# HTML5

**New Features**

HTML5 introduces a number of new elements and attributes that can help you in building modern websites. Here is a set of some of the most prominent features introduced in HTML5.

* **New Semantic Elements** − These are like <header>, <footer>, and <section>.
* **Forms 2.0** − Improvements to HTML web forms where new attributes have been introduced for <input> tag.
* **Persistent Local Storage** − To achieve without resorting to third-party plugins.
* **WebSocket** − A next-generation bidirectional communication technology for web applications.
* **Server-Sent Events** − HTML5 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE).
* **Canvas** − This supports a two-dimensional drawing surface that you can program with JavaScript.
* **Audio & Video** − You can embed audio or video on your webpages without resorting to third-party plugins.
* **Geolocation** − Now visitors can choose to share their physical location with your web application.
* **Microdata** − This lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.
* **Drag and drop** − Drag and drop the items from one location to another location on the same webpage.

HTML5 comes with a lot of flexibility and it supports the following features −

* Uppercase tag names.
* Quotes are optional for attributes.
* Attribute values are optional.
* Closing empty elements are optional.

<!DOCTYPE html>

The above syntax is case-insensitive.

## Character Encoding

<meta charset = "UTF-8">

## The <script> tag

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

## The <link> tag

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

* HTML5 tag names are case insensitive and may be written in all uppercase or mixed case, although the most common convention is to stick with lowercase.

## HTML5 Document

The following tags have been introduced for better structure −

* **section** − This tag represents a generic document or application section. It can be used together with h1-h6 to indicate the document structure.
* **article** − This tag represents an independent piece of content of a document, such as a blog entry or newspaper article.
* **aside** − This tag represents a piece of content that is only slightly related to the rest of the page.
* **header** − This tag represents the header of a section.
* **footer** − This tag represents a footer for a section and can contain information about the author, copyright information, et cetera.
* **nav** − This tag represents a section of the document intended for navigation.
* **dialog** − This tag can be used to mark up a conversation.
* **figure** − This tag can be used to associate a caption together with some embedded content, such as a graphic or video.

The markup for an HTML 5 document would look like the following −

<!DOCTYPE html>

<html>

<head>

<meta charset = "utf-8">

<title>...</title>

</head>

<body>

<header>...</header>

<nav>...</nav>

<article>

<section>

...

</section>

</article>

<aside>...</aside>

<footer>...</footer>

</body>

</html>

* Some attributes are defined globally and can be used on any element, while others are defined for specific elements only. All attributes have a name and a value and look like as shown below in the example.

Standard Attributes

## Standard Attributes

The attributes listed below are supported by almost all the HTML 5 tags and there are many more.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Options** | **Function** |
| align | right, left, center | Horizontally aligns tags |
| background | URL | Places an background image behind an element |
| bgcolor | numeric, hexidecimal, RGB values | Places a background color behind an element |
| class | User Defined | Classifies an element for use with Cascading Style Sheets. |
| height | Numeric Value | Specifies the height of tables, images, or table cells. |
| hidden | hidden | Specifies whether element should be visible or not. |
| id | User Defined | Names an element for use with Cascading Style Sheets. |

## Custom Attributes

A new feature being introduced in HTML 5 is the addition of custom data attributes.

A custom data attribute starts with **data-** and would be named based on your requirement. Here is a simple example −

<div class = "example" data-subject = "physics" data-level = "complex">

...

</div>

The above code will be perfectly valid HTML5 with two custom attributes called *datasubject* and *data-level*. You would be able to get the values of these attributes using JavaScript APIs or CSS in similar way as you get for standard attributes.

# HTML5 - Events

When users visit your website, they perform various activities such as clicking on text and images and links, hover over defined elements, etc. These are examples of what JavaScript calls **events**.

We can use the following set of attributes to trigger any **javascript** or vbscript code given as value, when there is any event that takes place for any HTML5 element.

We would cover element-specific events while discussing those elements in detail in subsequent chapters and there are many more.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value** | **Description** |
| offline | script | Triggers when the document goes offline |
| onabort | script | Triggers on an abort event |
| onbeforeonload | script | Triggers before the document loads |
| onblur | script | Triggers when the window loses focus |
| oncanplay | script | Triggers when media can start play, but might has to stop for buffering |
| oncanplaythrough | script | Triggers when media can be played to the end, without stopping for buffering |
| onchange | script | Triggers when an element changes |
| onclick | script | Triggers on a mouse click |
| oncontextmenu | script | Triggers when a context menu is triggered |
| ondblclick | script | Triggers on a mouse double-click |
| ondrag | script | Triggers when an element is dragged |

