JavaScript is the most popular and widely used client-side scripting language. Client-side scripting refers to scripts that run within your web browser. JavaScript is designed to add interactivity and dynamic effects to the web pages by manipulating the content returned from a web server.

**Adding JavaScript to Your Web Pages**

There are typically three ways to add JavaScript to a web page:

* Embedding the JavaScript code between a pair of <script> and </script> tag.
* Creating an external JavaScript file with the .js extension and then load it within the page through the src attribute of the <script> tag.
* Placing the JavaScript code directly inside an HTML tag using the special tag attributes such as onclick, onmouseover, onkeypress, onload, etc.

## Embedding the JavaScript Code

You can embed the JavaScript code directly within your web pages by placing it between the <script> and </script> tags. The <script> tag indicates the browser that the contained statements are to be interpreted as executable script and not HTML. Here's an example:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Embedding JavaScript</title>

</head>

<body>

<script>

var greet = "Hello World!";

document.write(greet); // Prints: Hello World!

</script>

</body>

</html>

## Calling an External JavaScript File

You can also place your JavaScript code into a separate file with a .js extension, and then call that file in your document through the src attribute of the <script> tag, like this:

<script src="js/hello.js"></script>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Including External JavaScript File</title>

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

</head>

<body>

<button type="button" onclick="myFunction()">Try it</button>

</body>

</html>

Hello.js

function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";

## Placing the JavaScript Code Inline

You can also place JavaScript code inline by inserting it directly inside the HTML tag using the special tag attributes such as onclick, onmouseover, onkeypress, onload, etc.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Inlining JavaScript</title>

</head>

<body>

<button onclick="alert('Hello World!')">Click Me</button>

</body>

</html>

## Understanding the JavaScript Syntax

The syntax of JavaScript is the set of rules that define a correctly structured JavaScript program.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Example of JavaScript Statements</title>

</head>

<body>

<script>

var x = 5;

var y = 10;

var sum = x + y;

document.write(sum); // Prints variable value

</script>

</body>

</html>

# JavaScript Variables

## What is Variable?

Variables are fundamental to all programming languages. Variables are used to store data, like string of text, numbers, etc. The data or value stored in the variables can be set, updated, and retrieved whenever needed. In general, variables are symbolic names for values.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Creating Variables in JavaScript</title>

</head>

<body>

<script>

// Creating variables

var name = "Peter Parker";

var age = 21;

var isMarried = false;

// Printing variable values

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

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

document.write(isMarried);

</script>

</body>

</html>

## Declaring Multiple Variables at Once

## <!DOCTYPE html>

## <html lang="en">

## <head>

## <meta charset="utf-8">

## <title>Declaring Multiple Variables in JavaScript</title>

## </head>

## <body>

## <script>

## // Declaring multiple Variables

## var name = "Peter Parker", age = 21, isMarried = false;

## 

## // Printing variable values

## document.write(name + "<br>");

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

## document.write(isMarried);

## </script>

## </body>

## </html>

**Naming Conventions for JavaScript Variables**

These are the following rules for naming a JavaScript variable:

* A variable name must start with a letter, underscore (\_), or dollar sign ($).
* A variable name cannot start with a number.
* A variable name can only contain alpha-numeric characters (A-z, 0-9) and underscores.
* A variable name cannot contain spaces.
* A variable name cannot be a JavaScript keyword or a [JavaScript reserved word](https://www.tutorialrepublic.com/javascript-reference/javascript-reserved-keywords.php).

## JavaScript Display

JavaScript can "display" data in different ways:

* Writing into an HTML element, using innerHTML.
* Writing into the HTML output using document.write().
* Writing into an alert box, using window.alert().
* Writing into the browser console, using console.log().

## Using innerHTML

o access an HTML element, JavaScript can use the document.getElementById(id) method.

The id attribute defines the HTML element. The innerHTML property defines the HTML content:

## <!DOCTYPE html>

## <html>

## <body>

## <h2>My First Web Page</h2>

## <p>My First Paragraph.</p>

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

## <script>

## document.getElementById("demo").innerHTML = 5 + 6;

## </script>

## </body>

## </html>

Example2:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Writing into an HTML Element with JavaScript</title>

</head>

<body>

<p id="greet"></p>

<p id="result"></p>

<script>

// Writing text string inside an element

document.getElementById("greet").innerHTML = "Hello World!";

// Writing a variable value inside an element

var x = 10;

var y = 20;

var sum = x + y;

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

</script>

</body>

</html>

## Using document.write()

<!DOCTYPE html>

<html>

<body>

<h2>My First Web Page</h2>

<p>My first paragraph.</p>

<p>Never call document.write after the document has finished loading.

It will overwrite the whole document.</p>

<script>

document.write(5 + 6);

</script>

</body>

</html>

Example2:

<!DOCTYPE html>

<html>

<body>

<h2>My First Web Page</h2>

<p>My first paragraph.</p>

<button type="button" onclick="document.write(5 + 6)">Try it</button>

</body>

</html>

## Using window.alert()

<!DOCTYPE html>

<html>

<body>

<h2>My First Web Page</h2>

<p>My first paragraph.</p>

<script>

window.alert(5 + 6);

</script>

</body>

</html>

# JavaScript Data Types

JavaScript variables can hold many **data types**: numbers, strings, objects and more:

var length = 16;                               // Number  
var lastName = "Johnson";                      // String  
var x = {firstName:"John", lastName:"Doe"};    // Object

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript</h2>

<p>JavaScript evaluates expressions from left to right. Different sequences can produce different results:</p>

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

<script>

var x = "Volvo" + 16 + 4;

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

</script>

</body>

</html>

## JavaScript Strings

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Strings</h2>

<p>Strings are written with quotes. You can use single or double quotes:</p>

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

<script>

var carName1 = "Volvo XC60";

var carName2 = 'Volvo XC60';

document.getElementById("demo").innerHTML =

carName1 + "<br>" +

carName2;

</script>

</body>

</html>

Example:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Strings</h2>

<p>You can use quotes inside a string, as long as they don't match the quotes surrounding the string:</p>

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

<script>

var answer1 = "It's alright";

var answer2 = "He is called 'Johnny'";

var answer3 = 'He is called "Johnny"';

document.getElementById("demo").innerHTML =

answer1 + "<br>" +

answer2 + "<br>" +

answer3;

</script>

</body>

</html>

Example:  
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Including Quotes inside the JavaScript String</title>

</head>

<body>

<script>

// Creating variables

var a = "Let's have a cup of coffee.";

var b = 'He said "Hello" and left.';

var c = 'We\'ll never give up.';

// Printing variable values

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

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

document.write(c);

</script>

</body>

</html>

## JavaScript Numbers

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Numbers</h2>

<p>Numbers can be written with, or without decimals:</p>

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

<script>

var x1 = 34.00;

var x2 = 34;

var x3 = 3.14;

document.getElementById("demo").innerHTML =

x1 + "<br>" + x2 + "<br>" + x3;

</script>

</body>

</html>

## JavaScript Booleans

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Booleans</h2>

<p>Booleans can have two values: true or false:</p>

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

<script>

var x = 5;

var y = 5;

var z = 6;

document.getElementById("demo").innerHTML =

(x == y) + "<br>" + (x == z);

</script>

</body>

</html>

## JavaScript Arrays

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Arrays</h2>

<p>Array indexes are zero-based, which means the first item is [0].</p>

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

<script>

var cars = ["Saab","Volvo","BMW"];

document.getElementById("demo").innerHTML = cars[0];

</script>

</body>

</html>

## JavaScript Objects

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Objects</h2>

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

<script>

var person = {

firstName : "John",

lastName : "Doe",

age : 50,

eyeColor : "blue"

};

document.getElementById("demo").innerHTML =

person.firstName + " is " + person.age + " years old.";

</script>

</body>

</html>

## The typeof Operator

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript typeof</h2>

<p>The typeof operator returns the type of a variable or an expression.</p>

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

<script>

document.getElementById("demo").innerHTML =

typeof "" + "<br>" +

typeof "John" + "<br>" +

typeof "John Doe";

