HTML 5 Tutorial

Tutorialspoint.com

HTML5 **is the latest and most enhanced version of HTML. Technically, HTML is not a programming language, but rather a markup language. This tutorial gives very good understanding on** ***HTML5*****.**

HTML5 Overview

HTML5 is the next major revision of the HTML standard superseding HTML 4.01, XHTML 1.0, and XHTML 1.1. HTML5 is a standard for structuring and presenting content on the World Wide Web.

HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).

The new standard incorporates features like video playback and drag-and-drop that have been previously dependent on third-party browser plug-ins such as Adobe Flash, Microsoft Silverlight, and Google Gears.

Browser Support:

The latest versions of Apple Safari, Google Chrome, Mozilla Firefox, and Opera all support many HTML5 features and Internet Explorer 9.0 will also have support for some HTML5 functionality.

The mobile web browsers that come pre-installed on iPhones, iPads, and Android phones all

have excellent support for HTML5.

New Features:

HTML5 introduces a number of new elements and attributes that helps in building a modern websites. Following are great 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 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 web pages 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 a the same webpage.

Backward Compatibility

HTML5 is designed, as much as possible, to be backward compatible with existing web browsers. New features build on existing features and allow you to provide fallback content for older browsers.

It is suggested to detect support for individual HTML5 features using a few lines of JavaScript.

If you are not familiar with any previous version of HTML, I would recommend to go through our HTML Tutorial before you explore further concepts of HTM5.

HTML5 Syntax

The HTML 5 language has a "custom" HTML syntax that is compatible with HTML 4 and XHTML1 documents published on the Web, but is not compatible with the more esoteric SGML features of HTML 4.

HTML 5 does not have the same syntax rules as XHTML where we needed lower case tag names, quoting our attributes,an attribute had to have a value and to close all empty elements.

But HTML5 is coming with lots of flexibility and would support the followings:

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

The DOCTYPE:

DOCTYPEs in older versions of HTML were longer because the HTML language was SGML based

and therefore required a reference to a DTD.

HTML 5 authors would use simple syntax to specify DOCTYPE as follows:

```
<!DOCTYPE html>
```

All the above syntax is case-insensitive.

Character Encoding:

HTML 5 authors can use simple syntax to specify Character Encoding as follows:

```
<meta charset="UTF-8">
```

All the above syntax is case-insensitive.

The <script> tag:

It's common practice to add a type attribute with a value of "text/javascript" to script elements as follows:

```
<script type="text/javascript" src="scriptfile.js"></script>
```

HTML 5 removes extra information required and you can use simply following syntax:

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

The <link> tag:

So far you were writing <link> as follows:

```
<link rel="stylesheet" type="text/css" href="stylefile.css">
```

HTML 5 removes extra information required and you can use simply following syntax:

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

HTML5 Elements:

HTML5 elements are marked up using start tags and end tags. Tags are delimited using angle brackets with the tag name in between.

The difference between start tags and end tags is that the latter includes a slash before the tag name.

Following is the example of an HTML5 element:

```
<p>...</p>
```

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.

Most of the elements contain some content like ... contains a paragraph. Some elements, however, are forbidden from containing any content at all and these are known as void elements. For example, br, hr, link and meta etc.

HTML5 Attributes:

Elements may contain attributes that are used to set various properties of an element.

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.

Following is the example of an HTML5 attributes which illustrates how to mark up a div element with an attribute named class using a value of "example":

```
<div class="example">...</div>
```

Attributes may only be specified within start tags and must never be used in end tags.

HTML5 attributes 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 HTM 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>
```

HTML5 Attributes

As explained in previous chapter, elements may contain attributes that are used to set various properties of an element.

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.

Following is the example of an HTML5 attributes which illustrates how to mark up a div element with an attribute named class using a value of "example":

