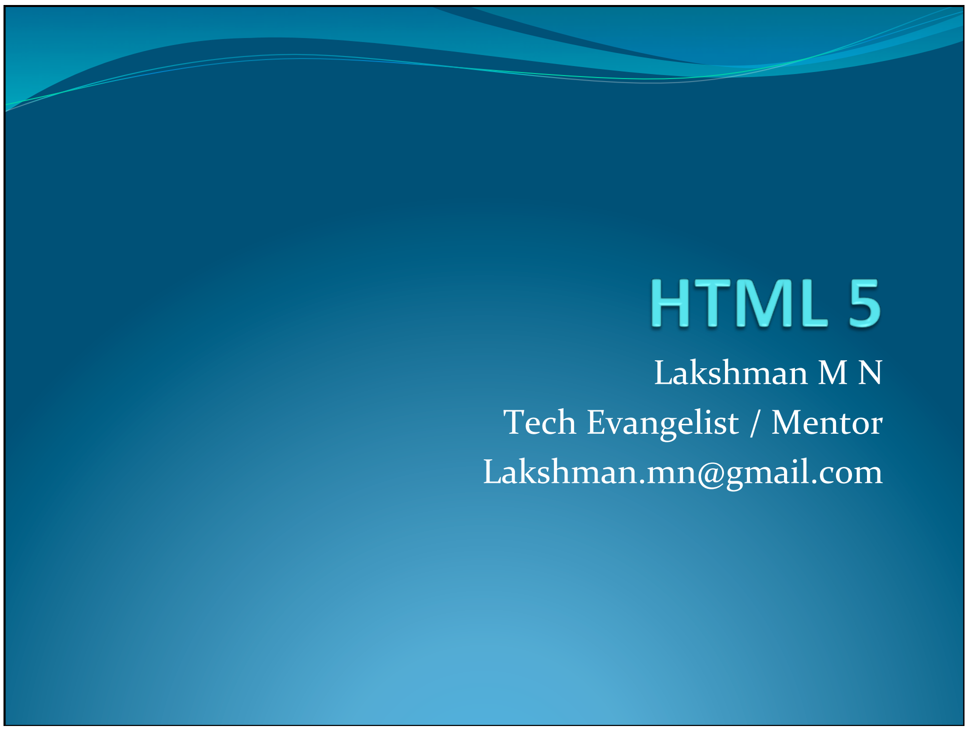
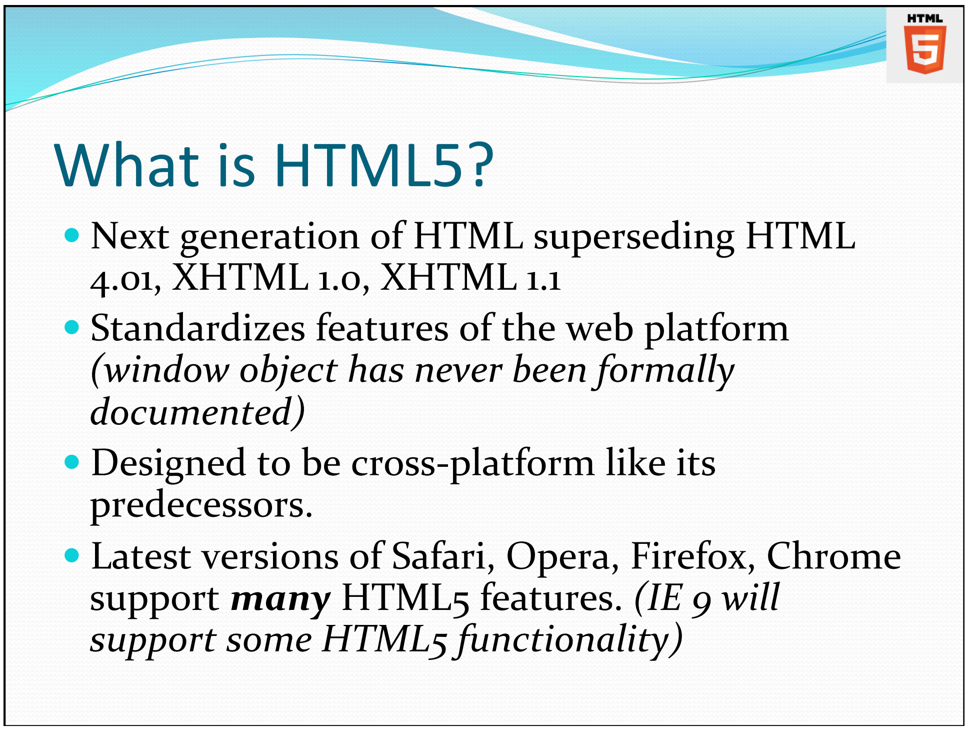