</script>

</body>

</html>

Example:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript typeof Operator</title>

</head>

<body>

<script>

// Numbers

document.write(typeof 15 + "<br>"); // Prints: "number"

document.write(typeof 42.7 + "<br>"); // Prints: "number"

document.write(typeof 2.5e-4 + "<br>"); // Prints: "number"

document.write(typeof Infinity + "<br>"); // Prints: "number"

document.write(typeof NaN + "<br>"); // Prints: "number". Despite being "Not-A-Number"

// Strings

document.write(typeof '' + "<br>"); // Prints: "string"

document.write(typeof 'hello' + "<br>"); // Prints: "string"

document.write(typeof '12' + "<br>"); // Prints: "string". Number within quotes is document.write(typeof string

// Booleans

document.write(typeof true + "<br>"); // Prints: "boolean"

document.write(typeof false + "<br>"); // Prints: "boolean"

// Undefined

document.write(typeof undefined + "<br>"); // Prints: "undefined"

document.write(typeof undeclaredVariable + "<br>"); // Prints: "undefined"

// Null

document.write(typeof Null + "<br>"); // Prints: "object"

// Objects

document.write(typeof {name: "John", age: 18} + "<br>"); // Prints: "object"

// Arrays

document.write(typeof [1, 2, 4] + "<br>"); // Prints: "object"

// Functions

document.write(typeof function(){}); // Prints: "function"

</script>

</body>

</html>

## The Undefined Data Type

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Undefined Data Type</title>

</head>

<body>

<script>

// Creating variables

var a;

var b = "Hello World!"

// Printing variable values

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

document.write(b);

</script>

</body>

</html>

## The Null Data Type

This is another special data type that can have only one value-the null value. A null value means that there is no value. It is not equivalent to an empty string ("") or 0, it is simply nothing.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Null Data Type</title>

</head>

<body>

<script>

var a = null;

document.write(a + "<br>"); // Print: null

var b = "Hello World!"

document.write(b + "<br>"); // Print: Hello World!

b = null;

document.write(b) // Print: null

</script>

</body>

</html>

Ques:

person = {"name": "Clark", "surname": "Kent", "age": "36"};

car = {

"modal": "BMW X3",

"color": "white",

"doors": 5

## The Function Data Type

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Function Data Type</title>

</head>

<body>

<script>

var greeting = function(){

return "Hello World!";

}

// Check the type of greeting variable

document.write(typeof greeting) // Output: function

document.write("<br>");

document.write(greeting()); // Output: Hello World!

</script>

</body>

</html>

# JavaScript Operators

For example, the addition (+) symbol is an operator that tells JavaScript engine to add two variables or values, while the equal-to (==), greater-than (>) or less-than (<) symbols are the operators that tells JavaScript engine to compare two variables or values, and so on.

**JavaScript Arithmetic Operators**

The arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc. Here's a complete list of JavaScript's arithmetic operators:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Result** |
| + | Addition | $x + $y | Sum of $x and $y |
| - | Subtraction | $x - $y | Difference of $x and $y. |
| \* | Multiplication | $x \* $y | Product of $x and $y. |
| / | Division | $x / $y | Quotient of $x and $y |
| % | Modulus | $x % $y | Remainder of $x divided by $y |

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Arithmetic Operators</title>

</head>

<body>

<script>

var x = 10;

var y = 4;

document.write(x + y); // Prints: 14

document.write("<br>");

document.write(x - y); // Prints: 6

document.write("<br>");

document.write(x \* y); // Prints: 40

document.write("<br>");

document.write(x / y); // Prints: 2.5

document.write("<br>");

document.write(x % y); // Prints: 2

</script>

</body>

</html>

**JavaScript Assignment Operators**

The assignment operators are used to assign values to variables.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Is The Same As** |
| = | Assign | x = y | x = y |
| += | Add and assign | x += $ | x = x + y |
| -= | Subtract and assign | x -= y | x = x - y |
| \*= | Multiply and assign | x \*= y | x = x \* y |
| /= | Divide and assign quotient | x /= y | x = x / y |
| %= | Divide and assign modulus | x %= y | x = x % y |

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Assignment Operators</title>

</head>

<body>

<script>

var x; // Declaring Variable

x = 10;

document.write(x + "<br>"); // Prints: 10

x = 20;

x += 30;

document.write(x + "<br>"); // Prints: 50

x = 50;

x -= 20;

document.write(x + "<br>"); // Prints: 30

x = 5;

x \*= 25;

document.write(x + "<br>"); // Prints: 125

x = 50;

x /= 10;

document.write(x + "<br>"); // Prints: 5

x = 100;

x %= 15;

document.write(x); // Prints: 10

</script>

</body>

</html>

**JavaScript String Operators**

There are two operators which can also used be for strings.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Result** |
| + | Concatenation | str1 + str2 | Concatenation of str1 and str2 |
| += | Concatenation assignment | str1 += str2  str1=str1+str2 | Appends the str2 to the str1 |

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript String Operators</title>

</head>

<body>

<script>

var str1 = "Hello";

var str2 = " World!";

document.write(str1 + str2 + "<br>"); // Outputs: Hello World!

str1 += str2;

document.write(str1); // Outputs: Hello World!

</script>

</body>

</html>

**JavaScript Incrementing and Decrementing Operators**

The increment/decrement operators are used to increment/decrement a variable's value.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Effect** |
| ++x | Pre-increment | Increments x by one, then returns x |
| x++ | Post-increment | Returns x, then increments x by one |
| --x | Pre-decrement | Decrements x by one, then returns x |
| x-- | Post-decrement | Returns x, then decrements x by one |

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Incrementing and Decrementing Operators</title>

</head>

<body>

<script>

var x; // Declaring Variable

x = 10;

document.write(++x); // Prints: 11

document.write("<p>" + x + "</p>"); // Prints: 11

x = 10;

document.write(x++); // Prints: 10

document.write("<p>" + x + "</p>"); // Prints: 11

x = 10;

document.write(--x); // Prints: 9

document.write("<p>" + x + "</p>"); // Prints: 9

x = 10;

document.write(x--); // Prints: 10

document.write("<p>" + x + "</p>"); // Prints: 9

</script>

</body>

</html>

**JavaScript Logical Operators**

The logical operators are typically used to combine conditional statements.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Result** |
| && | And | x && y | True if both x and y are true |
| || | Or | x || y | True if either x or y is true |
| ! | Not | !x | True if x is not true |

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Logical Operators</title>

</head>

<body>

<script>

var year = 2018;

// Leap years are divisible by 400 or by 4 but not 100

if((year % 400 == 0) || ((year % 100 != 0) && (year % 4 == 0))){

document.write(year + " is a leap year.");

} else{

document.write(year + " is not a leap year.");

}

</script>

</body>

</html>

## JavaScript Comparison Operators

The comparison operators are used to compare two values in a Boolean fashion.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Result** |
| == | Equal | x == y | True if x is equal to y |
| === | Identical | x === y | True if x is equal to y, and they are of the same [type](https://www.tutorialrepublic.com/javascript-tutorial/javascript-data-types.php) |
| != | Not equal | x != y | True if x is not equal to y |
| !== | Not identical | x !== y | True if x is not equal to y, or they are not of the same type |
| < | Less than | x < y | True if x is less than y |
| > | Greater than | x > y | True if x is greater than y |
| >= | Greater than or equal to | x >= y | True if x is greater than or equal to y |
| <= | Less than or equal to | x <= y | True if x is less than or equal to y |

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Comparison Operators</title>

</head>

<body>

<script>

var x = 25;

var y = 35;

var z = "25";

document.write(x == z); // Prints: true

document.write("<br>");

document.write(x === z); // Prints: false

document.write("<br>");

document.write(x != y); // Prints: true

document.write("<br>");

document.write(x !== z); // Prints: true

document.write("<br>");

document.write(x < y); // Prints: true

document.write("<br>");

document.write(x > y); // Prints: false

document.write("<br>");

document.write(x <= y); // Prints: true

document.write("<br>");

document.write(x >= y); // Prints: false

</script>

</body>

</html>

# JavaScript Functions

**Defining and Calling a Function**

The declaration of a function start with the function keyword, followed by the name of the function you want to create, followed by parentheses i.e. () and finally place your function's code between curly brackets {}. Here's the basic syntax for declaring a function:

function functionName() {  
    // Code to be executed  
}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Define and Call a Function</title>

</head>

<body>

<script>

// Defining function

function sayHello() {

document.write("Hello, welcome to this website!");

}

// Calling function

sayHello(); // Prints: Hello, welcome to this website!