# HTML5 - Web Forms 2.0

## The <input> element in HTML4

|  |  |
| --- | --- |
| **Sr.No.** | **Type & Description** |
| 1 | **text**  A free-form text field, nominally free of line breaks. |
| 2 | **password**  A free-form text field for sensitive information, nominally free of line breaks. |
| 3 | **checkbox**  A set of zero or more values from a predefined list. |
| 4 | **radio**  An enumerated value. |
| 5 | **submit**  A free form of button initiates form submission. |
| 6 | **file**  An arbitrary file with a MIME type and optionally a file name. |
| 7 | **image**  A coordinate, relative to a particular image's size, with the extra semantic that it must be the last value selected and initiates form submission. |
| 8 | **hidden**  An arbitrary string that is not normally displayed to the user. |
| 9 | **select**  An enumerated value, much like the radio type. |
| 10 | **textarea**  A free-form text field, nominally with no line break restrictions. |
| 11 | **button**  A free form of button which can initiates any event related to button. |

## The <input> element in HTML5

HTML5 input elements introduced several new values for the type attribute.

|  |  |
| --- | --- |
| **Sr.No.** | **Type & Description** |
| 1 | [**datetime**](https://www.tutorialspoint.com/html5/html5_datetime.htm)  A date and time (year, month, day, hour, minute, second, fractions of a second) encoded according to ISO 8601 with the time zone set to UTC. |
| 2 | [**datetime-local**](https://www.tutorialspoint.com/html5/html5_datetime_local.htm)  A date and time (year, month, day, hour, minute, second, fractions of a second) encoded according to ISO 8601, with no time zone information. |
| 3 | [**date**](https://www.tutorialspoint.com/html5/html5_date.htm)  A date (year, month, day) encoded according to ISO 8601. |
| 4 | [**month**](https://www.tutorialspoint.com/html5/html5_month.htm)  A date consisting of a year and a month encoded according to ISO 8601. |
| 5 | [**week**](https://www.tutorialspoint.com/html5/html5_week.htm)  A date consisting of a year and a week number encoded according to ISO 8601. |
| 6 | [**time**](https://www.tutorialspoint.com/html5/html5_time.htm)  A time (hour, minute, seconds, fractional seconds) encoded according to ISO 8601. |
| 7 | [**number**](https://www.tutorialspoint.com/html5/html5_number.htm)  It accepts only numerical value. The step attribute specifies the precision, defaulting to 1. |
| 8 | [**range**](https://www.tutorialspoint.com/html5/html5_range.htm)  The range type is used for input fields that should contain a value from a range of numbers. |
| 9 | [**email**](https://www.tutorialspoint.com/html5/html5_email.htm)  It accepts only email value. This type is used for input fields that should contain an e-mail address. If you try to submit a simple text, it forces to enter only email address in email@example.com format. |
| 10 | [**url**](https://www.tutorialspoint.com/html5/html5_url.htm)  It accepts only URL value. This type is used for input fields that should contain a URL address. If you try to submit a simple text, it forces to enter only URL address either in http://www.example.com format or in http://example.com format. |

## The <output> element

HTML5 introduced a new element <output> which is used to represent the result of different types of output, such as output written by a script.

You can use the for attribute to specify a relationship between the output element and other elements in the document that affected the calculation (for example, as inputs or parameters). The value of the for attribute is a space-separated list of IDs of other elements.

<!DOCTYPE HTML>

<html>

<head>

<script type = "text/javascript">

function showResult() {

x = document.forms["myform"]["newinput"].value;

document.forms["myform"]["result"].value = x;

}

</script>

</head>

<body>

<form action = "/cgi-bin/html5.cgi" method = "get" name = "myform">

Enter a value : <input type = "text" name = "newinput" />

<input type = "button" value = "Result" onclick = "showResult();" />

<output name = "result"></output>

</form>

</body>

</html>

## The placeholder attribute

HTML5 introduced a new attribute called **placeholder**. This attribute on <input> and <textarea> elements provide a hint to the user of what can be entered in the field. The placeholder text must not contain carriage returns or line-feeds.

## The autofocus attribute

This is a simple one-step pattern, easily programmed in JavaScript at the time of document load, automatically focus one particular form field.

Enter email : <input type = "text" name = "newinput" autofocus/>

## The required attribute

Now you do not need to have JavaScript for client-side validations like empty text box would never be submitted because HTML5 introduced a new attribute called **required.**

Enter email : <input type = "text" name = "newinput" required/>

# HTML5 - SVG

SVG stands for **S**calable **V**ector **G**raphics and it is a language for describing 2D-graphics and graphical applications in XML and the XML is then rendered by an SVG viewer.

SVG is mostly useful for vector type diagrams like Pie charts, Two-dimensional graphs in an X,Y coordinate system etc.

## Embedding SVG in HTML5

HTML5 allows embedding SVG directly using **<svg>...</svg>** tag which has following simple syntax −

<svg xmlns = "http://www.w3.org/2000/svg">

...

