HTML Forms

## **The <form> Element**

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

<form>  
.  
*form elements*  
.  
</form>

## **The <input> Element**

The <input> element is the most important form element.

The <input> element can be displayed in several ways, depending on the **type** attribute.

Here are some examples:

|  |  |
| --- | --- |
| **Type** | **Description** |
| <input type="text"> | Defines a one-line text input field |
| <input type="radio"> | Defines a radio button (for selecting one of many choices) |
| <input type="submit"> | Defines a submit button (for submitting the form) |

## **Text Input**

<input type="text"> defines a one-line input field for **text input**:

## **Radio Button Input**

<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

## **The Submit Button**

<input type="submit"> defines a button for **submitting** the form data to a **form-handler**.

The form-handler is typically a server page with a script for processing input data.

The form-handler is specified in the form's **action** attribute:

<form action="/action\_page.php">  
  First name:<br>  
  <input type="text" name="firstname" value="Mickey"><br>  
  Last name:<br>  
  <input type="text" name="lastname" value="Mouse"><br><br>  
  <input type="submit" value="Submit">  
</form>

## **The Action Attribute**

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 Target Attribute**

The target attribute specifies if the submitted result will open in a new browser tab, a frame, or in the current window.

The default value is "\_self" which means the form will be submitted in the current window.

## **The Method Attribute**

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

## **When to Use GET?**

The default method when submitting form data is GET.

However, when GET is used, the submitted form data will be **visible in the page address field**:

/action\_page.php?firstname=Mickey&lastname=Mouse

**Notes on GET:**

* Appends form-data into the URL in name/value pairs
* The length of a URL is limited (about 3000 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

## **When to Use POST?**

Always use POST if the form data contains sensitive or personal information. The POST method does not display the submitted form data in the page address field.

**Notes on POST:**

* POST has no size limitations, and can be used to send large amounts of data.
* Form submissions with POST cannot be bookmarked

## **The Name Attribute**

Each input field must have a name attribute to be submitted.

If the name attribute is omitted, the data of that input field will not be sent at all.

## **Grouping Form Data with <fieldset>**

The <fieldset> element is used to group related data in a form.

The <legend> element defines a caption for the <fieldset> element.

<form action="/action\_page.php">  
  <fieldset>  
    <legend>Personal information:</legend>  
    First name:<br>  
    <input type="text" name="firstname" value="Mickey"><br>  
    Last name:<br>  
    <input type="text" name="lastname" value="Mouse"><br><br>  
    <input type="submit" value="Submit">  
  </fieldset>  
</form>

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| accept-charset | Specifies the charset used in the submitted form (default: the page charset). |
| action | Specifies an address (url) where to submit the form (default: the submitting page). |
| autocomplete | Specifies if the browser should autocomplete the form (default: on). |
| enctype | Specifies the encoding of the submitted data (default: is url-encoded). |
| method | Specifies the HTTP method used when submitting the form (default: GET). |
| name | Specifies a name used to identify the form (for DOM usage: document.forms.name). |
| novalidate | Specifies that the browser should not validate the form. |
| target | Specifies the target of the address in the action attribute (default: \_self). |

# HTML Form Elements

## **The <input> Element**

<input name="firstname" type="text">

## **The <select> Element**

The <select> element defines a **drop-down list**

<select name="cars" **size="3"** **multiple** >  
  <option value="volvo">Volvo</option>  
  <option value="saab">Saab</option>  
  <option value="fiat" selected >Fiat</option>  
  <option value="audi">Audi</option>

</select>

## **The <textarea> Element**

<textarea name="message" rows="10" cols="30" style="width:200px; height:600px;">  
The cat was playing in the garden.  
</textarea>

## **The <button> Element**

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

## **HTML5 Form Elements**

HTML5 added the following form elements:

* <datalist>
* <output>

## **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.

<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>

## **HTML5 <output> Element**

<form action="/action\_page.php" oninput="x.value=parseInt(a.value)+parseInt(b.value)">  
  0  
  <input type="range"  id="a" name="a" value="50" oninput="y.value=parseInt(a.value)”>  
  100 +  
  <input type="number" id="b" name="b" value="50">  
  =  
  <output name="x" for="a b"></output>  
  <br><br>

<output name="y" for="a"></output>  
  <input type="submit">  
</form>

# HTML Input Types

* <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">

## **Input Type Text**

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

## **Input Type Password**

 User password:<br>  <input type="password" name="psw">

## **Input Type Submit**

<input type="submit"> defines a button for **submitting** form data to a **form-handler**.

The form-handler is typically a server page with a script for processing input data

<input type="submit" value="Submit">

## **Input Type Reset**

<input type="reset"> defines a **reset button** that will reset all form values to their default values:

<input type="submit" value="Submit">

## **Input Type Radio**

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

## **Input Type Checkbox**

<input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>

## **Input Type Button**

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

## **Input Type Color**

The <input type="color"> is used for input fields that should contain a color.

Depending on browser support, a color picker can show up in the input field.

Select your favorite color:  <input type="color" name="favcolor">

## **Input Type Date**

The <input type="date"> is used for input fields that should contain a date.

Depending on browser support, a date picker can show up in the input field.

 Birthday:  <input type="date" name="bday">

Enter a date before 1980-01-01:  
  <input type="date" name="bday" max="1979-12-31"><br>  
  Enter a date after 2000-01-01:  
  <input type="date" name="bday" min="2000-01-02"><br>

## **Input Type Datetime-local**

The <input type="datetime-local"> specifies a date and time input field, with no time zone.