</script>

</body>

</html>

**Adding Parameters to Functions**

function functionName(*parameter1*, *parameter2*, *parameter3*) {  
    // Code to be executed  
}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Add Parameters to a Function</title>

</head>

<body>

<script>

// Defining function

function displaySum(num1, num2) {

var total = num1 + num2;

document.write(total);

}

// Calling function

displaySum(6, 20); // Prints: 26

document.write("<br>");

displaySum(-5, 17); // Prints: 12

</script>

</body>

</html>

Example2:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Pass Arguments to a Function</title>

</head>

<body>

<script>

// Defining function

function showFullname(firstName, lastName) {

document.write(firstName + " " + lastName);

}

// Calling function

showFullname("Clark", "Kent"); // Prints: Clark Kent

document.write("<br>");

showFullname("John"); // Prints: John undefined

</script>

</body>

</html>

# JavaScript Events

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

With single quotes:

<*element* *event*=**'*some JavaScript*'**>

With double quotes:

<*element* *event*=**"*some JavaScript*"**>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Attaching Event Handlers Inline</title>

</head>

<body>

<button type="button" onclick="alert('Hello World!')">Click Me</button>

</body>

</html>

**With Functions:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Attaching Event Handlers in External File</title>

</head>

<body>

<button type="button" id="myBtn">Click Me</button>

<script>

function sayHello(){

alert('Hello World!');

}

document.getElementById("myBtn").onclick = sayHello;

</script>

</body>

</html>

Example:

<!DOCTYPE html>

<html>

<body>

<p>Click the button to display the date.</p>

<button onclick="displayDate()">The time is?</button>

<script>

function displayDate() {

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

}

</script>

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

</body>

</html>

**Mouse Events**

The Click Event (onclick)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Handling the Click Event</title>

</head>

<body>

<button type="button" onclick="alert('You have clicked a button!');">Click Me</button>

<a href="#" onclick="alert('You have clicked a link!');">Click Me</a>

</body>

</html>

**The Mouseover Event (onmouseover)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Handling the Mouseover Event</title>

</head>

<body>

<button type="button" onmouseover="alert('You have placed mouse pointer over a button!');">Place Mouse Over Me</button>

<a href="#" onmouseover="alert('You have placed mouse pointer over a link!');">Place Mouse Over Me</a>

</body>

</html>

**The Mouseout Event (onmouseout)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Handling the Mouseout Event</title>

</head>

<body>

<button type="button" onmouseout="alert('You have moved out of the button!');">Place Mouse Inside Me and Move Out</button>

<a href="#" onmouseout="alert('You have moved out of the link!');">Place Mouse Inside Me and Move Out</a>

</body>

</html>

**Keyboard Events**

**The Keydown Event (onkeydown)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Handling the Keydown Event</title>

</head>

<body>

<input type="text" onkeydown="alert('You have pressed a key inside text input!')">

<hr>

<textarea cols="30" onkeydown="alert('You have pressed a key inside textarea!')"></textarea>

<p><strong>Note:</strong> Try to enter some text inside input box and textarea.</p>

</body>

</html>

**The Keyup Event (onkeyup)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Handling the Keyup Event</title>

</head>

<body>

<input type="text" onkeyup="alert('You have released a key inside text input!')">

<hr>

<textarea cols="30" onkeyup="alert('You have released a key inside textarea!')"></textarea>

<p><strong>Note:</strong> Try to enter some text inside input box and textarea.</p>

</body>

</html>

**The Keypress Event (onkeypress)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Handling the Keypress Event</title>

</head>

<body>

<input type="text" onkeypress="alert('You have pressed a key inside text input!')">

<hr>

<textarea cols="30" onkeypress="alert('You have pressed a key inside textarea!')"></textarea>

<p><strong>Note:</strong> Try to enter some text inside input box and textarea.</p>

</body>

</html>

**Form Events**

**The Focus Event (onfocus)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Handling the Focus Event</title>

</head>

<body>

<script>

function highlightInput(elm){

elm.style.background = "yellow";

}

</script>

<input type="text" onfocus="highlightInput(this)">

<button type="button">Button</button>

</body>

</html>

**The Blur Event (onblur)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Handling the Blur Event</title>

</head>

<body>

<input type="text" onblur="alert('Text input loses focus!')">

<button type="button">Submit</button>

<p><strong>Note:</strong> First click inside the text input box then click outside to see how it works.</p>

</body>

</html>

**The Change Event (onchange)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Handling the Change Event</title>

</head>

<body>

<select onchange="alert('You have changed the selection!');">

<option>Select</option>

<option>Male</option>

<option>Female</option>

</select>

<p><strong>Note:</strong> Select any option in select box to see how it works.</p>

</body>

</html>

**The Submit Event (onsubmit)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Handling the Submit Event</title>

</head>

<body>

<form action="/examples/html/action.php" method="post" onsubmit="alert('Form data will be submitted to the server!');">

<label>First Name:</label>

<input type="text" name="first-name" required>

<input type="submit" value="Submit">

</form>

</body>

</html>

**Document/Window Events**

**The Load Event (onload)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Handling the Load Event</title>

</head>

<body onload="window.alert('Page is loaded successfully!');">

<h1>This is a heading</h1>

<p>This is paragraph of text.</p>

</body>

</html>

**The Unload Event (onunload)**

The unload event occurs when a user leaves the current web page.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Handling the Unload Event</title>

</head>

<body onunload="alert('Are you sure you want to leave this page?');">

<h1>This is a heading</h1>

<p>This is paragraph of text.</p>

<p><strong>Note:</strong> This example may not work. The unload event is not supported properly in most of the browsers.</p>

</body>

</html>

# JavaScript Strings

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Creating Strings in JavaScript</title>

</head>

<body>

<script>

// Creating variables

var myString = 'Hello World!'; // Single quoted string

var myString = "Hello World!"; // Double quoted string

// Printing variable values

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

document.write(myString);

</script>

</body>

</html>

Example2:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Using Quotes inside JavaScript Strings</title>

</head>

<body>

<script>

// Creating variables

var str1 = "it's okay";

var str2 = 'He said "Goodbye"';

var str3 = "She replied 'Calm down, please'";

// Printing variable values

document.write(str1 + "<br>");

document.write(str2 + "<br>");

document.write(str3);

</script>

</body>

</html>

Example3:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Escaping Quotes inside JavaScript Strings</title>

</head>

<body>

<script>

// Creating variables

var str1 = 'it\'s okay';

var str2 = "He said \"Goodbye\"";

var str3 = 'She replied \'Calm down, please\'';

// Printing variable values

document.write(str1 + "<br>");

document.write(str2 + "<br>");

document.write(str3);

</script>

</body>

</html>

**JavaScript Escape Sequences**

Escape sequences are also useful for situations where you want to use characters that can't be typed using a keyboard. Here are some other most commonly used escape sequences.