```
<div class="example">...</div>
```

Attributes may only be specified within start tags and must never be used in end tags.

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

Standard Attributes:

The attributes listed below are supported by almost all the HTML 5 tags.

Attribute	Options	Function
accesskey	User Defined	Specifies a keyboard shortcut to access an element.
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.
contenteditable	true, false	Specifies if the user can edit the element's content or not.

contextmenu	Menu id	Specifies the context menu for an element.
data-XXXX	User Defined	Custom attributes. Authors of a HTML document can define their own attributes. Must start with "data-".
draggable	true,false, auto	Specifies whether or not a user is allowed to drag an element.
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.
item	List of elements	Used to group elements.
itemprop	List of items	Used to group items.
spellcheck	true, false	Specifies if the element must have it's spelling or grammar checked.
style	CSS Style sheet	Specifies an inline style for an element.
subject	User define id	Specifies the element's corresponding item.
tabindex	Tab number	Specifies the tab order of an element.
title	User Defined	"Pop-up" title for your elements.
valign	top, middle, bottom	Vertically aligns tags within an HTML element.
width	Numeric Value	Specifies the width of tables, images, or table cells.

For a complete list of HTML5 Tags and related attributes please check reference to HTML5 Tags.

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. Following is the simple example:

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

The above will be perfectly valid HTML5 with two custom attributes called *data-subject* 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 a user visit your website, they do things like click on text and images and given links, hover over things etc. These are examples of what JavaScript calls events.

We can write our event handlers in Javascript or vbscript and you can specify these event handlers as a value of event tag attribute. The HTML5 specification defines various event attributes as listed below:

There are following attributes which can be used to trigger any **javascript** or **vbscript** code given as value, when there is any event occurs for any HTM5 element.

We would cover element specific events while discussing those elements in detail in subsequent chapters.

Attribute	Value	Description
offline	script	Triggers when the document goes offline
onabort	script	Triggers on an abort event
onafterprint	script	Triggers after the document is printed
onbeforeonload	script	Triggers before the document loads
onbeforeprint	script	Triggers before the document is printed
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
ondragend	script	Triggers at the end of a drag operation
ondragenter	script	Triggers when an element has been dragged to a valid drop target
ondragleave	script	Triggers when an element leaves a valid drop target
ondragover	script	Triggers when an element is being dragged over a valid drop target
ondragstart	script	Triggers at the start of a drag operation
ondrop	script	Triggers when dragged element is being dropped
ondurationchange	script	Triggers when the length of the media is changed

onemptied	script	Triggers when a media resource element suddenly becomes empty.
onended	script	Triggers when media has reach the end
onerror	script	Triggers when an error occur
onfocus	script	Triggers when the window gets focus
onformchange	script	Triggers when a form changes
onforminput	script	Triggers when a form gets user input
onhaschange	script	Triggers when the document has change
oninput	script	Triggers when an element gets user input
oninvalid	script	Triggers when an element is invalid
onkeydown	script	Triggers when a key is pressed
onkeypress	script	Triggers when a key is pressed and released
onkeyup	script	Triggers when a key is released
onload	script	Triggers when the document loads
onloadeddata	script	Triggers when media data is loaded
onloadedmetadata	script	Triggers when the duration and other media data of a media element is loaded
onloadstart	script	Triggers when the browser starts to load the media data
onmessage	script	Triggers when the message is triggered
onmousedown	script	Triggers when a mouse button is pressed
onmousemove	script	Triggers when the mouse pointer moves
onmouseout	script	Triggers when the mouse pointer moves out of an element
onmouseover	script	Triggers when the mouse pointer moves over an element
onmouseup	script	Triggers when a mouse button is released
onmousewheel	script	Triggers when the mouse wheel is being rotated
onoffline	script	Triggers when the document goes offline
onoine	script	Triggers when the document comes online
ononline	script	Triggers when the document comes online
onpagehide	script	Triggers when the window is hidden
onpageshow	script	Triggers when the window becomes visible
onpause	script	Triggers when media data is paused
onplay	script	Triggers when media data is going to start playing
onplaying	script	Triggers when media data has start playing
onpopstate	script	Triggers when the window's history changes
onprogress	script	Triggers when the browser is fetching the media data
onratechange	script	Triggers when the media data's playing rate has changed

onreadystatechange	script	Triggers when the ready-state changes
onredo	script	Triggers when the document performs a redo
onresize	script	Triggers when the window is resized
onscroll	script	Triggers when an element's scrollbar is being scrolled
onseeked	script	Triggers when a media element's seeking attribute is no longer true, and the seeking has ended
onseeking	script	Triggers when a media element's seeking attribute is true, and the seeking has begun
onselect	script	Triggers when an element is selected
onstalled	script	Triggers when there is an error in fetching media data
onstorage	script	Triggers when a document loads
onsubmit	script	Triggers when a form is submitted
onsuspend	script	Triggers when the browser has been fetching media data, but stopped before the entire media file was fetched
ontimeupdate	script	Triggers when media changes its playing position
onundo	script	Triggers when a document performs an undo
onunload	script	Triggers when the user leaves the document
onvolumechange	script	Triggers when media changes the volume, also when volume is set to "mute"
onwaiting	script	Triggers when media has stopped playing, but is expected to resume

HTML5 Web Forms 2.0

Web Forms 2.0 is an extension to the forms features found in HTML4. Form elements and attributes in HTML5 provide a greater degree of semantic mark-up than HTML4 and remove a great deal of the need for tedious scripting and styling that was required in HTML4.

The <input> element in HTML4

HTML4 input elements use the **type** attribute to specify the data type.HTML4 provides following types:

Type	Description
text	A free-form text field, nominally free of line breaks.
password	A free-form text field for sensitive information, nominally free of line breaks.
checkbox	A set of zero or more values from a predefined list.

radio	An enumerated value.
submit	A free form of button initiates form submission.
file	An arbitrary file with a MIME type and optionally a file name.
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.
hidden	An arbitrary string that is not normally displayed to the user.
select	An enumerated value, much like the radio type.
textarea	A free-form text field, nominally with no line break restrictions.
button	A free form of button which can initiates any event related to button.

Following is the simple example of using labels, radio buttons, and submit buttons:

```
...
<form action="http://example.com/cgiscript.pl" method="post">
    <p>
    <label for="firstname">first name: </label>
              <input type="text" id="firstname"><br />
    <label for="lastname">last name: </label>
              <input type="text" id="lastname"><br />
    <label for="email">email: </label>
              <input type="text" id="email"><br>
    <input type="radio" name="sex" value="male"> Male<br>
    <input type="radio" name="sex" value="female"> Female<br>
    <input type="submit" value="send"> <input type="reset">
    </p>
 </form>
 ...
```

The <input> element in HTML5

Apart from the above mentioned attributes, HTML5 input elements introduced sevral new values for the **type** attribute. These are listed below.

NOTE: Try all the following example using latest version of **Opera** browser.

Type	Description
datetime	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.

datetime-local	A date and time (year, month, day, hour, minute, second, fractions of a second) encoded according to ISO 8601, with no time zone information.
date	A date (year, month, day) encoded according to ISO 8601.
month	A date consisting of a year and a month encoded according to ISO 8601.
week	A date consisting of a year and a week number encoded according to ISO 8601.
time	A time (hour, minute, seconds, fractional seconds) encoded according to ISO 8601.
number	This accepts only numerical value. The step attribute specifies the precision, defaulting to 1.
range	The range type is used for input fields that should contain a value from a range of numbers.
email	This 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.
url	This 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.

The placeholder attribute

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

Here is the simple syntax for placeholder attribute:

```
<input type="text" name="search" placeholder="search the web"/>
```

This attribute is supported by latest versions of Mozilla, Safari and Crome browsers only.

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.

HTML5 introduced a new attribute called **autofocus** which would be used as follows:

```
<input type="text" name="search" autofocus/>
```

This attribute is supported by latest versions of Mozilla, Safari and Crome browsers only.

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** which would be used as follows and would insist to have a value:

```
<input type="text" name="search" required/>
```

This attribute is supported by latest versions of Mozilla, Safari and Crome browsers only.

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.

SVG became a W3C Recommendation 14. January 2003 and you can check latest version of SVG specification at SVG Specification.

Viewing SVG Files:

Most of the web browsers can display SVG just like they can display PNG, GIF, and JPG. Internet Explorer users may have to install the Adobe SVG Viewer to be able to view SVG in the browser.

Embeding SVG in HTML5

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

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

Firefox 3.7 has also introduced a configuration option ("about:config") where you can enable HTML5 using the following steps:

1. Type **about:config** in your Firefox address bar.
2. Click the "I'll be careful, I promise!" button on the warning message that appears (and make sure you adhere to it!).
3. Type **html5.enable** into the filter bar at the top of the page.

4. Currently it would be disabled, so click it to toggle the value to true.

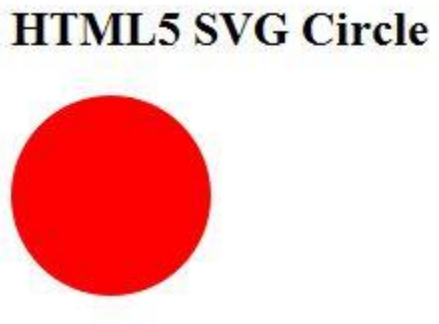
Now your Firefox HTML5 parser should now be enabled and you should be able to experiment with the following examples.

HTML5 - SVG Circle

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

```
<!DOCTYPE html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>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>
```

This would produce following result in HTML5 enabled latest version of Firefox.



HTML5 - SVG Rectangle

Following is the HTML5 version of an SVG example which would draw a rectangle using <rect> tag:

```
<!DOCTYPE html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Rectangle</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
    <rect id="redrect" width="300" height="100" fill="red" />
</svg>
</body>
</html>
```

This would produce following result in HTML5 enabled latest version of Firefox.

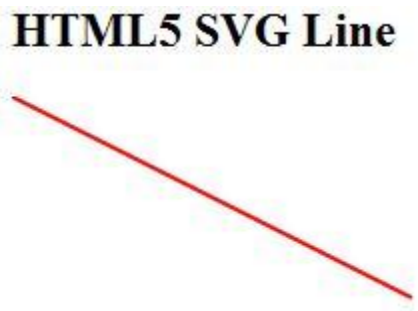
HTML5 - SVG Line

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

```
<!DOCTYPE html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Line</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
    <line x1="0" y1="0" x2="200" y2="100"
          style="stroke:red;stroke-width:2"/>
</svg>
</body>
</html>
```

You can use style attribute which allows you to set additional style information like stroke and fill colors, width of the stroke etc.

This would produce following result in HTML5 enabled latest version of Firefox.



HTML5 - SVG Ellipse

Following is the HTML5 version of an SVG example which would draw an ellipse using <ellipse> tag:

```
<!DOCTYPE html>
<head>
<title>SVG</title>
```

```
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Ellipse</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
    <ellipse cx="100" cy="50" rx="100" ry="50" fill="red" />
</svg>
</body>
</html>
```

This would produce following result in HTML5 enabled latest version of Firefox.

HTML5 - SVG Polygon

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

```
<!DOCTYPE html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Polygon</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
    <polygon  points="20,10 300,20, 170,50" fill="red" />
</svg>
</body>
</html>
```

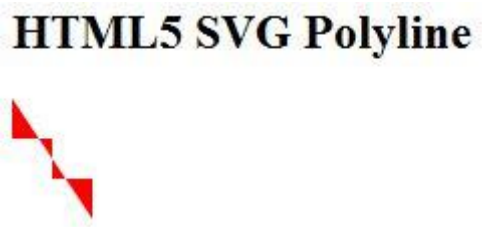
This would produce following result in HTML5 enabled latest version of Firefox.

HTML5 - SVG Polyline

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

```
<!DOCTYPE html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Polyline</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
 <polyline points="0,0 0,20 20,20 20,40 40,40 40,60" fill="red" />
</svg>
</body>
</html>
```

This would produce following result in HTML5 enabled latest version of Firefox.



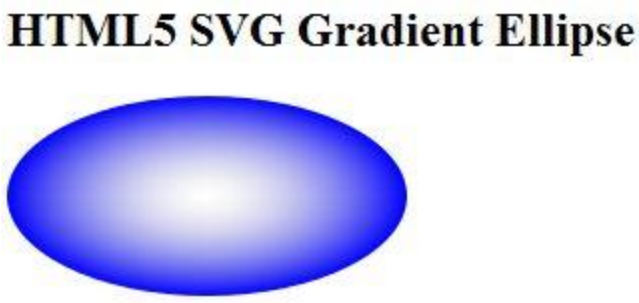
HTML5 - SVG Gradients

Following is the HTML5 version of an SVG example which would draw a ellipse using <ellipse> tag and would use <radialGradient> tag to define an SVG radial gradient.

Similar way you can use <linearGradient> tag to create SVG linear gradient.