Birthday (date and time):  <input type="datetime-local" name="bdaytime">

## **Input Type Email**

The <input type="email"> is used for input fields that should contain an e-mail address.

Depending on browser support, the e-mail address can be automatically validated when submitted.

Some smartphones recognize the email type, and add ".com" to the keyboard to match email input.

E-mail:  <input type="email" name="email">

## **Input Type File**

Select a file: <input type="file" name="myFile">

## **Input Type Month**

The <input type="month"> allows the user to select a month and year.

Depending on browser support, a date picker can show up in the input field.

Birthday (month and year): <input type="month" name="bdaymonth">

## **Input Type Number**

The <input type="number"> defines a **numeric** input field.

You can also set restrictions on what numbers are accepted.

Quantity (between 1 and 5):  <input type="number" name="quantity" min="1" max="5">

## **Input Restrictions**

Here is a list of some common input restrictions (some are new in HTML5):

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| disabled | Specifies that an input field should be disabled |
| max | Specifies the maximum value for an input field |
| maxlength | Specifies the maximum number of character for an input field |
| min | Specifies the minimum value for an input field |
| pattern | Specifies a regular expression to check the input value against |
| readonly | Specifies that an input field is read only (cannot be changed) |
| required | Specifies that an input field is required (must be filled out) |
| size | Specifies the width (in characters) of an input field |
| step | Specifies the legal number intervals for an input field |
| value | Specifies the default value for an input field |

Quantity:

<input type="number" name="points" min="0" max="100" step="10" value="30">

 <input type="range" name="points" min="0" max="10">

 Search Google:  <input type="search" name="googlesearch">

Telephone:  <input type="tel" name="phone" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">

## **Input Type Time**

The <input type="time"> allows the user to select a time (no time zone).

Depending on browser support, a time picker can show up in the input field.

Select a time:  <input type="time" name="usr\_time">

## **Input Type Url**

The <input type="url"> is used for input fields that should contain a URL address.

Depending on browser support, the url field can be automatically validated when submitted.

Some smartphones recognize the url type, and adds ".com" to the keyboard to match url input.

Add your homepage:  <input type="url" name="homepage">

## **Input Type Week**

The <input type="week"> allows the user to select a week and year.

Depending on browser support, a date picker can show up in the input field.

  Select a week:  <input type="week" name="week\_year">

# HTML Input Attributes

## **The value Attribute**

First name:<br>  <input type="text" name="firstname" value="John">

## **The readonly Attribute**

 First name:<br>  <input type="text" name="firstname" value="John" readonly>

## **The disabled Attribute**

First name:<br>  <input type="text" name="firstname" value="John" disabled>

## **The size Attribute**

First name:<br>  <input type="text" name="firstname" value="John" size="40">

## **The maxlength Attribute**

 First name:<br>  <input type="text" name="firstname" maxlength="10">

## **HTML5 Attributes**

HTML5 added the following attributes for <input>:

* autocomplete
* autofocus
* form
* formaction
* formenctype
* formmethod
* formnovalidate
* formtarget
* height and width
* list
* min and max
* multiple
* pattern (regexp)
* placeholder
* required
* step

and the following attributes for <form>:

* autocomplete
* novalidate

## **The autocomplete Attribute**

The autocomplete attribute specifies whether a form or input field should have autocomplete on or off.

When autocomplete is on, the browser automatically completes the input values based on values that the user has entered before.

**Tip:** It is possible to have autocomplete "on" for the form, and "off" for specific input fields, or vice versa.

The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

<form action="/action\_page.php" autocomplete="on">  
  First name:<input type="text" name="fname"><br>  
  Last name: <input type="text" name="lname"><br>  
  E-mail: <input type="email" name="email" autocomplete="off"><br>  
  <input type="submit">  
</form>

## **The novalidate Attribute**

The novalidate attribute is a <form> attribute.

<form action="/action\_page.php" novalidate>  
  E-mail: <input type="email" name="user\_email">  
  <input type="submit">  
</form>

## **The autofocus Attribute**

First name:<input type="text" name="fname" autofocus>

## **The form Attribute**

<form action="/action\_page.php" id="form1">  
  First name: <input type="text" name="fname"><br>  
  <input type="submit" value="Submit">  
</form>  
  
Last name: <input type="text" name="lname" form="form1">

## **The formaction Attribute**

The formaction attribute specifies the URL of a file that will process the input control when the form is submitted.

The formaction attribute overrides the action attribute of the <form> element.

The formaction attribute is used with type="submit" and type="image".

<form action="/action\_page.php">  
  First name: <input type="text" name="fname"><br>  
  Last name: <input type="text" name="lname"><br>  
  <input type="submit" value="Submit"><br>  
  <input type="submit" formaction="/action\_page2.php"  
  value="Submit as admin">  
</form>

## **The formenctype Attribute**

The formenctype attribute specifies how the form data should be encoded when submitted (only for forms with method="post").

The formenctype attribute overrides the enctype attribute of the <form> element.

The formenctype attribute is used with type="submit" and type="image".

Send form-data that is default encoded (the first submit button), and encoded as "multipart/form-data" (the second submit button):

<form action="/action\_page\_binary.asp" method="post">  
  First name: <input type="text" name="fname"><br>  
  <input type="submit" value="Submit">  
  <input type="submit" formenctype="multipart/form-data"  
  value="Submit as Multipart/form-data">  
</form>

## **The formmethod Attribute**

The formmethod attribute defines the HTTP method for sending form-data to the action URL.