* \n is replaced by the newline character
* \t is replaced by the tab character
* \r is replaced by the carriage-return character
* \b is replaced by the backspace character
* \\ is replaced by a single backslash (\)

**Performing Operations on Strings**

**Getting the Length of a String**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Get String Length in JavaScript</title>

</head>

<body>

<script>

var str1 = "This is a paragraph of text.";

document.write(str1.length + "<br>"); // Prints 28

var str2 = "This is a \n paragraph of text.";

document.write(str2.length); // Prints 30, because \n is only one character

</script>

</body>

</html>

**Finding a String Inside Another String**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Find the Position of Substring within a String</title>

</head>

<body>

<script>

var str = "If the facts don't fit the theory, change the facts.";

var pos = str.indexOf("facts");

document.write(pos); // 0utputs: 7

</script>

</body>

</html>

Similarly, you can use the lastIndexOf() method to get the index or position of the last occurrence of the specified string within a string, like this:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Find the Position of Substring within a String</title>

</head>

<body>

<script>

var str = "If the facts don't fit the theory, change the facts.";

var pos = str.lastIndexOf("facts");

document.write(pos); // 0utputs: 46

</script>

</body>

</html>

Both the indexOf(), and the lastIndexOf() methods return -1 if the substring is not found.

**Searching for a Pattern Inside a String**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Search Text or Pattern inside a String</title>

</head>

<body>

<script>

var str = "Color red looks brighter than color blue.";

// Case sensitive search

var pos1 = str.search("color");

document.write(pos1 + "<br>"); // 0utputs: 30

// Case insensitive search using regexp

var pos2 = str.search(/color/i);

document.write(pos2); // 0utputs: 0

</script>

</body>

</html>

**Extracting a Substring from a String**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Slice Out a Portion of a String</title>

</head>

<body>

<script>

var str = "The quick brown fox jumps over the lazy dog.";

var subStr = str.slice(4, 15);

document.write(subStr); // Prints: quick brown

</script>

</body>

</html>

You can also specify negative values. The negative value is treated as strLength + startIndex, where strLength is the length of the string (i.e. str.length), for example, if startIndex is -5 it is treated as strLength - 5. If startIndex is greater than or equal to the length of the string, slice() method returns an empty string. Also, if optional endIndex is not specified or omitted, the slice() method extracts to the end of the string.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Slice Strings Using Negative Indexes</title>

</head>

<body>

<script>

var str = "The quick brown fox jumps over the lazy dog.";

document.write(str.slice(-28, -19) + "<br>"); // Prints: fox jumps

document.write(str.slice(31)); // Prints: the lazy dog.

</script>

</body>

</html>

You can also use the substring() method to extract a section of the given string based on start and end indexes, like str.substring(startIndex, endIndex). The substring() method is very similar to the slice() method, except few differences:

* If either argument is less than 0 or is NaN, it is treated as 0.
* If either argument is greater than str.length, it is treated as if it were str.length.
* If startIndex is greater than endIndex, then substring() will swap those two arguments; for example, str.substring(5, 0) == str.substring(0, 5).

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Extract substring from a String</title>

</head>

<body>

<script>

var str = "The quick brown fox jumps over the lazy dog.";

document.write(str.substring(4, 15) + "<br>"); // Prints: quick brown

document.write(str.substring(9, 0) + "<br>"); // Prints: The quick

document.write(str.substring(-28, -19) + "<br>"); // Prints nothing

document.write(str.substring(31)); // Prints: the lazy dog.

</script>

</body>

</html>

EX

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Slice Out a Portion of a String</title>

</head>

<body>

<script>

var str = "The quick brown fox jumps over the lazy dog.";

//var subStr = str.substr(9,0);

//var subStr1 = str.slice(-28,-9);

//var subStr2 = str.substring(9,0);

var subStr2 = str.substring(17,9);

// document.write(subStr); // Prints: quick brown

//document.write(subStr1);

document.write(subStr2);

</script>

</body>

</html>

### Extracting a Fixed Number of Characters from a String

JavaScript also provide the substr() method which is similar to slice() with a subtle difference, the second parameter specifies the number of characters to extract instead of ending index, like str.substr(startIndex, length). If length is 0 or a negative number, an empty string is returned.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Extract Fixed Number of Characters from a String</title>

</head>

<body>

<script>

var str = "The quick brown fox jumps over the lazy dog.";

document.write(str.substr(4, 15) + "<br>"); // Prints: quick brown fox

document.write(str.substr(-28, -19) + "<br>"); // Prints nothing

document.write(str.substr(-28, 9) + "<br>"); // Prints: fox jumps

document.write(str.substr(31)); // Prints: the lazy dog.

</script>

</body>

</html>

**Replacing the Contents of a String**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Replace Part of a String with another String</title>

</head>

<body>

<script>

var str = "Color red looks brighter than color blue.";

var result = str.replace("color", "paint");

document.write(result); // 0utputs: Color red looks brighter than paint blue.

</script>

</body>

</html>

By default, the replace() method replaces only the first match, and it is case-sensitive. To replace the substring within a string in a case-insensitive manner you can use a regular expression (regexp) with an i modifier, as shown in the example below:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Replace Part of a String with another String</title>

</head>

<body>

<script>

var str = "Color red looks brighter than color blue.";

var result = str.replace(/color/i, "paint");

document.write(result); // 0utputs: paint red looks brighter than color blue.

</script>

</body>

</html>

Similarly, to replace all the occurrences of a substring within a string in a case-insensitive manner you can use the g modifier along with the i modifier, like this:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Replace All Occurrences of a Substring in a String</title>

</head>

<body>

<script>

var str = "Color red looks brighter than color blue.";

var result = str.replace(/color/ig, "paint");

document.write(result); // 0utputs: paint red looks brighter than paint blue.

</script>

</body>

</html>

**Converting a String to Uppercase or Lowercase**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Convert a String to Uppercase Characters</title>

</head>

<body>

<script>

var str = "Hello World!";

var result = str.toUpperCase();

document.write(result); // Prints: HELLO WORLD!

</script>

</body>

</html>

Example2:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Convert a String to Lowercase Characters</title>

</head>

<body>

<script>

var str = "Hello World!";

var result = str.toLowerCase();

document.write(result); // Prints: hello world!

</script>

</body>

</html>

**Concatenating Two or More Strings**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Join Two or More Strings</title>

</head>

<body>

<script>

var hello = "Hello";

var world = "World";

var greet = hello + " " + world;

document.write(greet + "<br>"); // Prints: Hello World

var wish = "Happy";

wish += " New Year";

document.write(wish); // Prints: Happy New Year

</script>

</body>

</html>

**Accessing Individual Characters from a String**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Extract a Single Character from a String</title>

</head>

<body>

<script>

var str = "Hello World!";

document.write(str.charAt() + "<br>"); // Prints: H

document.write(str.charAt(6) + "<br>"); // Prints: W

document.write(str.charAt(30) + "<br>"); // Prints nothing

document.write(str.charAt(str.length - 1)); // Prints: !

</script>

</body>

</html>

The charCodeAt() Method

The charCodeAt() method returns the unicode of the character at a specified index in a string:

<!DOCTYPE html>

<html>

<body>

<p>The charCodeAt() method returns the unicode of the character at a given position in a string:</p>

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

<script>

var str = "HELLO WORLD";

document.getElementById("demo").innerHTML = str.charCodeAt(0);

</script>

</body>

</html>

String.trim()

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

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript String.trim()</h2>

<p>Click the button to alert the string with removed whitespace.</p>

<button onclick="myFunction()">Try it</button>

<p><strong>Note:</strong> The trim() method is not supported in Internet Explorer 8 and earlier versions.</p>

<script>

function myFunction() {

var str = " Hello World! ";

alert(str.trim());

}

</script>

</body>

</html>

Converting a String to an Array

<!DOCTYPE html>

<html>

<body>

<p>Click "Try it" to display the first array element, after a string split.</p>

<button onclick="myFunction()">Try it</button>

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

<script>

function myFunction() {

var str = "a,b,c,d,e,f";

var arr = str.split(",");

document.getElementById("demo").innerHTML = arr[0];

