# 트리에 대해서 확인해 보자

# F:\2016Javascript\016장DOM\tree1.png

# 폴더구조, 사이틀이 없는 그래프 (노드와 간선으로 이루어진 데이터)

# 사이클이 없어서 하나의 노드에서 다른노드로 가는 방법이 오직 하나만 존재함

# F:\2016Javascript\016장DOM\tree2.png

# 위에있는 노드를 부모 아래있는 노드를 자식 부모가 같은 노드를 형제 노드라고 한다.

# F:\2016Javascript\016장DOM\tree3.png

# 루트 노드는?

# 리프노드는?

# 트리의 높이는?

# 레벨1의 노드는?

# J노드의 형제 노드는?

# L노드의 부모 노드는?

# A노드의 차수는?

# 트리의 차수는?

# F:\2016Javascript\016장DOM\tree4.png

# 돔이란?

# 웹브라우저에서 HTML문서를 구조화된 문서로 표현하는 방법 속성, 엘리먼트 등의 객체를 트리 구조로 표현

# 어디에 사용하는가?

# 메소드를 이용해 특정 엘리먼트,속성을 찾아서 엘리먼트나 속성을 삽입, 삭제, 변경하는 일을 할 때 사용한다.

### The HTML DOM Tree of Objects



상위그림을 보고 html문서를 만들어보자.

<!DOCTYPE html>

<html>

<head>

<title>INDEX</title>

</head>

<body>

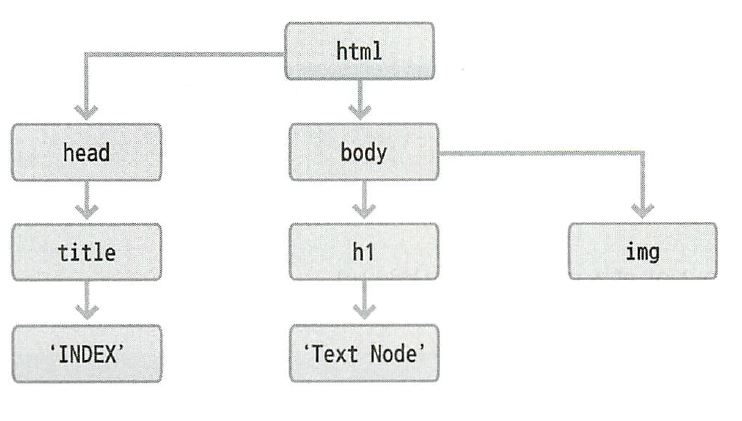
<h1>Text Node</h1>

<img src="image.jpg" />

</body>

</html>

다음을 그림으로 그려보자.



# JavaScript HTML DOM

With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

## 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**: 구성되어 있다.

With the object model, JavaScript gets all the power it needs to create dynamic HTML: 힘을 얻는다.동적

* 자바스크립트는 페이지 안의 HTML elements를 찾을 수 있다.
* 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 반응하다.
* JavaScript can create new HTML events in the page

## What You Will Learn

In the next chapters of this tutorial you will learn:

* How to change the content of HTML elements
* How to change the style (CSS) of HTML elements
* How to react to HTML DOM events 반응하다.
* How to add and delete HTML elements

## 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*

*언어중립적인 동적*

*dynamically access and update the content, structure, and style of a document."*

The W3C DOM standard is separated into 3 different parts:

분리되다 세분화된다.

* 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** 엘리먼트를 객체로
* 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

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

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

## The DOM Programming Interface

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

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

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

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

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

## Example

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

### Example

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

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_method)

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

## The getElementById Method

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

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

## The innerHTML Property

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

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

메소드를 이용해서 객체를 가지고 온 다음 객체안의 프로퍼티를 이용해서 값을 가져오거나 제거할 수 있다.

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

# JavaScript HTML DOM Document

The HTML DOM document object is the owner of all other objects in your web page.

## The HTML DOM Document Object

The document object represents your web page.표현하다

If you want to access any element in an HTML page, you always start with accessing the document object.

Below are some examples of how you can use the document object to access and manipulate HTML. 교묘히 능란하게 다루다.

## 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

|  |  |
| --- | --- |
| **Method** | **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.setAttribute(attribute, value) | Change the attribute value of an HTML element |
| element.style.property = new style | Change the style 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(element) | Replace an HTML element |
| document.write(text) | Write into the HTML output stream |

## Adding Events Handlers

|  |  |
| --- | --- |
| **Method** | **Description** |
| document.getElementById(id).onclick = function(){code} | Adding event handler code to an onclick event |

## Dom 객체 히스토리 강의 생략

## Finding HTML Objects

The first HTML DOM Level 1 (1998), defined 11 HTML objects, object collections, and properties. These are still valid in HTML5.

Later, in HTML DOM Level 3, more objects, collections, and properties were added.

|  |  |  |
| --- | --- | --- |
| **Property** | **Description** | **DOM** |
| document.anchors | Returns all <a> elements that have a name attribute | 1 |
| document.applets | Returns all <applet> elements (Deprecated in HTML5) | 1 |
| document.baseURI | Returns the absolute base URI of the document | 3 |
| document.body | Returns the <body> element | 1 |
| document.cookie | Returns the document's cookie | 1 |
| document.doctype | Returns the document's doctype | 3 |
| document.documentElement | Returns the <html> element | 3 |
| document.documentMode | Returns the mode used by the browser | 3 |
| document.documentURI | Returns the URI of the document | 3 |
| document.domain | Returns the domain name of the document server | 1 |
| document.domConfig | Obsolete. Returns the DOM configuration | 3 |
| document.embeds | Returns all <embed> elements | 3 |
| document.forms | Returns all <form> elements | 1 |
| document.head | Returns the <head> element | 3 |
| document.images | Returns all <img> elements | 1 |
| document.implementation | Returns the DOM implementation | 3 |
| document.inputEncoding | Returns the document's encoding (character set) | 3 |
| document.lastModified | Returns the date and time the document was updated | 3 |
| document.links | Returns all <area> and <a> elements that have a href attribute | 1 |
| document.readyState | Returns the (loading) status of the document | 3 |
| document.referrer | Returns the URI of the referrer (the linking document) | 1 |
| document.scripts | Returns all <script> elements | 3 |
| document.strictErrorChecking | Returns if error checking is enforced | 3 |
| document.title | Returns the <title> element | 1 |
| document.URL | Returns the complete URL of the document |  |

# JavaScript HTML DOM Elements

This page teaches you how to find and access HTML elements in an HTML page.