The formmethod attribute overrides the method attribute of the <form> element.

The formmethod attribute can be used with type="submit" and type="image".

<form action="/action\_page.php" method="get">  
  First name: <input type="text" name="fname"><br>  
  Last name: <input type="text" name="lname"><br>  
  <input type="submit" value="Submit">  
  <input type="submit" formmethod="post" value="Submit using POST">  
</form>

## **The formnovalidate Attribute**

The formnovalidate attribute overrides the novalidate attribute of the <form> element.

The formnovalidate attribute can be used with type="submit".

<form action="/action\_page.php">  
  E-mail: <input type="email" name="userid"><br>  
  <input type="submit" value="Submit"><br>  
  <input type="submit" formnovalidate value="Submit without validation">  
</form>

## **The formtarget Attribute**

The formtarget attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.

The formtarget attribute overrides the target attribute of the <form> element.

The formtarget attribute can be used with type="submit" and type="image".

<form action="/action\_page.php">  
  First name: <input type="text" name="fname"><br>  
  Last name: <input type="text" name="lname"><br>  
  <input type="submit" value="Submit as normal">  
  <input type="submit" formtarget="\_blank"  
  value="Submit to a new window">  
</form>

## **The height and width Attributes**

<input type="image" src="img\_submit.gif" alt="Submit" width="48" height="48">

## **The list Attribute**

The list attribute refers to a <datalist> element that contains pre-defined options for an <input> element.

<input list="browsers">  
  
<datalist id="browsers">  
  <option value="Internet Explorer">  
  <option value="Firefox">  
  <option value="Chrome">  
  <option value="Opera">  
  <option value="Safari">  
</datalist>

## **The min and max Attributes**

The min and max attributes specify the minimum and maximum values for an <input> element.

The min and max attributes work with the following input types: number, range, date, datetime-local, month, time and week.

Enter a date before 1980-01-01:  
<input type="date" name="bday" max="1979-12-31">  
  
Enter a date after 2000-01-01:  
<input type="date" name="bday" min="2000-01-02">  
  
Quantity (between 1 and 5):  
<input type="number" name="quantity" min="1" max="5">

## **The multiple Attribute**

The multiple attribute specifies that the user is allowed to enter more than one value in the <input> element.

The multiple attribute works with the following input types: email, and file.

Select images: <input type="file" name="img" multiple>

## **The pattern Attribute**

The pattern attribute specifies a regular expression that the <input> element's value is checked against.

The pattern attribute works with the following input types: text, search, url, tel, email, and password.

**Tip:** Use the global [title](https://www.w3schools.com/tags/att_global_title.asp) attribute to describe the pattern to help the user.

Country code: <input type="text" name="country\_code" pattern="[A-Za-z]{3}" title="Three letter country code">

## **The placeholder Attribute**

The placeholder attribute specifies a hint that describes the expected value of an input field (a sample value or a short description of the format).

The hint is displayed in the input field before the user enters a value.

The placeholder attribute works with the following input types: text, search, url, tel, email, and password.

<input type="text" name="fname" placeholder="First name">

## **The required Attribute**

The required attribute specifies that an input field must be filled out before submitting the form.

The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

Username: <input type="text" name="usrname" required>

## **The step Attribute**

The step attribute specifies the legal number intervals for an <input> element.

Example: if step="3", legal numbers could be -3, 0, 3, 6, etc.

<input type="number" name="points" step="3">

# HTML5 Introduction

# The DOCTYPE declaration for HTML5 is very simple:

<!DOCTYPE html>

The character encoding (charset) declaration is also very simple:

<meta charset="UTF-8">

## **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 HTML5 API's (Application Programming Interfaces)**

The most interesting new API's in HTML5 are:

* HTML Geolocation
* HTML Drag and Drop
* HTML Local Storage
* HTML Application Cache
* HTML Web Workers
* HTML SSE

## **HTML5 Browser Support**

HTML5 is supported in all modern browsers.

In addition, all browsers, old and new, automatically handle unrecognized elements as inline elements.

Because of this, you can "teach" older browsers to handle "unknown" HTML elements.

## **Define Semantic Elements as Block Elements**

HTML5 defines eight new **semantic** elements. All these are **block-level** elements.

To secure correct behavior in older browsers, you can set the CSS **display** property for these HTML elements to **block**:

header, section, footer, aside, nav, main, article, figure {  
  display: block;   
}

## **Add New Elements to HTML**

You can also add new elements to an HTML page with a browser trick.

<!DOCTYPE html>  
<html>  
<head>  
<script>document.createElement("myHero")</script>  
<style>  
myHero {  
  display: block;  
  background-color: #dddddd;  
  padding: 50px;  
  font-size: 30px;  
}   
</style>   
</head>  
<body>  
  
<h1>A Heading</h1>  
<myHero>My Hero Element</myHero>  
  
</body>  
</html>

## **Syntax For HTML5Shiv**

The HTML5Shiv is placed within the <head> tag.

The HTML5Shiv is a javascript file that is referenced in a <script> tag.

You should use the HTML5Shiv when you are using the new HTML5 elements such as: <article>, <section>, <aside>, <nav>, <footer>.