```
<!DOCTYPE html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Gradient Ellipse</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
   <defs>
      <radialGradient id="gradient" cx="50%" cy="50%" r="50%"
      fx="50%" fy="50%">
      <stop offset="0%" style="stop-color:rgb(200,200,200);
      stop-opacity:0"/>
      <stop offset="100%" style="stop-color:rgb(0,0,255);
      stop-opacity:1"/>
      </radialGradient>
   </defs>
   <ellipse cx="100" cy="50" rx="100" ry="50"
      style="fill:url(#gradient)" />
</svg>
</body>
</html>
```

This would produce following result in HTML5 enabled latest version of Firefox.



HTML5 – WebSockets

Web Sockets is a next-generation bidirectional communication technology for web applications which operates over a single socket and is exposed via a JavaScript interface in HTML 5 compliant browsers.

Once you get a Web Socket connection with the web server, you can send data from browser to server by calling a **send()** method, and receive data from server to browser by an **onmessage** event handler.

Following is the API which creates a new WebSocket object.

```
var Socket = new WebSocket(url, [protocal] );
```

Here first argument, url, specifies the URL to which to connect. The second attribute, protocol is optional, and if present, specifies a sub-protocol that the server must support for the connection to be successful.

WebSocket Attributes:

Following are the attribute of WebSocket object. Assuming we created Socket object as mentioned above:

Attribute	Description
Socket.readyState	The readonly attribute **readyState** represents the state of the connection. It can have the following values: 1. A value of 0 indicates that the connection has not yet been established. 2. A value of 1 indicates that the connection is established and communication is possible. 3. A value of 2 indicates that the connection is going through the closing handshake. 4. A value of 3 indicates that the connection has been closed or could not be opened.
Socket.bufferedAmount	The readonly attribute **bufferedAmount** represents the number of bytes of UTF-8 text that have been queued using send() method.

WebSocket Events:

Following are the events associated with WebSocket object. Assuming we created Socket object as mentioned above:

Event	Event Handler	Description
open	Socket.onopen	This event occurs when socket connection is established.
message	Socket.onmessage	This event occurs when client receives data from server.
error	Socket.onerror	This event occurs when there is any error in communication.
close	Socket.onclose	This event occurs when connection is closed.

WebSocket Methods:

Following are the methods associated with WebSocket object. Assuming we created Socket object as mentioned above:

Method	Description
Socket.send()	The send(data) method transmits data using the connection.
Socket.close()	The close() method would be used to terminate any existing connection.

WebSocket Example:

A WebSocket is a standard bidirectional TCP socket between the client and the server. The socket starts out as a HTTP connection and then "Upgrades" to a TCP socket after a HTTP handshake. After the handshake, either side can send data.

Client Side HTML & JavaScript Code:

At the time of writing this tutorial, there are only few web browsers supporting WebSocket() interface. You can try following example with latest version of Chrome, Mozilla, Opera and Safari.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function WebSocketTest()
{
  if ("WebSocket" in window)
  {
     alert("WebSocket is supported by your Browser!");
     // Let us open a web socket
     var ws = new WebSocket("ws://localhost:9998/echo");
     ws.onopen = function()
     {
```

```
        // Web Socket is connected, send data using send()
        ws.send("Message to send");
        alert("Message is sent...");
     };
     ws.onmessage = function (evt)
     {
        var received_msg = evt.data;
        alert("Message is received...");
     };
     ws.onclose = function()
     {
        // websocket is closed.
        alert("Connection is closed...");
     };
  }
  else
  {
     // The browser doesn't support WebSocket
     alert("WebSocket NOT supported by your Browser!");
  }
}
</script>
</head>
<body>
<div id="sse">
   <a href="javascript:WebSocketTest()">Run WebSocket</a>
</div>
</body>
</html>
```

Install pywebsocket:

Before you test above client program, you need a server which supports WebSocket. Download mod_pywebsocket-x.x.x.tar.gz from pywebsocket which aims to provide a Web Socket extension for Apache HTTP Server ans install it following these steps.