## Finding HTML Elements

Often, with JavaScript, you want to manipulate HTML elements.

To do so, you have to find the elements first. There are a couple of ways to do this:

* Finding HTML elements by id
* Finding HTML elements by tag name
* Finding HTML elements by class name
* Finding HTML elements by CSS selectors
* Finding HTML elements by HTML object collections

## Finding HTML Element by Id

The easiest way to find an HTML element in the DOM, is by using the element id.

This example finds the element with id="intro":

### Example

var myElement = document.getElementById("intro");

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_getelementbyid)

If the element is found, the method will return the element as an object (in myElement).

If the element is not found, myElement will contain null.

다음 예제를 확인해보자.

<!DOCTYPE html>

<html>

<body>

<p id="intro">Hello World!</p>

<p>This example demonstrates the <b>getElementById</b> method!</p>

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

<script>

var myElement = document.getElementById("intro");

document.getElementById("demo").innerHTML =

"The text from the intro paragraph is " + myElement.innerHTML;

</script>

</body>

</html>

## Finding HTML Elements by Tag Name

This example finds all <p> elements:

### Example

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

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_getelementsbytagname2)

This example finds the element with id="main", and then finds all <p> elements inside "main":

다음예제를 확인해보자.

<!DOCTYPE html>

<html>

<body>

<p>Hello World!</p>

<p>The DOM is very useful.</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 first paragraph (index 0) is: ' + x[0].innerHTML;

</script>

</body>

</html>

### Example

var x = document.getElementById("main");  
var y = x.getElementsByTagName("p");

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_getelementsbytagname)

<!DOCTYPE html>

<html>

<body>

<p>Hello World!</p>

<div id="main">

<p>The DOM is very useful.</p>

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

</div>

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

<script>

var x = document.getElementById("main");

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

document.getElementById("demo").innerHTML =

'The first paragraph (index 0) inside "main" is: ' + y[0].innerHTML;

</script>

</body>

</html>

## Finding HTML Elements by Class Name

If you want to find all HTML elements with the same class name, use getElementsByClassName().

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

### Example

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

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_getelementsbyclassname)

|  |  |
| --- | --- |
| **Note** | Finding elements by class name does not work in Internet Explorer 8 and earlier versions. |

<!DOCTYPE html>

<html>

<body>

<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>

## Finding HTML Elements by CSS Selectors

If you want to find all HTML elements that matches 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".

### Example

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

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_queryselectorall)

|  |  |
| --- | --- |
| **Note** | The querySelectorAll() method does not work in Internet Explorer 8 and earlier versions. |

<!DOCTYPE html>

<html>

<body>

<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>

## Finding HTML Elements by HTML Object Collections

This example finds the form element with id="frm1", in the forms collection, and displays all element values:

### Example

var x = document.forms["frm1"];  
var text = "";  
var i;  
for (i = 0; i < x.length; i++) {  
    text += x.elements[i].value + "<br>";  
}  
document.getElementById("demo").innerHTML = text;

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_form_elements)

<!DOCTYPE html>

<html>

<body>

<form id="frm1" action="form\_action.asp">

First name: <input type="text" name="fname" value="Donald"><br>

Last name: <input type="text" name="lname" value="Duck"><br><br>

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

</form>

<p>Click "Try it" to display the value of each element in the form.</p>

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

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

<script>

function myFunction() {

var x = document.forms["frm1"];

var text = "";

var i;

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

text += x.elements[i].value + "<br>";

}

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

}

</script>

</body>

</html>

The following HTML objects (and object collections) are also accessible:

* [document.anchors](http://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_anchors)
* [document.body](http://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_body)
* [document.documentElement](http://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_element)
* [document.embeds](http://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_embeds)
* [document.forms](http://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_forms)
* [document.head](http://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_head)
* [document.images](http://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_images)
* [document.links](http://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_links)
* [document.scripts](http://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_scripts)
* [document.title](http://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_title)

## Test Yourself with Exercises!

[Exercise 1 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_elements1)  [Exercise 2 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_elements2)  [Exercise 3 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_elements3)  [Exercise 4 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_elements4)  [Exercise 5 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_elements5)

[« Previous](http://www.w3schools.com/js/js_htmldom_document.asp)

[Next Chapter »](http://www.w3schools.com/js/js_htmldom_html.asp)

# JavaScript HTML DOM - Changing HTML

[« Previous](http://www.w3schools.com/js/js_htmldom_elements.asp)

[Next Chapter »](http://www.w3schools.com/js/js_htmldom_css.asp)

The HTML DOM allows JavaScript to change the content of HTML elements.

## Changing the HTML Output Stream

JavaScript can create dynamic HTML content:

**Date: Fri Nov 27 2015 12:17:17 GMT+0900 (대한민국 표준시)**

In JavaScript, document.write() can be used to write directly to the HTML output stream:

### Example

<!DOCTYPE html>  
<html>  
<body>  
  
<script>  
document.write(Date());  
</script>  
  
</body>  
</html>

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_date)

|  |  |
| --- | --- |
| **Note** | Never use document.write() after the document is loaded. It will overwrite the document. |

## Changing HTML Content

The easiest way to modify the content of an HTML element is by using the **innerHTML** property.

