

Sign up for our free weekly [Web Developer Newsletter](#).



Your Skills. Our Tools.
Let's build great apps, together.
Try our ASP.NET Controls risk-free for 30 days.



home articles quick answers discussions features community help Search for articles, questions, tips

Articles » Web Development » HTML / CSS » HTML

Next →

Article

Browse Code

Stats

Revisions

Alternatives

Comments &
Discussions (99)

HTML5 Quick Start Web Application

By **Sandeep Mewara**, 15 Feb 2013



4.96 (72 votes)

Prize winner in Competition "Best Web Dev article of February 2013"

[Download HTML5QuickStart.zip - 7 MB](#)

Background

I started learning more about HTML5 from December last year. Since it was new, talked about and had real good features, I got a jump start from the thought of using it in our upcoming projects and possibly giving a session on HTML5 to my team once I have some good insight on the same. While learning about the HTML5, I started developing a demo web application that was fully HTML5 based with all the new major features that we will be discussing in this article going ahead.

Once the thought of sharing the features in a quick and easy manner hit me, I tried to develop a HTML5 enabled ASP.NET application that would be self sufficient in explaining and showcasing the new HTML5 features - kind of self starter kit to see and play around with the feature implementation straight away post download.

Table of Contents



- [Introduction](#)
- [Tools](#)
- [Structure](#)
- [Forms](#)
- [Drag & Drop](#)
- [Audio & Video](#)
- [GeoLocation](#)
- [Web Storage](#)
- [Web Workers](#)
- [Web Socket](#)
- [Canvas](#)
- [Offline Application](#)
- [Device Support](#)

Introduction

About Article

HTML5 self learning center - Explore and understand the new features!

Type	Article
Licence	CPOL
First Posted	15 Feb 2013
Views	44,715
Downloads	3,042
Bookmarked	182 times

TabletPC C# ASP.NET
.NET HTML Mobile Dev
+



Ruby for programmers.



HTML5 is the fifth revision and the newest version of the HTML standards. Previous version (HTML 4.01) was released in 1999. In 2006, WHATWG (Web Hypertext Application Technology Working Group) & W3C (World Wide Web Consortium) came together to define the new version of HTML. Earlier, WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0.

They discussed and laid out [HTML Design Principles](#). They were addressed towards better experience for user and easy adoption for developer which would be independent of device and plugins with a graceful fallback mechanism. HTML5 specifications were finalized on 17th December 2012 last year.

Tools

HTML5 is still a work in progress. Browsers are implementing HTML5 specifications as per their individual plans, thus we have various level of support on different browsers.

In order to know what features are supported by a particular browser, few websites share the information in detail about HTML5 compatibility with different versions of a browser:

- [HTML5 Test](#) - provides a HTML5 score with detail on what can be used
- [Can I Use](#) - provides a table comparison of major browsers for different features

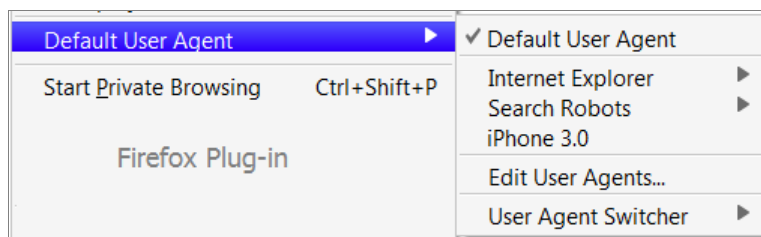
Current support scores for various browsers are:

MOBILE BROWSERS			Score	Bonus
Opera Mobile 12.10 »			406	12
Chrome »			390	11
Firefox Mobile 16 »			388	10
iOS 6.0 »			386	9
Windows Phone 8 »			320	6
Android 4.0 »			297	3
BlackBerry OS 7 »			288	3
Bada 2.0 »			283	9
Nokia Belle FP 2 »			272	9
Android 2.3 »			200	1

TABLET BROWSERS			Score	Bonus
RIM Tablet OS 2.1 »			411	9
Opera Mobile 12.10 »			406	12
Chrome »			390	11
Firefox Mobile 16 »			388	10
iOS 6.0 »			386	9
Silk 2.2 »			358	13
Internet Explorer 10 »			320	6
Android 4.0 »			297	3
webOS 3.0 »			224	7

DESKTOP BROWSERS			Score	Bonus
Maxthon 3.4.5 »			457	15
Chrome 23 »			448	13
Opera 12.10 »			419	9
Firefox 17 »			392	10
Safari 6.0 »			378	8
Internet Explorer 10 »			320	6

In order to develop for various browsers, we can use **User Agent Switcher**:



In order to have a fallback plan, while developing, we need to first check if a particular HTML5 feature is supported on the browser being used. Traditional approach of creating the dynamic element in JavaScript and checking for any of its properties existence is good enough. Another option could be to use an open source [JavaScript library Modernizr](#) (MIT-licensed). It can be used in the following manner:

```
<script src="modernizr.min.js"></script>
<script type="text/javascript">
if (Modernizr.canvas) {
```

[Collapse](#) | [Copy Code](#)

Top News

[R.I.P. Windows](#)

Get the [Insider News](#) free each morning.

