(); }

/\* JavaScript Forward Function\*/

function forwardFunction(){ history.forward(); }

/\* JavaScript Go Function\*/

function goFunction(){ history.go(1); }

</script>

</body>

</html>

[Previous](https://www.yahubaba.com/javascript/js-location-object)[Next](https://www.yahubaba.com/javascript/httpswwwyoutubecomwatchvdenkf4u9y5o)