To change the content of an HTML element, use this syntax:

document.getElementById(*id*).innerHTML = *new HTML*

This example changes the content of a <p> element:

### Example

<html>  
<body>  
  
<p id="p1">Hello World!</p>  
  
<script>  
document.getElementById("p1").innerHTML = "New text!";  
</script>  
  
</body>  
</html>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_change_innerhtml)

This example changes the content of an <h1> element:

### Example

<!DOCTYPE html>  
<html>  
<body>  
  
<h1 id="header">Old Header</h1>  
  
<script>  
var element = document.getElementById("header");  
element.innerHTML = "New Header";  
</script>  
  
</body>  
</html>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_innerhtml)

Example explained:

* The HTML document above contains an <h1> element with id="header"
* We use the HTML DOM to get the element with id="header"
* A JavaScript changes the content (innerHTML) of that element

## Changing the Value of an Attribute

To change the value of an HTML attribute, use this syntax:

document.getElementById(*id*).*attribute=new value*

This example changes the value of the src attribute of an <img> element:

### Example

<!DOCTYPE html>  
<html>  
<body>  
  
<img id="myImage" src="smiley.gif">  
  
<script>  
document.getElementById("myImage").src = "landscape.jpg";  
</script>  
  
</body>  
</html>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_image)

Example explained:

* The HTML document above contains an <img> element with id="myImage"
* We use the HTML DOM to get the element with id="myImage"
* A JavaScript changes the src attribute of that element from "smiley.gif" to "landscape.jpg"

## Test Yourself with Exercises!

[Exercise 1 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_change1)  [Exercise 2 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_change2)  [Exercise 3 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_change3)  [Exercise 4 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_change4)  [Exercise 5 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_change5)

[« Previous](http://www.w3schools.com/js/js_htmldom_elements.asp)

[Next Chapter »](http://www.w3schools.com/js/js_htmldom_css.asp)

# JavaScript HTML DOM - Changing CSS

[« Previous](http://www.w3schools.com/js/js_htmldom_html.asp)

[Next Chapter »](http://www.w3schools.com/js/js_htmldom_animate.asp)

The HTML DOM allows JavaScript to change the style of HTML elements.

## Changing HTML Style

To change the style of an HTML element, use this syntax:

document.getElementById(*id*).style.*property*=*new style*

The following example changes the style of a <p> element:

### Example

<html>  
<body>  
  
<p id="p2">Hello World!</p>  
  
<script>  
document.getElementById("p2").style.color = "blue";  
</script>  
  
<p>The paragraph above was changed by a script.</p>  
  
</body>  
</html>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_change_style)

## Using Events

The HTML DOM allows you to execute code when an event occurs.

Events are generated by the browser when "things happen" to HTML elements:

* An element is clicked on
* The page has loaded
* Input fields are changed

You will learn more about events in the next chapter of this tutorial.

This example changes the style of the HTML element with id="id1", when the user clicks a button:

### Example

<!DOCTYPE html>  
<html>  
<body>  
  
<h1 id="id1">My Heading 1</h1>  
  
<button type="button"   
onclick="document.getElementById('id1').style.color = 'red'">  
Click Me!</button>  
  
</body>  
</html>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_color2)

## More Examples

[Visibility](http://www.w3schools.com/js/tryit.asp?filename=tryjs_visibility) How to make an element invisible. Do you want to show the element or not?

## HTML DOM Style Object Reference

For all HTML DOM style properties, look at our complete [HTML DOM Style Object Reference](http://www.w3schools.com/jsref/dom_obj_style.asp).

## Test Yourself with Exercises!

[Exercise 1 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_css1)  [Exercise 2 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_css2)  [Exercise 3 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_css3)  [Exercise 4 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_css4)  [Exercise 5 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_css5)

# JavaScript HTML DOM Animation

[« Previous](http://www.w3schools.com/js/js_htmldom_css.asp)

[Next Chapter »](http://www.w3schools.com/js/js_htmldom_events.asp)

Learn to create HTML animations using JavaScript.

## A Basic Web Page

To demonstrate how to create HTML animations with JavaScript, we will use a simple web page:

### Example

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First JavaScript Animation</h1>  
  
<div id="animation">My animation will go here</div>  
  
</body>  
<html>

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_animate_1)

## Create an Animation Container

All animations should be relative to a container element.

### Example

<div id ="container">  
    <div id ="animate">My animation will go here</div>  
</div>

## Style the Elements

The container element should be created with style = "position: relative".

The animation element should be created with style = "position: absolute".

### Example

#container {  
    width: 400px;  
    height: 400px;  
    position: relative;  
    background: yellow;  
}  
#animate {  
    width: 50px;  
    height: 50px;  
    position: absolute;  
    background: red;  
}

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_animate_2)

<!Doctype html>

<html>

<style>

#container {

width: 400px;

height: 400px;

position: relative;

background: yellow;

}

#animate {

width: 50px;

height: 50px;

position: absolute;

background: red;

}

</style>

<body>

<h1>My First JavaScript Animation</h1>

<div id="container">

<div id="animate"></div>

</div>

</body>

</html>

## Animation Code

JavaScript animations are done by programming gradual changes in an element's style.

The changes are called by a timer. When the timer interval is small, the animation looks continuous.