Related Videos



Related Articles

[Semantic HTML5 Page Layout](#)

[Building Apps with HTML5: What You Need to Know](#)

[HTML5/CSS3 graphic enhancement: buttons, inputs](#)

[Accessing WPF apps over a Web browser via WebSockets and HTML5 Canvas](#)

[Learn HTML5 in 5 Minutes!](#)

[Add Image Borders using HTML5/CSS3](#)

[HTML5 Canvas - Aqua Gauge](#)

[Using the "Copy XAML" Feature in Expression Design to Create HTML5 SVG Path Data](#)

[HTML5 Zero Footprint Viewer for DICOM and PACS - Part 2](#)

[HTML5 New Attributes](#)

[Why the Web Is Ready for Responsive Web Design](#)

[Building C++ Applications with HTML5](#)

[HTML5 Web Storage \(Example : TODO\)](#)

[The Complete Guide to Building HTML5 games with Canvas & SVG](#)

[Fun with HTML5 Canvas, WebSocket, JQuery and ASP.NET. End-result: A live white board on a web page!](#)

[Building a basic HTML5 client/server application](#)

[Building Rich Mobile Apps with HTML5, CSS3 and JavaScript](#)

[HTML5 in Action: Changing the Face of Web Forms with New Input Types](#)

[Top 5 Best Practices for Building HTML5 Games...In Action!](#)

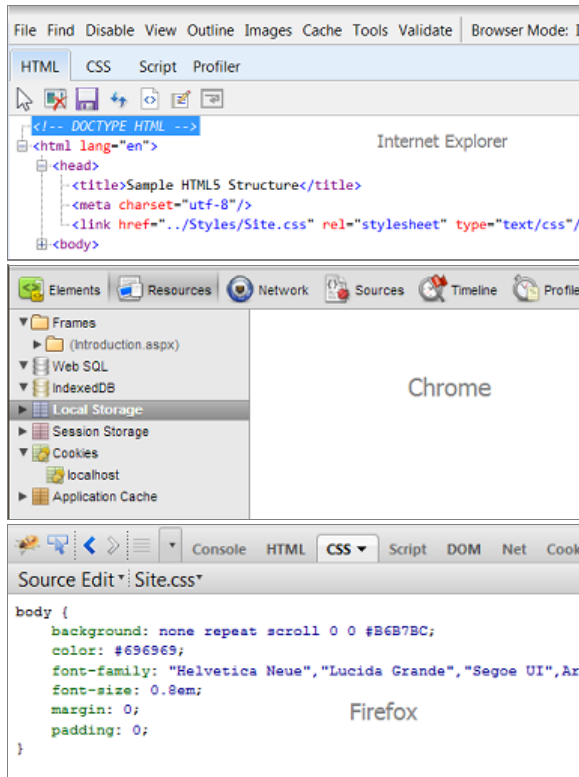
[HTML5 Zero Footprint Viewer for DICOM and PACS](#)

```

    // Congrats! Canvas is supported on your browser.
} else {
    // Sorry! No native canvas support available on your browser.
}
</script>

```

In order to have a look at internal details of the web page, explore and understand what's happening on client side in browser, **Developer Tools** of the browsers are extremely helpful:



Structure

HTML5 is designed such that it is not based on SGML (Standard Generalized Markup Language), thus they does not require a reference to a DTD (Document Type Definition). So, for passing the HTML version instruction to web browser, following works:

[Collapse](#) | [Copy Code](#)

```
<!DOCTYPE HTML>
```

A new short way to declare character encoding in the webpage:

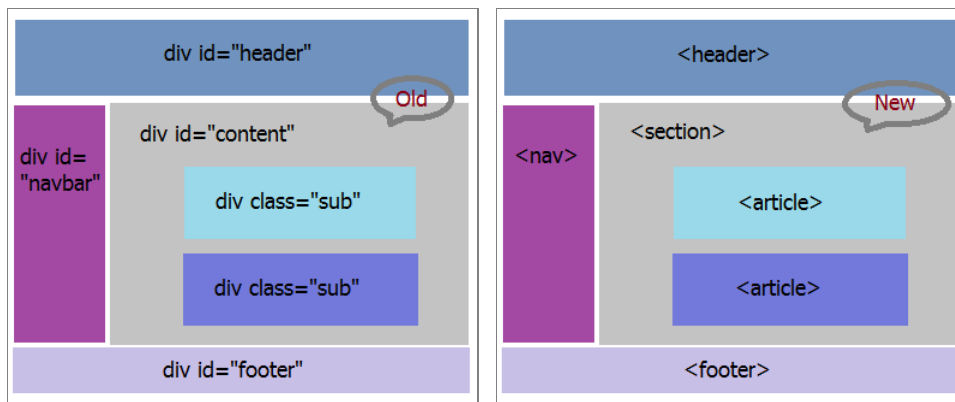
[Collapse](#) | [Copy Code](#)

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

Moving forward, HTML5 has new web page elements that adds semantic to the page. Like:

- Header
- Nav
- Footer
- Section
- Article

Following is the sample web page structure comparison with HTML4 design:



Here is the code snippet of an example using new structural semantics:

[Collapse](#) | [Copy Code](#)

```
<hgroup>
  <h1>Penning Everything Interesting...</h1>
</hgroup>
<article id="MyBlog">
  <header>
    <h2>My first blog entry!</h2>
  </header>
  <p>I was thinking what to write about. I read few articles at times that I feel like
  sharing across, probably ...</p>
  <br/>
  <footer>
    <section>
      <h4>One Thought on 'My first blog entry...'</h4>
      <article id="comment_1">
        <link href="#comment_1" />
        <h5>Comment by: Guruji</h5>
        <p>Very well written. Love the enthu and energy you have for sharing your
        knowledge...</p>
      </article>
      <article id="comment_2">
        <link href="#comment_2" />
        <h5>Comment by: SuperStar</h5>
        <p>Looks like a good start...Good going!</p>
      </article>
    </section>
  </footer>
</article>
```

Along with addition, few elements & attributes are deprecated and are no more thereas per specification:

Tags (Elements)	Description
<acronym>	Defines an acronym
<applet>	Defines an applet
<basefont>	Defines an base font for the page.
<big>	Defines big text
<center>	Defines centered text
<dir>	Defines a directory list
	Defines text font, size, and color
<frame>	Defines a frame
<frameset>	Defines a set of frames
<isindex>	Defines a single-line input field
<noframes>	Defines a noframe section
<s>	Defines strikethrough text
<strike>	Defines strikethrough text
<tt>	Defines teletype text
<u>	Defines underlined text

These above tags are not to be used by developer. Still, it has been decided that User agents will have to support them.

Forms

Various new form based tags and attributes has been defined in HTML5, like:

- input types and attributes

- email
- url
- number
- date time
- search
- range
- color
- control elements
 - datalist
 - keygen
 - output
- attributes
 - spellcheck
 - autofocus
 - list
 - required
 - placeholder
 - min and max
 - multiple
- events
 - orientationchange
 - resize
 - online
 - offline

email :

! Please enter a comma separated list of email addresses.

url :

! Please enter a URL.

datalist :

- Sandeep
- Sandy

E-mail:

Previous Next AutoFill Done

Q W E R T Y U I O P

A S D F G H J K L

Z X C V B N M

123 space @ . Go

Few samples using them:

[Collapse](#) | [Copy Code](#)

```
email: <input type="email" multiple required autofocus placeholder="sample@sample.com" />
url: <input type="url" />
number: <input type="number" />
date time: <input type="date"/>
           <input type="time"/>
search: <input type="search"/>
range: <input type="range" max="100" min="80" step="2" title="Your AES score"
name="rscore"/>
       <output name="score">90</output>
color: <input type="color"/>
datalist: <input type="text" id="somenames" list="fullList" />
         <datalist id="fullList">
           <option value="Sandeep">
           <option value="Deepak">
           <option value="Sandy">
           <option value="Deepu">
         </datalist>
spellcheck:<input type="text" spellcheck="false" placeholder="Without Spellcheck">
           <input type="text" spellcheck="true" placeholder="With Spellcheck">
```

Further, HTML5 has improved way of finding controls with new selectors. Just like jQuery selectors, HTML5 has two new selectors that help in finding the needed control quickly:

- `document.querySelector()` - Return the first element in the page which matches the specified selector rules

[Collapse](#) | [Copy Code](#)

```
// Returns the first element in the document with the class "myclass"
var e1 = document.querySelector(".myclass");
```

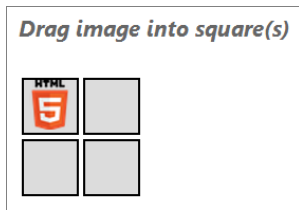
- `document.querySelectorAll()` - Returns all elements which match the specified rules

