**Introduction to HTML**

HTML is an acronym which stands for **Hyper Text Markup Language** which is used for creating web pages and web applications.

**Hyper Text:** HyperText simply means "Text linked with Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

**Markup language:** A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

**Web Page:** A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. **With the help of HTML only, we can create static web pages**.

Hence, HTML is a markup language which is used for creating attractive web pages with the help of styling, and which looks in a nice format on a web browser. An HTML document is made of many HTML tags and each HTML tag contains different content.

Sample HTML Page

**<html>**

**<head>**

**<title>Web page title</title>**

**</head>**

**<body>**

**<h1>Write Your First Heading</h1>**

**<p>Write Your First Paragraph.</p>**

**</body>**

**</html>**

**Editor - Visual Studio Code**

**Building blocks of HTML**

An HTML document consist of its basic building blocks which are:

**Tags:** An HTML tag surrounds the content and apply meaning to it. It is written between < and > brackets.

**Attribute:** An attribute in HTML provides extra information about the element, and it is applied within the start tag. An HTML attribute contains two fields: name & value.

**Syntax**

<tag name attribute\_name= " attr\_value"> content </ tag name>

**Elements:** An HTML element is an individual component of an HTML file. In an HTML file, everything written within tags are termed as HTML elements.

HTML Tag Examples

<p> Paragraph Tag </p>

<h2> Heading Tag </h2>

<b> Bold Tag </b>

<i> Italic Tag </i>

<u> Underline Tag</u>

—————————————————————————————————-

**HTML Attribute**

HTML attributes are special words which provide additional information about the elements. They are the modifier of the HTML element.

Each element or tag can have multiple attributes, which defines the behaviour of that element.

The Attribute should always be applied with its name and value pair.

The Attributes name and values are case sensitive, and it is recommended by W3C that it should be written in Lowercase only.

You can add multiple attributes in one HTML element, but need to give space between two attributes.

**Syntax**

<element attribute\_name="value">content</element>

**Example**

<html>

<head>

</head>

<body>

<h1> This is Style attribute</h1>

<p style="height: 50px; color: blue">It will add style property in element</p>

<p style="color: red">It will change the color of content</p>

</body>

</html>

**The title attribute**

**Description:** The title attribute is used as text tooltip in most of the browsers. It display its text when user move the cursor over a link or any text. You can use it with any text or link to show the description about that link or text. In our example, we are taking this with paragraph tag and heading tag.

**Example**

<h1 title="This is heading tag">Example of title attribute</h1>

<p title="This is paragraph tag">Move the cursor over the heading and paragraph, and you will see a description as a tooltip</p>

**HTML Elements**

An HTML file is made of elements. These elements are responsible for creating web pages and define content in that webpage. Technically, an element is a collection of start tag, attributes, end tag, content between them.

Example:

<p> Hello world!!! </p>

**Some Void elements** are <br> (represents a line break) , <hr>(represents a horizontal line), etc.

**Nested HTML Elements**: HTML can be nested, which means an element can contain another element.

**Block-level element:**

These are the elements, which structure main part of web page, by dividing a page into blocks.

A block-level element always start with new line and takes the full width of web page, from left to right.

These elements can contain block-level as well as inline elements.

Following are the block-level elements in HTML.

<address>, <article>, <aside>, <blockquote>, <canvas>, <dd>, <div>, <dl>, <dt>, <fieldset>, <figcaption>, <figure>, <footer>, <form>, <h1>-<h6>, <header>, <hr>, <li>, <main>, <nav>, <noscript>, <ol>, <output>, <p>, <pre>, <section>, <table>, <tfoot>, <ul> and <video>.

**Formatting Elements**

**<b>** This is a physical tag, which is used to bold the text written between it.

**<strong>** This is a logical tag, which tells the browser that the text is important.

**<i>** This is a physical tag which is used to make text italic.

**<em>** This is a logical tag which is used to display content in emphasized similar to italic.

**<mark>** This tag is used to highlight text.

**<u>** This tag is used to underline text written between it.

**<sup>** It displays the content slightly above the normal line.

**<sub>** It displays the content slightly below the normal line.

**<del>** This tag is used to display the deleted content.

**<ins>** This tag displays the content which is added

**<big>** This tag is used to increase the font size by one conventional unit.

**<small>** This tag is used to decrease the font size by one unit from base font size.

**Example** (htm2\_formatting\_tags.html)

————————————————————

<html>

<p> <b>Write Your First Paragraph in bold text.</b></p>

<p><strong>This is an important content</strong>, and this is normal content</p>

<p> <i>Write Your First Paragraph in italic text.</i></p>

<p><em>This is an important content</em>, which displayed in italic font.</p>

<h2> I want to put a <mark> Mark</mark> on your face</h2>

<p> <u>Write Your First Paragraph in underlined text.</u></p>

<p>Hello <tt>Write Your First Paragraph in monospaced font.</tt></p>

<p>Hello <sup>Write Your First Paragraph in superscript.</sup></p>

<p>Hello <sub>Write Your First Paragraph in subscript.</sub></p>

<p>Hello <del>Delete your first paragraph.</del></p>

<p> <del>Delete your first paragraph.</del><ins>Write another paragraph.</ins></p>

<p>Hello <big>Write the paragraph in larger font.</big></p>

<p>Hello <small>Write the paragraph in smaller font.</small></p>

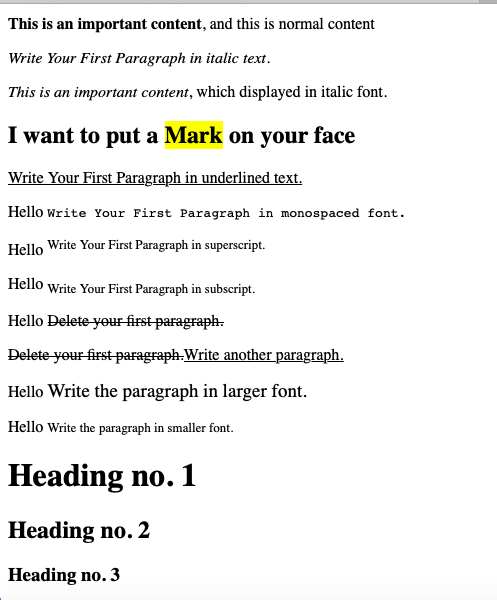
<h1>Heading no. 1</h1>

<h2>Heading no. 2</h2>

<h3>Heading no. 3</h3>

<h4>Heading no. 4</h4>

<h5>Heading no. 5</h5>

<h6>Heading no. 6</h6>

</html>

Output

——————-

**HTML Paragraph**

HTML paragraph or HTML p tag is used to define a paragraph in a webpage.If you put a lot of spaces inside the HTML p tag, browser removes extra spaces and extra line while displaying the page. The browser counts number of spaces and lines as a single one.

**Example**

<p>This is first paragraph.</p>

<p>

I am

going to provide

you a tutorial on HTML

and hope that it will

be very beneficial for you.

</p>

**An HTML <br>** tag is used for line break and it can be used with paragraph elements

<p><br>Papa and mama, and baby and Dot,

<br>Willie and me?the whole of the lot

<br>Of us all went over in Bimberlie's sleigh,

<br>To grandmama's house on Christmas day.

</p>

**An HTML <hr>** tag is used to apply a horizontal line between two statements or two paragraphs.

**The HTML <blockquote>** element shows that the enclosed content is quoted from another source. The Source URL can be given using the cite attribute, and text representation of source can display using <cite> ..... </cite>element.

**Example**

<blockquote cite="https://www.keepinspiring.me/famous-quotes/"><p>?The first step toward success is taken when you refuse to be a captive of the environment in which you first find yourself.?</p></blockquote>

<cite>-Mark Caine</cite>

**The HTML <code>** </code> element is used to display the part of computer code. It will display the content in monospaced font.

<p>First Java program</p>

<p><code>class Simple{ public static void main(String args[]){

System.out.println("Hello Java"); }} </code>

</p>

**The HTML <pre>** element defines pre-formatted text.

The text inside a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks:

**Example**

<pre>

My Bonnie lies over the ocean.

My Bonnie lies over the sea.

My Bonnie lies over the ocean.

Oh, bring back my Bonnie to me.

</pre>

**The HTML Style Attribute**

Setting the style of an HTML element, can be done with the style attribute.

The HTML style attribute has the following syntax:

<tagname style="property:value;">

The property is a CSS property. The value is a CSS value.

Example

<body style="background-color:powderblue;">

<h1>This is a plane heading</h1>

<p>This is a plane paragraph.</p>

<h1 style="font-size:20;color: blueviolet">This is a styled heading</h1>

<p style="font-size:60;color: chocolate">This is a styled paragraph.</p>

</body>

Comments

<!-- Write your comments here —>

HTML Links

The **HTML anchor tag** defines *a hyperlink that links one page to another page*. It can create hyperlink to other web page as well as files, location, or any URL. The "href" attribute ok anchor tag links to destination page or URL.

**<a** href="second.html"**>**Click for Second Page**</a>**

second.html should be in the same folder

<a href="<https://codebetter.in/>">Visit our HTML tutorial</a>

***target*** Attribute

The target attribute specifies where to open the linked document.

The target attribute can have one of the following values:

* \_blank - Opens the linked document in a new window or tab
* \_self - Opens the linked document in the same window/tab as it was clicked (this is default)
* \_parent - Opens the linked document in the parent frame
* \_top - Opens the linked document in the full body of the window
* *framename* - Opens the linked document in a named frame

<a href="<https://codebetter.in/>" target=“\_top”>Learn coding!</a>

Image Links

<a href="default.asp">

  <img src="smiley.gif" alt="HTML tutorial" style=“width:42px;height:42px;border:0;”></a>

Link Titles

The title attribute specifies a tooltip text when the mouse moves over the element.

<a href="https://www.codebetter.in/html/" title=“Learn HTML”>Visit our HTML Tutorial</a>

This example links to a page located in the html folder on the current web site:

<a href="html/html1.html">HTML tutorial</a>

Create a Bookmark

HTML bookmarks are used to allow readers to jump to specific parts of a Web page.

Bookmarks can be useful if your webpage is very long.

To make a bookmark, you must first create the bookmark, and then add a link to it.

When the link is clicked, the page will scroll to the location with the bookmark.