The basic code is:

### Example

var id = setInterval(frame, 5);  
  
function frame() {  
    if (/\* test for finished \*/) {  
        clearInterval(id);  
    } else {  
        /\* code to change the element style \*/    
    }  
}

## Create the Animation Using JavaScript

### Example

function myMove() {  
    var elem = document.getElementById("animate");   
    var pos = 0;  
    var id = setInterval(frame, 5);  
    function frame() {  
        if (pos == 350) {  
            clearInterval(id);  
        } else {  
            pos++;   
            elem.style.top = pos + 'px';   
            elem.style.left = pos + 'px';   
        }  
    }  
}

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_animate_3)

<!DOCTYPE html>

<html>

<style>

#container {

width: 400px;

height: 400px;

position: relative;

background: yellow;

}

#animate {

width: 50px;

height: 50px;

position: absolute;

background-color: red;

}

</style>

<body>

<p>

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

</p>

<div id ="container">

<div id ="animate"></div>

</div>

<script>

function myMove() {

var elem = document.getElementById("animate");

var pos = 0;

var id = setInterval(frame, 5);

function frame() {

if (pos == 350) {

clearInterval(id);

} else {

pos++;

elem.style.top = pos + 'px';

elem.style.left = pos + 'px';

}

}

}

</script>

</body>

</html>

# [« Previous](http://www.w3schools.com/js/js_htmldom_css.asp)JavaScript HTML DOM Events

[« Previous](http://www.w3schools.com/js/js_htmldom_animate.asp)

[Next Chapter »](http://www.w3schools.com/js/js_htmldom_eventlistener.asp)

HTML DOM allows JavaScript to react to HTML events:

**Mouse Over Me**

**Click Me**

## Reacting to Events

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.

To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

onclick=*JavaScript*

Examples of HTML events:

* When a user clicks the mouse
* When a web page has loaded
* When an image has been loaded
* When the mouse moves over an element
* When an input field is changed
* When an HTML form is submitted
* When a user strokes a key

In this example, the content of the <h1> element is changed when a user clicks on it:

### Example

<!DOCTYPE html>  
<html>  
<body>  
  
<h1 onclick="this.innerHTML='Ooops!'">Click on this text!</h1>  
  
</body>  
</html>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onclick2)

In this example, a function is called from the event handler:

### Example

<!DOCTYPE html>  
<html>  
<body>  
  
<h1 onclick="changeText(this)">Click on this text!</h1>  
  
<script>  
function changeText(id) {   
    id.innerHTML = "Ooops!";  
}  
</script>  
  
</body>  
</html>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onclick3)

## HTML Event Attributes

To assign events to HTML elements you can use event attributes.

### Example

Assign an onclick event to a button element:

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

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events1)

<!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>

In the example above, a function named displayDate will be executed when the button is clicked.

## Assign Events Using the HTML DOM

The HTML DOM allows you to assign events to HTML elements using JavaScript:

### Example

Assign an onclick event to a button element:

<script>  
document.getElementById("myBtn").onclick = displayDate;  
</script>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events2)

<!DOCTYPE html>

<html>

<body>

<p>Click "Try it" to execute the displayDate() function.</p>

<button id="myBtn">Try it</button>

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

<script>

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

function displayDate() {

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

}

</script>

</body>

</html>

In the example above, a function named displayDate is assigned to an HTML element with the id="myBtn".

The function will be executed when the button is clicked.

## The onload and onunload Events

The onload and onunload events are triggered when the user enters or leaves the page.

The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

The onload and onunload events can be used to deal with cookies.

### Example

<body onload="checkCookies()">

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onload)

<!DOCTYPE html>

<html>

<body onload="checkCookies()">

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

<script>

function checkCookies() {

var text = "";

if (navigator.cookieEnabled == true) {

text = "Cookies are enabled.";

} else {

text = "Cookies are not enabled.";

}

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

}

</script>

</body>

</html>

## The onchange Event

The onchange event are often used in combination with validation of input fields.

Below is an example of how to use the onchange. The upperCase() function will be called when a user changes the content of an input field.

### Example

<input type="text" id="fname" onchange="upperCase()">

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_onchange)

<!DOCTYPE html>

<html>

<head>

<script>

function myFunction() {

var x = document.getElementById("fname");

x.value = x.value.toUpperCase();

}

</script>

</head>

<body>

Enter your name: <input type="text" id="fname" onchange="myFunction()">

<p>When you leave the input field, a function is triggered which transforms the input text to upper case.</p>

</body>

</html>

## The onmouseover and onmouseout Events

The onmouseover and onmouseout events can be used to trigger a function when the user mouses over, or out of, an HTML element:

Mouse Over Me

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_mouseover)

<!DOCTYPE html>

<html>

<body>

<div onmouseover="mOver(this)" onmouseout="mOut(this)"

style="background-color:#D94A38;width:120px;height:20px;padding:40px;">

Mouse Over Me</div>

<script>

function mOver(obj) {

obj.innerHTML = "Thank You"

}

function mOut(obj) {

obj.innerHTML = "Mouse Over Me"

}

</script>

</body>

</html>

## The onmousedown, onmouseup and onclick Events

The onmousedown, onmouseup, and onclick events are all parts of a mouse-click. First when a mouse-button is clicked, the onmousedown event is triggered, then, when the mouse-button is released, the onmouseup event is triggered, finally, when the mouse-click is completed, the onclick event is triggered.

Click Me

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_mousedown)

<!DOCTYPE html>

<html>

<body>

<div onmousedown="mDown(this)" onmouseup="mUp(this)"

style="background-color:#D94A38;width:90px;height:20px;padding:40px;">

Click Me</div>

<script>

function mDown(obj) {

obj.style.backgroundColor = "#1ec5e5";

obj.innerHTML = "Release Me";

}

function mUp(obj) {

obj.style.backgroundColor="#D94A38";

obj.innerHTML="Thank You";

}

</script>

</body>

</html>

## More Examples

[onmousedown and onmouseup](http://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onmousedown)  
Change an image when a user holds down the mouse button.