1. Unzip and untar the downloaded file.
2. Go inside **pywebsocket-x.x.x/src/** directory.
3. $python setup.py build
4. $sudo python setup.py install
5. Then read document by:
    - $pydoc mod_pywebsocket

This will install it into your python environment.

Start the Server

Go to the **pywebsocket-x.x.x/src/mod_pywebsocket** folder and run the following command:

```
$sudo python standalone.py -p 9998 -w ../example/
```

This will start the server listening at port 9998 and use the handlers directory specified by the -w option where our echo_wsh.py resides.

Now using Chrome browser open the html file your created in the beginning. If your browser supports WebSocket(), then you would get alert indicating that your browser supports

WebSocket and finally when you click on "Run WebSocket" you would get Goodbye message sent by the server script.

HTML5 – Canvas

HTML5 element <canvas> gives you an easy and powerful way to draw graphics using JavaScript. It can be used to draw graphs, make photo compositions or do simple (and not so simple) animations.

Here is a simple <canvas> element which has only two specific attributes **width** and **height** plus all the core HTML5 attributes like id, name and class etc.

```
<canvas id="mycanvas" width="100" height="100"></canvas>
```

You can easily find that <canvas> element in the DOM using *getElementById()* method as follows:

```
var canvas  = document.getElementById("mycanvas");
```

Let us see a simple example on using <canvas> element in HTML5 document.

```
<!DOCTYPE HTML>
<html>
<head>
<style>
#mycanvas{
   border:1px solid red;
}
</style>
</head>
<body>
   <canvas id="mycanvas" width="100" height="100"></canvas>
</body>
</html>
```

The Rendering Context:

The <canvas> is initially blank, and to display something, a script first needs to access the rendering context and draw on it.

The canvas element has a DOM method called **getContext**, used to obtain the rendering context and its drawing functions. This function takes one parameter, the type of context **2d**.

Following is the code to get required context along with a check if your browser supports <canvas> element:

```
var canvas  = document.getElementById("mycanvas");
if (canvas.getContext){
   var ctx = canvas.getContext('2d');
   // drawing code here
} else {
   // canvas-unsupported code here
}
```

Browser Support

The latest versions of Firefox, Safari, Chrome and Opera all support for HTML5 Canvas but IE8 does not support canvas natively.

You can use ExplorerCanvas to have canvas support through Internet Explorer. You just need to include this javascript as follows:

```
<!--[if IE]><script src="excanvas.js"></script><![endif]-->
```

HTML5 Canvas Examples:

This tutorial covers following examples related to HTML5 <canvas> element.

Examples	Description
Drawing Rectangles	Learn how to draw rectangle using HTML5 <canvas> element
Drawing Paths	Learn how to make shapes using paths in HTML5 <canvas> element
Drawing Lines	Learn how to draw lines using HTML5 <canvas> element
Drawing Bezier	Learn how to draw bezier curve using HTML5 <canvas> element
Drawing Quadratic	Learn how to draw quadratic curve using HTML5 <canvas> element
Using Images	Learn how to use images with HTML5 <canvas> element
Create Gradients	Learn how to create gradients using HTML5 <canvas> element
Styles and Colors	Learn how to apply styles and colors using HTML5 <canvas> element
Text and Fonts	Learn how to draw amazing text using different fonts and their size.
Pattern and Shadow	Learn how to draw different patterns and drop shadows.
Canvas States	Learn how to save and restore canvas states while doing complex drawings on a canvas.
Canvas Translation	This method is used to move the canvas and its origin to a different point in the grid.
Canvas Rotation	This method is used to rotate the canvas around the current origin.
Canvas Scaling	This method is used to increase or decrease the units in a canvas grid.
Canvas Transform	These methods allow modifications directly to the transformation matrix.

Canvas Composition	This method is used to mask off certain areas or clear sections from the canvas.
Canvas Animation	Learn how to create basic animation using HTML5 canvas and Javascript.

For complete Tutorial: http://www.tutorialspoint.com/html5