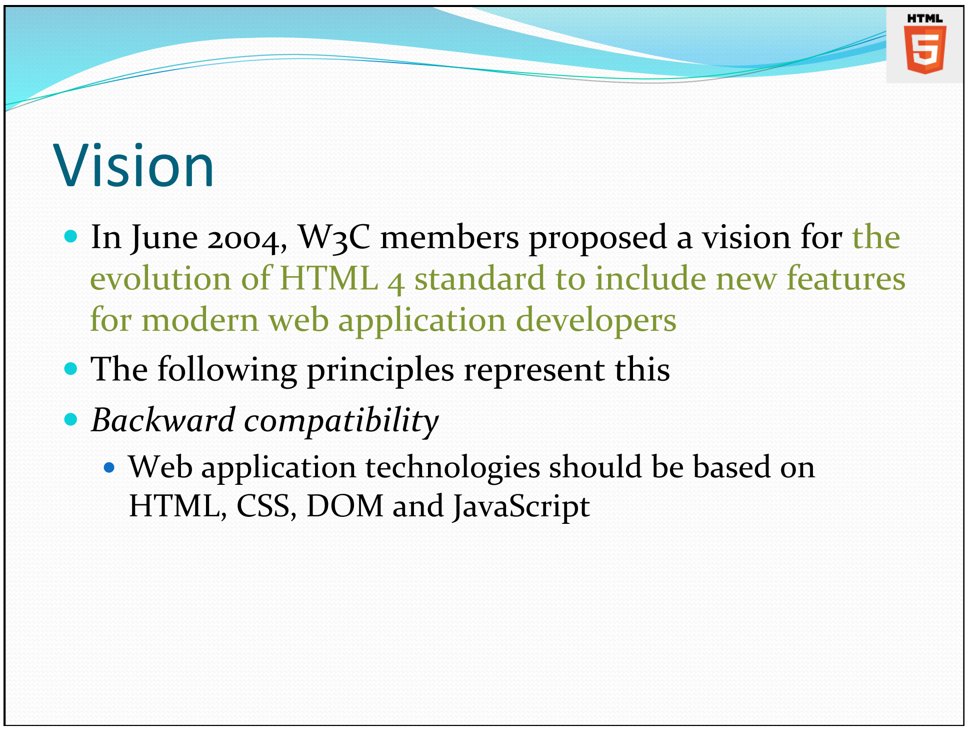
HTML 5

Lakshman M N
Tech Evangelist / Mentor
Lakshman.mn@gmail.com



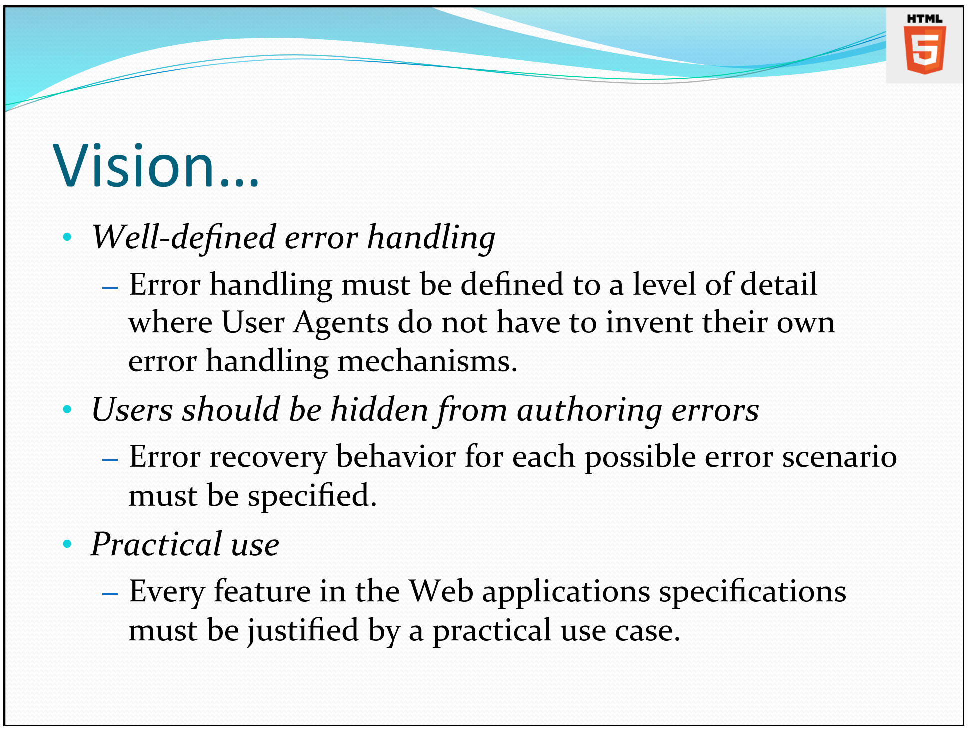
What is HTML5?

- Next generation of HTML superseding HTML 4.01, XHTML 1.0, XHTML 1.1
- Standardizes features of the web platform *(window object has never been formally documented)*
- Designed to be cross-platform like its predecessors.
- Latest versions of Safari, Opera, Firefox, Chrome support ***many*** HTML5 features. *(IE 9 will support some HTML5 functionality)*



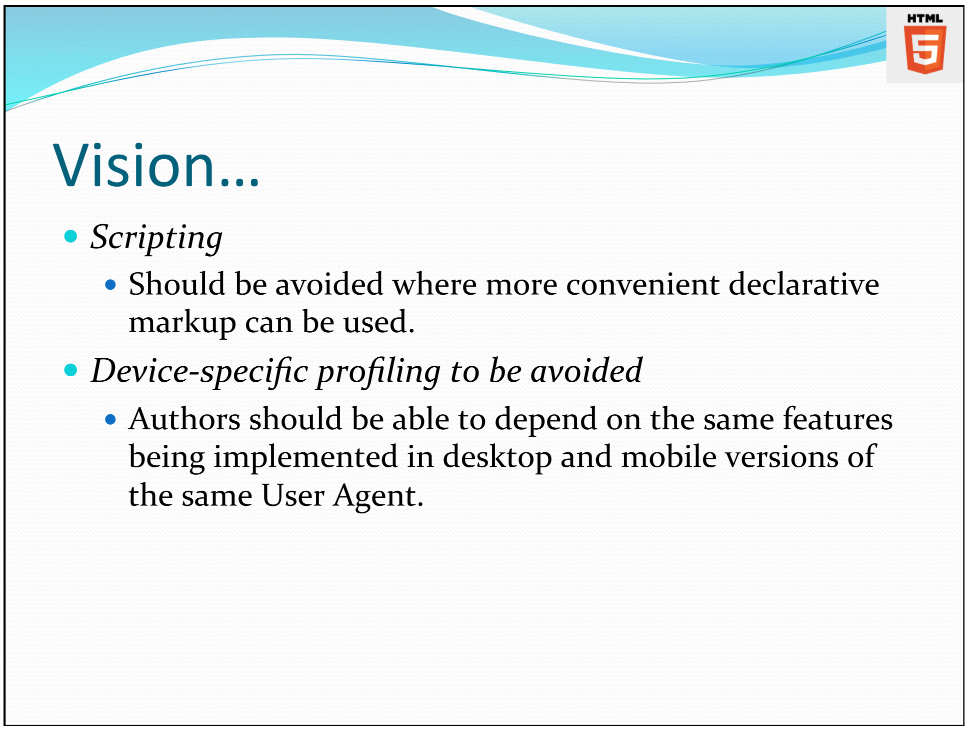
Vision

- In June 2004, W3C members proposed a vision for the evolution of HTML 4 standard to include new features for modern web application developers
- The following principles represent this
- *Backward compatibility*
  - Web application technologies should be based on HTML, CSS, DOM and JavaScript



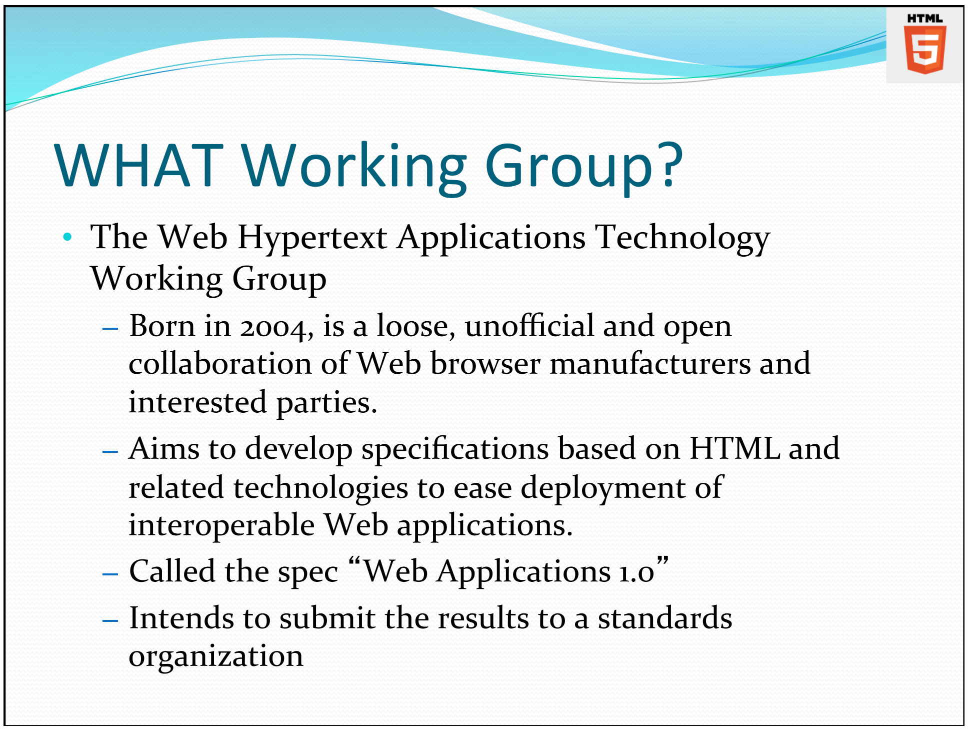
Vision…

- *Well-defined error handling*
  - Error handling must be defined to a level of detail where User Agents do not have to invent their own error handling mechanisms.
- *Users should be hidden from authoring errors*
  - Error recovery behavior for each possible error scenario must be specified.
- *Practical use*
  - Every feature in the Web applications specifications must be justified by a practical use case.