[onload](http://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onload)  
Display an alert box when the page has finished loading.

[onfocus](http://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onfocus)  
Change the background-color of an input field when it gets focus.

[Mouse Events](http://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onmouse)  
Change the color of an element when the cursor moves over it.

## HTML DOM Event Object Reference

For a list of all HTML DOM events, look at our complete [HTML DOM Event Object Reference](http://www.w3schools.com/jsref/dom_obj_event.asp).

## Test Yourself with Exercises!

[Exercise 1 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_events1)   [Exercise 2 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_events2)   [Exercise 3 »](http://www.w3schools.com/js/exercise.asp?filename=exercise_dom_events3)

# JavaScript HTML DOM EventListener

## The addEventListener() method

### Example

Add an event listener that fires when a user clicks a button:

document.getElementById("myBtn").addEventListener("click", displayDate);

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_displaydate)

<!DOCTYPE html>

<html>

<body>

<p>This example uses the addEventListener() method to attach a click event to a button.</p>

<button id="myBtn">Try it</button>

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

<script>

document.getElementById("myBtn").addEventListener("click", displayDate);

function displayDate() {

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

}

</script>

</body>

</html>

The addEventListener() method attaches an event handler to the specified element.

The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.

You can add many event handlers to one element.

You can add many event handlers of the same type to one element, i.e two "click" events.

You can add event listeners to any DOM object not only HTML elements. i.e the window object.

The addEventListener() method makes it easier to control how the event reacts to bubbling.

When using the addEventListener() method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML markup.

You can easily remove an event listener by using the removeEventListener() method.

## Syntax

element.addEventListener(event, function, useCapture);

The first parameter is the type of the event (like "click" or "mousedown").

The second parameter is the function we want to call when the event occurs.

The third parameter is a boolean value specifying whether to use event bubbling or event capturing. This parameter is optional.

|  |  |
| --- | --- |
| **Note** | Note that you don't use the "on" prefix for the event; use "click" instead of "onclick". |

## Add an Event Handler to an Element

### Example

Alert "Hello World!" when the user clicks on an element:

element.addEventListener("click", function(){ alert("Hello World!"); });

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add)

<!DOCTYPE html>

<html>

<body>

<p>This example uses the addEventListener() method to attach a click event to a button.</p>

<button id="myBtn">Try it</button>

<script>

document.getElementById("myBtn").addEventListener("click", function(){

alert("Hello World!");

});

</script>

</body>

</html>

You can also refer to an external "named" function:

### Example

Alert "Hello World!" when the user clicks on an element:

element.addEventListener("click", myFunction);  
  
function myFunction() {  
    alert ("Hello World!");  
}

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add2)

<!DOCTYPE html>

<html>

<body>

<p>This example uses the addEventListener() method to execute a function when a user clicks on a button.</p>

<button id="myBtn">Try it</button>

<script>

document.getElementById("myBtn").addEventListener("click", myFunction);

function myFunction() {

alert ("Hello World!");

}

</script>

</body>

</html>

## Add Many Event Handlers to the Same Element

The addEventListener() method allows you to add many events to the same element, without overwriting existing events:

### Example

element.addEventListener("click", myFunction);  
element.addEventListener("click", mySecondFunction);

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add_many)

<!DOCTYPE html>

<html>

<body>

<p>This example uses the addEventListener() method to add two click events to the same button.</p>

<button id="myBtn">Try it</button>

<script>

var x = document.getElementById("myBtn");

x.addEventListener("click", myFunction);

x.addEventListener("click", someOtherFunction);

function myFunction() {

alert ("Hello World!");

}

function someOtherFunction() {

alert ("This function was also executed!");

}

</script>

</body>

</html>

You can add events of different types to the same element:

### Example

element.addEventListener("mouseover", myFunction);  
element.addEventListener("click", mySecondFunction);  
element.addEventListener("mouseout", myThirdFunction);

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add_many2)

<!DOCTYPE html>

<html>

<body>

<p>This example uses the addEventListener() method to add many events on the same button.</p>

<button id="myBtn">Try it</button>

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

<script>

var x = document.getElementById("myBtn");

x.addEventListener("mouseover", myFunction);

x.addEventListener("click", mySecondFunction);

x.addEventListener("mouseout", myThirdFunction);

function myFunction() {

document.getElementById("demo").innerHTML += "Moused over!<br>";

}

function mySecondFunction() {

document.getElementById("demo").innerHTML += "Clicked!<br>";

}

function myThirdFunction() {

document.getElementById("demo").innerHTML += "Moused out!<br>";

}

</script>

</body>

</html>

## Add an Event Handler to the Window Object

The addEventListener() method allows you to add event listeners on any HTML DOM object such as HTML elements, the HTML document, the window object, or other objects that supports events, like the xmlHttpRequest object.

### Example

Add an event listener that fires when a user resizes the window:

window.addEventListener("resize", function(){  
    document.getElementById("demo").innerHTML = sometext;  
});

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_dom)

<!DOCTYPE html>

<html>

<body>

<p>This example uses the addEventListener() method on the window object.</p>

<p>Try resizing this browser window to trigger the "resize" event handler.</p>

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

<script>

window.addEventListener("resize", function(){

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

});

</script>

</body>

</html>

## Passing Parameters

When passing parameter values, use an "anonymous function" that calls the specified function with the parameters:

### Example

element.addEventListener("click", function(){ myFunction(p1, p2); });

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_parameters)

<!DOCTYPE html>

<html>

<body>

<p>This example demonstrates how to pass parameter values when using the

addEventListener() method.</p>

<p>Click the button to perform a calculation.</p>

<button id="myBtn">Try it</button>

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

<script>

var p1 = 5;

var p2 = 7;

document.getElementById("myBtn").addEventListener("click", function() {

myFunction(p1, p2);

});

function myFunction(a, b) {

var result = a \* b;

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

}

</script>

</body>

</html>

## Event Bubbling or Event Capturing?

There are two ways of event propagation in the HTML DOM, bubbling and capturing.