Example

First, create a bookmark with the id attribute:

<h2 id="C4">Chapter 4</h2>

Then, add a link to the bookmark ("Jump to Chapter 4"), from within the same page:

<a href="#C4">Jump to Chapter 4</a>

Or, add a link to the bookmark ("Jump to Chapter 4"), from another page:

Example

<a href="html\_demo.html#C4">Jump to Chapter 4</a>

HTML Link Colors

By default, a link will appear like this (in all browsers):

* An unvisited link is underlined and blue
* A visited link is underlined and purple
* An active link is underlined and red

You can change the default colors, by using CSS:

Example

<style>

a:link {

  color: green;

  background-color: transparent;

  text-decoration: none;

}

a:visited {

  color: pink;

  background-color: transparent;

  text-decoration: none;

}

a:hover {

  color: red;

  background-color: transparent;

  text-decoration: underline;

}

a:active {

  color: yellow;

  background-color: transparent;

  text-decoration: underline;

}

</style>

A links can also be styled as a button, by using CSS:

[This is a link](javascript:void(0))

Example

<style>

a:link, a:visited {

  background-color: #f44336;

  color: white;

  padding: 15px 25px;

  text-align: center;

  text-decoration: none;

  display: inline-block;

}

a:hover, a:active {

  background-color: red;

}

</style>

———————————————————————————-

Image

<img src="offer6.png" alt=“20 % chashback”>

The <img> tag is empty, it contains attributes only, and does not have a closing tag.

The src attribute specifies the URL (web address) of the image:

The alt attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).

The value of the alt attribute should describe the image:

<img src="offer9.png" alt="Italian Trulli”>

Width and Height

<img src="offer6.png" alt="Girl in a jacket" style="width:100px;height:100px;">

<img src="offer9.png" alt="Girl in a jacket" width="300" height="300">

<img src="/images/html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">

Image as a Link

<a href="default.asp">

<img src=“offer3.png" alt="HTML tutorial" style="width:42px;height:42px;border:0;">

</a>

**HTML Table**

**—————————-**

An HTML table is defined with the <table> tag.

Each table row is defined with the <tr> tag. A table header is defined with the <th> tag. By default, table headings are bold and centered. A table data/cell is defined with the <td> tag.

<table >

<tr>

<th>Firstname</th>

<th>Lastname</th>

<th>Age</th>

</tr>

<tr>

<td>Jill</td>

<td>Smith</td>

<td>50</td>

</tr>

<tr>

<td>Eve</td>

<td>Jackson</td>

<td>94</td>

</tr>

</table>

<table style="width: 300px;" border=“1" align="center">

<tr>

<th>Name</th>

<th>City</th>

<th>Gender</th>

</tr>

<tr>

<td align="center">Ankit</td>

<td align="center">Pune</td>

<td align="center"> M</td>

</tr>

<tr>

<td align="center">Reena</td>

<td align="center">Delhi</td>

<td align="center">F</td>

</tr>

</table>

rowspan and caption

<table style="width:100%" border="1">

<caption>Contact Information</caption>

<tr>

<th>Name:</th>

<td>Bill Gates</td>

</tr>

<tr>

<th rowspan="2">Telephone:</th>

<td>55577854</td>

</tr>

<tr>

<td>55577855</td>

</tr>

</table>

HTML Lists

Unordered HTML List

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

The list items will be marked with bullets (small black circles) by default:

Example

<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

The CSS list-style-type property is used to define the style of the list item marker:

<ul style="list-style-type:lower-roman;">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

**Ordered HTML List**

An ordered list starts with the **<ol>** tag. Each list item starts with the <li> tag.

The list items will be marked with numbers by default:

Example

<ol>

<li>Pune</li>

<li>Mumbai</li>

<li>Delhi</li>

</ol>

The type attribute of the <ol> tag, defines the type of the list item marker:

Type Description

type="1" The list items will be numbered with numbers (default)

type="A" The list items will be numbered with uppercase letters

type="a" The list items will be numbered with lowercase letters

type="I" The list items will be numbered with uppercase roman numbers

type="i" The list items will be numbered with lowercase roman numbers

Numbers:

<ol type="1">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

<ol start="50">

  <li>Coffee</li>

  <li>Tea</li>

  <li>Milk</li>

</ol>

Description Lists

A description list is a list of terms, with a description of each term.

The [<dl>](https://www.w3schools.com/tags/tag_dl.asp) tag defines the description list, the [<dt>](https://www.w3schools.com/tags/tag_dt.asp) tag defines the term (name), and the [<dd>](https://www.w3schools.com/tags/tag_dd.asp) tag describes each term:

Example

<dl>

  <dt>Coffee</dt>

  <dd>- black hot drink</dd>

  <dt>Milk</dt>

  <dd>- white cold drink</dd>

</dl>

Horizontal List with CSS

HTML lists can be styled in many different ways with CSS.

One popular way is to style a list horizontally, to create a navigation menu:

Example

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333333;

}

li {

float: left;

}

li a {

display: block;

color: white;

text-align: center;

padding: 16px;

text-decoration: none;

}

li a:hover {

background-color: #111111;

}

</style>

</head>

<body>

<ul>

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

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

</body>

</html>

HTML Block and Inline Elements

Every HTML element has a default display value depending on what type of element it is.

The two display values are: block and inline.

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The <div> element is a block-level element.

Example

<div>Hello World</div>

[<address>](https://www.w3schools.com/tags/tag_address.asp) [<article>](https://www.w3schools.com/tags/tag_article.asp) [<aside>](https://www.w3schools.com/tags/tag_aside.asp) [<blockquote>](https://www.w3schools.com/tags/tag_blockquote.asp)

[<canvas>](https://www.w3schools.com/tags/tag_canvas.asp) [<dd>](https://www.w3schools.com/tags/tag_dd.asp) [<div>](https://www.w3schools.com/tags/tag_div.asp) [<dl>](https://www.w3schools.com/tags/tag_dl.asp) [<dt>](https://www.w3schools.com/tags/tag_dt.asp) [<fieldset>](https://www.w3schools.com/tags/tag_fieldset.asp) [<figcaption>](https://www.w3schools.com/tags/tag_figcaption.asp) [<figure>](https://www.w3schools.com/tags/tag_figure.asp) [<footer>](https://www.w3schools.com/tags/tag_footer.asp) [<form>](https://www.w3schools.com/tags/tag_form.asp) [<h1>-<h6>](https://www.w3schools.com/tags/tag_hn.asp)

[<header>](https://www.w3schools.com/tags/tag_header.asp) [<hr>](https://www.w3schools.com/tags/tag_hr.asp) [<li>](https://www.w3schools.com/tags/tag_li.asp) [<main>](https://www.w3schools.com/tags/tag_main.asp) [<nav>](https://www.w3schools.com/tags/tag_nav.asp) [<noscript>](https://www.w3schools.com/tags/tag_noscript.asp) [<ol>](https://www.w3schools.com/tags/tag_ol.asp)

[<p>](https://www.w3schools.com/tags/tag_p.asp) [<pre>](https://www.w3schools.com/tags/tag_pre.asp) [<section>](https://www.w3schools.com/tags/tag_section.asp) [<table>](https://www.w3schools.com/tags/tag_table.asp) [<tfoot>](https://www.w3schools.com/tags/tag_tfoot.asp) [<ul>](https://www.w3schools.com/tags/tag_ul.asp) [<video>](https://www.w3schools.com/tags/tag_video.asp)

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

<span>Hello World</span>

[<a>](https://www.w3schools.com/tags/tag_a.asp) [<abbr>](https://www.w3schools.com/tags/tag_abbr.asp) [<acronym>](https://www.w3schools.com/tags/tag_acronym.asp) [<b>](https://www.w3schools.com/tags/tag_b.asp) [<bdo>](https://www.w3schools.com/tags/tag_bdo.asp) [<big>](https://www.w3schools.com/tags/tag_big.asp) [<br>](https://www.w3schools.com/tags/tag_br.asp) [<button>](https://www.w3schools.com/tags/tag_button.asp)

[<cite>](https://www.w3schools.com/tags/tag_cite.asp) [<code>](https://www.w3schools.com/tags/tag_code.asp) [<dfn>](https://www.w3schools.com/tags/tag_dfn.asp) [<em>](https://www.w3schools.com/tags/tag_em.asp) [<i>](https://www.w3schools.com/tags/tag_i.asp) [<img>](https://www.w3schools.com/tags/tag_img.asp) [<input>](https://www.w3schools.com/tags/tag_input.asp) [<kbd>](https://www.w3schools.com/tags/tag_kbd.asp) [<label>](https://www.w3schools.com/tags/tag_label.asp) [<map>](https://www.w3schools.com/tags/tag_map.asp) [<object>](https://www.w3schools.com/tags/tag_object.asp) [<output>](https://www.w3schools.com/tags/tag_output.asp) [<q>](https://www.w3schools.com/tags/tag_q.asp) [<samp>](https://www.w3schools.com/tags/tag_samp.asp) [<script>](https://www.w3schools.com/tags/tag_script.asp) [<select>](https://www.w3schools.com/tags/tag_select.asp) [<small>](https://www.w3schools.com/tags/tag_small.asp) [<span>](https://www.w3schools.com/tags/tag_span.asp) [<strong>](https://www.w3schools.com/tags/tag_strong.asp) [<sub>](https://www.w3schools.com/tags/tag_sub.asp)

[<sup>](https://www.w3schools.com/tags/tag_sup.asp) [<textarea>](https://www.w3schools.com/tags/tag_textarea.asp) [<time>](https://www.w3schools.com/tags/tag_time.asp) [<tt>](https://www.w3schools.com/tags/tag_tt.asp) [<var>](https://www.w3schools.com/tags/tag_var.asp)

The <div> Element

The <div> element is often used as a container for other HTML elements.

The <div> element has no required attributes, but style, class and id are common.

When used together with CSS, the <div> element can be used to style blocks of content:

Example

<div style="background-color:black;color:white;padding:20px;">

  <h2>London</h2>

  <p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>

</div>

**page\_design\_with\_div**

<html>

<body>

<div style="background-color: darkgoldenrod; width: 100%;height: 20%;"></div>

<div style="background-color: rgb(224, 233, 208);width: 100%;height: 60%;">

<div style="background-color: rgb(11, 158, 184);width: 25%;height: 100%;float: left;"></div>

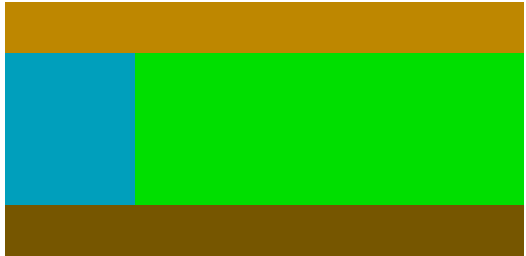
<div style="background-color: rgb(63, 212, 33);width: 75%;height: 100%;float: right;"></div>

</div>

<div style="background-color: rgb(114, 85, 11); width: 100%;height: 20%;"></div>

</body>

</html>



The <span> Element

The <span> element is often used as a container for some text.

The <span> element has no required attributes, but style, class and id are common.

When used together with CSS, the <span> element can be used to style parts of the text:

Example

<h1>My <span style="color:red">Important</span> Heading</h1>

Using The id Attribute

The id attribute specifies a unique id for an HTML element (the value must be unique within the HTML document).