You can [download the latest version of HTML5shiv from github](https://github.com/aFarkas/html5shiv) or reference the CDN version at<https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js>

<head>  
  <!--[if lt IE 9]>  
    <script src="/js/html5shiv.js"></script>  
  <![endif]-->  
</head>

## **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  <article>   <h1>Google Chrome</h1>   <p>Google Chrome is a free, open-source web browser developed by Google, released in 2008.</p> </article>   * Forum post * Blog post * News story * Comment |
| [<aside>](https://www.w3schools.com/tags/tag_aside.asp) | Defines content aside from the page content  <aside>   <h4>Epcot Center</h4>   <p>The Epcot Center is a theme park in Disney World, Florida.</p> </aside>  The <aside> tag defines some content aside from the content it is placed in.  The aside content should be related to the surrounding 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  <details>   <summary>Copyright 1999-2018.</summary>   <p> - by Refsnes Data. All Rights Reserved.</p>   <p>All content and graphics on this web site are the property of the company Refsnes Data.</p> </details>  The <details> tag specifies additional details that the user can view or hide on demand.  The <details> tag can be used to create an interactive widget that the user can open and close. Any sort of content can be put inside the <details> tag.  The content of a <details> element should not be visible unless the open attribute is set. |
| [<dialog>](https://www.w3schools.com/tags/tag_dialog.asp) | Defines a dialog box or window  <tr>   <th>January <dialog open>This is an open dialog window</dialog></th>   <th>February</th>   <th>March</th> </tr>  The <dialog> tag defines a dialog box or window.  The <dialog> element makes it easy to create popup dialogs and modals on a web page. |
| [<figcaption>](https://www.w3schools.com/tags/tag_figcaption.asp) | Defines a caption for a <figure> element  <figure>   <img src="pic\_trulli.jpg" alt="Trulli" style="width:100%">   <figcaption>Fig.1 - Trulli, Puglia, Italy.</figcaption> </figure>  The <figcaption> tag defines a caption for a [<figure>](https://www.w3schools.com/tags/tag_figure.asp) element.  The <figcaption> element can be placed as the first or last child of the <figure> element. |
| [<figure>](https://www.w3schools.com/tags/tag_figure.asp) | Defines self-contained content  <figure>   <img src="img\_pulpit.jpg" alt="The Pulpit Rock" width="304" height="228"> </figure>  The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc. |
| [<footer>](https://www.w3schools.com/tags/tag_footer.asp) | Defines 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> |
| [<header>](https://www.w3schools.com/tags/tag_header.asp) | Defines a header for a document or section  <article>   <header>     <h1>Most important heading here</h1>     <h3>Less important heading here</h3>     <p>Some additional information here</p>   </header>   <p>Lorem Ipsum dolor set amet....</p> </article> |
| [<main>](https://www.w3schools.com/tags/tag_main.asp) | Defines the main content of a document  <main>   <h1>Web Browsers</h1>   <p>Google Chrome, Firefox, and Internet Explorer are the most used browsers today.</p>    <article>     <h1>Google Chrome</h1>     <p>Google Chrome is a free, open-source web browser developed by Google,     released in 2008.</p>   </article>    <article>     <h1>Internet Explorer</h1>     <p>Internet Explorer is a free web browser from Microsoft, released in 1995.</p>   </article>    <article>     <h1>Mozilla Firefox</h1>     <p>Firefox is a free, open-source web browser from Mozilla, released in 2004.</p>   </article> </main> |
| [<mark>](https://www.w3schools.com/tags/tag_mark.asp) | Defines marked/highlighted text  <p>Do not forget to buy <mark>milk</mark> today.</p> |
| [<meter>](https://www.w3schools.com/tags/tag_meter.asp) | Defines a scalar measurement within a known range (a gauge)  <meter value="2" min="0" max="10">2 out of 10</meter><br> <meter value="0.6">60%</meter> |
| [<nav>](https://www.w3schools.com/tags/tag_nav.asp) | Defines navigation links  <nav>   <a href="/html/">HTML</a> |   <a href="/css/">CSS</a> |   <a href="/js/">JavaScript</a> |   <a href="/jquery/">jQuery</a> </nav> |
| [<progress>](https://www.w3schools.com/tags/tag_progress.asp) | Represents the progress of a task  <progress value="22" max="100"></progress> |
| [<rp>](https://www.w3schools.com/tags/tag_rp.asp) | Defines what to show in browsers that do not support ruby annotations  <ruby> 漢 <rt><rp>(</rp>ㄏㄢˋ<rp>)</rp></rt> </ruby>  The <rp> tag can be used to provide parentheses around a ruby text, to be shown by browsers that do not support ruby annotations.  Use the <rp> tag together with the [<ruby>](https://www.w3schools.com/tags/tag_ruby.asp) and the [<rt>](https://www.w3schools.com/tags/tag_rt.asp) tags: The <ruby> element consists of one or more characters that needs an explanation/pronunciation, and an <rt> element that gives that information, and an optional <rp> element that defines what to show for browsers that not support ruby annotations. |
| [<rt>](https://www.w3schools.com/tags/tag_rt.asp) | Defines an explanation/pronunciation of characters (for East Asian typography)  The <rt> tag defines an explanation or pronunciation of characters (for East Asian typography) in a ruby annotation. |
| [<ruby>](https://www.w3schools.com/tags/tag_ruby.asp) | Defines a ruby annotation (for East Asian typography)  The <ruby> tag specifies a ruby annotation.  A ruby annotation is a small extra text, attached to the main text to indicate the pronunciation or meaning of the corresponding characters. This kind of annotation is often used in Japanese publications. |
| [<section>](https://www.w3schools.com/tags/tag_section.asp) | Defines a section in a document  <section>   <h1>WWF</h1>   <p>The World Wide Fund for Nature (WWF) is....</p> </section>  The <section> tag defines sections in a document, such as chapters, headers, footers, or any other sections of the document. |
| [<summary>](https://www.w3schools.com/tags/tag_summary.asp) | Defines a visible heading for a <details> element  <details>   <summary>Copyright 1999-2014.</summary>   <p> - by Refsnes Data. All Rights Reserved.</p>   <p>All content and graphics on this web site are the property of the company Refsnes Data.</p> </details>  The <summary> tag defines a visible heading for the [<details>](https://www.w3schools.com/tags/tag_details.asp) element. The heading can be clicked to view/hide the details. |
| [<time>](https://www.w3schools.com/tags/tag_time.asp) | Defines a date/time  <p>We open at <time>10:00</time> every morning.</p>  he <time> tag defines a human-readable date/time. |
| [<wbr>](https://www.w3schools.com/tags/tag_wbr.asp) | Defines a possible line-break  <p>This is a veryveryveryveryveryveryveryveryveryveryveryveryveryveryv  eryveryveryvery<wbr>longwordthatwillbreakatspecific<wbr>  placeswhenthebrowserwindowisresized.</p> |