Event propagation is a way of defining the element order when an event occurs. If you have a <p> element inside a <div> element, and the user clicks on the <p> element, which element's "click" event should be handled first?

In bubbling the inner most element's event is handled first and then the outer: the <p> element's click event is handled first, then the <div> element's click event.

In capturing the outer most element's event is handled first and then the inner: the <div> element's click event will be handled first, then the <p> element's click event.

With the addEventListener() method you can specify the propagation type by using the "useCapture" parameter:

addEventListener(event, function, useCapture);

The default value is false, which will use the bubbling propagation, when the value is set to true, the event uses the capturing propagation.

### Example

document.getElementById("myP").addEventListener("click", myFunction, true);  
document.getElementById("myDiv").addEventListener("click", myFunction, true);

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_usecapture)

<!DOCTYPE html>

<html>

<head>

<style>

div {

background-color: coral;

border: 1px solid;

padding: 50px;

}

</style>

</head>

<body>

<p>This example demonstrates the difference between bubbling and capturing when adding event listeners.</p>

<div id="myDiv">

<p id="myP">Click this paragraph, I am Bubbling.</p>

</div><br>

<div id="myDiv2">

<p id="myP2">Click this paragraph, I am Capturing.</p>

</div>

<script>

document.getElementById("myP").addEventListener("click", function() {

alert("You clicked the P element!");

}, false);

document.getElementById("myDiv").addEventListener("click", function() {

alert("You clicked the DIV element!");

}, false);

document.getElementById("myP2").addEventListener("click", function() {

alert("You clicked the P element!");

}, true);

document.getElementById("myDiv2").addEventListener("click", function() {

alert("You clicked the DIV element!");

}, true);

</script>

</body>

</html>

## The removeEventListener() method

The removeEventListener() method removes event handlers that have been attached with the addEventListener() method:

### Example

element.removeEventListener("mousemove", myFunction);

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_remove)

<!DOCTYPE html>

<html>

<head>

<style>

#myDIV {

background-color: coral;

border: 1px solid;

padding: 50px;

color: white;

}

</style>

</head>

<body>

<div id="myDIV">This div element has an onmousemove event handler that displays a random number every time you move your mouse inside this orange field.

<p>Click the button to remove the DIV's event handler.</p>

<button onclick="removeHandler()" id="myBtn">Try it</button>

</div>

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

<script>

document.getElementById("myDIV").addEventListener("mousemove", myFunction);

function myFunction() {

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

}

function removeHandler() {

document.getElementById("myDIV").removeEventListener("mousemove", myFunction);

}

</script>

</body>

</html>

## Browser Support

The numbers in the table specifies the first browser version that fully supports these methods.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** |  |  |  |  |  |
| addEventListener() | 1.0 | 9.0 | 1.0 | 1.0 | 7.0 |
| removeEventListener() | 1.0 | 9.0 | 1.0 | 1.0 | 7.0 |

**Note:** The addEventListener() and removeEventListener() methods are not supported in IE 8 and earlier versions and Opera 6.0 and earlier versions. However, for these specific browser versions, you can use the attachEvent() method to attach an event handlers to the element, and the detachEvent() method to remove it:

element.attachEvent(event, function);element.detachEvent(event, function);

### Example

Cross-browser solution:

var x = document.getElementById("myBtn");  
if (x.addEventListener) {                    // For all major browsers, except IE 8 and earlier  
    x.addEventListener("click", myFunction);  
} else if (x.attachEvent) {                  // For IE 8 and earlier versions  
    x.attachEvent("onclick", myFunction);  
}

[Try it yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_crossbrowser)

<!DOCTYPE html>

<html>

<body>

<p>The addEventListener() method is not supported Internet Explorer 8 and earlier versions.</p>

<p>This example demonstrates a solution that will work for all browsers.</p>

<button id="myBtn">Try it</button>

<script>

var x = document.getElementById("myBtn");

if (x.addEventListener) {

x.addEventListener("click", myFunction);

} else if (x.attachEvent) {

x.attachEvent("onclick", myFunction);

}

function myFunction() {

alert("Hello World!");

}

</script>

</body>

</html>

## HTML DOM Event Object Reference

For a list of all HTML DOM events, look at our complete [HTML DOM Event Object Reference](http://www.w3schools.com/jsref/dom_obj_event.asp).

[« Previous](http://www.w3schools.com/js/js_htmldom_events.asp)

# JavaScript HTML DOM Navigation

[« Previous](http://www.w3schools.com/js/js_htmldom_eventlistener.asp)

[Next Chapter »](http://www.w3schools.com/js/js_htmldom_nodes.asp)

With the HTML DOM, you can navigate the node tree using node relationships.

## DOM Nodes

According to the W3C HTML DOM standard, everything in an HTML document is a node:

* The entire document is a document node
* Every HTML element is an element node
* The text inside HTML elements are text nodes
* Every HTML attribute is an attribute node
* All comments are comment nodes



With the HTML DOM, all nodes in the node tree can be accessed by JavaScript.

New nodes can be created, and all nodes can be modified or deleted.

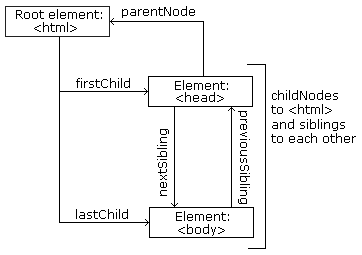
## Node Relationships

The nodes in the node tree have a hierarchical relationship to each other.

The terms parent, child, and sibling are used to describe the relationships.