Vision…

- *Scripting*
  - Should be avoided where more convenient declarative markup can be used.
- *Device-specific profiling to be avoided*
  - Authors should be able to depend on the same features being implemented in desktop and mobile versions of the same User Agent.



WHAT Working Group?

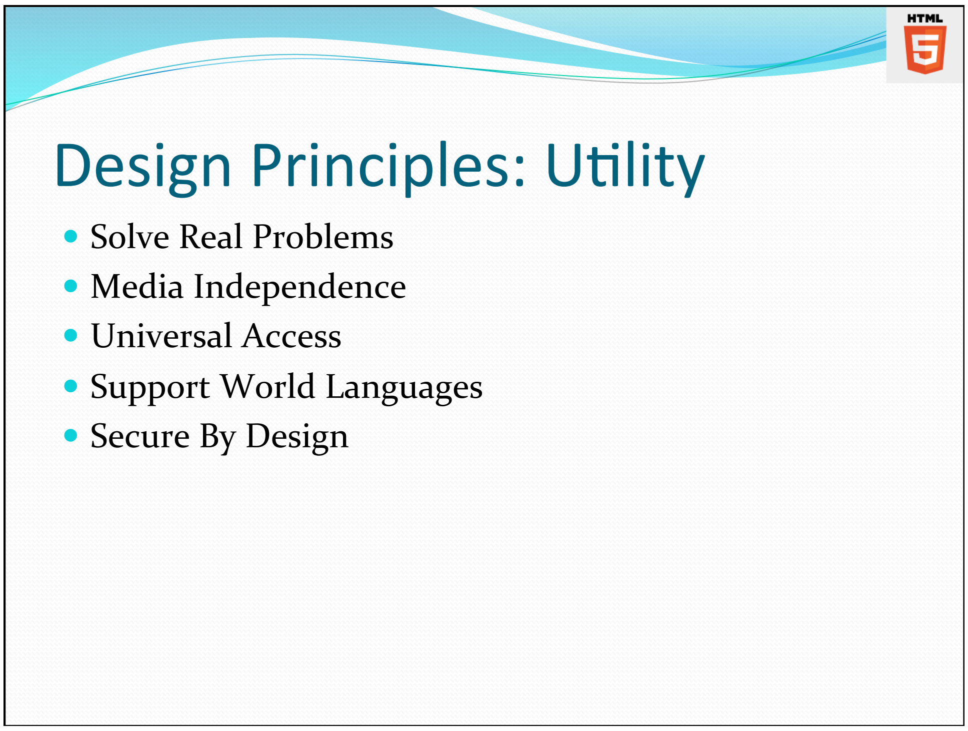
- The Web Hypertext Applications Technology Working Group
  - Born in 2004, is a loose, unofficial and open collaboration of Web browser manufacturers and interested parties.
  - Aims to develop specifications based on HTML and related technologies to ease deployment of interoperable Web applications.
  - Called the spec "Web Applications 1.0"
  - Intends to submit the results to a standards organization

WHATWG and W3C

- Tim Berners-Lee, the founder of W3C announced in Oct 2006 that the W3C would work together with the WHAT Working Group.
- The objective being to make incremental improvements to HTML.
- It was decided to rename "Web Applications 1.0" to "HTML5".
- The draft specifications at WHATWG is a superset of the HTML5 work being published at the W3C.

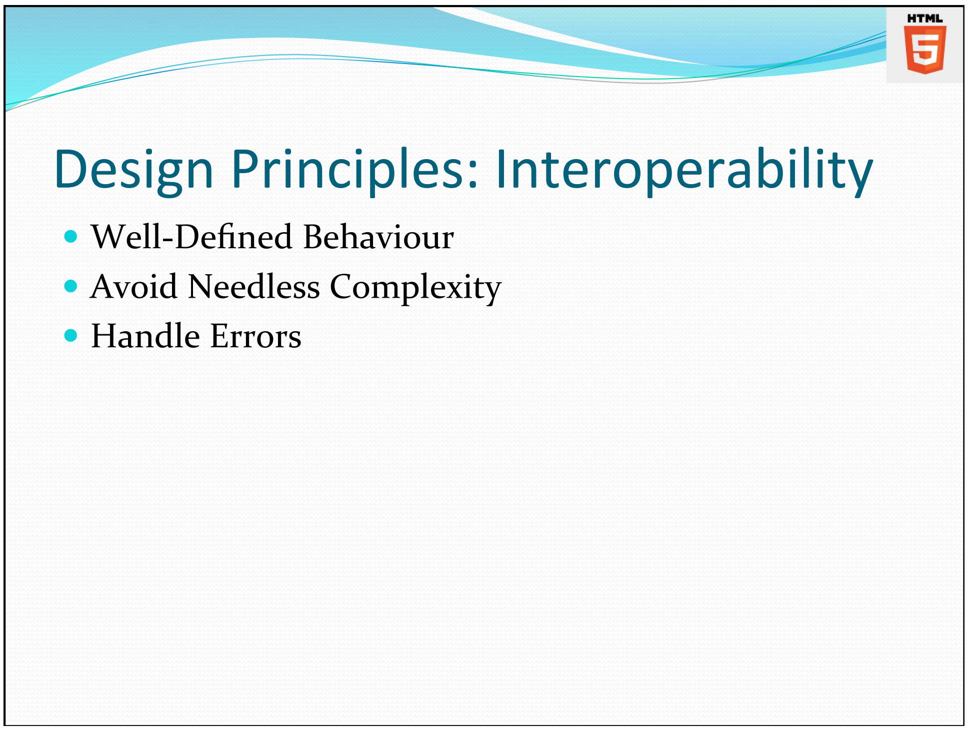
Design Principles: Compatibility

- Support Existing Content
- Degrade Gracefully
- Don't Reinvent the Wheel
- Evolution, not Revolution



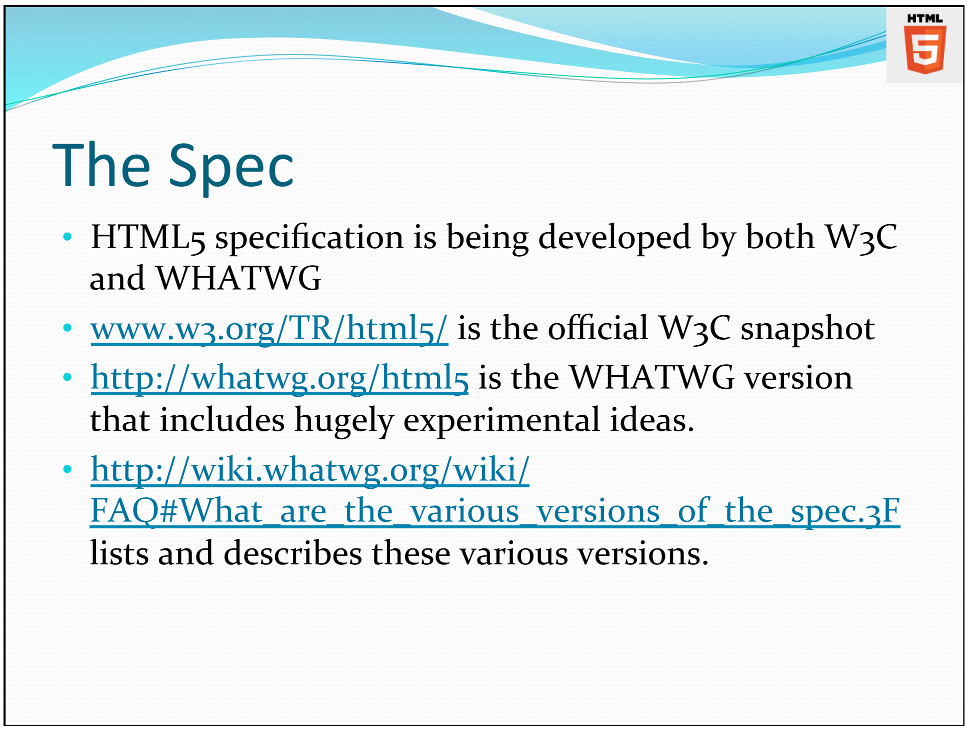
Design Principles: Utility

- Solve Real Problems
- Media Independence
- Universal Access
- Support World Languages
- Secure By Design