The id value can be used by CSS and JavaScript to perform certain tasks for the element with the specific id value.

<h1 id="myHeader">My Header</h1>

Using The class Attribute

The HTML class attribute is used to define equal styles for elements with the same class name.

So, all HTML elements with the same class attribute will get the same style.

<!DOCTYPE html>

<html>

<head>

<style>

.cities {

  background-color: black;

  color: white;

  margin: 20px;

  padding: 20px;

}

</style>

</head>

<body>

<div class="cities">

  <h2>London</h2>

  <p>London is the capital of England.</p>

</div>

<div class="cities">

  <h2>Paris</h2>

  <p>Paris is the capital of France.</p>

</div>

<div class="cities">

  <h2>Tokyo</h2>

  <p>Tokyo is the capital of Japan.</p>

</div>

</body>

</html>

HTML Iframes

An iframe is used to display a web page within a web page.

Example

<iframe src="demo\_iframe.htm" height="200" width="300"></iframe>

<iframe src="demo\_iframe.htm" style=“height:200px;width:300px;"></iframe>

<iframe src="demo\_iframe.htm" style=“border:none;"></iframe>

<iframe src="demo\_iframe.htm" style="border:2px solid red;”></iframe>

Iframe - Target for a Link

An iframe can be used as the target frame for a link.

The target attribute of the link must refer to the name attribute of the iframe:

Example

<iframe src="demo\_iframe.htm" name="iframe\_a"></iframe>

<p><a href="https://www.w3schools.com" target="iframe\_a">W3Schools.com</a></p>

The <form> Element

The HTML <form> element defines a form that is used to collect user input:

<form>

.

*form elements*

.

</form>

An HTML form contains **form elements**.

Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.

|  |  |
| --- | --- |
| <input type="text"> | Defines a one-line text input field |
| <input type="radio"> | Defines a radio button |
| <input type="submit"> | Defines a submit button for submitting the form |

Form element attributes

The action attribute defines the action to be performed when the form is submitted.

Normally, the form data is sent to a web page on the server when the user clicks on the submit button

The method attribute specifies the HTTP method (**GET**or **POST**) to be used when submitting the form data:

<form **action=“/action\_page.php**” **target=“\_blank" method=“get”**>

……..

…….

</form>

**GET type request**

* Appends form-data into the URL in name/value pairs
* The length of a URL is limited (2048 characters)
* Never use GET to send sensitive data! (will be visible in the URL)
* Useful for form submissions where a user wants to bookmark the result
* GET is better for non-secure data, like query strings in Google

**POST type request**

* POST has no size limitations, and can be used to send large amounts of data.
* Form submissions with POST cannot be bookmarked
* The POST method does not display the submitted form data in the page address field.

<html>

<body>

<form action="/somepage.php" target="\_blank" method="GET">

First name:<br>

<input type="text" name="firstname"><br>

Last name:<br>

<input type="text" name="lastname"><br>

Gender : <br>

<input type="radio" name="gender" value="male" checked> Male<br>

<input type="radio" name="gender" value="female"> Female<br>

<input type="radio" name="gender" value="other"> Other <br>

<br>

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

</form>

</body>

</html>

HTML5 <datalist> Element

The <datalist> element specifies a list of pre-defined options for an <input> element.

Users will see a drop-down list of the pre-defined options as they input data.

The list attribute of the <input> element, must refer to the id attribute of the <datalist> element.

<form action="/action\_page.php">

  <input list="browsers">

  <datalist id="browsers">

    <option value="Internet Explorer">

    <option value="Firefox">

    <option value="Chrome">

    <option value="Opera">

    <option value="Safari">

  </datalist>

</form>

Here are the different input types you can use in HTML:

* <input type="button">
* <input type="checkbox">
* <input type="color">
* <input type="date">
* <input type="datetime-local">
* <input type="email">
* <input type="file">
* <input type="hidden">
* <input type="image">
* <input type="month">
* <input type="number">
* <input type="password">
* <input type="radio">
* <input type="range">
* <input type="reset">
* <input type="search">
* <input type="submit">
* <input type="tel">
* <input type="text">
* <input type="time">
* <input type="url">
* <input type="week">

Example form. - html\_forms.html

New HTML5 Elements

The most interesting new HTML5 elements are:

New **semantic elements** like <header>, <footer>, <article>, and <section>.

New **attributes of form elements** like number, date, time, calendar, and range.

New **graphic elements**: <svg> and <canvas>.

New **multimedia elements**: <audio> and <video>.

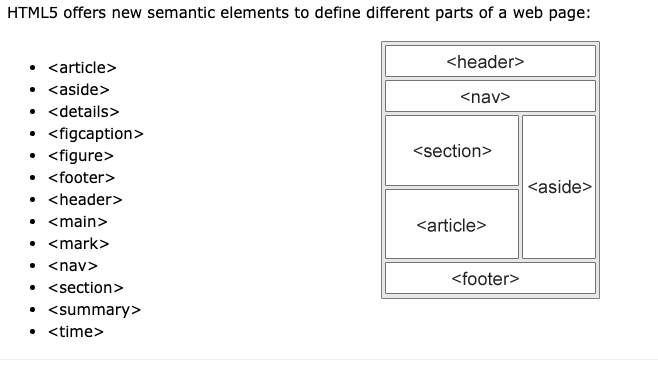
New Elements in HTML5

Below is a list of the new HTML5 elements, and a description of what they are used for.

New Semantic/Structural Elements

HTML5 offers new elements for better document structure:

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<article>](https://www.w3schools.com/tags/tag_article.asp) | Defines an article in a document |
| [<aside>](https://www.w3schools.com/tags/tag_aside.asp) | Defines content aside from the page content |
| [<bdi>](https://www.w3schools.com/tags/tag_bdi.asp) | Isolates a part of text that might be formatted in a different direction from other text outside it |
| [<details>](https://www.w3schools.com/tags/tag_details.asp) | Defines additional details that the user can view or hide |
| [<dialog>](https://www.w3schools.com/tags/tag_dialog.asp) | Defines a dialog box or window |
| [<figcaption>](https://www.w3schools.com/tags/tag_figcaption.asp) | Defines a caption for a <figure> element |
| [<figure>](https://www.w3schools.com/tags/tag_figure.asp) | Defines self-contained content |
| [<footer>](https://www.w3schools.com/tags/tag_footer.asp) | Defines a footer for a document or section |
| [<header>](https://www.w3schools.com/tags/tag_header.asp) | Defines a header for a document or section |
| [<main>](https://www.w3schools.com/tags/tag_main.asp) | Defines the main content of a document |
| [<mark>](https://www.w3schools.com/tags/tag_mark.asp) | Defines marked/highlighted text |
| [<meter>](https://www.w3schools.com/tags/tag_meter.asp) | Defines a scalar measurement within a known range (a gauge) |
| [<nav>](https://www.w3schools.com/tags/tag_nav.asp) | Defines navigation links |
| [<progress>](https://www.w3schools.com/tags/tag_progress.asp) | Represents the progress of a task |
| [<rp>](https://www.w3schools.com/tags/tag_rp.asp) | Defines what to show in browsers that do not support ruby annotations |
| [<rt>](https://www.w3schools.com/tags/tag_rt.asp) | Defines an explanation/pronunciation of characters (for East Asian typography) |
| [<ruby>](https://www.w3schools.com/tags/tag_ruby.asp) | Defines a ruby annotation (for East Asian typography) |
| [<section>](https://www.w3schools.com/tags/tag_section.asp) | Defines a section in a document |
| [<summary>](https://www.w3schools.com/tags/tag_summary.asp) | Defines a visible heading for a <details> element |
| [<time>](https://www.w3schools.com/tags/tag_time.asp) | Defines a date/time |
| [<wbr>](https://www.w3schools.com/tags/tag_wbr.asp) | Defines a possible line-break |



<section> Element

The <section> element defines a section in a document.

According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."

A home page could normally be split into sections for introduction, content, and contact information.

Example

<section>

  <h1>WWF</h1>

  <p>The World Wide Fund for Nature (WWF) is....</p>

</section>

<section>

<h1>WWF</h1>

<p>The World Wide Fund for Nature (WWF) is....</p>

</section>

<section>

<h1>WWF2</h1>

<p>The World Wide Fund for Nature (WWF) is....</p>

</section>

**The <article>** element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.

Examples of where an <article> element can be used:

* Forum post
* Blog post
* Newspaper article

<article>

<h1>What Does WWF Do?</h1>

<p>WWF's mission is to stop the degradation of our planet's natural environment, and build a future in which humans live in harmony with nature.</p>

</article>

<header> Element

The <header> element specifies a header for a document or section.

<article>

<header>

<h1>What Does WWF Do?</h1>

<p>WWF's mission:</p>

</header>

<p>WWF's mission is to stop the degradation of our planet's natural environment,

and build a future in which humans live in harmony with nature.</p>

</article>

<footer> Element

The <footer> element specifies a footer for a document or section.

<footer>

<p>Posted by: Hege Refsnes</p>

<p>Contact information: <a href="mailto:someone@example.com">

someone@example.com</a>.</p>

</footer>

<nav> Element

The <nav> element defines a set of navigation links.

<nav>

  <a href="/html/">HTML</a> |

  <a href="/css/">CSS</a> |

  <a href="/js/">JavaScript</a> |

  <a href="/jquery/">jQuery</a>

</nav>

HTML5 <aside> Element

The <aside> element defines some content aside from the content it is placed in (like a sidebar).

The <aside> content should be related to the surrounding content.

<aside>

<h4>Epcot Center</h4>

<p>The Epcot Center is a theme park in Disney World, Florida.</p>

</aside>

New Input Types

|  |  |
| --- | --- |
| **New Input Types** | **New Input Attributes** |
| * color * date * datetime * datetime-local * email * month * number * range * search * tel * time * url * week | * autocomplete * autofocus * form * formaction * formenctype * formmethod * formnovalidate * formtarget * height and width * list * min and max * multiple * pattern (regexp) * placeholder * required * step |

HTML5 - New Attribute Syntax

HTML5 allows four different syntaxes for attributes.