* In a node tree, the top node is called the root (or root node)
* Every node has exactly one parent, except the root (which has no parent)
* A node can have a number of children
* Siblings (brothers or sisters) are nodes with the same parent

<html>  
  
  <head>  
      <title>DOM Tutorial</title>  
  </head>  
  
  <body>  
      <h1>DOM Lesson one</h1>  
      <p>Hello world!</p>  
  </body>  
  
</html>



From the HTML above you can read:

* <html> is the root node
* <html> has no parents
* <html> is the parent of <head> and <body>
* <head> is the first child of <html>
* <body> is the last child of <html>

and:

* <head> has one child: <title>
* <title> has one child (a text node): "DOM Tutorial"
* <body> has two children: <h1> and <p>
* <h1> has one child: "DOM Lesson one"
* <p> has one child: "Hello world!"
* <h1> and <p> are siblings

## Navigating Between Nodes

You can use the following node properties to navigate between nodes with JavaScript:

* parentNode
* childNodes[nodenumber]
* firstChild
* lastChild
* nextSibling
* previousSibling

## Warning !

A common error in DOM processing is to expect an element node to contain text.

In this example: **<title>DOM Tutorial</title>**, the element node < title> does not contain text. It contains a **text node** with the value "DOM Tutorial".

The value of the text node can be accessed by the node's **innerHTML** property, or the **nodeValue**.

## Child Nodes and Node Values

In addition to the innerHTML property, you can also use the childNodes and nodeValue properties to get the content of an element.

The following example collects the node value of an <h1> element and copies it into a <p> element:

### Example

<html>  
<body>  
  
<h1 id="intro">My First Page</h1>  
  
<p id="demo">Hello!</p>  
  
<script>  
var myText = document.getElementById("intro").childNodes[0].nodeValue;  
document.getElementById("demo").innerHTML = myText;  
</script>  
  
</body>  
</html>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_nodevalue)

In the example above, getElementById is a method, while childNodes and nodeValue are properties.

In this tutorial we use the innerHTML property. However, learning the method above is useful for understanding the tree structure and the navigation of the DOM.

Using the firstChild property is the same as using childNodes[0]:

### Example

<html>  
<body>  
  
<h1 id="intro">My First Page</h1>  
  
<p id="demo">Hello World!</p>  
  
<script>  
myText = document.getElementById("intro").firstChild.nodeValue;  
document.getElementById("demo").innerHTML = myText;  
</script>  
  
</body>  
</html>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_firstchild)

## DOM Root Nodes

There are two special properties that allow access to the full document:

* document.body - The body of the document
* document.documentElement - The full document

### Example

<html>  
<body>  
  
<p>Hello World!</p>  
<div>  
<p>The DOM is very useful!</p>  
<p>This example demonstrates the <b>document.body</b> property.</p>  
</div>  
  
<script>  
alert(document.body.innerHTML);  
</script>  
  
</body>  
</html>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_body)

### Example

<html>  
<body>  
  
<p>Hello World!</p>  
<div>  
<p>The DOM is very useful!</p>  
<p>This example demonstrates the <b>document.documentElement</b> property.</p>  
</div>  
  
<script>  
alert(document.documentElement.innerHTML);  
</script>  
  
</body>  
</html>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_document)

## The nodeName Property

The nodeName property specifies the name of a node.

* nodeName is read-only
* nodeName of an element node is the same as the tag name
* nodeName of an attribute node is the attribute name
* nodeName of a text node is always #text
* nodeName of the document node is always #document

**Note:** nodeName always contains the uppercase tag name of an HTML element.

## The nodeValue Property

The nodeValue property specifies the value of a node.

* nodeValue for element nodes is undefined
* nodeValue for text nodes is the text itself
* nodeValue for attribute nodes is the attribute value

## The nodeType Property

The nodeType property returns the type of node. nodeType is read only.

The most important node types are:

|  |  |
| --- | --- |
| **Element type** | **NodeType** |
| Element | 1 |
| Attribute | 2 |
| Text | 3 |
| Comment | 8 |
| Document | 9 |

[« Previous](http://www.w3schools.com/js/js_htmldom_eventlistener.asp)

# JavaScript HTML DOM Elements (Nodes)

[« Previous](http://www.w3schools.com/js/js_htmldom_navigation.asp)

[Next Chapter »](http://www.w3schools.com/js/js_htmldom_nodelist.asp)

Adding and Removing Nodes (HTML Elements)

## Creating New HTML Elements (Nodes)

To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

### Example

<div id="div1">  
<p id="p1">This is a paragraph.</p>  
<p id="p2">This is another paragraph.</p>  
</div>  
  
<script>  
var para = document.createElement("p");  
var node = document.createTextNode("This is new.");  
para.appendChild(node);  
  
var element = document.getElementById("div1");  
element.appendChild(para);  
</script>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_elementcreate)

<!DOCTYPE html>

<html>

<body>

<div id="div1">

<p id="p1">This is a paragraph.</p>

<p id="p2">This is another paragraph.</p>

</div>

<script>

var para = document.createElement("p");

var node = document.createTextNode("This is new.");

para.appendChild(node);

var element = document.getElementById("div1");

element.appendChild(para);

</script>

</body>

</html>

## Example Explained

This code creates a new <p> element:

var para = document.createElement("p");

To add text to the <p> element, you must create a text node first. This code creates a text node:

var node = document.createTextNode("This is a new paragraph.");

Then you must append the text node to the <p> element:

para.appendChild(node);

Finally you must append the new element to an existing element.

This code finds an existing element:

var element = document.getElementById("div1");

This code appends the new element to the existing element:

element.appendChild(para);

## Creating new HTML Elements - insertBefore()

The appendChild() method in the previous example, appended the new element as the last child of the parent.

