HTML5 Canvas

- ✓ The HTML5 <canvas> element is used to draw graphics, on the fly, via scripting (usually JavaScript).
- ✓ The <canvas> element is only a container for graphics. You must use a script 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”></canvas>
```

- ✓ For providing the border/add a border, use style attributes

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

HTML5 Canvas

- 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 **context2d**.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
}
```

HTML5 Canvas (Rectangle Example)

```
<html><head>
   <script type="text/javascript">
      function drawShape() {
         // Get the canvas element using the DOM
         var canvas = document.getElementById('mycanvas');
         // Make sure we don't execute when canvas isn't supported
         if (canvas.getContext) {
            // use getContext to use the canvas for drawing
            var ctx = canvas.getContext('2d');
            // Draw shapes
            ctx.fillRect(25, 25, 100, 100);
            ctx.clearRect(45, 45, 60, 60);
            ctx.strokeRect(50, 50, 50, 50);
         } else {
            alert('You need Safari or Firefox 1.5+ to see this demo.');
         }
      }
   </script></head>
<body onload="drawShape();">
   <canvas id="mycanvas" width="200px" height="200px"></canvas>
</body></html>
```

HTML5 Inline SVG

- ✓ SVG stands for Scalable Vector Graphics; 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 used to define vector-based graphics for the Web like Pie charts, two-dimensional graphs
- ✓ SVG defines the graphics in XML format
- ✓ SVG graphics do **NOT** lose any quality if they are zoomed or resized
- ✓ Every element and every attribute in SVG files can be animated
- ✓ SVG is a W3C recommendation

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

SVG Advantages

Advantages of using SVG over other image formats (like JPEG and GIF) are:

- ✓ SVG images can be created and edited with any text editor
- ✓ SVG images can be searched, indexed, scripted, and compressed
- ✓ SVG images are scalable
- ✓ SVG images can be printed with high quality at any resolution
- ✓ SVG images are zoomable (and the image can be zoomed without degradation)

Difference Between SVG & Canvas

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

HTML5 SVG (Rectangle Example)

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      #svgelem {
        position: relative;
        left: 50%;
        transform: translateX(-50%);
      }
    </style>
    <title>SVG</title>
    <meta charset = "utf-8" />
  </head><body>
    <h2 align = "center">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>
```

HTML Media

- Multimedia on the web is sound, music, videos, movies, and animations and comes in many different formats.
- TML5 features include native audio and video support without the need for Flash
- The most common way to discover the type of a file, is to look at the file extension like: .wav, .mp3, .mp4 etc.
- The MP4, WebM, and Ogg formats are supported by HTML.
- The MP4 format is recommended by YouTube

Tag	Description
<audio>	Defines sound content
<video>	Defines a video or movie
<source>	Defines multiple media resources for <video> and <audio>
<embed>	Defines a container for an external application or interactive content (a plug-in)
<track>	Defines text tracks for <video> and <audio>

HTML5 Video & Audio

- The HTML5 <audio> and <video> tags make it simple to add media to a website

```
<video  width = "300" height = "200" controls autoplay>
        <source src = "/html5/foo.ogg" type ="video/ogg" />
        <source src = "/html5/foo.mp4" type = "video/mp4" />
        Your browser does not support the <video> element.
</video>

<audio controls autoplay>
        <source src = "/html5/audio.ogg" type = "audio/ogg" />
        <source src = "/html5/audio.wav" type = "audio/wav" />
        Your browser does not support the <audio> element.
</audio>
```

Video Formats and Browser Support

Browser	MP4	WebM	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	NO **Update:** Firefox 21 running on Windows 7, Windows 8, Windows Vista, and Android now supports MP4	YES	YES
Safari	YES	NO	NO
Opera	NO	YES	YES

Audio Formats and Browser Support

Browser	MP3	Wav	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	NO **Update:** Firefox 21 running on Windows 7, Windows 8, Windows Vista, and Android now supports MP3	YES	YES
Safari	YES	YES	NO
Opera	NO	YES	YES

HTML5 YouTube

- The easiest way to play videos in HTML, is to use YouTube; Converting videos to different formats can be difficult and time-consuming.
- For Youtube AutoPlay without controls and but muted

```
<iframe width="420" height="345"
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1&mute=1&controls=0">
</iframe>
```

HTML5 Geolocation

- ✓ The HTML5 Geolocation API is used to get the geographical position of a user.
- ✓ Since this can compromise user privacy, the position is not available unless the user approves it.
- ✓ The **getCurrentPosition()** method is used to return the user's position.

```
function getLocation() {
  var geolocation = navigator.geolocation;
  geolocation.getCurrentPosition(showLocation, errorHandler);
}
```

Information you get from Geolocation API

Property	Description
coords.latitude	The latitude as a decimal number
coords.longitude	The longitude as a decimal number
coords.accuracy	The accuracy of position
coords.altitude	The altitude in meters above the mean sea level
coords.altitudeAccuracy	The altitude accuracy of position
coords.heading	The heading as degrees clockwise from North
coords.speed	The speed in meters per second
timestamp	The date/time of the response

HTML5 Input Types

HTML5 has several new input types for forms. These new features allow better input control and validation.

- ✓ color
- ✓ Date
- ✓ datetime
- ✓ datetime-local
- ✓ email
- ✓ month
- ✓ number
- ✓ range
- ✓ search
- ✓ tel
- ✓ time
- ✓ url
- ✓ week

HTML5 Form Elements

HTML5 has the following new form elements:

- ✓ <datalist>
- ✓ <keygen>
- ✓ <output>

HTML5 <datalist> Element

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

The <datalist> element is used to provide an "autocomplete" feature on <input> elements. Users will see a drop-down list of pre-defined options as they input data.

Use the <input> element's list attribute to bind it together with a <datalist> element.

HTML5 <datalist> Element