[Collapse](#) | [Copy Code](#)

```
// Returns the list of all the element in the document with the class "myclass"
var els = document.querySelectorAll(".myclass");
```

Drag & Drop

HTML5 has made drag-drop feature as part of standards and any element can be made draggable and then dragged to defined location. Attributes, methods and events related to it are:



- **draggable**
- **ondragstart**
- **setData**
- **getData**
- **ondragover**
- **ondrop**

Sample code for above logo drag-drop will look like:

[Collapse](#) | [Copy Code](#)

```
function allowDropDragOver(e) {
    e.preventDefault();
}

function dragStart(e) {
    e.dataTransfer.setData("dragItemID", e.target.id);
}

function dropIt(e) {
    e.preventDefault();
    var dataId = e.dataTransfer.getData("dragItemID");
    e.target.appendChild( document.getElementById(dataId));
}

<table>
  <tr>
    <td>
      <div id="bin1" ondrop="dropIt(event)" ondragover="allowDropDragOver(event)"
class="dragDropBin">
        
      </div>
    </td>
    <td><div id="bin2" ondrop="dropIt(event)" ondragover="allowDropDragOver(event)"
class="dragDropBin"></div></td>
  </tr>
  <tr>
    <td><div id="bin3" ondrop="dropIt(event)" ondragover="allowDropDragOver(event)"
class="dragDropBin"></div></td>
    <td><div id="bin4" ondrop="dropIt(event)" ondragover="allowDropDragOver(event)"
class="dragDropBin"></div></td>
  </tr>
</table>
```

Audio & Video

This is one of the major features of HTML5 that has brought a big difference in having audio or video feature on a webpage. They will kill usage of plugins (like flash or silverlight) just based on performance. Guess what, it is super easy to implement too with just a few lines of code. Attributes, methods and events related to it are:

- **Attributes & Methods**
 - **play()**: To play the media
 - **pause()**: To pause the media
 - **canPlayType()**: To check if browser can play a certain media resource type
 - **src**: Path of the media file. One can also use the `<source>` element
 - **autoplay**: To play the media automatically
 - **controls**: To display the controls (play/pause/volume/etc) on the media
 - **loop**: To play in loop
 - **muted**: To mute the volume
 - **playbackRate**: To vary the speed of the media
- **Events**
 - **play**: raised when the media is played
 - **pause**: raised when the media is paused
 - **timeupdate**: raised when the current playback position has changed
 - **volumechange**: raised when the volume of the media has changed
 - **ratechange**: raised when the playback rate of the



- media has changed
- **ended**: raised when the media has finished playing
- **seeking**: raised when the end user is seeking in the media
- **seeked**: raised when the seeking attribute has changed to false

Currently, there are defined formats supported by different browsers. HTML5 based supported formats are:

- Audio: MP3, Wav, and Ogg
- Video: MP4, WebM, and Ogg

Sample code for audio & video implementation looks like:

[Collapse](#) | [Copy Code](#)

```
<audio id="sampleAudio" controls>
  <source src="../Files/AudioSample.mp3">
  <source src="../Files/AudioSample.wav">
  <source src="../Files/AudioSample.ogg">
  Your browser does not support the audio element.
</audio>

<video id="sampleVideo" controls src="../Files/VideoSample.mp4">
  <track src="../Files/sampleSubtitles.vtt" srclang="en" kind="subtitles" label="English subtitles">
  Your browser does not support the audio element.
</video>

<!-- For runtime button based operations -->
function CMute(controlId) {
  $(controlId).attr('muted', true);
}

function CUnMute(controlId) {
  $(controlId).attr('muted', false);
}

function CPause(controlId) {
  if (!$(controlId).get(0).paused)
    $(controlId).get(0).pause();
}

function CPlay(controlId) {
  $(controlId).get(0).play();
}

function CFullScreen(controlId) {
  // Only Chrome code currently.
  $(controlId)[0].webkitRequestFullScreen();
}

<input type="button" value="Mute" onclick="CMute(sampleAudio);"/>
<input type="button" value="Unmute" onclick="CUnMute(sampleAudio);"/>
<input type="button" value="Pause" onclick="CPause(sampleAudio);"/>
<input type="button" value="Play" onclick="CPlay(sampleAudio);"/>
```

Geo Location

HTML5 Geolocation is used to locate a user's geographical position. Since this is related to specific individual usage, it can compromise user privacy. Thus, user permission is necessary in order to locate the position. In order to use Geolocation, we would need to know about **navigator.geolocation** object, it's methods and parameters:

- Methods
 - **getCurrentPosition**: get's location once
 - **watchPosition**: regularly keeps polling
 - **clearWatch**: stop's watching position
- Parameters
 - **position**
 - coords (Coordinates): contains information on user location
 - latitude
 - longitude
 - altitude

CURRENT POSITION:

Latitude: 12.9221826

Longitude: 77.6820787

- accuracy
- altitudeAccuracy
- speed
- heading
- timestamp (DOMTimeStamp) : time when user location was obtained
- **PositionError** : error callback object argument
 - UNKNOWN_ERROR = 0
 - PERMISSION_DENIED = 1
 - POSITION_UNAVAILABLE = 2
 - TIMEOUT = 3
 - code
 - message
- **PositionOptions** : specifies options when getting user location
 - enableHighAccuracy
 - timeout
 - maximumAge



Sample code for GeoLocation - Latitude & Longitude, looks like:

[Collapse](#) | [Copy Code](#)

```
// navigator.geolocation.getCurrentPosition(success_callback, error_callback, {other
parameters});
// The watchPosition() function has the same structure as getCurrentPosition()
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(updateLocation, handleLocationError, {
            timeout: 1000000,
            maximumAge: 1000,
            enableHighAccuracy: false
        });
    }
    else {
        document.getElementById("supportstatus").innerHTML = "HTML5 Geolocation is not
supported in your browser.";
    }
}

function updateLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var timeTaken = new Date(position.timestamp);
    ...
}

function handleLocationError(error) {
    switch(error.code)
    {
        ...
    }
}
```

Sample code for GeoLocation - Map display, looks like:

[Collapse](#) | [Copy Code](#)

```
<div id="mapView" class="mapDimension1"></div>

// include the Maps API JavaScript using a script tag.
<script type="text/javascript" src="http://maps.google.com/maps/api/js?
sensor=false"></script>

// navigator.geolocation.getCurrentPosition
function showMap() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showMapLocation, handleLocationError, {
            timeout: 1000000,
            maximumAge: 1000,
            enableHighAccuracy: true
        });
    }
}

// create an object literal to hold map properties and create a "map" object & "marker"
objects
function showMapLocation(position) {
    var latlng = new google.maps.LatLng(position.coords.latitude,
position.coords.longitude);
    var myOptions = {
        zoom: 16,
```



```

        center: latlng,
        mapTypeControl: false,
        navigationControlOptions: { style: google.maps.NavigationControlStyle.SMALL },
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var map = new google.maps.Map(document.getElementById("mapView"), myOptions);

    var marker = new google.maps.Marker({
        position: latlng,
        map: map,
        title: "You are here! (Accuracy: " + position.coords.accuracy + " meter
radius)"
    });
}

```

Web Storage

HTML5 provides better local storage within the browser. Currently, we use cookies for storing information on client browser but with few limitations like they are slow, unencrypted, limited to about 4KB & need to be included with every HTTP request. Web Storage is more secure and faster. It has good storage space (~5MB) on the client without affecting the website's performance. It persists and is transmitted to the server only when asked. Currently, it supports two types of storage:

- **localStorage**: stores data with no expiration date
 - Key/Value pair storage system
 - Shared between window/tab for a same domain
 - Data available until explicitly deleted by the user
- **sessionStorage**: stores data for one session
 - Key/Value pair storage system
 - Specific to particular window/tab

Web Storage follows the storage interface that exposes API to create, read, update or delete data on the client side:

- **getItem(DOMString key)** : Get the value of an item identified by its key
- **setItem(DOMString key, any data)** : used to create or modify an item
- **removeItem(DOMString key)** : Delete an item identified by its key
- **clear()** : Delete all items
- **length** : Number of items.
- **key(unsigned long index)** : Get the key of an item with a zero based index

USAGE	
Local Session	
Set:	<input type="button" value="Go"/>
Get:	<input type="button" value="Go"/>
Clear:	<input type="button" value="Go"/>

Sample code for implementing the web storage looks like.

[Collapse](#) | [Copy Code](#)

```

function SaveToSessionStorage() {
    window.sessionStorage.mySessionKeyX = 'Saved Directly in Session';
    window.sessionStorage.setItem("mySessionKeyY", "Saved Using Set Item");
}

function SaveToLocalStorage() {
    window.localStorage.myLocalKeyA = 'Saved Directly in Local';
    window.localStorage.setItem("myLocalKeyB", "Saved Using Set Item");
}

function GetFromSessionStorage() {
    alert("Value X: " + window.sessionStorage.getItem("mySessionKeyX"));
}

function GetFromLocalStorage() {
    alert("Value B: " + window.localStorage.getItem("myLocalKeyB"));
}

function RemoveFromSessionStorage() {
    window.sessionStorage.removeItem("mySessionKeyX");
}

function RemoveFromLocalStorage() {
    window.localStorage.removeItem("myLocalKeyB");
}

function clearAll() {
    localStorage.clear();
    sessionStorage.clear();
}