## **New Form Elements**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<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 |

## **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'**> |

## **HTML5 Graphics**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<canvas>](https://www.w3schools.com/tags/tag_canvas.asp) | Draw graphics, on the fly, via scripting (usually JavaScript) |
| [<svg>](https://www.w3schools.com/tags/tag_svg.asp) | Draw scalable vector graphics |

## **New Media Elements**

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<audio>](https://www.w3schools.com/tags/tag_audio.asp) | Defines sound content |
| [<embed>](https://www.w3schools.com/tags/tag_embed.asp) | Defines a container for an external (non-HTML) application |
| [<source>](https://www.w3schools.com/tags/tag_source.asp) | Defines multiple media resources for media elements (<video> and <audio>) |
| [<track>](https://www.w3schools.com/tags/tag_track.asp) | Defines text tracks for media elements (<video> and <audio>) |
| [<video>](https://www.w3schools.com/tags/tag_video.asp) | Defines video or movie |

## **New Semantic Elements in HTML5**

Many web sites contain HTML code like: <div id="nav"> <div class="header"> <div id="footer">  
to indicate navigation, header, and footer.

HTML5 offers new semantic elements to define different parts of a web page:

* <article>
* <aside>
* <details>
* <figcaption>
* <figure>
* <footer>
* <header>
* <main>
* <mark>
* <nav>
* <section>
* <summary>
* <time>



# HTML5 Migration

## **Migration from HTML4 to HTML5**

This chapter is entirely about how to **migrate** from **HTML4** to **HTML5**.

You can migrate from XHTML to HTML5, using the same recipe.

|  |  |
| --- | --- |
| **Typical HTML4** | **Typical HTML5** |
| <div id="header"> | <header> |
| <div id="menu"> | <nav> |
| <div id="content"> | <section> |
| <div class="article"> | <article> |
| <div id="footer"> | <footer> |

**HTML 4 Page**

<!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>

**HTML 5 Page**

<!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>

</style>

</head>

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;  
}

<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..</p>  
  </article>  
  <article>  
    <h2>News Article</h2>  
    <p>Lorem ipsum dolor sit amet..</p>  
  </article>  
</section>  
  
<footer>  
  <p>&copy; 2014 Monday Times. All rights reserved.</p>  
</footer>  
  
</body>

</html>

## **The Difference Between <article> <section> and <div>**

There is a confusing (lack of) difference in the HTML5 standard, between <article> <section> and <div>.

In the HTML5 standard, the <section> element is defined as a block of related elements.

The <article> element is defined as a complete, self-contained block of related elements.

The <div> element is defined as a block of children elements.

# HTML5 Style Guide and Coding Conventions

# Use Correct Document Type

# Use Lower Case Element Names

# Close All HTML Elements

# Close Empty HTML Elements

# Use Lower Case Attribute Names

# Quote Attribute Values

# Image Attributes

# Spaces and Equal Signs

# Avoid Long Code Lines

# Blank Lines and Indentation

# Omitting <html> and <body>?

# Omitting <head>?

# Meta Data

# Setting The Viewport

# HTML Comments

# Style Sheets

# Loading JavaScript in HTML

# Accessing HTML Elements with JavaScript

# Use Lower Case File Names

# File Extensions

# Differences Between .htm and .html

# HTML5 Canvas

The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.

The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;"></canvas>

var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");  
ctx.moveTo(0, 0);  
ctx.lineTo(200, 100);  
ctx.stroke();

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");  
ctx.beginPath();  
ctx.arc(95, 50, 40, 0, 2 \* Math.PI);  
ctx.stroke();

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");  
ctx.font = "30px Arial";  
ctx.fillText("Hello World", 10, 50);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");  
ctx.font = "30px Arial";  
ctx.fillText("Hello World", 10, 50);

var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");  
  
// Create gradient  
var grd = ctx.createLinearGradient(0, 0, 200, 0);  
grd.addColorStop(0, "red");  
grd.addColorStop(1, "white");  
  
// Fill with gradient  
ctx.fillStyle = grd;  
ctx.fillRect(10, 10, 150, 80);

var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");  
  
// Create gradient  
var grd = ctx.createRadialGradient(75, 50, 5, 90, 60, 100);  
grd.addColorStop(0, "red");  
grd.addColorStop(1, "white");  
  