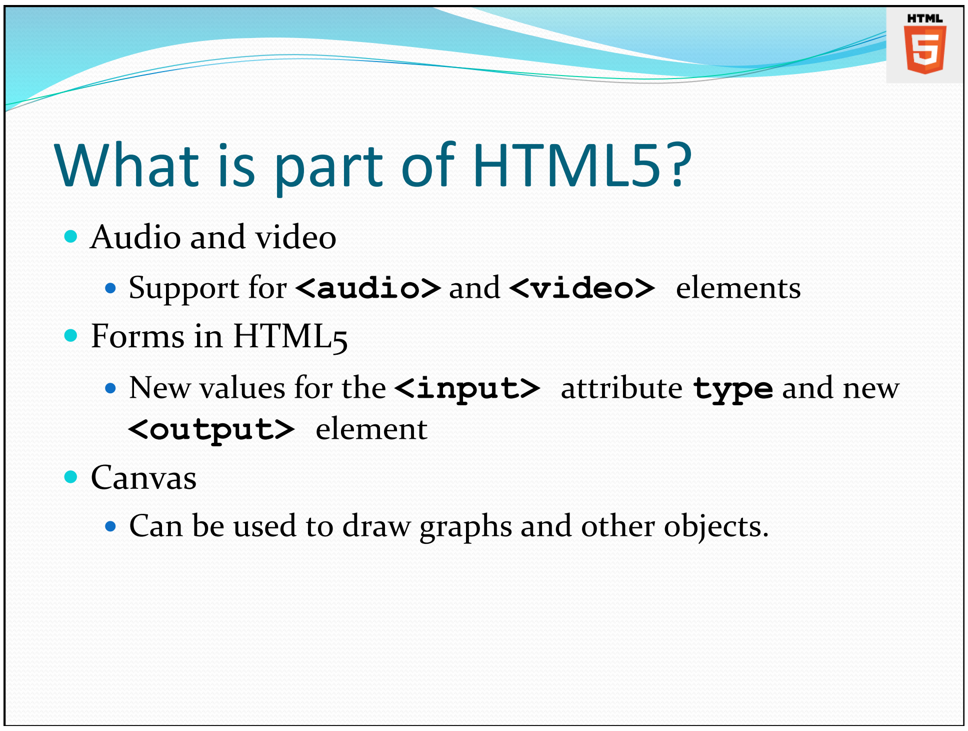
Design Principles: Interoperability

- Well-Defined Behaviour
- Avoid Needless Complexity
- Handle Errors



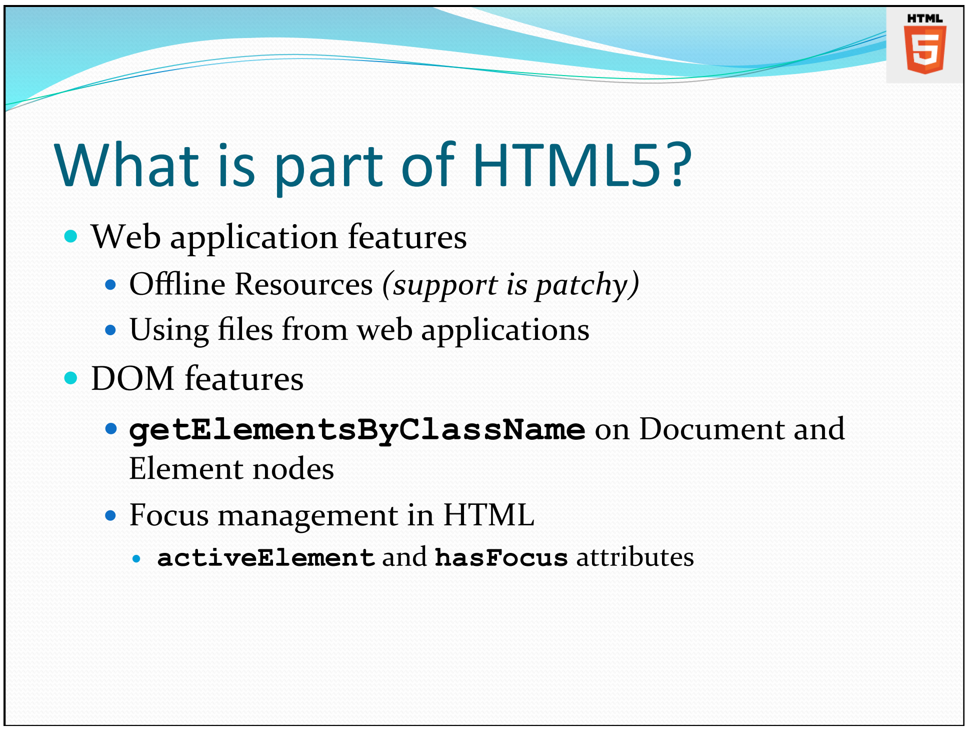
The Spec

- HTML5 specification is being developed by both W3C and WHATWG
- www.w3.org/TR/html5/ is the official W3C snapshot
- http://whatwg.org/html5 is the WHATWG version that includes hugely experimental ideas.
- http://wiki.whatwg.org/wiki/FAQ#What_are_the_various_versions_of_the_spec.3F lists and describes these various versions.



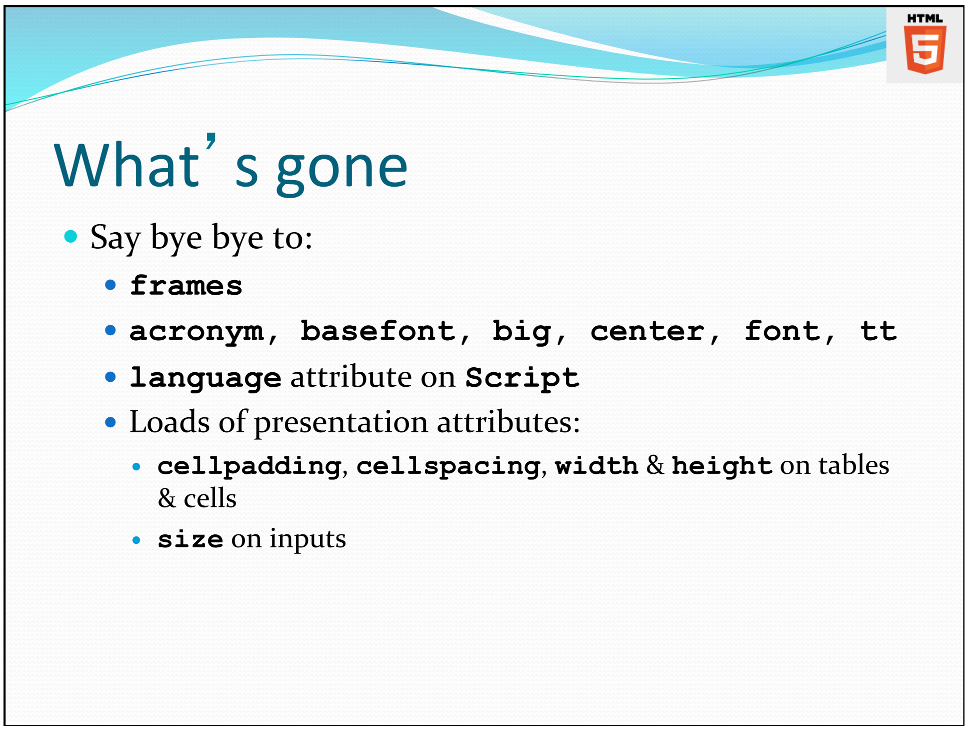
What is part of HTML5?