```

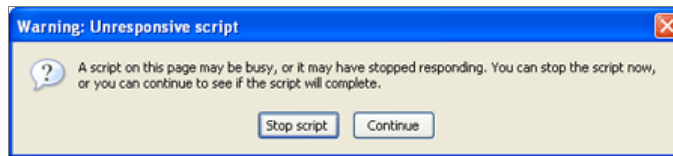
}

Web Workers

HTML5 Web Workers is a JavaScript script that runs in the background after being executed from a webpage. It runs independently from other userscripts without affecting the performance of the page.

When executing scripts in an HTML page, the page becomes unresponsive until the script is finished. Web Workers allows to perform tasks in a background process in parallel of the main browser process. Thus the page is responsive even when a worker is running. Since, web workers are in external files, they cannot access the following JavaScript objects:

- **window** object
- **document** object
- **parent** object



Sample code to implement workers would look like:

[Collapse](#) | [Copy Code](#)

```
// create a new Worker object
var worker = new Worker('WebWorkerSample.js');

// start it by calling the postMessage() method
worker.postMessage();

// Communication between a worker and its parent page is done using an event model via postMessage()
// Main script:
worker.addEventListener('message', function(e) {
    console.log('Worker said: ', e.data);
}, false);

//Internal Script (the worker):
self.addEventListener('message', function(e) {
    self.postMessage(e.data);
}, false);
```

Web Socket

HTML5 Web Socket provides a two-way communication over one TCP connection (socket). The WebSocket API standards are defined by the W3C (World Wide Web Consortium), and the WebSocket protocol has been standardized by the IETF (Internet Engineering Task Force). For it, new protocols are introduced - **WS**: & **WSS**:

It has simplified complexities around client-server communication. In order to establish a connection between client and server, for the first time when the request is sent to server, it is sent as an upgrade request from a HTTP call. Once it is upgraded, and handshake is successful, client and server can talk to each other using frames (2 bytes). Server notifies the client of new messages. General use-case of it is:

- create web socket
- define event listeners
- send message

Attributes, methods and events related to it are:

```
// browser request to the server
GET /WSDemo HTTP/1.1
Upgrade: WebSocket
Connection: Upgrade
Host: mySample.com
Origin: http://mySample.com
WebSocket-Protocol: sample

// server response
HTTP/1.1 101 Web Socket Protocol Handshake
Upgrade: WebSocket
Connection: Upgrade
WebSocket-Origin: http://mySample.com
WebSocket-Location: ws://mySample.com/WSDemo
WebSocket-Protocol: sample
```

- **readyState**
- **bufferedAmount**
- **onopen**
- **onmessage**
- **onclose**
- **onerror**
- **send()**
- **close()**

Sample code related to this feature looks like:

[Collapse](#) | [Copy Code](#)

```
function WebSocketTest() {
    if ("WebSocket" in window) {
        // Open a Web Socket
        var ws = new WebSocket("ws://myServer/HTML5Demo/");
        ws.onopen = function () {
            // Send data
            ws.send("Message sent to server");
            alert("Message was sent to server.");
        };
        ws.onmessage = function (evt) {
            // Recieve data
            var receivedmsg = evt.data;
            alert("Message is received from server"+ receivedmsg);
        };
        ws.onclose = function () {
            // Close a WebSocket
            alert("Connection is closed.");
        };
        ws.onerror = function (evt) {
            // Log Error
            alert("ERROR!!! " + evt.message);
        };
    }
}
```

It is efficient, has low latency (almost instant) and size. Scalable and real time applications can be easily made based on it.

Canvas

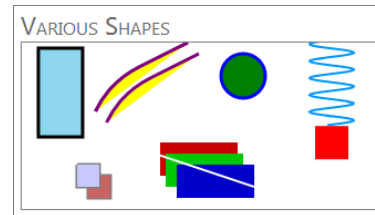
Canvas is another feature of HTML5 that is used to draw graphics on a webpage at runtime. Canvas only the container of the graphics, the whole drawing is done via JavaScript. Thus, it is mandatory to define ID, height & width of the canvas. It has methods for drawing 2-D shapes, texts, images using Paths.