// Fill with gradient  
ctx.fillStyle = grd;  
ctx.fillRect(10, 10, 150, 80);

var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");  
var img = document.getElementById("scream");  
ctx.drawImage(img, 10, 10);

## **What is SVG?**

* SVG stands for Scalable Vector Graphics
* SVG is used to define graphics for the Web
* SVG is a W3C recommendation

<svg width="100" height="100">  
  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />  
</svg>

<svg width="400" height="100">  
  <rect width="400" height="100" style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)" />  
</svg>

<svg width="400" height="180">  
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"  
  style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />  
</svg>

<svg width="300" height="200">  
  <polygon points="100,10 40,198 190,78 10,78 160,198"  
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />  
</svg>

## **Differences Between SVG and Canvas**

SVG is a language for describing 2D graphics in XML.

Canvas draws 2D graphics, on the fly (with a JavaScript).

SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.

In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.

Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

## **Comparison of Canvas and SVG**

The table below shows some important differences between Canvas and SVG:

|  |  |
| --- | --- |
| **Canvas** | **SVG** |
| * Resolution dependent * No support for event handlers * Poor text rendering capabilities * You can save the resulting image as .png or .jpg * Well suited for graphic-intensive games | * Resolution independent * Support for event handlers * Best suited for applications with large rendering areas (Google Maps) * Slow rendering if complex (anything that uses the DOM a lot will be slow) * Not suited for game applications |

# HTML Multimedia

Multimedia comes in many different formats. It can be almost anything you can hear or see.

Examples: Images, music, sound, videos, records, films, animations, and more.

Web pages often contain multimedia elements of different types and formats.

The first web browsers had support for text only, limited to a single font in a single color.

Later came browsers with support for colors and fonts, and images!

Audio, video, and animation have been handled differently by the major browsers. Different formats have been supported, and some formats require extra helper programs (plug-ins) to work.

## **Multimedia Formats**

Multimedia elements (like audio or video) are stored in media files.

The most common way to discover the type of a file, is to look at the file extension.

Multimedia files have formats and different extensions like: .swf, .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

MP4 is the new and upcoming format for internet video.  
  
MP4 is recommended by YouTube.  
  
MP4 is supported by Flash Players.  
  
MP4 is supported by HTML5.

# HTML5 Video

<video width="320" height="240" controls>  
  <source src="movie.mp4" type="video/mp4">  
  <source src="movie.ogg" type="video/ogg">  
Your browser does not support the video tag.  
</video>

<video width="320" height="240" autoplay>  
  <source src="movie.mp4" type="video/mp4">  
  <source src="movie.ogg" type="video/ogg">  
Your browser does not support the video tag.  
</video>

The controls attribute adds video controls, like play, pause, and volume.

It is a good idea to always include width and height attributes. If height and width are not set, the page might flicker while the video loads.

The <source> element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.

<!DOCTYPE html>

<html>

<body>

<div style="text-align:center">

<button onclick="playPause()">Play/Pause</button>

<button onclick="makeBig()">Big</button>

<button onclick="makeSmall()">Small</button>

<button onclick="makeNormal()">Normal</button>

<br><br>

<video id="video1" width="420">

<source src="mov\_bbb.mp4" type="video/mp4">

<source src="mov\_bbb.ogg" type="video/ogg">

Your browser does not support HTML5 video.

</video>

</div>

<script>

var myVideo = document.getElementById("video1");

function playPause() {

if (myVideo.paused)

myVideo.play();

else

myVideo.pause();

}

function makeBig() {

myVideo.width = 560;

}

function makeSmall() {

myVideo.width = 320;

}

function makeNormal() {

myVideo.width = 420;

}

</script>

<p>Video courtesy of <a href="https://www.bigbuckbunny.org/" target="\_blank">Big Buck Bunny</a>.</p>

</body>

<video width="320" height="240" controls>  
  <source src="forrest\_gump.mp4" type="video/mp4">  
  <source src="forrest\_gump.ogg" type="video/ogg">  
  <track src="subtitles\_en.vtt" kind="subtitles" srclang="en" label="English">  
  <track src="subtitles\_no.vtt" kind="subtitles" srclang="no" label="Norwegian">  
</video>

## **Definition and Usage**

The <track> tag specifies text tracks for media elements (<audio> and <video>).

This element is used to specify subtitles, caption files or other files containing text, that should be visible when the media is playing.

# HTML5 Audio

## **Audio on the Web**

Before HTML5, audio files could only be played in a browser with a plug-in (like flash).

The HTML5 <audio> element specifies a standard way to embed audio in a web page.

<audio controls>  
  <source src="horse.ogg" type="audio/ogg">  
  <source src="horse.mp3" type="audio/mpeg">  
Your browser does not support the audio element.  
</audio>

# HTML Plug-ins

## **HTML Helpers (Plug-ins)**

Helper applications (plug-ins) are computer programs that extend the standard functionality of a web browser.

Examples of well-known plug-ins are Java applets.

Plug-ins can be added to web pages with the <object> tag or the <embed> tag.

Plug-ins can be used for many purposes: display maps, scan for viruses, verify your bank id, etc.

## **The <object> Element**

The <object> element is supported by all browsers.

The <object> element defines an embedded object within an HTML document.

<object width="400" height="50" data="bookmark.swf"></object>

<object width="100%" height="500px" data="snippet.html"></object>

<object data="audi.jpeg"></object>

## **The <embed> Element**

The <embed> element is supported in all major browsers.