- Audio and video
  - Support for **`<audio>`** and **`<video>`** elements
- Forms in HTML5
  - New values for the **`<input>`** attribute **`type`** and new **`<output>`** element
- Canvas
  - Can be used to draw graphs and other objects.



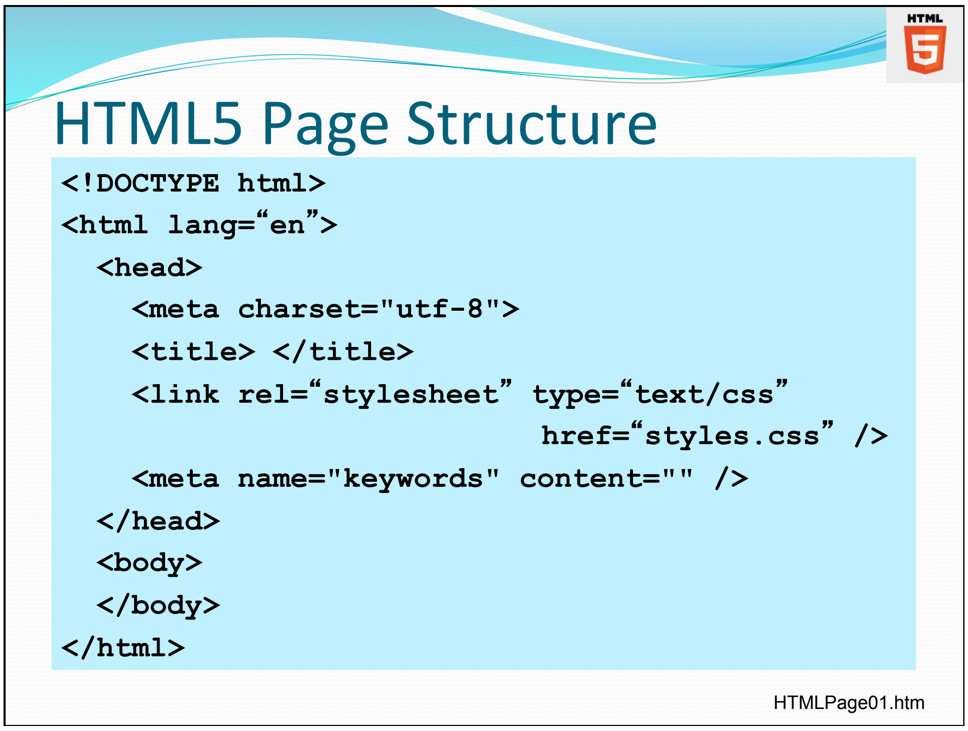
What is part of HTML5?

- Web application features
  - Offline Resources *(support is patchy)*
  - Using files from web applications
- DOM features
  - **`getElementsByClassName`** on Document and Element nodes
  - Focus management in HTML
    - **`activeElement`** and **`hasFocus`** attributes



What' s gone

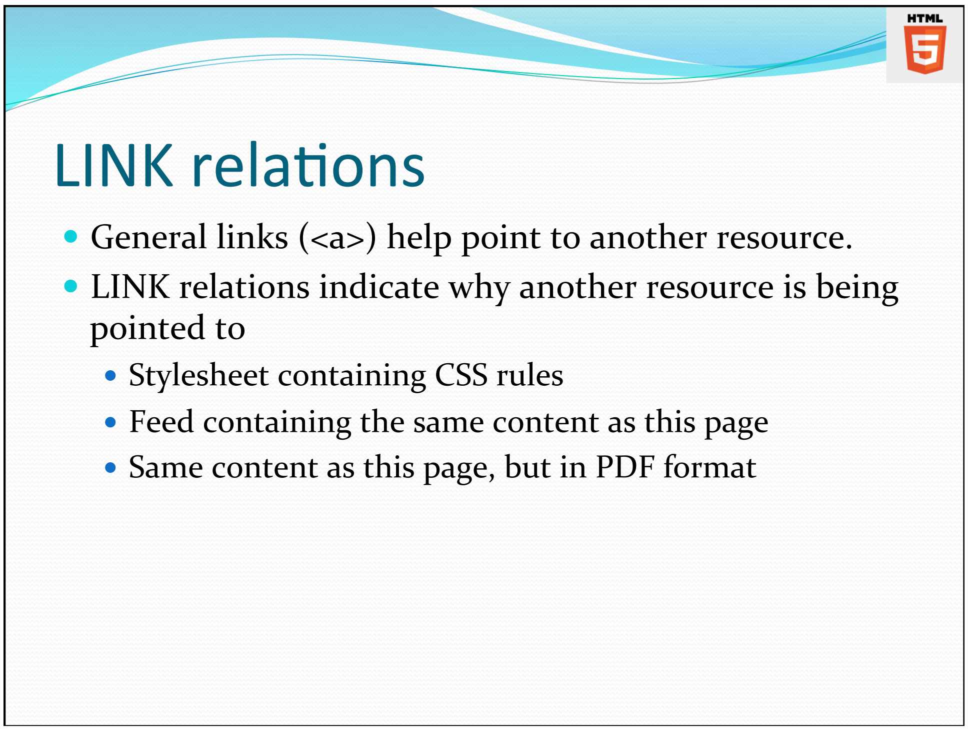
- Say bye bye to:
  - **`frames`**
  - **`acronym, basefont, big, center, font, tt`**
  - **`language`** attribute on **`Script`**
  - Loads of presentation attributes:
    - **`cellpadding`**, **`cellspacing`**, **`width`** & **`height`** on tables & cells
    - **`size`** on inputs



HTML5 Page Structure

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title> </title>
    <link rel="stylesheet" type="text/css"
                                href="styles.css" />
    <meta name="keywords" content="" />
  </head>
  <body>
  </body>
</html>
```

HTMLPage01.htm



LINK relations

- General links (<a>) help point to another resource.
- LINK relations indicate why another resource is being pointed to
  - Stylesheet containing CSS rules
  - Feed containing the same content as this page
  - Same content as this page, but in PDF format

LINK relations

- **rel = stylesheet**

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

  - Used for pointing to CSS rules stored in a separate file.

- **rel = alternate**

```
<link rel="alternate"
      type="application/atom+xml"
       title="My Blog Feed" href="/feed/" />
```

  - Used to enable feed auto-discovery to allow feed readers to discover that a site has a news feed.

HTMLPage02.htm

LINK relations…

- **rel = shortcut icon**