If you don't want that you can use the insertBefore() method:

### Example

<div id="div1">  
<p id="p1">This is a paragraph.</p>  
<p id="p2">This is another paragraph.</p>  
</div>  
  
<script>  
var para = document.createElement("p");  
var node = document.createTextNode("This is new.");  
para.appendChild(node);  
  
var element = document.getElementById("div1");  
var child = document.getElementById("p1");  
element.insertBefore(para,child);  
</script>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_elementcreate2)

<!DOCTYPE html>

<html>

<body>

<div id="div1">

<p id="p1">This is a paragraph.</p>

<p id="p2">This is another paragraph.</p>

</div>

<script>

var para = document.createElement("p");

var node = document.createTextNode("This is new.");

para.appendChild(node);

var element = document.getElementById("div1");

var child = document.getElementById("p1");

element.insertBefore(para,child);

</script>

</body>

</html>

## Removing Existing HTML Elements

To remove an HTML element, you must know the parent of the element:

### Example

<div id="div1">  
<p id="p1">This is a paragraph.</p>  
<p id="p2">This is another paragraph.</p>  
</div>  
  
<script>  
var parent = document.getElementById("div1");  
var child = document.getElementById("p1");  
parent.removeChild(child);  
</script>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_elementremove)

<!DOCTYPE html>

<html>

<body>

<div id="div1">

<p id="p1">This is a paragraph.</p>

<p id="p2">This is another paragraph.</p>

</div>

<script>

var parent = document.getElementById("div1");

var child = document.getElementById("p1");

parent.removeChild(child);

</script>

</body>

</html>

## Example Explained

This HTML document contains a <div> element with two child nodes (two <p> elements):

<div id="div1">  
< p id="p1">This is a paragraph.</p>  
< p id="p2">This is another paragraph.</p>  
< /div>

Find the element with id="div1":

var parent = document.getElementById("div1");

Find the <p> element with id="p1":

var child = document.getElementById("p1");

Remove the child from the parent:

parent.removeChild(child);

|  |  |
| --- | --- |
| **Note** | It would be nice to be able to remove an element without referring to the parent. But sorry. The DOM needs to know both the element you want to remove, and its parent. |

Here is a common workaround: Find the child you want to remove, and use its parentNode property to find the parent:

var child = document.getElementById("p1");  
child.parentNode.removeChild(child);

## Replacing HTML Elements

To replace an element to the HTML DOM, use the replaceChild() method:

### Example

<div id="div1">  
<p id="p1">This is a paragraph.</p>  
<p id="p2">This is another paragraph.</p>  
</div>  
  
<script>  
var para = document.createElement("p");  
var node = document.createTextNode("This is new.");  
para.appendChild(node);  
  
var parent = document.getElementById("div1");  
var child = document.getElementById("p1");  
parent.replaceChild(para,child);  
</script>

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_elementreplace)

<!DOCTYPE html>

<html>

<body>

<div id="div1">

<p id="p1">This is a paragraph.</p>

<p id="p2">This is another paragraph.</p>

</div>

<script>

var parent = document.getElementById("div1");

var child = document.getElementById("p1");

var para = document.createElement("p");

var node = document.createTextNode("This is new.");

para.appendChild(node);

parent.replaceChild(para,child);

</script>

</body>

</html>

[« Previous](http://www.w3schools.com/js/js_htmldom_navigation.asp)

[Next Chapter »](http://www.w3schools.com/js/js_htmldom_nodelist.asp)

# JavaScript HTML DOM Node List

[« Previous](http://www.w3schools.com/js/js_htmldom_nodes.asp)

[Next Chapter »](http://www.w3schools.com/js/js_window.asp)

A node list is a collection of nodes

## HTML DOM Node List

The getElementsByTagName() method returns a **node list**. A node list is an array-like collection of nodes.

The following code selects all <p> nodes in a document:

### Example

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

The nodes can be accessed by an index number. To access the second <p> node you can write:

y = x[1];

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_nodelist)

<!DOCTYPE html>

<html>

<body>

<p>Hello World!</p>

<p>The DOM is very useful!</p>

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

<script>

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

document.getElementById("demo").innerHTML =

"The innerHTML of the second paragraph is: " +

myNodelist[1].innerHTML;

</script>

</body>

</html>

**Note:** The index starts at 0.

## HTML DOM Node List Length

The length property defines the number of nodes in a node list:

### Example

var myNodelist = document.getElementsByTagName("p");  
document.getElementById("demo").innerHTML = myNodelist.length;

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_nodelist_length)

<!DOCTYPE html>

<html>

<body>

<p>Hello World!</p>

<p>How many paragraphs in this document?</p>

<p>This example demonstrates the length property of a nodelist.</p>

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

<script>

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

document.getElementById("demo").innerHTML = myNodelist.length;

</script>

</body>

</html>

Example explained:

1. Get all <p> elements in a node list
2. Display the length of the node list

The length property is useful when you want to loop through the nodes in a node list:

### Example

Change the background color of all <p> elements in a node list:

var myNodelist = document.getElementsByTagName("p");  
var i;  
for (i = 0; i < myNodelist.length; i++) {  
    myNodelist[i].style.backgroundColor = "red";  
}

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_nodelist_loop)

<!DOCTYPE html>

<html>

<body>

<p>This is a p element</p>

<p>This is also a p element.</p>

<p>This is also a p element - Click the button to change the background color of all p elements in this document.</p>

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

<script>

function myFunction() {

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

var i;

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

myNodelist[i].style.backgroundColor = "red";

}

}

</script>

</body>

</html>

|  |  |
| --- | --- |
| **Note** | **A node list is not an array!**  A node list may look like an array, but it is not. You can loop through the node list and refer to its nodes like an array. However, you cannot use Array Methods, like valueOf() or join() on the node list. |