}

</script>

</body>

</html>

# JavaScript Numbers

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Numbers</title>

</head>

<body>

<script>

// Creating variables

var x = 2; // integer number

var y = 3.14; // floating-point number

var z = 0xff; // hexadecimal number

// Printing variable values

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

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

document.write(z);

</script>

</body>

</html>

Example2:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Representing Numbers in Exponential Notation</title>

</head>

<body>

<script>

// Creating variables

var x = 1. e574; // same as 1.5700

var y = 4.25e+6; // same as 4.25e6 or 4.250000

var z = 4.25e-6; // same as 0.00000425

// Printing variable values

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

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

document.write(z);

</script>

</body>

</html>

Example:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Numbers</h2>

<p>Integers (numbers without a period or exponent notation) are accurate up to 15 digits:</p>

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

<script>

var x = 999999999999999;

var y = 9999999999999999;

document.getElementById("demo").innerHTML = x + "<br>" + y;

</script>

</body>

</html>

**Operating on Numbers and Strings**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Adding Numbers and Strings</title>

</head>

<body>

<script>

// Creating variables

var x = 10;

var y = 20;

var z = "30";

// Adding a number with a number, the result will be sum of numbers

document.write(x + y); // 30

document.write("<br>");

// Adding a string with a string, the result will be string concatenation

document.write(z + z); // '3030'

document.write("<br>");

// Adding a number with a string, the result will be string concatenation

document.write(x + z); // '1030'

document.write("<br>");

// Adding a string with a number, the result will be string concatenation

document.write(z + x); // '3010'

document.write("<br>");

// Adding strings and numbers, the result will be string concatenation

document.write("The result is: " + x + y); // 'The result is: 1020'

document.write("<br>");

// Adding numbers and strings, calculation performed from left to right

document.write(x + y + z); // 'The result is: 3030'

</script>

</body>

</html>

Number Methods and Properties

**Parsing Integers from Strings**

parseInt() parses a string and returns a whole number. Spaces are allowed. Only the first number is returned:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Parse Integers from Strings</title>

</head>

<body>

<script>

document.write(parseInt("3.14") + "<br>"); // 3

document.write(parseInt("50px") + "<br>"); // 50

document.write(parseInt("12pt") + "<br>"); // 12

document.write(parseInt("0xFF", 16) + "<br>"); // 255

document.write(parseInt("20 years") + "<br>"); // 20

document.write(parseInt("Year 2048") + "<br>"); // NaN

document.write(parseInt("10 12 2018")); // 10

</script>

</body>

</html>

Similarly, you can use the parseFloat() method to parse floating-point number from a string. The parseFloat() method works the same way as the parseInt() method, except that it retrieves both integers and numbers with decimals.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Parse Floating Point Numbers from Strings</title>

</head>

<body>

<script>

document.write(parseFloat("3.14") + "<br>"); // 3.14

document.write(parseFloat("50px") + "<br>"); // 50

document.write(parseFloat("1.6em") + "<br>"); // 1.6

document.write(parseFloat("124.5 lbs") + "<br>"); // 124.5

document.write(parseFloat("weight 124.5 lbs") + "<br>"); // NaN

document.write(parseFloat("6.5 acres")); // 6.5

</script>

</body>

</html>

**Converting Numbers to Strings**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Convert Numbers to Strings</title>

</head>

<body>

<script>

var x = 10;

var y = x.toString();

document.write(y); // '10'

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

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

document.write((12).toString() + "<br>"); // '12'

document.write((15.6).toString() + "<br>"); // '15.6'

document.write((6).toString(2) + "<br>"); // '110'

document.write((255).toString(16)); // 'ff'

</script>

</body>

</html>

**Formatting Numbers in Exponential Notation**

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Number Methods</h2>

<p>The toExponential() method returns a string, with the number rounded and written using exponential notation.</p>

<p>An optional parameter defines the number of digits behind the decimal point.</p>

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

<script>

var x = 9.656;

document.getElementById("demo").innerHTML =

x.toExponential() + "<br>" +

x.toExponential(2) + "<br>" +

x.toExponential(4) + "<br>" +

x.toExponential(6);

</script>

</body>

</html>

**Formatting Numbers to Fixed Decimals**

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Number Methods</h2>

<p>The toFixed() method rounds a number to a given number of digits.</p>

<p>For working with money, toFixed(2) is perfect.</p>

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

<script>

var x = 9.656;

document.getElementById("demo").innerHTML =

x.toFixed(0) + "<br>" +

x.toFixed(2) + "<br>" +

x.toFixed(4) + "<br>" +

x.toFixed(6);

</script>

</body>

</html>

**Formatting Numbers with Precision**

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Number Methods</h2>

<p>The toPrecision() method returns a string, with a number written with a specified length:</p>

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

<script>

var x = 9.656;

document.getElementById("demo").innerHTML =

x.toPrecision() + "<br>" +

x.toPrecision(2) + "<br>" +

x.toPrecision(4) + "<br>" +

x.toPrecision(6);

</script>

</body>

</html>

The valueOf() Method

valueOf() returns a number as a number.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Number Methods</h2>

<p>The valueOf() method returns a number as a number:</p>

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

<script>

var x = 123;

document.getElementById("demo").innerHTML =

x.valueOf() + "<br>" +

(123).valueOf() + "<br>" +

(100 + 23).valueOf();

</script>

</body>

</html>

The Number() Method

Number() can be used to convert JavaScript variables to numbers:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Global Methods</h2>

<p>The Number() method converts variables to numbers:</p>

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

<script>

document.getElementById("demo").innerHTML =

Number(true) + "<br>" +

Number(false) + "<br>" +

Number("10") + "<br>" +

Number(" 10") + "<br>" +

Number("10 ") + "<br>" +

Number(" 10 ") + "<br>" +

Number("10.33") + "<br>" +

Number("10,33") + "<br>" +

Number("10 33") + "<br>" +

Number("John");

</script>

</body>

</html>

# JavaScript Arrays

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Storing Single Values</title>

</head>

<body>

<script>

// Creating variables

var color1 = "Red";

var color2 = "Green";

var color3 = "Blue";

// Printing variable values

document.write(color1 + "<br>");

document.write(color2 + "<br>");

document.write(color3);

</script>

</body>

</html>

**Creating an Array**

The simplest way to create an array in JavaScript is enclosing a comma-separated list of values in square brackets ([]), as shown in the following syntax:

var myArray = [*element0*, *element1*, ..., *elementN*];

Array can also be created using the Array() constructor as shown in the following syntax. However, for the sake of simplicity previous syntax is recommended.

var myArray = new Array(*element0*, *element1*, ..., *elementN*);

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Creating Arrays in JavaScript</title>

</head>

<body>

<script>

// Creating variables

var colors = ["Red", "Green", "Blue"];

var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

var cities = ["London", "Paris", "New York"];

var person = ["John", "Wick", 32];

// Printing variable values

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

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

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

document.write(person);

</script>

</body>

</html>

Changing an Array Element

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Arrays</h2>

<p>JavaScript array elements are accessed using numeric indexes (starting from 0).</p>

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

<script>

var cars = ["Saab", "Volvo", "BMW"];

cars[0] = "Opel";

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

</script>

</body>

</html>

**Accessing the Elements of an Array**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Access Individual Elements of an Array</title>

</head>

<body>

<script>

var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

document.write(fruits[0] + "<br>"); // Prints: Apple

document.write(fruits[1] + "<br>"); // Prints: Banana

document.write(fruits[2] + "<br>"); // Prints: Mango

document.write(fruits[fruits.length - 1]); // Prints: Papaya

</script>

</body>

</html>

**Getting the Length of an Array**

The length property returns the length of an array, which is the total number of elements contained in the array. Array length is always greater than the index of any of its element.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Get the Length of an Array</title>

</head>

<body>

<script>

var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

document.write(fruits.length); // 0utputs: 5

</script>

</body>

</html>

**Looping Through Array Elements**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Loop Through an Array Using For Loop</title>