This example demonstrates the different syntaxes used in an <input> tag:

|  |  |
| --- | --- |
| **Type** | **Example** |
| Empty | <input type="text" value="John" **disabled**> |
| Unquoted | <input type="text" **value=John**> |
| Double-quoted | <input type="text" **value="John Doe"**> |
| Single-quoted | <input type="text" **value='John Doe'**> |

**Typical HTML 4 Page design**

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html lang="en">

<head>

<meta http-equiv="Content-Type" content="text/html;charset=utf-8">

<title>HTML4</title>

<style>

body {

  font-family: Verdana,sans-serif;

  font-size: 0.9em;

}

div#header, div#footer {

  padding: 10px;

  color: white;

  background-color: black;

}

div#content {

  margin: 5px;

  padding: 10px;

  background-color: lightgrey;

}

div.article {

  margin: 5px;

  padding: 10px;

  background-color: white;

}

div#menu ul {

  padding: 0;

}

div#menu ul li {

  display: inline;

  margin: 5px;

}

</style>

</head>

<body>

<div id="header">

  <h1>Monday Times</h1>

</div>

<div id="menu">

  <ul>

    <li>News</li>

    <li>Sports</li>

    <li>Weather</li>

  </ul>

</div>

<div id="content">

  <h2>News Section</h2>

  <div class="article">

    <h2>News Article</h2>

    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.</p>

  </div>

  <div class="article">

    <h2>News Article</h2>

    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.</p>

  </div>

</div>

<div id="footer">

  <p>&amp;copy; 2016 Monday Times. All rights reserved.</p>

</div>

</body>

</html>

Bookmarks with ID and Links

HTML bookmarks are used to allow readers to jump to specific parts of a Web page.

Bookmarks can be useful if your webpage is very long.

To make a bookmark, you must first create the bookmark, and then add a link to it.

When the link is clicked, the page will scroll to the location with the bookmark.

Example

First, create a bookmark with the id attribute:

<h2 id="C4">Chapter 4</h2>

Then, add a link to the bookmark ("Jump to Chapter 4"), from within the same page:

<a href="#C4">Jump to Chapter 4</a>

Or, add a link to the bookmark ("Jump to Chapter 4"), from another page:

Example

<a href="html\_demo.html#C4">Jump to Chapter 4</a>

**HTML Form Elements**

HTML Input Types

[Input type text](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_text)

[Input type password](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_password)

[Input type radio](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_radio)

[Input type checkbox](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_checkbox)

[Input type button](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_button)

[Input type number - with restrictions](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_number)

[Input type number - with steps](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_number_step)

[Input type date - with date picker](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_date)

[Input type date - with restrictions](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_date_max_min)

[Input type color - with color picker](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_color)

[Input type range](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_range)

[Input type month](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_month)

[Input type week](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_week)

[Input type time](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_time)

[Input type datetime](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_datetime)

[Input type datetime-local](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_datetime-local)

[Input type email](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_email)

[Input type search](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_search)

[Input type tel](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_tel)

[Input type url](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_url)

|  |  |
| --- | --- |
| [<form>](https://www.w3schools.com/tags/tag_form.asp) | Defines an HTML form for user input |
| [<input>](https://www.w3schools.com/tags/tag_input.asp) | Defines an input control |
| [<textarea>](https://www.w3schools.com/tags/tag_textarea.asp) | Defines a multiline input control (text area) |
| [<label>](https://www.w3schools.com/tags/tag_label.asp) | Defines a label for an <input> element |
| [<fieldset>](https://www.w3schools.com/tags/tag_fieldset.asp) | Groups related elements in a form |
| [<legend>](https://www.w3schools.com/tags/tag_legend.asp) | Defines a caption for a <fieldset> element |
| [<select>](https://www.w3schools.com/tags/tag_select.asp) | Defines a drop-down list |
| [<optgroup>](https://www.w3schools.com/tags/tag_optgroup.asp) | Defines a group of related options in a drop-down list |
| [<option>](https://www.w3schools.com/tags/tag_option.asp) | Defines an option in a drop-down list |
| [<button>](https://www.w3schools.com/tags/tag_button.asp) | Defines a clickable button |
| [<datalist>](https://www.w3schools.com/tags/tag_datalist.asp) | Specifies a list of pre-defined options for input controls |
| [<output>](https://www.w3schools.com/tags/tag_output.asp) | Defines the result of a calculation |

Multimedia

——————————

<!DOCTYPE html>

<html>

<body>

<video width="400" controls >

<source src="tech.3gp" type="video/mp4">

<source src="tech.3gp" type="video/ogg">

Your browser does not support HTML5 video.

</video>

<p>

Video courtesy of

<a href=“https://www.bigbuckbunny.org/" target=“\_blank" > Big Buck Bunny</a>.

</p>

<div>

<audio controls>

<source src="waka.ogg" type="audio/ogg">

<source src="waka.mp3" type="audio/mpeg">

Your browser does not support the audio element.

</audio>

</div>

</body>

</html>

HTML5 Geolocation

HTML5 WebStorage

**CSS**

CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**.

CSS **saves a lot of work**. It can control the layout of multiple web pages all at once.

CSS can be added to HTML elements in 3 ways:

* **Inline** - by using the style attribute in HTML elements
* **Internal** - by using a <style> element in the <head> section
* **External** - by using an external CSS file

**Inline CSS**

<h1 **style="color:blue;"**>This is a Blue Heading</h1>

Internal CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the <head> section of an HTML page, within a <style> element:

<html>

<head>

<style>

body {background-color: powderblue;}

h1 {color: blue;}

p {color: red;}

</style>

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>

External CSS

An external style sheet is used to define the style for many HTML pages.

**With an external style sheet, you can change the look of an entire web site, by changing one file!**

To use an external style sheet, add a link to it in the <head> section of the HTML page:

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>

**CSS Fonts**

<html>

<head>

<style>

h1 {

color: blue;

font-family: verdana;

font-size: 300%;

}

p {

color: red;

font-family: courier;

font-size: 160%;

}

</style>

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>

**Border**

p {

  border: 1px solid powderblue;

}

**Padding and Margin**

p {

  border: 1px solid powderblue;

  padding: 30px;

}

p {

  border: 1px solid powderblue;

  margin: 50px;

}

The id Attribute

To define a specific style for one special element, add an id attribute to the element:

<p id="p01">I am different</p>

then define a style for the element with the specific id:

Example

#p01 {

  color: blue;

}

**Example 2**

<html>

<head>

<style>

#myHeader {

background-color: lightblue;

color: black;

padding: 40px;

text-align: center;

}

</style>

</head>

<body>

<h2>The id Attribute</h2>

<p>Use CSS to style an element with the id "myHeader":</p>

<h1 id="myHeader">My Header</h1>

</body>

</html>

The class Attribute

To define a style for special types of elements, add a class attribute to the element:

<p class="error">I am different</p>

then define a style for the elements with the specific class:

Example

p.error {

  color: red;

}

All CSS Simple Selectors

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Example description** |
| [.](https://www.w3schools.com/cssref/sel_class.asp)*[class](https://www.w3schools.com/cssref/sel_class.asp)* | .intro | Selects all elements with class="intro" |
| [#](https://www.w3schools.com/cssref/sel_id.asp)*[id](https://www.w3schools.com/cssref/sel_id.asp)* | #firstname | Selects the element with id="firstname" |
| [\*](https://www.w3schools.com/cssref/sel_all.asp) | \* | Selects all elements |
| *[element](https://www.w3schools.com/cssref/sel_element.asp)* | p | Selects all <p> elements |
| *[element,element,..](https://www.w3schools.com/cssref/sel_element_comma.asp)* | div, p | Selects all <div> elements and all <p> elements |

External References

External style sheets can be referenced with a full URL or with a path relative to the current web page.

This example uses a full URL to link to a style sheet:

Example

<link rel=“stylesheet" href="https://www.mywebsite.com/html/styles.css">

**HTML Link Colors**

<!DOCTYPE html>

<html>

<head>

<style>

a:link {

color: green;

background-color: transparent;

text-decoration: none;

}

a:visited {

color: pink;

background-color: transparent;

text-decoration: none;

}

a:hover {

color: red;

background-color: transparent;

text-decoration: underline;

}

a:active {

color: yellow;

background-color: transparent;

text-decoration: underline;

}

</style>

</head>

<body>

<h2>Link Colors</h2>

<p>You can change the default colors of links</p>

<a href="html\_images.asp" target="\_blank">HTML Images</a>

</body>

</html>

**Multiple Classes**

<html>

<style>

.city {

background-color: tomato;

color: white;

padding: 10px;

}

.main {

text-align: center;

}

</style>

<body>

<h2>Multiple Classes</h2>

<p>All three headers have the class name "city". In addition, London also have the class name "main", which center-aligns the text.</p>

<h2 class="city main">London</h2>

<h2 class="city">Paris</h2>

<h2 class="city">Tokyo</h2>

</body>

</html>

**Colors**

<h1 style="background-color:rgb(255, 99, 71);">...</h1>

<h1 style="background-color:#ff6347;">...</h1>

<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>

<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>

<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>

**Background Image**

background-image: url(“paper.gif");

background-repeat: repeat-x; // or no-repeat

background-position: right top;

background-attachment: fixed;

**Borders**

The border-style property specifies what kind of border to display.

The following values are allowed:

* dotted - Defines a dotted border
* dashed - Defines a dashed border
* solid - Defines a solid border
* double - Defines a double border
* groove - Defines a 3D grooved border. The effect depends on the border-color value
* ridge - Defines a 3D ridged border. The effect depends on the border-color value
* inset - Defines a 3D inset border. The effect depends on the border-color value
* outset - Defines a 3D outset border. The effect depends on the border-color value
* none - Defines no border
* hidden - Defines a hidden border

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

p.one {

  border-style: solid;

  border-width: 5px;

border-color: red;

}

p.three {

  border-style: solid;

  border-width: 2px 10px 4px 20px;

}

p {

  border-top-style: dotted;

  border-right-style: solid;

}

shorthand syntax

p {

  border: 5px solid red;

}

**Margin and Padding**

p {

  margin-top: 100px;

  margin-bottom: 100px;

}

shorthand syntax. top right bottom left

p {

  margin: 25px 50px 75px 100px;

}

div {

  width: 300px;

  margin: auto; //set horizontal in container

  border: 1px solid red;

}

**Padding**

**div {**

**padding-top: 50px;**

**padding-right: 30px;**

**padding-bottom: 50px;**

**padding-left: 80px;**

**}**

**div {**

**padding: 25px 50px 75px 100px;**

**}**

Here, the <div> element is given a width of 300px. However, the actual width of the <div> element will be 350px (300px + 25px of left padding + 25px of right padding):

div {

  width: 300px;

  padding: 25px;

}

Use the box-sizing property to keep the width at 300px, no matter the amount of padding:

div {

  width: 300px;

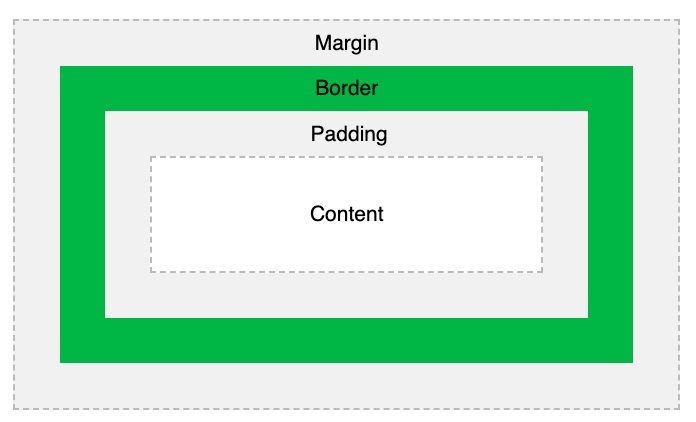
  padding: 25px;

  box-sizing: border-box;