General use-case of it is:

- define **canvas** element
- get 2-D **context**
- draw

Canvas provides methods which allow modifications to the transformation matrix:

- **save()** & **restore()** - to save and restore different transformation states
- **translate()** - to translate the HTML5 Canvas context
- **rotate()** - to rotate the HTML5 Canvas
- **scale()** - to scale the HTML5 Canvas



SVG (Scalable Vector Graphics) is another alternative. It is vector-based, relies on files whereas Canvas manipulates pixels, uses pure scripting. SVG has slower rendering due to SVG's integration into the DOM. Thus, probably not a good option for dynamic applications like games. Depending on the type of work, one need to select which to use.

Sample code related to same looks like:

[Collapse](#) | [Copy Code](#)

```
// Define Canvas tag
<canvas id="myCanvas" width="200" height="100"></canvas>

// Access in JavaScript to draw in it at runtime
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
context.fillStyle='Red';
context.fillRect(0,0,150,75);
context.beginPath();
context.moveTo(150, 10);
context.arcTo(80, 45, 3, 200, 90);
context.fillStyle = 'Yellow';
context.fill();
context.strokeStyle = 'Purple';
context.stroke();
context.closePath();
```

Offline Application

HTML5 Offline is another major feature that helps in browsing a web application even when there is no network connection. Web application has a reference to a manifest file that holds the information about the files that needs to be cached and kept, not to be cached and fallback rules for non cached files. Local copies are cached and kept without any expiration rule. If application finds any change in manifest when it goes online, local files are updated to make it up-to-date. In absence of network connection (an **onLine** flag present in DOM tells it), webpages and related files are served from the local cached copies.

Sample Cache Manifest file looks like:

[Collapse](#) | [Copy Code](#)

```
CACHE MANIFEST
# Cache manifest - version & date
# version 2.1.24
# 02.01.2013

# Cache - resources to cache with html page
CACHE:
Cache-Control: no-cache, private
Styles/Site.css
Images/HTML5.png
Scripts/jquery-1.4.1.min.js
Scripts/jquery-1.4.1-vsdoc.js
Scripts/Utilities.js
Scripts/OfflineApp.js
offline.manifest
# Images/HTML5LogoSmall.png

# Network - files that will not be cached and will always be online
# Following resources require network connection.
# * - it's a special flag (meaning all resources that are not cached will require a
connection.)
NETWORK:
*

# Fallback - display for all uncached pages/resources [precedence below Network]
# FALLBACK:
Images/ShowFullscreen.png
#site/images/ images/offline.jpg
#*.html offline.html
```

ApplicationCache has various states that tells the status of the cache based on the manifest file:

- 0 UNCACHED : cache is empty
- 1 IDLE : cache is latest
- 2 CHECKING : checking for a new updated version of the cache
- 3 DOWNLOADING : downloading the new cache
- 4 UPDATEREADY : ready to be used
- 5 OBSOLETE : cache is marked obsolete

Sample code for tracking Offline capability is like:

[Collapse](#) | [Copy Code](#)

```
// offline.appcache manifest file reference in the webpage
// <html lang="en" manifest="<%=ResolveClientUrl("offline.appcache")%>">

var cacheStatusMessages = [];
cacheStatusMessages[0] = 'uncached';
cacheStatusMessages[1] = 'idle';
cacheStatusMessages[2] = 'checking';
cacheStatusMessages[3] = 'downloading';
cacheStatusMessages[4] = 'updateready';
cacheStatusMessages[5] = 'obsolete';

var appCache = window.applicationCache;
appCache.addEventListener('cached', logEvent, false);
appCache.addEventListener('checking', logEvent, false);
appCache.addEventListener('noupdate', logEvent, false);

appCache.addEventListener('downloading', logEvent, false);
appCache.addEventListener('progress', logEvent, false);
appCache.addEventListener('updateready', function() {
    window.applicationCache.swapCache();
    console.log('Update was ready. Swap Cache done.');
```

```

appCache.addEventListener('obsolete', logEvent, false);
appCache.addEventListener('error', logEvent, false);

function logEvent(e) {
    var online, status, type, message;
    online = (navigator.onLine) ? 'yes' : 'no';
    status = cacheStatusMessages[appCache.status];
    type = e.type;

    message = 'online: ' + online;
    message += ', event: ' + type;
    message += ', status: ' + status;

    console.log(message);

    if (type == 'obsolete')
        alert('Manifest got deleted. Offline will not work anymore.');
```

Device Support

HTML5 have defined ways to support multiple devices. It's a key to mobile page optimization.

- Viewport Meta Tag - it controls browsers page display, by telling the browser how content should fit on the device's screen. Further, it informs the browser that the site is optimized for mobile devices. Sample use:

[Collapse](#) | [Copy Code](#)

```
<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0, user-scalable=yes"/>
```

- CSS Media Queries - these are media based stylesheets, that enables the content's presentation to be customized to a specific range of output devices without having to change the content itself.