The <embed> element also defines an embedded object within an HTML document.

<embed width="400" height="50" src="bookmark.swf">

<embed width="100%" height="500px" src="snippet.html">

<embed src="audi.jpeg">

# HTML YouTube Videos

## **Struggling with Video Formats?**

Earlier in this tutorial, you have seen that you might have to convert your videos to different formats to make them play in all browsers.

Converting videos to different formats can be difficult and time-consuming.

An easier solution is to let YouTube play the videos in your web page.

## **YouTube Video Id**

YouTube will display an id (like tgbNymZ7vqY), when you save (or play) a video.

You can use this id, and refer to your video in the HTML code.

## **Playing a YouTube Video in HTML**

To play your video on a web page, do the following:

* Upload the video to YouTube
* Take a note of the video id
* Define an <iframe> element in your web page
* Let the src attribute point to the video URL
* Use the width and height attributes to specify the dimension of the player
* Add any other parameters to the URL (see below)

<iframe width="420" height="315"  
src="https://www.youtube.com/embed/tgbNymZ7vqY">  
</iframe>

## **YouTube Autoplay**

**Note:** Take careful consideration when deciding to autoplay your videos. Automatically starting a video can annoy your visitor and end up causing more harm than good.

Value 0 (default): The video will not play automatically when the player loads.

Value 1: The video will play automatically when the player loads.

<iframe width="420" height="315"  
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1">  
</iframe>

## **YouTube Playlist**

A comma separated list of videos to play (in addition to the original URL).

## **YouTube Loop**

Value 0 (default): The video will play only once.

Value 1: The video will loop (forever).

<iframe width="420" height="315"  
src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNymZ7vqY&loop=1">  
</iframe>

## **YouTube Controls**

Value 0: Player controls does not display.

Value 1 (default): Player controls display.

<iframe width="420" height="315"  
src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0">  
</iframe>

## **YouTube - Using <object> or <embed>**

<object width="420" height="315"  
data="https://www.youtube.com/embed/tgbNymZ7vqY">  
</object>

<embed width="420" height="315"  
src="https://www.youtube.com/embed/tgbNymZ7vqY">

# HTML5 Geolocation

## **Locate the User's Position**

The HTML Geolocation API is used to get the geographical position of a user.

Since this can compromise privacy, the position is not available unless the user approves it.

## **Using HTML Geolocation**

The getCurrentPosition() method is used to return the user's position.

<p>Click the button to get your coordinates.</p>

<button onclick="getLocation()">Try It</button>

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

<script>

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

function getLocation() {

if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(showPosition, showError);

} else {

x.innerHTML = "Geolocation is not supported by this browser.";

}

}

function showPosition(position) {

x.innerHTML = "Latitude: " + position.coords.latitude +

"<br>Longitude: " + position.coords.longitude;

}

function showError(error) {

switch(error.code) {

case error.PERMISSION\_DENIED:

x.innerHTML = "User denied the request for Geolocation."

break;

case error.POSITION\_UNAVAILABLE:

x.innerHTML = "Location information is unavailable."

break;

case error.TIMEOUT:

x.innerHTML = "The request to get user location timed out."

break;

case error.UNKNOWN\_ERROR:

x.innerHTML = "An unknown error occurred."

break;

}

}

</script>

## **Location-specific Information**

This page has demonstrated how to show a user's position on a map.

Geolocation is also very useful for location-specific information, like:

* Up-to-date local information
* Showing Points-of-interest near the user
* Turn-by-turn navigation (GPS)

## **The getCurrentPosition() Method - Return Data**

The getCurrentPosition() method returns an object on success. The latitude, longitude and accuracy properties are always returned. The other properties are returned if available:

|  |  |
| --- | --- |
| **Property** | **Returns** |
| coords.latitude | The latitude as a decimal number (always returned) |
| coords.longitude | The longitude as a decimal number (always returned) |
| coords.accuracy | The accuracy of position (always returned) |
| coords.altitude | The altitude in meters above the mean sea level (returned if available) |
| coords.altitudeAccuracy | The altitude accuracy of position (returned if available) |
| coords.heading | The heading as degrees clockwise from North (returned if available) |
| coords.speed | The speed in meters per second (returned if available) |
| timestamp | The date/time of the response (returned if available) |

## **Geolocation Object - Other interesting Methods**

* watchPosition() - Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).
* clearWatch() - Stops the watchPosition() method.

<script>  
var x = document.getElementById("demo");  
function getLocation() {  
  if (navigator.geolocation) {  
    navigator.geolocation.watchPosition(showPosition);  
  } else {  
    x.innerHTML = "Geolocation is not supported by this browser.";  
  }  
}  
function showPosition(position) {  
  x.innerHTML = "Latitude: " + position.coords.latitude +   
  "<br>Longitude: " + position.coords.longitude;   
}  
</script>

# HTML5 Drag and Drop

Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.