}

CSS height/width Values

The height and width properties may have the following values:

* auto - This is default. The browser calculates the height and width
* length - Defines the height/width in px, cm etc.
* % - Defines the height/width in percent of the containing block
* initial - Sets the height/width to its default value
* inherit - The height/width will be inherited from its parent value

**Inherit css values**

<html>

<head>

<style>

div {

border: 1px solid red;

margin-left: 100px;

}

p.ex1 {

margin-left: **inherit**;

}

</style>

</head>

<body>

<h2>Use of the inherit value</h2>

<p>Let the left margin be inherited from the parent element:</p>

<div>

<p class="ex1">This paragraph has an inherited left margin (from the div element).</p>

</div>

</body>

</html>

**CSS BOX Model**

Example

This <div> element will have a total width of 350px:

div {

  width: 320px;

  padding: 10px;

  border: 5px solid gray;

  margin: 0;

}

320px (width)

+ 20px (left + right padding)

+ 10px (left + right border)

+ 0px (left + right margin)

**= 350px**

The total width of an element should be calculated like this:

**Total element width** = width + left padding + right padding + left border + right border + left margin + right margin

The total height of an element should be calculated like this:

**Total element height** = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

Outline

An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".

**CSS has the following outline properties:**

outline-style

outline-color

outline-width

outline-offset

outline

p.ex4 {

  border: 1px solid black;

  outline-style: solid;

  outline-color: red;

  outline-width: 4px;

}

p.ex4 {outline: thick ridge pink;}

**Text Properties**

**h1 {**

**text-align: center;**

**}**

**a {**

**text-decoration: none;**

**}**

**The text-indent property is used to specify the indentation of the first line of a text:**

**p {**

**text-indent: 50px;**

**}**

**h1 {**

**letter-spacing: 3px;**

**}**

**h1 {**

**word-spacing: 10px;**

**}**

**h1 {**

**text-shadow: 3px 2px red;**

**}**

**Text Overflow**

div{

white-space: nowrap;

width: 50px;

overflow: hidden;

text-overflow: clip;

}

text-overflow: ellipsis;

**Font**

p {

  font-family: "Times New Roman", Times, serif;

font-style: normal;

font-size: 30px;

}

**Font - icons**

<!DOCTYPE html>

<html>

<head>

<script src="https://kit.fontawesome.com/a076d05399.js"></script>

</head>

<body>

<i class="fas fa-cloud"></i>

<i class="fas fa-heart"></i>

<i class="fas fa-car"></i>

<i class="fas fa-file"></i>

<i class="fas fa-bars"></i>

</body>

</html>

**Link Colors**

/\* unvisited link \*/

a:link {

  color: red;

}

/\* visited link \*/

a:visited {

  color: green;

text-decoration: none;

}

/\* mouse over link \*/

a:hover {

  color: hotpink;

text-decoration: none;

}

/\* selected link \*/

a:active {

  color: blue;

text-decoration: underline;

}

**Link Button**

a:link, a:visited {

  background-color: #f44336;

  color: white;

  padding: 14px 25px;

  text-align: center;

  text-decoration: none;

  display: inline-block;

}

a:hover, a:active {

  background-color: red;

}

<a href=“demo.html" target="\_blank">This is a link</a>

HTML Table CSS Properties

table, th, td {

  border: 1px solid black;

}

table { // for single line border

**border-collapse: collapse**;

}

th {

  text-align: left;

}

td {

  height: 50px;

  vertical-align: bottom;

}

th, td {

  padding: 15px;

  text-align: left;

}

th, td {

  border-bottom: 1px solid #ddd;

}

tr:hover {background-color: #f5f5f5;}

//striped row

tr:nth-child(even) {background-color: #f2f2f2;}

**Responsive Table**

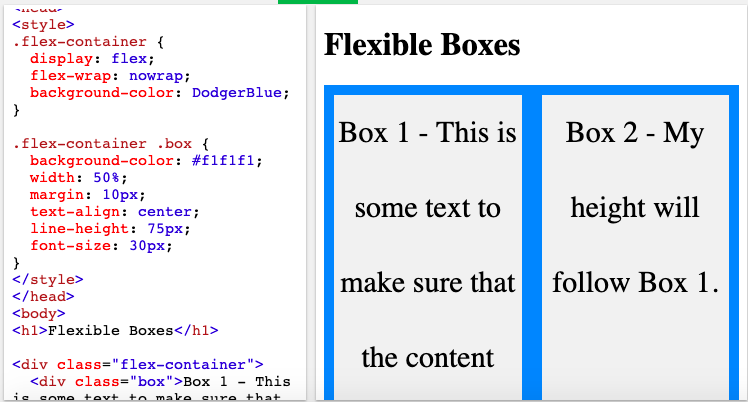
<div style="overflow-x:auto;">

<table>

... table content ...

</table>

</div>



**The float and** Clear **property** can have one of the following values:

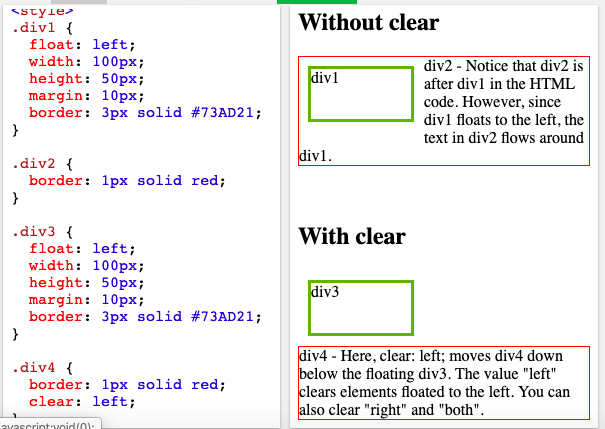
* left - The element floats to the left of its container
* right - The element floats to the right of its container
* none - The element does not float (will be displayed just where it occurs in the text). This is default
* inherit - The element inherits the float value of its parent

img {

float: left;

}

<p><img src="pineapple.jpg" alt="Pineapple" style="width:170px;height:170px;margin-right:15px;">

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque </p>

**The CSS overflow property** controls what happens to content that is too big to fit into an area.

The overflow property has the following values:

* visible - Default. The overflow is not clipped. The content renders outside the element's box
* hidden - The overflow is clipped, and the rest of the content will be invisible
* scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
* auto - Similar to scroll, but it adds scrollbars only when necessary

**Note:** The overflow property only works for block elements with a specified height.

div {

  width: 200px;

  height: 50px;

  background-color: #eee;

  overflow: visible;

}

div {

overflow-x: hidden; /\*Hide horizontal scrollbar \*/

  overflow-y: scroll; /\* Add vertical scrollbar \*/

}

CSS Positioning Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| [bottom](https://www.w3schools.com/cssref/pr_pos_bottom.asp) | Sets the bottom margin edge for a positioned box |
| [clip](https://www.w3schools.com/cssref/pr_pos_clip.asp) | Clips an absolutely positioned element |
| [left](https://www.w3schools.com/cssref/pr_pos_left.asp) | Sets the left margin edge for a positioned box |
| [position](https://www.w3schools.com/cssref/pr_class_position.asp) | Specifies the type of positioning for an element |
| [right](https://www.w3schools.com/cssref/pr_pos_right.asp) | Sets the right margin edge for a positioned box |
| [top](https://www.w3schools.com/cssref/pr_pos_top.asp) | Sets the top margin edge for a positioned box |
| [z-index](https://www.w3schools.com/cssref/pr_pos_z-index.asp) | Sets the stack order of an element |

The position Property

The position property specifies the type of positioning method used for an element.

There are five different position values:

* static
* relative
* fixed
* absolute
* sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

position: static;

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

This <div> element has position: static;

Here is the CSS that is used:

Example

div.static {

  position: static;

  border: 3px solid #73AD21;

}

position: relative;

An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

This <div> element has position: relative;

Here is the CSS that is used:

Example

div.relative {

  position: relative;

  left: 30px;

  border: 3px solid #73AD21;

}

position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

Example

**div.fixed {**

**position: fixed;**

**bottom: 0;**

**right: 0;**

**width: 300px;**

**border: 3px solid #73AD21;**

**}**

This <div> element has position: fixed;

position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

**Note:** A "positioned" element is one whose position is anything except static.

Here is a simple example:

This <div> element has position: relative;

This <div> element has position: absolute;

Here is the CSS that is used:

Example

div.relative {

  position: relative;

  width: 400px;

  height: 200px;

  border: 3px solid #73AD21;

}

**div.absolute {**

**position: absolute;**

**top: 80px;**

**right: 0;**

**width: 200px;**

**height: 100px;**

**border: 3px solid #73AD21;**

**}**

position: sticky;

An element with position: sticky; is positioned based on the user's scroll position.

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

**Note:**Internet Explorer, Edge 15 and earlier versions do not support sticky positioning. Safari requires a -webkit- prefix (see example below). You must also specify at least one of top, right, bottom or left for sticky positioning to work.

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

Example

div.sticky {

  position: -webkit-sticky; /\* Safari \*/

  position: sticky;

  top: 0;

  background-color: green;

  border: 2px solid #4CAF50;

}

Overlapping Elements

When elements are positioned, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

An element can have a positive or negative stack order:

This is a heading



Because the image has a z-index of -1, it will be placed behind the text.

Example

img {

  position: absolute;

  left: 0px;

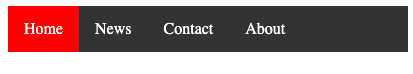
  top: 0px;

  z-index: -1;

}

An element with greater stack order is always in front of an element with a lower stack order.