```
<link rel="shortcut icon" sizes="16x16"
  type="image/x-icon" href="myicon.ico"/>
```

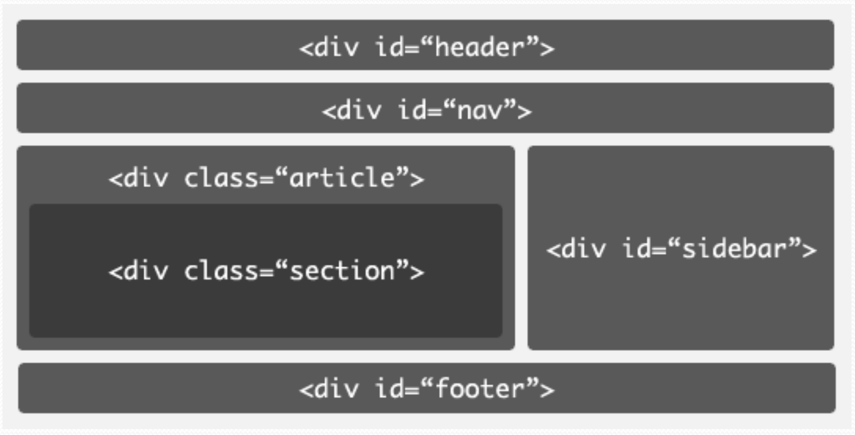
  - Used to associate an icon with the page.
  - displayed in the browser' s location bar next to the URL.

HTMLPage03.htm

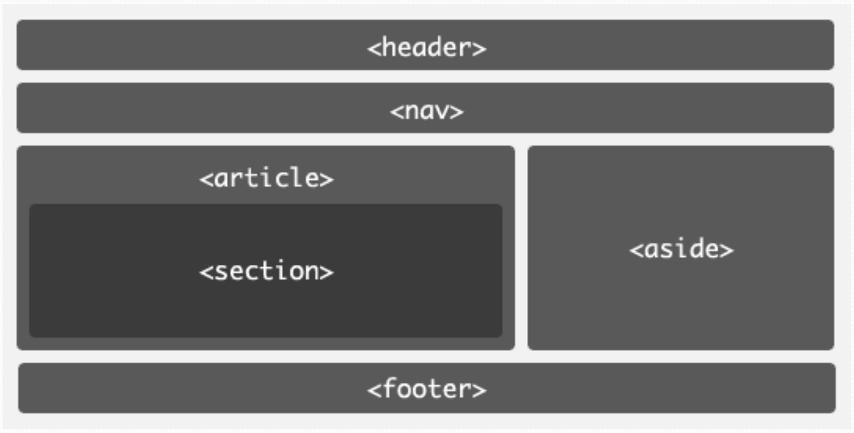
Need for new Structural Elements

- HTML4 provided semantic elements to define different features of a web page such as
  - Forms
  - Lists
  - Paragraphs
- **`<div>`** and **`<span>`** elements with different id and class attributes are used to define features such as
  - Headers
  - Footers
  - Side bars

HTML4 Structure



HTML5 Structure



HTML5 Structural Elements…

- **`<hgroup>`**
  - Represents the heading of a section.
  - Can be used to wrap multiple headings like a main heading and a sub heading.
- **`<time>`**
  - Used to markup times and dates
- **`<mark>`**
  - Used to highlight terms or parts of content that need attention. (like highlighting with a highlighter pen)

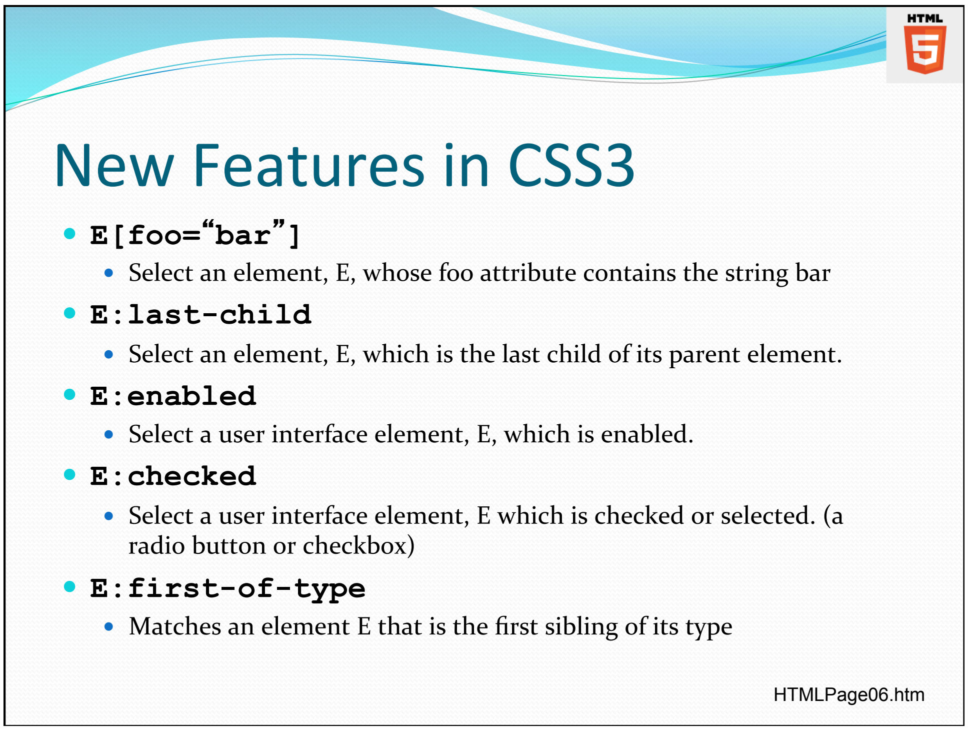
HTMLPage04.htm
HTMLPage05.htm

Deprecated Elements

- A number of tags and attributes have been deprecated in HTML5 and are supported only for backward compatibility.
- A partial list includes
  - **`<applet>`** replaced by **`<object>`**
  - **`<bgsound>`** replaced by **`<audio>`**
  - **`<center>`** replaced by **`text-align:center`** in CSS
  - **`<font>`** replaced by relevant font properties in CSS.
  - **`<frameset>`**, **`<frame>`** are no longer valid in HTML5

Deprecations in HTML5

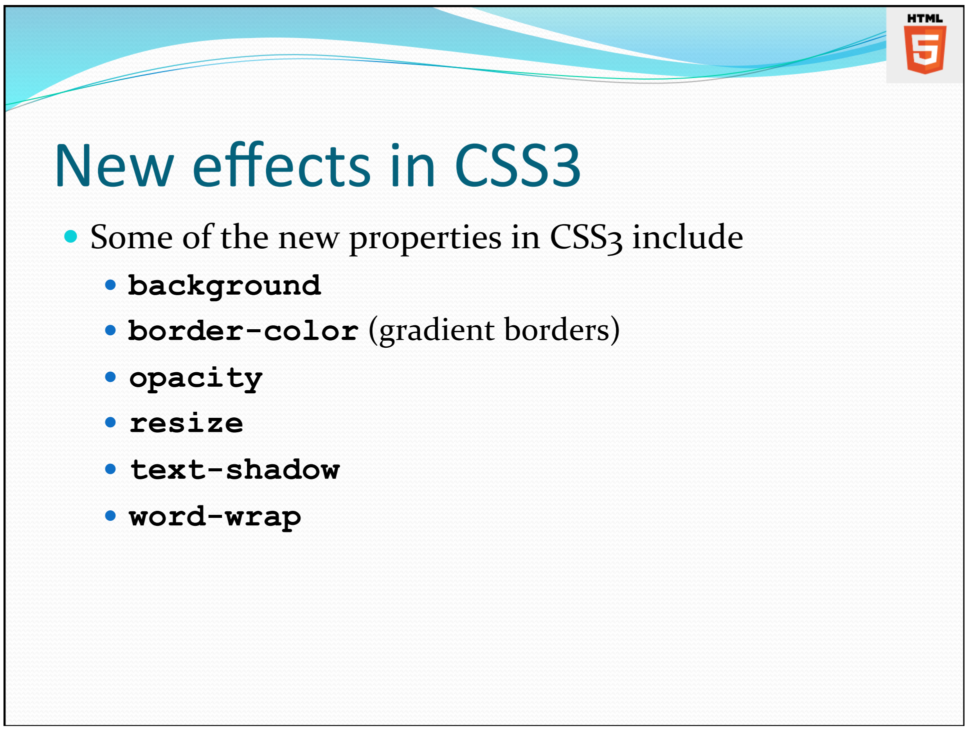
- The following indicates a partial list of attributes that are deprecated in HTML5
  - align, border of the **`<IMG>`** tag
  - language attribute of the **`<SCRIPT>`** tag
- The non-breaking space HTML entity **`&nbsp;`** should be avoided.
  - The decimal entity **`&#160;`** should be used.



New Features in CSS3

- `E[foo="bar"]`
  - Select an element, E, whose foo attribute contains the string bar
- `E:last-child`
  - Select an element, E, which is the last child of its parent element.
- `E:enabled`
  - Select a user interface element, E, which is enabled.
- `E:checked`
  - Select a user interface element, E which is checked or selected. (a radio button or checkbox)
- `E:first-of-type`
  - Matches an element E that is the first sibling of its type

HTMLPage06.htm

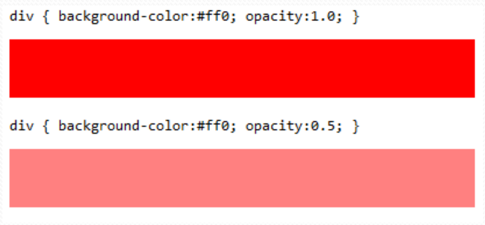


New effects in CSS3

- Some of the new properties in CSS3 include
  - `background`
  - `border-color` (gradient borders)
  - `opacity`
  - `resize`
  - `text-shadow`
  - `word-wrap`

Color - Opacity

- Adjusts the opacity of the selected element’ s presentation on screen.
- Takes values between 0.0 (fully transparent) and 1.0 (fully opaque)



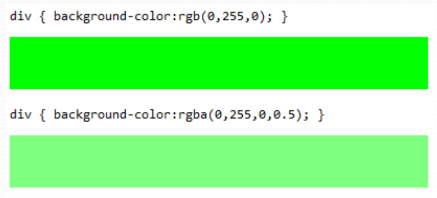
```
div { background-color:#ff0; opacity:1.0; }
```