- External stylesheet :

[Collapse](#) | [Copy Code](#)

```
<link rel="stylesheet" media="handheld, only screen and (max-device-width: 320px)" href="phone.css">
<link rel="stylesheet" media="only screen and (min-width: 641px) and (max-width: 800px)" href="ipad.css">
```

- Within the `<style>` element (or in css file) as a media rule:

[Collapse](#) | [Copy Code](#)

```
@media only screen and (max-width: 480px){
    /* rules defined inside here are only applied to browsers that support CSS
    media queries */
    /* and the browser window is 480px or smaller */
}
```

- Imported stylesheet within the `<style>` element:

[Collapse](#) | [Copy Code](#)

```
@import "smallscreen.css" only screen and (max-width: 480px);
```

Points of Interest

I learned a lot of things while developing the starter kit. Reading is okay but implementing is where I found having more fun and trouble. I would suggest anyone to try and implement features by themselves once they know about it to learn more. I have a list of what all I discovered while development and plan to put them on my blog.

References

Thanks to all the knowledge-base articles shared at the following links that helped me learn and develop this quick start application:

<http://www.w3.org/>
<http://www.w3schools.com/>
<http://caniuse.com/>
<http://html5test.com/>
<http://www.modernizr.com/>
<http://html5doctor.com/>
<http://msdn.microsoft.com/en-us/hh969243.aspx>
http://marakana.com/static/bookshelf/html5_tutorial/index.html

[http://www.whatwg.org/specs/web-apps/current-work/multipage/offline.html\[^](http://www.whatwg.org/specs/web-apps/current-work/multipage/offline.html[^)
[http://www.javascriptkit.com/dhtmltutors/cssmediaqueries.shtml\[^](http://www.javascriptkit.com/dhtmltutors/cssmediaqueries.shtml[^)

History

Version 1: 16-Feb-2013 First Draft!

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

About the Author



Sandeep Mewara

Software Developer (Senior)
India



I did my B.Tech from IIT Kharagpur and joined IT industry in June 2005. Since then, I have been working on various projects using Microsoft Technology. Started capturing my learning a little late but am happy enough to share few of them with the world in form of CodeProject articles.

Overall, I have 8+ years of development experience and have played roles like Team Lead, Module owner, Technical Analyst and Client Interface

Our team won in category 'Solving a critical pain point' in CTOF 2012 @ Intuit for implementing an idea MyTaxDocuments – windows, web & mobile solution.

I have worked both on Web and Desktop applications. I am honored to be CP MVP from last 3 years(MVP 2011, MVP 2012, MVP 2013) and Microsoft MVP from last 2 years(MVP 2012, 2013).

I like watching movies during my free time.

Started my tech based blog this year: <http://smewara.wordpress.com/>

Current location: Bangalore, India.

Follow on Twitter

[Article Top](#)

Like

47

12

Tweet

50

[Sign Up to vote](#) Poor ☐ ☐ ☐ ☐ ☐ Excellent





















Comments and Discussions

Hint: For improved responsiveness ensure Javascript is enabled and choose 'Normal' from the Layout dropdown and hit 'Update'.
 You must [Sign In](#) to use this message board.

Search this forum

☒ Profile popups Spacing Relaxed ▼ Noise Very High ▼ Layout Normal ▼ Per page 10 ▼

First Prev Next

 Permission for publishing your article	 Marla Suresh	29-Aug-13 17:26
 Re: Permission for publishing your article	 Sandeep Mewara	29-Aug-13 20:57
 My vote of 5	 damodara naidu betha	1-Aug-13 19:17
 Re: My vote of 5	 Sandeep Mewara	1-Aug-13 19:40
 My Vote of 5	 Anoop Kr Sharma	7-Jul-13 17:27
 Re: My Vote of 5	 Sandeep Mewara	8-Jul-13 6:22
 My vote of 5	 csharpbd	3-Jul-13 17:39
 Re: My vote of 5	 Sandeep Mewara	4-Jul-13 1:03
 My vote of 5	 Tejas Patel	15-Jun-13 6:33
 Re: My vote of 5	 Sandeep Mewara	16-Jun-13 4:30

Last Visit: 31-Dec-99 18:00 Last Update: 8-Sep-13 17:12 [Refresh](#) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) Next »

 General  News  Suggestion  Question  Bug  Answer  Joke  Rant  Admin

[Permalink](#) | [Advertise](#) | [Privacy](#) | [Mobile](#)
 Web02 | 2.6.130903.1 | Last Updated 15 Feb 2013

Layout: [fixed](#) | [fluid](#)

Article Copyright 2013 by Sandeep Mewara
 Everything else Copyright © CodeProject, 1999-2013
[Terms of Use](#)