**Navigation Menu**

****

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

li {

float: left;

}

li a {

display: inline-block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover {

background-color: #111;

}

.active {

background-color: red;

}

</style>

</head>

<body>

<ul>

<li><a href="#home" class="active">Home</a></li>

<li><a href="#news">News</a></li>

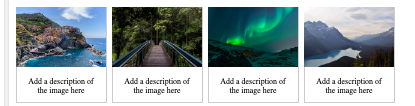
<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

</body>

</html>

**Image Gallery**

<!DOCTYPE html>

<html>

<head>

<style>

div.gallery {

margin: 5px;

border: 1px solid #ccc;

float: left;

width: 180px;

}

div.gallery:hover {

border: 1px solid #777;

}

div.gallery img {

width: 100%;

height: auto;

}

div.desc {

padding: 15px;

text-align: center;

}

</style>

</head>

<body>

<div class="gallery">

<a target="\_blank" href="images/offer2.png">

<img src="images/offer2.png" alt="Cinque Terre" width="600" height="400">

</a>

<div class="desc">Add a description of the image here</div>

</div>

<div class="gallery">

<a target="\_blank" href="images/offer3.png">

<img src="images/offer3.png" alt="Forest" width="600" height="400">

</a>

<div class="desc">Add a description of the image here</div>

</div>

<div class="gallery">

<a target="\_blank" href="images/offer4.png">

<img src="images/offer4.png" alt="Northern Lights" width="600" height="400">

</a>

<div class="desc">Add a description of the image here</div>

</div>

<div class="gallery">

<a target="\_blank" href="images/offer5.png">

<img src="images/offer5.png" alt="Mountains" width="600" height="400">

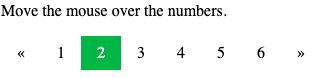
</a>

<div class="desc">Add a description of the image here</div>

</div>

</body>

</html>

**Pagination CSS**

<!DOCTYPE html>

<html>

<head>

<style>

.pagination {

display: inline-block;

}

.pagination a {

color: black;

float: left;

padding: 8px 16px;

text-decoration: none;

}

.pagination a.active {

background-color: #4CAF50;

color: white;

}

.pagination a:hover:not(.active) {background-color: #ddd;}

</style>

</head>

<body>

<h2>Active and Hoverable Pagination</h2>

<p>Move the mouse over the numbers.</p>

<div class="pagination">

<a href="#">&laquo;</a>

<a href="#">1</a>

<a class="active" href="#">2</a>

<a href="#">3</a>

<a href="#">4</a>

<a href="#">5</a>

<a href="#">6</a>

<a href="#">&raquo;</a>

</div>

</body>

</html>

**HTML 5 Page design**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>HTML5</title>

<!--[if lt IE 9]>

<script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>

</script>

<![endif]-->

<style>

body {

font-family: Verdana,sans-serif;

font-size: 0.9em;

}

header, footer {

padding: 10px;

color: white;

background-color: black;

}

section {

margin: 5px;

padding: 10px;

background-color: lightgrey;

}

article {

margin: 5px;

padding: 10px;

background-color: white;

}

nav ul {

padding: 0;

}

nav ul li {

display: inline;

margin: 5px;

}

</style>

</head>

<body>

<header>

<h1>Monday Times</h1>

</header>

<nav>

<ul>

<li>News</li>

<li>Sports</li>

<li>Weather</li>

</ul>

</nav>

<section>

<h2>News Section</h2>

<article>

<h2>News Article</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.</p>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.</p>

</article>

<article>

<h2>News Article</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.</p>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in porta lorem. Morbi condimentum est nibh, et consectetur tortor feugiat at.</p>

</article>

</section>

</body>

</html>

—————————————————————————

**JavaScript**

DOM Model

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().

//——————————————————————————————————————

<html>

<body>

<h2>My First Web Page</h2>

**<script>**

**document.write(5 + 6);**

**</script>**

</body>

</html>

//=====================================

<h2>JavaScript Statements</h2>

<p>A <b>JavaScript program</b> is a list of <b>statements</b> to be executed by a computer.</p>

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

<script>

var x, y, z; // Statement 1

x = 5; // Statement 2

y = 6; // Statement 3

z = x + y; // Statement 4

document.getElementById("demo").innerHTML =

"The value of z is " + z + ".";

</script>

========================================

document.getElementById('demo').innerHTML = Date()

document.getElementById("demo").style.fontSize = "35px";

document.getElementById("demo").style.display = "none"; //hide

document.getElementById("demo").style.display = "block"; //show

<script>

document.getElementById("demo").innerHTML = "My First JavaScript";

</script>

=====================================================

<html>

<body>

<h2>**JavaScript in Body**</h2>

<p id="demo">A Paragraph.</p>

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

<script>

function myFunction() {

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

}

</script>

</body>

</html>

==============================================

JavaScript Keywords

JavaScript statements often start with a **keyword** to identify the JavaScript action to be performed.

Here is a list of some of the keywords you will learn about in this tutorial:

|  |  |
| --- | --- |
| **Keyword** | **Description** |
| break | Terminates a switch or a loop |
| continue | Jumps out of a loop and starts at the top |
| debugger | Stops the execution of JavaScript, and calls (if available) the debugging function |
| do ... while | Executes a block of statements, and repeats the block, while a condition is true |
| for | Marks a block of statements to be executed, as long as a condition is true |
| function | Declares a function |
| if ... else | Marks a block of statements to be executed, depending on a condition |
| return | Exits a function |
| switch | Marks a block of statements to be executed, depending on different cases |
| try ... catch | Implements error handling to a block of statements |
| var | Declares a variable |

================================================

<!DOCTYPE html>

<html>

<head>  
<script>  
function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";  
}  
</script>  
</head>  
<body>

<h1>A Web Page</h1>  
<p id="demo">A Paragraph</p>  
<button type="button" onclick="myFunction()">Try it</button>

</body>  
</html>

//———————————————————

All operators arithmetic, logical, relational, bitwise are available

//——————————typeof and undefined

<script>

var car = "Volvo";

car = undefined;

document.getElementById("demo").innerHTML =

car + "<br>" + typeof car;

</script>

//—————————————

**JavaScript Arrays**

**=================**

**<script>**

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

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

**</script>**

**<script>**

**var cars = new Array("Saab", "Volvo", "BMW");**

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

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

cars[0] = "Opel";

var x = cars.length;

var y = cars.sort();

**</script>**

**————————**

<script>

var fruits, text, fLen, i;

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

fLen = fruits.length;

text = "<ul>";

**for (i = 0; i < fLen; i++) {**

**text += "<li>" + fruits[i] + "</li>";**

**}**

text += "</ul>";

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

</script>

================

<script>

var fruits, text;

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

text = "<ul>";

fruits.forEach(myFunction);

text += "</ul>";

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

function myFunction(value) {

text += "<li>" + value + "</li>";

}

</script>

————————Add new Element at End————

<script>

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

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

function myFunction() {

fruits[fruits.length] = "Lemon";

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

}

</script>

//——————————————————

Associative Arrays

Many programming languages support arrays with named indexes.

Arrays with named indexes are called associative arrays (or hashes).

JavaScript does **not** support arrays with named indexes.

In JavaScript, **arrays** always use **numbered indexes**.

Example

var person = [];

person[0] = "John";

person[1] = "Doe";

person[2] = 46;

var x = person.length;     // person.length will return 3

var y = person[0];

//——————————

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

</script>

<script>

var person = [];

person["firstName"] = "Ankit";

person["lastName"] = "Mishra";

person["age"] = 26;

document.getElementById("demo").innerHTML =

person["age"] + " " + person.length;

<script>

**Objects**

**var person = {**

**firstName:"Ankit",**

**lastName:"Sharma",**

**age:50,**

**eyeColor:”blue" };**

// Display some data from the object:

document.getElementById("demo").innerHTML =

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

</script>

HTML Events

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

Here are some examples of HTML events:

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

<button onclick="document.getElementById('demo').innerHTML=Date()">The time is?</button>

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

Common HTML Events

Here is a list of some common HTML events:

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

What can JavaScript Do?

Event handlers can be used to handle, and verify, user input, user actions, and browser actions:

* Things that should be done every time a page loads
* Things that should be done when the page is closed
* Action that should be performed when a user clicks a button
* Content that should be verified when a user inputs data
* And more ...

Many different methods can be used to let JavaScript work with events:

* HTML event attributes can execute JavaScript code directly
* HTML event attributes can call JavaScript functions
* You can assign your own event handler functions to HTML elements
* You can prevent events from being sent or being handled

HTML DOM Events

HTML DOM events allow JavaScript to register different event handlers on elements in an HTML document.

Events are normally used in combination with functions, and the function will not be executed before the event occurs (such as when a user clicks a button).

For a tutorial about Events, read our [JavaScript Events Tutorial](https://www.w3schools.com/js/js_events.asp).