```
div { background-color:#ff0; opacity:0.5; }
```

CSS01.htm

RGBA Color

- Like RGB color definitions, but allows a fourth field, defining the alpha value of the color being applied.
- Like opacity, the alpha value is between 0.0 (fully transparent) and 1.0 (fully opaque)



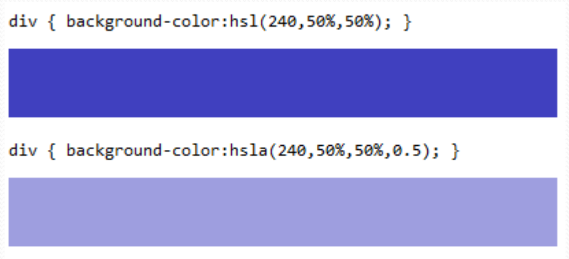
```
div { background-color:rgb(0,255,0); }
```

```
div { background-color:rgba(0,255,0,0.5); }
```

CSS02.htm

HSL/A Color

- HSL color definitions accept three arguments: hue is a degree on a color wheel (0-360), saturation is a percentage, and lightness is a percentage.
- HSLA is like HSL color, but allows a fourth field, defining the alpha value of the color being applied.



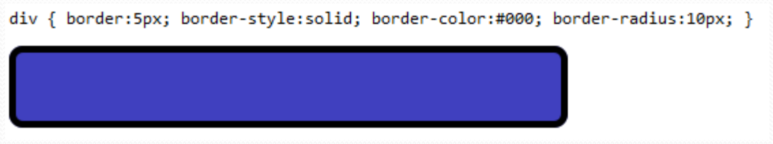
```
div { background-color:hsl(240,50%,50%); }
```