</svg>

HTML5 − SVG Circle

Following is the HTML5 version of an SVG example which would draw a circle using <circle> tag −

[Live Demo](http://tpcg.io/7DX1Da)

<!DOCTYPE html>

<html>

<head>

<style>

#svgelem {

position: relative;

left: 50%;

-webkit-transform: translateX(-20%);

-ms-transform: translateX(-20%);

transform: translateX(-20%);

}

</style>

<title>SVG</title>

<meta charset = "utf-8" />

</head>

<body>

<h2 align = "center">HTML5 SVG Circle</h2>

<svg id = "svgelem" height = "200" xmlns = "http://www.w3.org/2000/svg">

<circle id = "redcircle" cx = "50" cy = "50" r = "50" fill = "red" />

</svg>

</body>

</html>

You can create rectangle, line, ellipse, polygon, polyline, gradients, star etc.

# HTML5 - MathML

The HTML syntax of HTML5 allows for MathML elements to be used inside a document using <math>...</math> tags.

## MathML Examples

<!doctype html>

<html>

<head>

<meta charset = "UTF-8">

<title>Pythagorean theorem</title>

</head>

<body>

<math xmlns = "http://www.w3.org/1998/Math/MathML">

<mrow>

<msup><mi>a</mi><mn>2</mn></msup>

<mo>+</mo>

<msup><mi>b</mi><mn>2</mn></msup>

<mo> = </mo>

<msup><mi>c</mi><mn>2</mn></msup>

</mrow>

</math>

</body>

</html>

Result:

a2 + b2 = c2

# HTML5 - Web Storage

HTML5 introduces two mechanisms, similar to HTTP session cookies, for storing structured data on the client side and to overcome following drawbacks.

The two storages are **session storage** and **local storage** and they would be used to handle different situations.

## Session Storage

The *Session Storage* is designed for scenarios where the user is carrying out a single transaction, but could be carrying out multiple transactions in different windows at the same time.

### Example

*For example, if a user buying plane tickets in two different windows, using the same site. If the site used cookies to keep track of which ticket the user was buying, then as the user clicked from page to page in both windows, the ticket currently being purchased would "leak" from one window to the other, potentially causing the user to buy two tickets for the same flight without really noticing.*

HTML5 introduces the *sessionStorage* attribute which would be used by the sites to add data to the session storage, and it will be accessible to any page from the same site opened in that window, i.e., **session** and as soon as you close the window, the session would be lost.

Following is the code which would set a session variable and access that variable −

[Live Demo](http://tpcg.io/4rYmLB)

<!DOCTYPE HTML>

<html>

<body>

<script type = "text/javascript">

if( sessionStorage.hits ) {

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

} else {

sessionStorage.hits = 1;

}

document.write("Total Hits :" + sessionStorage.hits );

</script>

<p>Refresh the page to increase number of hits.</p>

<p>Close the window and open it again and check the result.</p>

</body>

</html>

## Local Storage

The *Local Storage* is designed for storage that spans multiple windows, and lasts beyond the current session. In particular, Web applications may wish to store megabytes of user data, such as entire user-authored documents or a user's mailbox, on the client side for performance reasons.

Again, cookies do not handle this case well, because they are transmitted with every request.

### Example

HTML5 introduces the *localStorage* attribute which would be used to access a page's local storage area without no time limit and this local storage will be available whenever you would use that page.

Following is the code which would set a local storage variable and access that variable every time this page is accessed, even next time, when you open the window −

<!DOCTYPE HTML>

<html>

<body>

<script type = "text/javascript">

if( localStorage.hits ) {

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

} else {

localStorage.hits = 1;

}

document.write("Total Hits :" + localStorage.hits );

</script>

<p>Refresh the page to increase number of hits.</p>

<p>Close the window and open it again and check the result.</p>

</body>

</html>

**Delete Web Storage**

Storing sensitive data on local machine could be dangerous and could leave a security hole.

The *Session Storage Data* would be deleted by the browsers immediately after the session gets terminated.

To clear a local storage setting you would need to call **localStorage.remove('key')**; where 'key' is the key of the value you want to remove. If you want to clear all settings, you need to call **localStorage.clear()** method.

Following is the code which would clear complete local storage −

[Live Demo](http://tpcg.io/dGhyzX)

<!DOCTYPE HTML>

<html>

<body>

<script type = "text/javascript">

localStorage.clear();

// Reset number of hits.

if( localStorage.hits ) {

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

} else {

localStorage.hits = 1;

}

document.write("Total Hits :" + localStorage.hits );

</script>

<p>Refreshing the page would not to increase hit counter.</p>

<p>Close the window and open it again and check the result.</p>

</body>

</html>