|  |  |  |
| --- | --- | --- |
| **Event** | **Description** | **Belongs To** |
| [abort](https://www.w3schools.com/jsref/event_onabort_media.asp) | The event occurs when the loading of a media is aborted | [UiEvent](https://www.w3schools.com/jsref/obj_uievent.asp), [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [afterprint](https://www.w3schools.com/jsref/event_onafterprint.asp) | The event occurs when a page has started printing, or if the print dialogue box has been closed | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [animationend](https://www.w3schools.com/jsref/event_animationend.asp) | The event occurs when a CSS animation has completed | [AnimationEvent](https://www.w3schools.com/jsref/obj_animationevent.asp) |
| [animationiteration](https://www.w3schools.com/jsref/event_animationiteration.asp) | The event occurs when a CSS animation is repeated | [AnimationEvent](https://www.w3schools.com/jsref/obj_animationevent.asp) |
| [animationstart](https://www.w3schools.com/jsref/event_animationstart.asp) | The event occurs when a CSS animation has started | [AnimationEvent](https://www.w3schools.com/jsref/obj_animationevent.asp) |
| [beforeprint](https://www.w3schools.com/jsref/event_onbeforeprint.asp) | The event occurs when a page is about to be printed | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [beforeunload](https://www.w3schools.com/jsref/event_onbeforeunload.asp) | The event occurs before the document is about to be unloaded | [UiEvent](https://www.w3schools.com/jsref/obj_uievent.asp), [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [blur](https://www.w3schools.com/jsref/event_onblur.asp) | The event occurs when an element loses focus | [FocusEvent](https://www.w3schools.com/jsref/obj_focusevent.asp) |
| [canplay](https://www.w3schools.com/jsref/event_oncanplay.asp) | The event occurs when the browser can start playing the media (when it has buffered enough to begin) | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [canplaythrough](https://www.w3schools.com/jsref/event_oncanplaythrough.asp) | The event occurs when the browser can play through the media without stopping for buffering | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [change](https://www.w3schools.com/jsref/event_onchange.asp) | The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>) | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [click](https://www.w3schools.com/jsref/event_onclick.asp) | The event occurs when the user clicks on an element | [MouseEvent](https://www.w3schools.com/jsref/obj_mouseevent.asp) |
| [contextmenu](https://www.w3schools.com/jsref/event_oncontextmenu.asp) | The event occurs when the user right-clicks on an element to open a context menu | [MouseEvent](https://www.w3schools.com/jsref/obj_mouseevent.asp) |
| [copy](https://www.w3schools.com/jsref/event_oncopy.asp) | The event occurs when the user copies the content of an element | [ClipboardEvent](https://www.w3schools.com/jsref/obj_clipboardevent.asp) |
| [cut](https://www.w3schools.com/jsref/event_oncut.asp) | The event occurs when the user cuts the content of an element | [ClipboardEvent](https://www.w3schools.com/jsref/obj_clipboardevent.asp) |
| [dblclick](https://www.w3schools.com/jsref/event_ondblclick.asp) | The event occurs when the user double-clicks on an element | [MouseEvent](https://www.w3schools.com/jsref/obj_mouseevent.asp) |
| [drag](https://www.w3schools.com/jsref/event_ondrag.asp) | The event occurs when an element is being dragged | [DragEvent](https://www.w3schools.com/jsref/obj_dragevent.asp) |
| [dragend](https://www.w3schools.com/jsref/event_ondragend.asp) | The event occurs when the user has finished dragging an element | [DragEvent](https://www.w3schools.com/jsref/obj_dragevent.asp) |
| [dragenter](https://www.w3schools.com/jsref/event_ondragenter.asp) | The event occurs when the dragged element enters the drop target | [DragEvent](https://www.w3schools.com/jsref/obj_dragevent.asp) |
| [dragleave](https://www.w3schools.com/jsref/event_ondragleave.asp) | The event occurs when the dragged element leaves the drop target | [DragEvent](https://www.w3schools.com/jsref/obj_dragevent.asp) |
| [dragover](https://www.w3schools.com/jsref/event_ondragover.asp) | The event occurs when the dragged element is over the drop target | [DragEvent](https://www.w3schools.com/jsref/obj_dragevent.asp) |
| [dragstart](https://www.w3schools.com/jsref/event_ondragstart.asp) | The event occurs when the user starts to drag an element | [DragEvent](https://www.w3schools.com/jsref/obj_dragevent.asp) |
| [drop](https://www.w3schools.com/jsref/event_ondrop.asp) | The event occurs when the dragged element is dropped on the drop target | [DragEvent](https://www.w3schools.com/jsref/obj_dragevent.asp) |
| [durationchange](https://www.w3schools.com/jsref/event_ondurationchange.asp) | The event occurs when the duration of the media is changed | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [ended](https://www.w3schools.com/jsref/event_onended.asp) | The event occurs when the media has reach the end (useful for messages like "thanks for listening") | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [error](https://www.w3schools.com/jsref/event_onerror.asp) | The event occurs when an error occurs while loading an external file | [ProgressEvent](https://www.w3schools.com/jsref/obj_progressevent.asp),[UiEvent](https://www.w3schools.com/jsref/obj_uievent.asp), [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [focus](https://www.w3schools.com/jsref/event_onfocus.asp) | The event occurs when an element gets focus | [FocusEvent](https://www.w3schools.com/jsref/obj_focusevent.asp) |
| [focusin](https://www.w3schools.com/jsref/event_onfocusin.asp) | The event occurs when an element is about to get focus | [FocusEvent](https://www.w3schools.com/jsref/obj_focusevent.asp) |
| [focusout](https://www.w3schools.com/jsref/event_onfocusout.asp) | The event occurs when an element is about to lose focus | [FocusEvent](https://www.w3schools.com/jsref/obj_focusevent.asp) |
| [fullscreenchange](https://www.w3schools.com/jsref/event_fullscreenchange.asp) | The event occurs when an element is displayed in fullscreen mode | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [fullscreenerror](https://www.w3schools.com/jsref/event_fullscreenerror.asp) | The event occurs when an element can not be displayed in fullscreen mode | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [hashchange](https://www.w3schools.com/jsref/event_onhashchange.asp) | The event occurs when there has been changes to the anchor part of a URL | [HashChangeEvent](https://www.w3schools.com/jsref/obj_hashchangeevent.asp) |
| [input](https://www.w3schools.com/jsref/event_oninput.asp) | The event occurs when an element gets user input | [InputEvent](https://www.w3schools.com/jsref/obj_inputevent.asp), [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [invalid](https://www.w3schools.com/jsref/event_oninvalid.asp) | The event occurs when an element is invalid | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [keydown](https://www.w3schools.com/jsref/event_onkeydown.asp) | The event occurs when the user is pressing a key | [KeyboardEvent](https://www.w3schools.com/jsref/obj_keyboardevent.asp) |
| [keypress](https://www.w3schools.com/jsref/event_onkeypress.asp) | The event occurs when the user presses a key | [KeyboardEvent](https://www.w3schools.com/jsref/obj_keyboardevent.asp) |
| [keyup](https://www.w3schools.com/jsref/event_onkeyup.asp) | The event occurs when the user releases a key | [KeyboardEvent](https://www.w3schools.com/jsref/obj_keyboardevent.asp) |
| [load](https://www.w3schools.com/jsref/event_onload.asp) | The event occurs when an object has loaded | [UiEvent](https://www.w3schools.com/jsref/obj_uievent.asp), [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [loadeddata](https://www.w3schools.com/jsref/event_onloadeddata.asp) | The event occurs when media data is loaded | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [loadedmetadata](https://www.w3schools.com/jsref/event_onloadedmetadata.asp) | The event occurs when meta data (like dimensions and duration) are loaded | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [loadstart](https://www.w3schools.com/jsref/event_onloadstart.asp) | The event occurs when the browser starts looking for the specified media | [ProgressEvent](https://www.w3schools.com/jsref/obj_progressevent.asp) |
| [message](https://www.w3schools.com/jsref/event_onmessage_sse.asp) | The event occurs when a message is received through the event source | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [mousedown](https://www.w3schools.com/jsref/event_onmousedown.asp) | The event occurs when the user presses a mouse button over an element | [MouseEvent](https://www.w3schools.com/jsref/obj_mouseevent.asp) |
| [mouseenter](https://www.w3schools.com/jsref/event_onmouseenter.asp) | The event occurs when the pointer is moved onto an element | [MouseEvent](https://www.w3schools.com/jsref/obj_mouseevent.asp) |
| [mouseleave](https://www.w3schools.com/jsref/event_onmouseleave.asp) | The event occurs when the pointer is moved out of an element | [MouseEvent](https://www.w3schools.com/jsref/obj_mouseevent.asp) |
| [mousemove](https://www.w3schools.com/jsref/event_onmousemove.asp) | The event occurs when the pointer is moving while it is over an element | [MouseEvent](https://www.w3schools.com/jsref/obj_mouseevent.asp) |
| [mouseover](https://www.w3schools.com/jsref/event_onmouseover.asp) | The event occurs when the pointer is moved onto an element, or onto one of its children | [MouseEvent](https://www.w3schools.com/jsref/obj_mouseevent.asp) |
| [mouseout](https://www.w3schools.com/jsref/event_onmouseout.asp) | The event occurs when a user moves the mouse pointer out of an element, or out of one of its children | [MouseEvent](https://www.w3schools.com/jsref/obj_mouseevent.asp) |
| [mouseup](https://www.w3schools.com/jsref/event_onmouseup.asp) | The event occurs when a user releases a mouse button over an element | [MouseEvent](https://www.w3schools.com/jsref/obj_mouseevent.asp) |
| mousewheel | Deprecated. Use the [wheel](https://www.w3schools.com/jsref/event_onwheel.asp) event instead | [WheelEvent](https://www.w3schools.com/jsref/obj_wheelevent.asp) |
| [offline](https://www.w3schools.com/jsref/event_onoffline.asp) | The event occurs when the browser starts to work offline | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [online](https://www.w3schools.com/jsref/event_ononline.asp) | The event occurs when the browser starts to work online | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [open](https://www.w3schools.com/jsref/event_onopen_sse.asp) | The event occurs when a connection with the event source is opened | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [pagehide](https://www.w3schools.com/jsref/event_onpagehide.asp) | The event occurs when the user navigates away from a webpage | [PageTransitionEvent](https://www.w3schools.com/jsref/obj_pagetransitionevent.asp) |
| [pageshow](https://www.w3schools.com/jsref/event_onpageshow.asp) | The event occurs when the user navigates to a webpage | [PageTransitionEvent](https://www.w3schools.com/jsref/obj_pagetransitionevent.asp) |
| [paste](https://www.w3schools.com/jsref/event_onpaste.asp) | The event occurs when the user pastes some content in an element | [ClipboardEvent](https://www.w3schools.com/jsref/obj_clipboardevent.asp) |
| [pause](https://www.w3schools.com/jsref/event_onpause.asp) | The event occurs when the media is paused either by the user or programmatically | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [play](https://www.w3schools.com/jsref/event_onplay.asp) | The event occurs when the media has been started or is no longer paused | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [playing](https://www.w3schools.com/jsref/event_onplaying.asp) | The event occurs when the media is playing after having been paused or stopped for buffering | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| popstate | The event occurs when the window's history changes | [PopStateEvent](https://www.w3schools.com/jsref/obj_popstateevent.asp) |
| [progress](https://www.w3schools.com/jsref/event_onprogress.asp) | The event occurs when the browser is in the process of getting the media data (downloading the media) | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [ratechange](https://www.w3schools.com/jsref/event_onratechange.asp) | The event occurs when the playing speed of the media is changed | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [resize](https://www.w3schools.com/jsref/event_onresize.asp) | The event occurs when the document view is resized | [UiEvent](https://www.w3schools.com/jsref/obj_uievent.asp), [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [reset](https://www.w3schools.com/jsref/event_onreset.asp) | The event occurs when a form is reset | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [scroll](https://www.w3schools.com/jsref/event_onscroll.asp) | The event occurs when an element's scrollbar is being scrolled | [UiEvent](https://www.w3schools.com/jsref/obj_uievent.asp), [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [search](https://www.w3schools.com/jsref/event_onsearch.asp) | The event occurs when the user writes something in a search field (for <input="search">) | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [seeked](https://www.w3schools.com/jsref/event_onseeked.asp) | The event occurs when the user is finished moving/skipping to a new position in the media | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [seeking](https://www.w3schools.com/jsref/event_onseeking.asp) | The event occurs when the user starts moving/skipping to a new position in the media | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [select](https://www.w3schools.com/jsref/event_onselect.asp) | The event occurs after the user selects some text (for <input> and <textarea>) | [UiEvent](https://www.w3schools.com/jsref/obj_uievent.asp), [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [show](https://www.w3schools.com/jsref/event_onshow.asp) | The event occurs when a <menu> element is shown as a context menu | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [stalled](https://www.w3schools.com/jsref/event_onstalled.asp) | The event occurs when the browser is trying to get media data, but data is not available | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| storage | The event occurs when a Web Storage area is updated | [StorageEvent](https://www.w3schools.com/jsref/obj_storageevent.asp) |
| [submit](https://www.w3schools.com/jsref/event_onsubmit.asp) | The event occurs when a form is submitted | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [suspend](https://www.w3schools.com/jsref/event_onsuspend.asp) | The event occurs when the browser is intentionally not getting media data | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [timeupdate](https://www.w3schools.com/jsref/event_ontimeupdate.asp) | The event occurs when the playing position has changed (like when the user fast forwards to a different point in the media) | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [toggle](https://www.w3schools.com/jsref/event_ontoggle.asp) | The event occurs when the user opens or closes the <details> element | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [touchcancel](https://www.w3schools.com/jsref/event_touchcancel.asp) | The event occurs when the touch is interrupted | [TouchEvent](https://www.w3schools.com/jsref/obj_touchevent.asp) |
| [touchend](https://www.w3schools.com/jsref/event_touchend.asp) | The event occurs when a finger is removed from a touch screen | [TouchEvent](https://www.w3schools.com/jsref/obj_touchevent.asp) |
| [touchmove](https://www.w3schools.com/jsref/event_touchmove.asp) | The event occurs when a finger is dragged across the screen | [TouchEvent](https://www.w3schools.com/jsref/obj_touchevent.asp) |
| [touchstart](https://www.w3schools.com/jsref/event_touchstart.asp) | The event occurs when a finger is placed on a touch screen | [TouchEvent](https://www.w3schools.com/jsref/obj_touchevent.asp) |
| [transitionend](https://www.w3schools.com/jsref/event_transitionend.asp) | The event occurs when a CSS transition has completed | [TransitionEvent](https://www.w3schools.com/jsref/obj_transitionevent.asp) |
| [unload](https://www.w3schools.com/jsref/event_onunload.asp) | The event occurs once a page has unloaded (for <body>) | [UiEvent](https://www.w3schools.com/jsref/obj_uievent.asp), [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [volumechange](https://www.w3schools.com/jsref/event_onvolumechange.asp) | The event occurs when the volume of the media has changed (includes setting the volume to "mute") | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [waiting](https://www.w3schools.com/jsref/event_onwaiting.asp) | The event occurs when the media has paused but is expected to resume (like when the media pauses to buffer more data) | [Event](https://www.w3schools.com/jsref/obj_event.asp) |
| [wheel](https://www.w3schools.com/jsref/event_onwheel.asp) | The event occurs when the mouse wheel rolls up or down over an element | [WheelEvent](https://www.w3schools.com/jsref/obj_wheelevent.asp) |