<!DOCTYPE HTML>  
<html>  
<head>  
<script>  
function allowDrop(ev) {  
  ev.preventDefault();  
}  
  
function drag(ev) {  
  ev.dataTransfer.setData("text", ev.target.id);  
}  
  
function drop(ev) {  
  ev.preventDefault();  
  var data = ev.dataTransfer.getData("text");  
  ev.target.appendChild(document.getElementById(data));  
}  
</script>  
</head>  
<body>  
  
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>  
  
<img id="drag1" src="img\_logo.gif" draggable="true" ondragstart="drag(event)" width="336" height="69">  
  
</body>  
</html>

### **Drag image back and forth**

<!DOCTYPE HTML>

<html>

<head>

<style>

#div1, #div2 {

float: left;

width: 100px;

height: 35px;

margin: 10px;

padding: 10px;

border: 1px solid black;

}

</style>

<script>

function allowDrop(ev) {

ev.preventDefault();

}

function drag(ev) {

ev.dataTransfer.setData("text", ev.target.id);

}

function drop(ev) {

ev.preventDefault();

var data = ev.dataTransfer.getData("text");

ev.target.appendChild(document.getElementById(data));

}

</script>

</head>

<body>

<h2>Drag and Drop</h2>

<p>Drag the image back and forth between the two div elements.</p>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">

<img src="img\_w3slogo.gif" draggable="true" ondragstart="drag(event)" id="drag1" width="88" height="31">

</div>

<div id="div2" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

</body>

</html>

# HTML5 Web Storage

With web storage, web applications can store data locally within the user's browser.

Before HTML5, application data had to be stored in cookies, included in every server request. Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.

Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

## **HTML Web Storage Objects**

HTML web storage provides two objects for storing data on the client:

* window.localStorage - stores data with no expiration date
* window.sessionStorage - stores data for one session (data is lost when the browser tab is closed)

if (typeof(Storage) !== "undefined") {  
  // *Code for localStorage/sessionStorage.*  
} else {  
  // Sorry! No Web Storage support..  
}

## **The localStorage Object**

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

<script>

if (typeof(Storage) !== "undefined") {

localStorage.setItem("lastname", "Smith");

document.getElementById("result").innerHTML = localStorage.getItem("lastname");

} else {

document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Storage...";

}

</script>

// Store  
localStorage.setItem("lastname", "Smith");  
  
// Retrieve  
document.getElementById("result").innerHTML = localStorage.getItem("lastname");

// Store  
localStorage.lastname = "Smith";  
// Retrieve  
document.getElementById("result").innerHTML = localStorage.lastname;

localStorage.removeItem("lastname");

<script>

function clickCounter() {

if (typeof(Storage) !== "undefined") {

if (localStorage.clickcount) {

localStorage.clickcount = Number(localStorage.clickcount)+1;

} else {

localStorage.clickcount = 1;

}

document.getElementById("result").innerHTML = "You have clicked the button " + localStorage.clickcount + " time(s).";

} else {

document.getElementById("result").innerHTML = "Sorry, your browser does not support web storage...";

}

}

</script>

</head>

<body>

<p><button onclick="clickCounter()" type="button">Click me!</button></p>

<div id="result"></div>

<p>Click the button to see the counter increase.</p>

<p>Close the browser tab (or window), and try again, and the counter will continue to count (is not reset).</p>

## **The sessionStorage Object**

The sessionStorage object is equal to the localStorage object, **except** that it stores the data for only one session. The data is deleted when the user closes the specific browser tab.

<script>

function clickCounter() {

if (typeof(Storage) !== "undefined") {

if (sessionStorage.clickcount) {

sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;

} else {

sessionStorage.clickcount = 1;

}

document.getElementById("result").innerHTML = "You have clicked the button " + sessionStorage.clickcount + " time(s) in this session.";

} else {

document.getElementById("result").innerHTML = "Sorry, your browser does not support web storage...";

}

}

</script>

</head>

<body>

<p><button onclick="clickCounter()" type="button">Click me!</button></p>

<div id="result"></div>

<p>Click the button to see the counter increase.</p>

<p>Close the browser tab (or window), and try again, and the counter is reset.</p>

# HTML5 Web Workers

A web worker is a JavaScript running in the background, without affecting the performance of the page.

When executing scripts in an HTML page, the page becomes unresponsive until the script is finished.

A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

<body>  
<p>Count numbers: <output id="result"></output></p>  
<button onclick="startWorker()">Start Worker</button>   
<button onclick="stopWorker()">Stop Worker</button>  
  
<script>  
var w;  
function startWorker() {  
  if (typeof(Worker) !== "undefined") {  
    if (typeof(w) == "undefined") {  
      w = new Worker("demo\_workers.js");  
    }  
    w.onmessage = function(event) {  
      document.getElementById("result").innerHTML = event.data;  
    };  
  } else {  
    document.getElementById("result").innerHTML = "Sorry! No Web Worker support.";  
  }  
}  
function stopWorker() {   
  w.terminate();  
  w = undefined;  
}  
</script>  
</body>

demo\_workers.js

var i=0;

function timedCount() {

i=i+1;

postMessage(i);

setTimeout("timedCount()", 500);

}

timedCount();

## **Web Workers and the DOM**

Since web workers are in external files, they do not have access to the following JavaScript objects:

* The window object
* The document object
* The parent object

# HTML5 Server-Sent Events

Server-Sent Events allow a web page to get updates from a server.

## **Server-Sent Events - One Way Messaging**

A server-sent event is when a web page automatically gets updates from a server.

This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.

Examples: Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

## **Receive Server-Sent Event Notifications**

<body>

<h1>Getting server updates</h1>

<div id="result"></div>

<script>

if(typeof(EventSource) !== "undefined") {

var source = new EventSource("demo\_sse.php");

source.onmessage = function(event) {

document.getElementById("result").innerHTML += event.data + "<br>";

};

} else {

document.getElementById("result").innerHTML = "Sorry, your browser does not support server-sent events...";

}

</script>

</body>