</head>

<body>

<script>

var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

// Iterates over array elements

for(var i = 0; i < fruits.length; i++){

document.write(fruits[i] + "<br>"); // Print array element

}

</script>

</body>

</html>

You can also iterate over the array elements using for-in loop, like this:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Loop Through an Array Using For-In Loop</title>

</head>

<body>

<script>

var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

// Loop through all the elements in the array

for(var i in fruits) {

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

}

</script>

</body>

</html>

**Adding New Elements to an Array**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Add a New Element at the End of an Array</title>

</head>

<body>

<script>

var colors = ["Red", "Green", "Blue"];

colors.push("Yellow");

document.write(colors + "<br>"); // Prints: Red,Green,Blue,Yellow

document.write(colors.length); // Prints: 4

</script>

</body>

</html>

Similarly, to add a new element at the beginning of an array use the unshift() method, like this:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Add a New Element at the Beginning of an Array</title>

</head>

<body>

<script>

var colors = ["Red", "Green", "Blue"];

colors.unshift("Yellow");

document.write(colors + "<br>"); // Prints: Yellow,Red,Green,Blue

document.write(colors.length); // Prints: 4

</script>

</body>

</html>

You can also add multiple elements at once using the push() and unshift() methods, like this:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Add Multiple elements to an Array At Once</title>

</head>

<body>

<script>

var colors = ["Red", "Green", "Blue"];

colors.push("Pink", "Voilet");

colors.unshift("Yellow", "Grey");

document.write(colors + "<br>"); // Prints: Yellow,Grey,Red,Green,Blue,Pink,Voilet

document.write(colors.length); // Prints: 7

</script>

</body>

</html>

**Removing Elements from an Array**

To remove the last element from an array you can use the pop() method. This method returns the value that was popped out. Here's an example:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Remove the Last Element from an Array</title>

</head>

<body>

<script>

var colors = ["Red", "Green", "Blue"];

var last = colors.pop();

document.write(last + "<br>"); // Prints: Blue

document.write(colors.length); // Prints: 2

</script>

</body>

</html>

Similarly, you can remove the first element from an array using the shift() method. This method also returns the value that was shifted out. Here's an example:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Remove the First Element from an Array</title>

</head>

<body>

<script>

var colors = ["Red", "Green", "Blue"];

var first = colors.shift();

document.write(first + "<br>"); // Prints: Red

document.write(colors.length); // Prints: 2

</script>

</body>

</html>

**Adding or Removing Elements at Any Position**

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Array Methods</h2>

<h2>splice()</h2>

<p>The splice() method adds new elements to an array.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo1"></p>

<p id="demo2"></p>

<script>

var fruits = ["Banana", "Orange", "Apple", "Mango"];

document.getElementById("demo1").innerHTML = "Original Array:<br>" + fruits;

function myFunction() {

fruits.splice(2, 0, "Lemon", "Kiwi");

document.getElementById("demo2").innerHTML = "New Array:<br>" + fruits;

}

</script>

</body>

</html>

The first parameter (2) defines the position **where** new elements should be **added** (spliced in).

The second parameter (0) defines **how many** elements should be **removed**.

The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be **added**.

The splice() method returns an array with the deleted items:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Array Methods</h2>

<h2>splice()</h2>

<p>The splice() method adds new elements to an array, and returns an array with the deleted elements (if any).</p>

<button onclick="myFunction()">Try it</button>

<p id="demo1"></p>

<p id="demo2"></p>

<p id="demo3"></p>

<script>

var fruits = ["Banana", "Orange", "Apple", "Mango"];

document.getElementById("demo1").innerHTML = "Original Array:<br> " + fruits;

function myFunction() {

var removed = fruits.splice(2, 2, "Lemon", "Kiwi");

document.getElementById("demo2").innerHTML = "New Array:<br>" + fruits;

document.getElementById("demo3").innerHTML = "Removed Items:<br> " + removed;

}

</script>

</body>

</html>

Merging (Concatenating) Arrays

The concat() method creates a new array by merging (concatenating) existing arrays:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Array Methods</h2>

<h2>concat()</h2>

<p>The concat() method is used to merge (concatenate) arrays:</p>

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

<script>

var myGirls = ["Cecilie", "Lone"];

var myBoys = ["Emil", "Tobias", "Linus"];

var myChildren = myGirls.concat(myBoys);

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

</script>

</body>

</html>

The concat() method can take any number of array arguments:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Array Methods</h2>

<h2>concat()</h2>

<p>The concat() method is used to merge (concatenate) arrays:</p>

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

<script>

var arr1 = ["Cecilie", "Lone"];

var arr2 = ["Emil", "Tobias", "Linus"];

var arr3 = ["Robin", "Morgan"];

var myChildren = arr1.concat(arr2, arr3);

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

</script>

</body>

</html>

The concat() method can also take strings as arguments:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Array Methods</h2>

<h2>concat()</h2>

<p>The concat() method can also merge string values to arrays:</p>

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

<script>

var arr1 = ["Emil", "Tobias", "Linus"];

var myChildren = arr1.concat("Peter");

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

</script>

</body>

</html>

Slicing an Array

The slice() method slices out a piece of an array into a new array.

This example slices out a part of an array starting from array element 1 ("Orange"):

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Array Methods</h2>

<h2>slice()</h2>

<p>This example slices out a part of an array starting from array element 1 ("Orange"):</p>

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

<script>

var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];

var citrus = fruits.slice(1);

document.getElementById("demo").innerHTML = fruits + "<br><br>" + citrus;

</script>

</body>

</html>

The slice() method can take two arguments like slice(1, 3).

The method then selects elements from the start argument, and up to (but not including) the end argument.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Array Methods</h2>

<h2>slice()</h2>

<p>When the slice() method is given two arguments, it selects array elements from the start argument, and up to (but not included) the end argument:</p>

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

<script>

var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];

var citrus = fruits.slice(1,3);

document.getElementById("demo").innerHTML = fruits + "<br><br>" + citrus;

</script>

</body>

</html>

**Creating a String from an Array**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Join All Elements of an Array into a String</title>

</head>

<body>

<script>

var colors = ["Red", "Green", "Blue"];

document.write(colors.join() + "<br>"); // Prints: Red,Green,Blue

document.write(colors.join("") + "<br>"); // Prints: RedGreenBlue

document.write(colors.join("-") + "<br>"); // Prints: Red-Green-Blue

document.write(colors.join(", ")); // Prints: Red, Green, Blue

</script>

</body>

</html>

**Searching Through an Array**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Search an Array for a Specific Value</title>

</head>

<body>

<script>

var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

document.write(fruits.indexOf("Apple") + "<br>"); // Prints: 0

document.write(fruits.indexOf("Banana") + "<br>"); // Prints: 1

document.write(fruits.indexOf("Pineapple")); // Prints: -1

</script>

</body>

</html>

# JavaScript Sorting Arrays

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Sort an Array Alphabetically</title>

</head>

<body>

<script>

var fruits = ["Banana", "Orange", "Apple", "Papaya", "Mango"];

var sorted = fruits.sort();

document.write(fruits + "<br>"); // Outputs: Apple,Banana,Mango,Orange,Papaya

document.write(sorted); // Outputs: Apple,Banana,Mango,Orange,Papaya

</script>

</body>

</html>

**Reversing an Array**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Reverse the Order of an Array</title>

</head>

<body>

<script>

var counts = ["one", "two", "three", "four", "five"];

var reversed = counts.reverse();

document.write(counts + "<br>"); // Outputs: five,four,three,two,one

document.write(reversed); // Output: five,four,three,two,one

</script>

</body>

</html>

**Sorting Numeric Arrays**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Sort a Numeric Array</title>

</head>

<body>

<script>

var numbers = [5, 20, 10, 75, 50, 100];

numbers.sort(); // Sorts numbers array

document.write(numbers); // Outputs: 10,100,20,5,50,75

</script>

</body>

</html>

# JavaScript Date Objects

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript new Date()</h2>

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

<script>

var d = new Date();

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

</script>

</body>

</html>

Creating Date Objects

Date objects are created with the new Date() constructor.