typeof "John"                 // Returns "string"

typeof 3.14                   // Returns "number"

typeof NaN                    // Returns "number"

typeof false                  // Returns "boolean"

typeof [1,2,3,4]              // Returns "object"

typeof {name:'John', age:34}  // Returns "object"

typeof new Date()             // Returns "object"

typeof function () {}         // Returns "function"

typeof myCar                  // Returns "undefined" \*

typeof null                   // Returns "object"

What Is a Regular Expression?

A regular expression is a sequence of characters that forms a **search pattern**.

When you search for data in a text, you can use this search pattern to describe what you are searching for.

A regular expression can be a single character, or a more complicated pattern.

Regular expressions can be used to perform all types of **text search** and **text replace** operations.

Syntax

/*pattern*/*modifiers*;

Example

var patt = /cdac/i;

Using String Methods

In JavaScript, regular expressions are often used with the two **string methods**: search() and replace().

The search() method uses an expression to search for a match, and returns the position of the match.

The replace() method returns a modified string where the pattern is replaced.

Using String search() With a String

The search() method searches a string for a specified value and returns the position of the match:

Example

Use a string to do a search for "W3schools" in a string:

var str = "Visit W3Schools!";

var n = str.search("W3Schools");

Using String search() With a Regular Expression

Example

Use a regular expression to do a case-insensitive search for "w3schools" in a string:

var str = "Visit W3Schools";

var n = str.search(/w3schools/i);

Use String replace() With a Regular Expression

Example

Use a case insensitive regular expression to replace Microsoft with W3Schools in a string:

var str = "Visit Microsoft!";

var res = str.replace(/microsoft/i, "W3Schools")

//=================================================

The delete keyword deletes a property from an object:

Example

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

delete person.age;   // or delete person["age"]

//=================================

**Object Methods**

**var person = {**

**firstName: "John",**

**lastName : "Doe",**

**id       : 5566,**

**fullName : function() {**

**return this.firstName + " " + this.lastName;**

**}**

**};**

//—————————

person.fullName()

//=================

Adding a Method to an Object

Adding a new method to an object is easy:

Example

person.name = function () {

  return this.firstName + " " + this.lastName;

};

//===========================================

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

<script>

// **Constructor function**

//for Person objects

**function Person(first, last, age, eye) {**

**this.firstName = first;**

**this.lastName = last;**

**this.age = age;**

**this.eyeColor = eye;**

**}**

**// Create a Person object**

**var myFather = new Person("John", "Doe", 50, "blue");**

// Display age

document.getElementById("demo").innerHTML =

"My father is " + myFather.age + ".";

</script>

The **this** Keyword

In JavaScript, the thing called this is the object that "owns" the code.

The value of this, when used in an object, is the object itself.

In a constructor function this does not have a value. It is a substitute for the new object. The value of this will become the new object when a new object is created

The Function() Constructor

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

<script>

var myFunction = function (a, b) {return a \* b}

document.getElementById("demo").innerHTML = myFunction(4, 3);

</script>

Arrow Functions

Arrow functions allows a short syntax for writing function expressions.

You don't need the function keyword, the return keyword, and the **curly brackets**.

Example

// ES5

var x = function(x, y) {

  return x \* y;

}

// ES6

const x = (x, y) => x \* y;

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 | 1 |

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.elements.length; i++) {

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

}

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

//=================================

<body>

<img id="myImage" src="smiley.gif">

<script>

document.getElementById("myImage").src = "landscape.jpg";

</script>

</body>

**Animations=========**

Timing Events

The window object allows execution of code at specified time intervals.

These time intervals are called timing events.

The two key methods to use with JavaScript are:

* setTimeout(*function, milliseconds*)  
  Executes a function, after waiting a specified number of milliseconds.
* setInterval(*function, milliseconds*)  
  Same as setTimeout(), but repeats the execution of the function continuously.

The setTimeout() and setInterval() are both methods of the HTML DOM Window object.

The setTimeout() Method

window.setTimeout(*function*,*milliseconds*);

The window.setTimeout() method can be written without the window prefix.

The first parameter is a function to be executed.

The second parameter indicates the number of milliseconds before execution.

Example

Click a button. Wait 3 seconds, and the page will alert "Hello":

<button onclick="setTimeout(myFunction, 3000)">Try it</button>

<script>

function myFunction() {

  alert('Hello');

}

</script>

[Try it Yourself »](https://www.w3schools.com/Js/tryit.asp?filename=tryjs_timing1)

How to Stop the Execution?

The clearTimeout() method stops the execution of the function specified in setTimeout().

window.clearTimeout(*timeoutVariable*)

The window.clearTimeout() method can be written without the window prefix.

The clearTimeout() method uses the variable returned from setTimeout():

myVar = setTimeout(*function*,*milliseconds*);

clearTimeout(myVar);

If the function has not already been executed, you can stop the execution by calling the clearTimeout() method:

Example

Same example as above, but with an added "Stop" button:

<button onclick="myVar = setTimeout(myFunction, 3000)">Try it</button>

<button onclick="clearTimeout(myVar)">Stop it</button>

[Try it Yourself »](https://www.w3schools.com/Js/tryit.asp?filename=tryjs_settimeout2)

The setInterval() Method

The setInterval() method repeats a given function at every given time-interval.

window.setInterval(*function*,*milliseconds*);

The window.setInterval() method can be written without the window prefix.

The first parameter is the function to be executed.

The second parameter indicates the length of the time-interval between each execution.

This example executes a function called "myTimer" once every second (like a digital watch).

Example

Display the current time:

var myVar = setInterval(myTimer, 1000);

function myTimer() {

  var d = new Date();

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

}

[Try it Yourself »](https://www.w3schools.com/Js/tryit.asp?filename=tryjs_setinterval2)

There are 1000 milliseconds in one second.

How to Stop the Execution?

The clearInterval() method stops the executions of the function specified in the setInterval() method.

window.clearInterval(*timerVariable*)

The window.clearInterval() method can be written without the window prefix.

The clearInterval() method uses the variable returned from setInterval():

myVar = setInterval(*function*,*milliseconds*);

clearInterval(myVar);

Example

Same example as above, but we have added a "Stop time" button:

<p id="demo"></p>

<button onclick="clearInterval(myVar)">Stop time</button>

<script>

var myVar = setInterval(myTimer, 1000);

function myTimer() {

  var d = new Date();

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

}

</script>

//——————————**Animation Example———————**

<!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>

The addEventListener() method

Example

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

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

The removeEventListener() method

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

Example

*element*.removeEventListener("mousemove", myFunction);

Navigating Between Nodes

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

* parentNode
* childNodes[*nodenumber*]
* firstChild
* lastChild
* nextSibling
* previousSibling
* Example
* <html>
* <body>
* <h1 id="id01">My First Page</h1>
* <p id="id02"></p>
* <script>
* document.getElementById("id02").innerHTML = document.getElementById("id01").firstChild.nodeValue;
* </script>
* </body>
* </html>

Example

Change the background color of all <p> elements:

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

var i;

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

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

}

Window Screen

The window.screen object can be written without the window prefix.

Properties:

* screen.width
* screen.height
* screen.availWidth
* screen.availHeight
* screen.colorDepth
* screen.pixelDepth

Window Location

The window.location object can be written without the window prefix.

Some examples:

* window.location.href returns the href (URL) of the current page
* window.location.hostname returns the domain name of the web host
* window.location.pathname returns the path and filename of the current page
* window.location.protocol returns the web protocol used (http: or https:)
* window.location.assign loads a new document

Window Location Assign

The window.location.assign() method loads a new document.

Example

Load a new document:

<html>

<head>

<script>

function newDoc() {

  window.location.assign("https://www.w3schools.com")

}

</script>

</head>

<body>

<input type="button" value="Load new document"onclick="newDoc()">

</body>

</html>

Window History

The window.history object can be written without the window prefix.

To protect the privacy of the users, there are limitations to how JavaScript can access this object.

Some methods:

* history.back() - same as clicking back in the browser
* history.forward() - same as clicking forward in the browser

Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.

Syntax

window.confirm("*sometext*");

The window.confirm() method can be written without the window prefix.

Example

if (confirm("Press a button!")) {

  txt = "You pressed OK!";

} else {

  txt = "You pressed Cancel!";

}

Prompt Box

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Prompt</h2>

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

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

<script>

function myFunction() {

var txt;

var person = prompt("Please enter your name:", "Harry Potter");

if (person == null || person == "") {

txt = "User cancelled the prompt.";

} else {

txt = "Hello " + person + "! How are you today?";

}

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

}

</script>

</body>

</html>

//——————————————————