```
<!DOCTYPE html>
<html><body>
<h2>The datalist Element</h2>
<p>The datalist element specifies a list of pre-defined options for an input element.</p>

<form action="/action_page.php">
  <input list="browsers" name="browser">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
  <input type="submit">
</form>

<p><b>Note:</b> The datalist tag is not supported in Safari prior version 12.1.</p>
</body></html>
```

HTML5 <keygen> Element

- ✓ The purpose of the <keygen> element is to provide a secure way to authenticate users.
- ✓ The <keygen> tag specifies a key-pair generator field in a form.
- ✓ When the form is submitted, two keys are generated, one private and one public.
- ✓ The private key is stored locally, and the public key is sent to the server. The public key could be used to generate a client certificate to authenticate the user in the future.

HTML5 <keygen> Element

```
<!DOCTYPE html>
<html>
<body>

<form action="/action_page.php" method="get">
  Username: <input type="text" name="usr_name">
  Encryption: <keygen name="security" challenge = "0987654321">
  <input type="submit">
</form>

<p><strong>Note:</strong> The keygen tag is not supported in Internet
Explorer.</p>

</body>
</html>
```

HTML5 <output> Element

The <output> element represents the result of a calculation (like one performed by a script).

```
<!DOCTYPE html>
<html>
<body>

<h1>The output element</h1>

<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">
<input type="range" id="a" value="50">
+<input type="number" id="b" value="25">
=<output name="x" for="a b"></output>
</form>

<p><strong>Note:</strong> The output element is not supported in Edge 12 (or earlier).</p>

</body>
</html>
```

HTML5 Web Storage

- ✓ With HTML5, web pages can store data locally within the user's browser.
- ✓ Earlier, this was done with cookies. However, Web Storage is more secure and faster. The data is not included with every server request, but used ONLY when asked for. It is also possible to store large amounts of data, without affecting the website's performance.
- ✓ The data is stored in key/value pairs, and a web page can only access data stored by itself.

HTML5 Web Storage

- There are two new objects for storing data on the client:
  - localStorage - stores data with no expiration date
  - sessionStorage - stores data for one session
- 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 browser window.

Before using web storage, check browser support for localStorage and sessionStorage

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

HTML5 Web Storage

```
<!DOCTYPE HTML>

<html>
  <body>
    <script type = "text/javascript">
      //sessionStorage for session storage
      //to clear local storage
      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>
```

HTML5 Application Cache

HTML5 introduces application cache, which means that a web application is cached, and accessible without an internet connection.

- ✓ Application cache gives an application three advantages:
- ✓ **Offline browsing** - users can use the application when they're offline
- ✓ **Speed** - cached resources load faster
- ✓ **Reduced server load** - the browser will only download updated/changed resources from the server

HTML5 Cache Manifest Example

The example below shows an HTML document with a cache manifest (for offline browsing):

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">

<body>
The content of the document.....
</body>

</html>
```

Cache Manifest Basics

To enable application cache, include the manifest attribute in the document's <html> tag.

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

Every page with the manifest attribute specified will be cached when the user visits it. If the manifest attribute is not specified, the page will not be cached (unless the page is specified directly in the manifest file).

The recommended file extension for manifest files is: ".appcache”

The Manifest File

The manifest file is a simple text file, which tells the browser what to cache (and what to never cache).

The manifest file has three sections:

- ✓ **CACHE MANIFEST** - Files listed under this header will be cached after they are downloaded for the first time
- ✓ **NETWORK** - Files listed under this header require a connection to the server, and will never be cached
- ✓ **FALLBACK** - Files listed under this header specifies fallback pages if a page is inaccessible

CACHE MANIFEST

The first line, CACHE MANIFEST, is required:

```
CACHE MANIFEST
/theme.css
/logo.gif
/main.js
```

The manifest file above lists three resources: a CSS file, a GIF image, and a JavaScript file. When the manifest file is loaded, the browser will download the three files from the root directory of the web site. Then, whenever the user is not connected to the internet, the resources will still be available.

NETWORK

The NETWORK section below specifies that the file "login.asp" should never be cached, and will not be available offline.

```
NETWORK:
login.asp
```

An asterisk can be used to indicate that all other resources/files require an internet connection:

```
NETWORK:
*
```

FALLBACK

The FALLBACK section below specifies that "offline.html" will be served in place of all files in the /html/ catalog, in case an internet connection cannot be established:

FALLBACK:
/html/ /offline.html

Updating the Cache

Once an application is cached, it remains cached until one of the following happens:

- ✓ The user clears the browser's cache
- ✓ The manifest file is modified (see tip below)
- ✓ The application cache is programmatically updated

HTML5 Web Workers

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.
- 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.,

HTML5 Server sent events

Conventional web applications generate events which are dispatched to the web server. For example, a simple click on a link requests a new page from the server.

Server-Sent Events (SSE) allow a web page to get 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.

HTML5 WebSockets

WebSockets is a next-generation bidirectional communication technology for web applications which operates over a single socket.

- 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. For example : Chat messenger

Websockets and **SSE (Server Sent Events)** are both capable of pushing data to browsers, however they are not competing technologies.

- **Websockets** connections can both send data to the browser and receive data from the browser.
- **SSE** connections can only push data to the browser.

HTML5 WebSockets

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.