There are **4 ways** to create a new date object:

new Date()  
new Date(year, month, day, hours, minutes, seconds, milliseconds)  
new Date(milliseconds)  
new Date(date string)

new Date(*year, month, ...*)

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript new Date()</h2>

<p>Using new Date(7 numbers), creates a new date object with the specified date and time:</p>

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

<script>

var d = new Date(2018, 11, 24, 10, 33, 30, 0);

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

</script>

</body>

</html>

Example:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript new Date()</h2>

<p>A Date object can be created with a specified date and time:</p>

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

<script>

var d = new Date("October 13, 2014 11:13:00");

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

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript new Date()</h2>

<p>3 numbers specify year, month, and day:</p>

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

<script>

var d = new Date(2018, 11, 24);

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

</script>

</body>

</html>

JavaScript Date Input

There are generally 3 types of JavaScript date input formats:

|  |  |
| --- | --- |
| **Type** | **Example** |
| ISO Date | "2015-03-25" (The International Standard) |
| Short Date | "03/25/2015" |
| Long Date | "Mar 25 2015" or "25 Mar 2015" |

These methods can be used for getting information from a date object:

|  |  |
| --- | --- |
| **Method** | **Description** |
| getFullYear() | Get the **year** as a four digit number (yyyy) |
| getMonth() | Get the **month** as a number (0-11) |
| getDate() | Get the **day** as a number (1-31) |
| getHours() | Get the **hour** (0-23) |
| getMinutes() | Get the **minute** (0-59) |
| getSeconds() | Get the **second** (0-59) |
| getMilliseconds() | Get the **millisecond** (0-999) |
| getTime() | Get the time (milliseconds since January 1, 1970) |
| getDay() | Get the weekday as a number (0-6) |
| Date.now() | Get the time. ECMAScript 5. |

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript getTime()</h2>

<p>The internal clock in JavaScript counts from midnight January 1, 1970.</p>

<p>The getTime() function returns the number of milliseconds since then:</p>

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

<script>

var d = new Date();

document.getElementById("demo").innerHTML = d.getTime();

</script>

</body>

</html>

Set Date Methods

Set Date methods are used for setting a part of a date:

|  |  |
| --- | --- |
| **Method** | **Description** |
| setDate() | Set the day as a number (1-31) |
| setFullYear() | Set the year (optionally month and day) |
| setHours() | Set the hour (0-23) |
| setMilliseconds() | Set the milliseconds (0-999) |
| setMinutes() | Set the minutes (0-59) |
| setMonth() | Set the month (0-11) |
| setSeconds() | Set the seconds (0-59) |
| setTime() | Set the time (milliseconds since January 1, 1970) |

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript setFullYear()</h2>

<p>The setFullYear() method sets the year of a date object:</p>

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

<script>

var d = new Date();

d.setFullYear(2020);

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

</script>

</body>

</html>

**The JavaScript Math Object**

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Math.PI</h2>

<p>Math.PI returns the ratio of a circle's circumference to its diameter:</p>

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

<script>

document.getElementById("demo").innerHTML = Math.PI;

</script>

</body>

</html>

Math.round()

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Math.round()</h2>

<p>Math.round(x) returns the value of x rounded to its nearest integer:</p>

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

<script>

document.getElementById("demo").innerHTML = Math.round(4.4);

</script>

</body>

</html>

Math.Power

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Math.pow()</h2>

<p>Math.pow(x,y) returns the value of x to the power of y:</p>

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

<script>

document.getElementById("demo").innerHTML = Math.pow(8,2);

</script>

</body>

</html>

Math.sqrt()

Math.abs()

Math.abs(-4.7);

Math.ceil()

Math.ceil(6.4);

Math.floor()

Math.floor(x) returns the value of x rounded **down** to its nearest integer:

Math.sin()

Math.sin(x) returns the sine (a value between -1 and 1) of the angle x (given in radians).

Math.min() and Math.max()

Math.min() and Math.max() can be used to find the lowest or highest value in a list of arguments:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Math.min()</h2>

<p>Math.min() returns the lowest value in a list of arguments:</p>

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

<script>

document.getElementById("demo").innerHTML =

Math.min(0, 150, 30, 20, -8, -200);

</script>

</body>

</html>

Math.random()

Math.random() returns a random number between 0 and 1:

# JavaScript if else and else if

**JavaScript Conditional Statements**

There are several conditional statements in JavaScript that you can use to make decisions:

* The **if** statement
* The **if...else** statement
* The **if...else if....else** statement
* The **switch...case** statement
* **The if Statement**
* The *if* statement is used to execute a block of code only if the specified condition evaluates to true. This is the simplest JavaScript's conditional statements and can be written like:
* if(condition) {  
      *// Code to be executed*  
  }

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript If Statement</title>

</head>

<body>

<script>

var now = new Date();

var dayOfWeek = now.getDay(); // Sunday - Saturday : 0 - 6

if(dayOfWeek == 5) {

document.write("Have a nice weekend!");

}

</script>

<p><strong>Note:</strong> This example will print "Have a nice weekend!" if the current day is Friday.</p>

</body>

</html>

**The if...else Statement**

if(condition) {  
    // Code to be executed if condition is true  
} else {  
    // Code to be executed if condition is false  
}

**The if...else if...else Statement**

<!DOCTYPE html>

<html>

<body>

<p>Click the button to get a time-based greeting:</p>

<button onclick="myFunction()">Try it</button>

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

<script>

function myFunction() {

var greeting;

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

if (time < 10) {

greeting = "Good morning";

} else if (time < 20) {

greeting = "Good day";

} else {

greeting = "Good evening";

}

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

}

</script>

</body>

</html>

# JavaScript Switch...Case Statements

switch(x){  
    case value1:  
        // Code to be executed if x === value1  
        break;  
    case value2:  
        // Code to be executed if x === value2  
        break;  
    ...  
    default:  
        // Code to be executed if x is different from all values  
}

Consider the following example, which display the name of the day of the week.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Switch Case Statement</title>

</head>

<body>

<script>

var d = new Date();

switch(d.getDay()) {

case 0:

document.write("Today is Sunday.");

break;

case 1:

document.write("Today is Monday.");

break;

case 2:

document.write("Today is Tuesday.");

break;

case 3:

document.write("Today is Wednesday.");

break;

case 4:

document.write("Today is Thursday.");

break;

case 5:

document.write("Today is Friday.");

break;

case 6:

document.write("Today is Saturday.");

break;

default:

document.write("No information available for that day.");

break;

}

</script>

</body>

</html>

Example:

# JavaScript Loops

**Different Types of Loops in JavaScript**

Loops are used to execute the same block of code again and again, as long as a certain condition is met. The basic idea behind a loop is to automate the repetitive tasks within a program to save the time and effort. JavaScript now supports five different types of loops:

* **while** — loops through a block of code as long as the condition specified evaluates to true.
* **do…while** — loops through a block of code once; then the condition is evaluated. If the condition is true, the statement is repeated as long as the specified condition is true.
* **for** — loops through a block of code until the counter reaches a specified number.
* **for…in** — loops through the properties of an object.

**The for Loop**

The for loop repeats a block of code as long as a certain condition is met. It is typically used to execute a block of code for certain number of times. Its syntax is:

for(*initialization*; *condition*; *increment*) {  
    *// Code to be executed*  
}

The parameters of the for loop statement have following meanings:

* ***initialization*** — it is used to initialize the counter variables, and evaluated once unconditionally before the first execution of the body of the loop.
* ***condition*** — it is evaluated at the beginning of each iteration. If it evaluates to true, the loop statements execute. If it evaluates to false, the execution of the loop ends.
* ***increment*** — it updates the loop counter with a new value each time the loop runs.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript For Loop</title>

</head>

<body>

<script>

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

document.write("<p>The number is " + i + "</p>");

}

</script>

</body>

</html>

**The for...in Loop**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Iterate Over an Array Using For Loop</title>

</head>

<body>

<script>

// An object with some properties

var person = {"name": "Clark", "surname": "Kent", "age": "36"};

// Loop through all the properties in the object

for(var prop in person) {

document.write("<p>" + prop + " = " + person[prop] + "</p>");

}

</script>

</body>

</html>

## The 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>

<body>

<h2>JavaScript While Loop</h2>

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

<script>

var text = "";

var i = 0;

while (i < 10) {

text += "<br>The number is " + i;

i++;

}

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

</script>

</body>

</html>

## The Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

### Syntax

do {  
*// code block to be executed*}  
while (condition);

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Do/While Loop</h2>

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

<script>

var text = ""

var i = 0;

do {

text += "<br>The number is " + i;

i++;

}

while (i < 10);

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

</script>

</body>

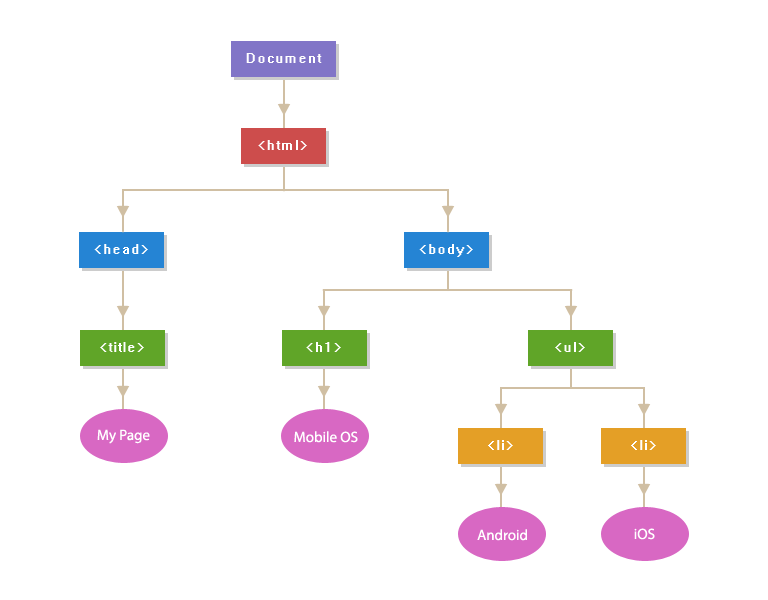
</html>

# JavaScript DOM Nodes

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

In other words:**The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**



# JavaScript - HTML DOM Methods

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

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

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

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

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

<!DOCTYPE html>

<html>

<body>

<h2>My First Page</h2>

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

<script>

document.getElementById("demo").innerHTML = "Hello World!";

</script>

</body>

</html>

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

Finding HTML Elements

|  |  |
| --- | --- |
| **Method** | **Description** |
| document.getElementById(*id*) | Find an element by element id |
| document.getElementsByTagName(*name*) | Find elements by tag name |
| document.getElementsByClassName(*name*) | Find elements by class name |

Changing HTML Elements

|  |  |
| --- | --- |
| **Property** | **Description** |
| *element*.innerHTML =  *new html content* | Change the inner HTML of an element |
| *element*.*attribute = new value* | Change the attribute value of an HTML element |
| *element*.style.*property = new style* | Change the style of an HTML element |
| **Method** | **Description** |
| *element*.setAttribute*(attribute, value)* | Change the attribute value of an HTML element |

Adding and Deleting Elements

|  |  |
| --- | --- |
| **Method** | **Description** |
| document.createElement(*element*) | Create an HTML element |
| document.removeChild(*element*) | Remove an HTML element |
| document.appendChild(*element*) | Add an HTML element |
| document.replaceChild(*new, old*) | Replace an HTML element |
| document.write(*text*) | Write into the HTML output stream |

**Finding HTML Elements by Tag Name**

<!DOCTYPE html>

<html>

<body>

<h2>Finding HTML Elements by Tag Name</h2>

<p>Hello World!</p>

<p>This example demonstrates the <b>getElementsByTagName</b> method.</p>

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

<script>

var x = document.getElementsByTagName("p");

document.getElementById("demo").innerHTML =

'The text in first paragraph (index 0) is: ' + x[0].innerHTML;

</script>

</body>

</html>

Example2:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Select Elements by Tag Name</title>

</head>

<body>

<p>This is a paragraph of text.</p>

<div class="test">This is another paragraph of text.</div>

<p>This is one more paragraph of text.</p>

<hr>

<script>

// Selecting all paragraph elements

var matches = document.getElementsByTagName("p");

// Printing the number of selected paragraphs

document.write("Number of selected elements: " + matches.length);

// Highlighting each paragraph's background through loop

for(var elem in matches) {

matches[elem].style.background = "yellow";

}

</script>

</body>

</html>

Example:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Select Elements by Tag Name</title>

</head>

<body>

<p>This is a paragraph of text.</p>

<div class="test">This is another paragraph of text.</div>

<p>This is one more paragraph of text.</p>

<hr>

<script>

// Selecting all paragraph elements

var matches = document.getElementsByTagName("p");

// Printing the number of selected paragraphs

document.write("Number of selected elements: " + matches.length);

//matches[0].style.background = "yellow";

// Highlighting each paragraph's background through loop

// for(var i in matches) {

// matches[i].style.background = "yellow";

//}

for( var i=0; i< matches.length; i++)

{

matches[i].style.background = "yellow";

}

</script>

</body>

</html>

**Finding HTML Elements by Class Name**

<!DOCTYPE html>

<html>

<body>

<h2>Finding HTML Elements by Class Name</h2>

<p>Hello World!</p>

<p class="intro">The DOM is very useful.</p>

<p class="intro">This example demonstrates the <b>getElementsByClassName</b> method.</p>

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

<script>

var x = document.getElementsByClassName("intro");

document.getElementById("demo").innerHTML =

'The first paragraph (index 0) with class="intro": ' + x[0].innerHTML;

</script>

</body>

</html>

Example2:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Select Elements by Class Name</title>

</head>

<body>

<p class="test">This is a paragraph of text.</p>

<div class="block test">This is another paragraph of text.</div>

<p>This is one more paragraph of text.</p>

<hr>

<script>

// Selecting elements with class test

var matches = document.getElementsByClassName("test");

// Displaying the selected elements count

document.write("Number of selected elements: " + matches.length);

// Applying bold style to first element in selection

matches[0].style.fontWeight = "bold";

// Applying italic style to last element in selection

matches[matches.length - 1].style.fontStyle = "italic";

// Highlighting each element's background through loop

for(var elem in matches) {

matches[elem].style.background = "yellow";

}

</script>

</body>

</html>

Finding HTML Elements by CSS Selectors

If you want to find all HTML elements that match a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method.

This example returns a list of all <p> elements with class="intro".

<!DOCTYPE html>

<html>

<body>

<h2>Finding HTML Elements by Query Selector</h2>

<p>Hello World!</p>

<p class="intro">The DOM is very useful.</p>

<p class="intro">This example demonstrates the <b>querySelectorAll</b> method.</p>

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

<script>

var x = document.querySelectorAll("p.intro");

document.getElementById("demo").innerHTML =

'The first paragraph (index 0) with class="intro": ' + x[0].innerHTML;

</script>

</body>

</html>

Example2:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript Select Elements with CSS Selectors</title>

</head>

<body>

<ul>

<li>Bread</li>

<li class="tick">Coffee</li>

<li>Pineapple Cake</li>

</ul>

<script>

// Selecting all li elements

var matches = document.querySelectorAll("ul li");

// Printing the number of selected li elements

document.write("Number of selected elements: " + matches.length + "<hr>")

// Printing the content of selected li elements

for(var elem of matches) {

document.write(elem.innerHTML + "<br>");

}

// Applying line through style to first li element with class tick

matches = document.querySelectorAll("ul li.tick");

matches[0].style.textDecoration = "line-through";

</script>

</body>

</html>