```
div { background-color:hsla(240,50%,50%,0.5); }
```

CSS03.htm

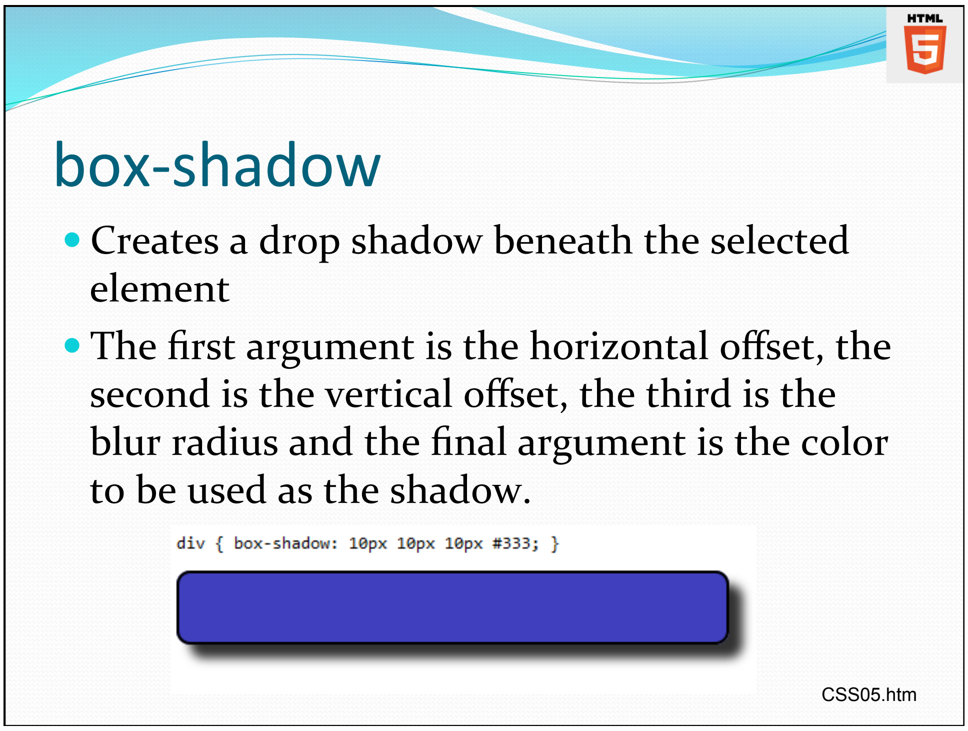
border

- border-color
  - Allows for multiple border colors to be specified.
- border-radius
  - Curves the corners of the border using the radius given, often in pixels.



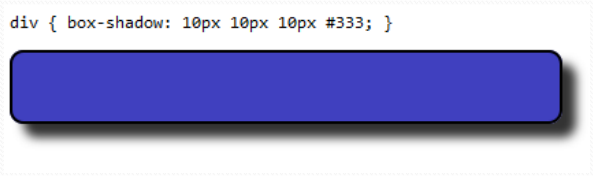
```
div { border:5px; border-style:solid; border-color:#000; border-radius:10px; }
```

CSS04.htm



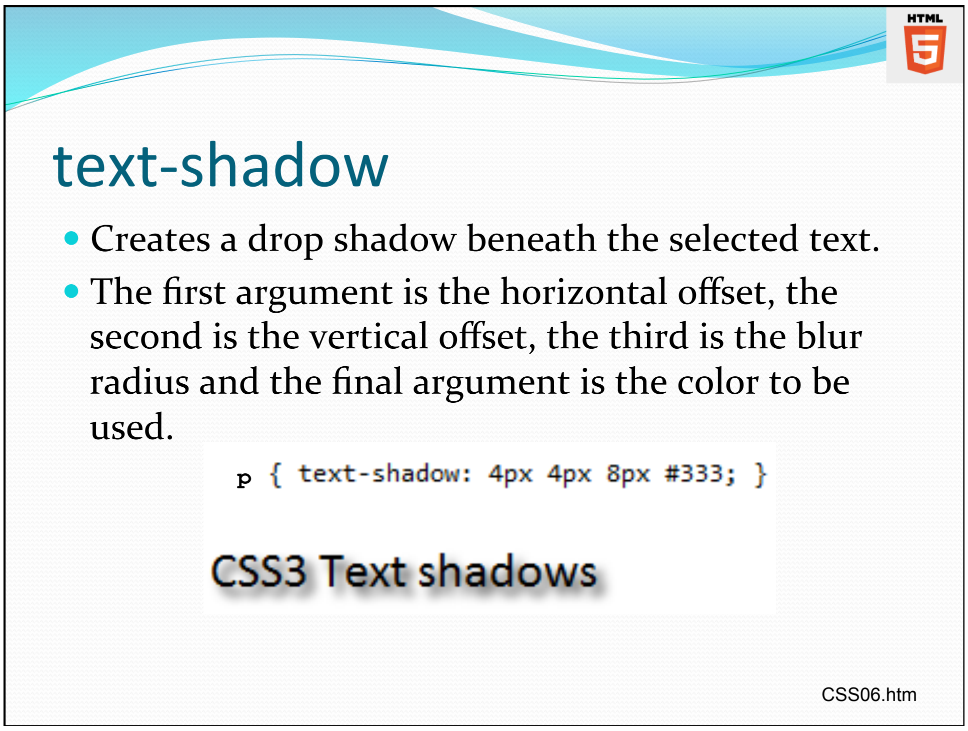
box-shadow

- Creates a drop shadow beneath the selected element
- The first argument is the horizontal offset, the second is the vertical offset, the third is the blur radius and the final argument is the color to be used as the shadow.



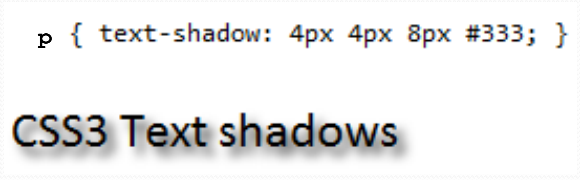
```
div { box-shadow: 10px 10px 10px #333; }
```

CSS05.htm



text-shadow

- Creates a drop shadow beneath the selected text.
- The first argument is the horizontal offset, the second is the vertical offset, the third is the blur radius and the final argument is the color to be used.



```
p { text-shadow: 4px 4px 8px #333; }
```

CSS3 Text shadows

CSS06.htm

background

- Multiple background images for box elements can be specified using a comma-separated list.
- Each value in the list generates a separate "background layer"
- The first value in the list represents the top layer (closest to the user)

```
div.class1
 {
   width: 1024px;
   height: 768px;
   background-image: url(sheep.png), url(stonehenge.jpg);
   background-position: center bottom, left top;
   background-repeat: no-repeat;
 }
```

CSS07.htm

Transitions

- CSS3 helps developers create smooth transitions of CSS properties for elements.
- Basic CSS properties
  - **`transition-property`**
    - The CSS property to animate eg. `background-color`
  - **`transition-duration`**
    - Specified in seconds eg.2s
  - **`transition-delay`**
    - Delay before the transition start, specified in seconds eg. 1s
  - **`transition-timing-function`**
    - Used for appearance of the transition
    - Values include **`ease (default), linear, ease-out, ease-in-out`**

CSS08.htm

CSS3 - Columns

- CSS3 allows for the creation of multiple columns for text layout.
- **`column-count`**
  - Specifies the number of columns an element should be divided into
- **`column-gap`**
  - Specifies the gap between the columns
- **`column-rule`**
  - Sets the width, color and style of the rule between columns.

CSS-MulticolumnDemo.htm

UI Properties - appearance

- CSS3 User Interface module introduces several new features that enable web designers to enhance the user experience.

`appearance`

- Allows changing any element into one of the standard UI elements.
- **`appearance: normal | icon | window | button | menu | field`**

CSS09.htm

UI Properties – box-sizing

- **`box-sizing`**
  - Allows defining of certain elements to fit an area in a certain manner.

```
box-sizing : content-box | border-box |
inherit
```

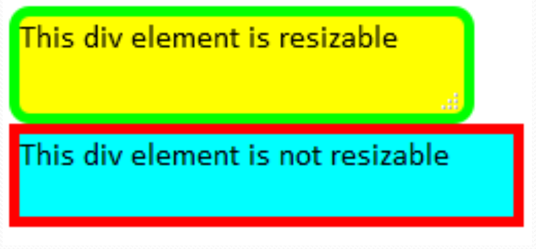
  - **`content-box`** – default
  - **`border-box`** – the width specified for the element is the total width of the element including border if any.
  - **`inherit`** – box-sizing value inherited from the parent element.

CSS10.htm

UI Properties – resize

- The resize property grants users control over the size display of an element on a webpage.

```
resize : none| horizontal | vertical |
both
```



This div element is resizable

This div element is not resizable

CSS11.htm

Generated Content Properties

- CSS3 provides properties to insert and move content in a document
- Inserted content can include counters and strings.
- `content`
  - Used in conjunction with the :before or :after pseudo-elements, inserts generated content.
  - The generated content is only rendered, it does not appear in the DOM tree.

```
content:url(bottle.jpeg);
```

CSS13.htm

CSS Media Queries

- Media queries allow styling of elements for specific form factors using attributes like
  - Browser dimensions (width, height, aspect ratio)
  - Device dimensions (device-width, device-height)
  - Browser orientation
  - Color information (color, color index)
  - Device-specific details (resolution)

* Not all these properties are currently supported.

Media Features

- Media features are use to target media with values like “screen”
- **`min-width, max-width`**
  - This calls for a special stylesheet if the media device is a screen and the max-width of the viewing area is 800 pixels

```
<link rel=“stylesheet” media=“screen and (max-
  width: 800px)” href=“styles1.css”/>
```

  - CSS declarations in the stylesheet can contain declarations specific to a device or width

```
@media screen and (max-width: 800px) {
body div {width: 760px;}
header nav ul {width: 740px;}
}
```

Media Features…

- Values can be combined by including “**`min-width`**” and “**`max-width`**” and are intended to serve different styles

```
@media screen and (min-width: 800px) and
  (max-width: 1200px)
{
   section {width: 100px;}
}
```

Media Features…

- `max-device-width`
  - This value is used to target mobile devices

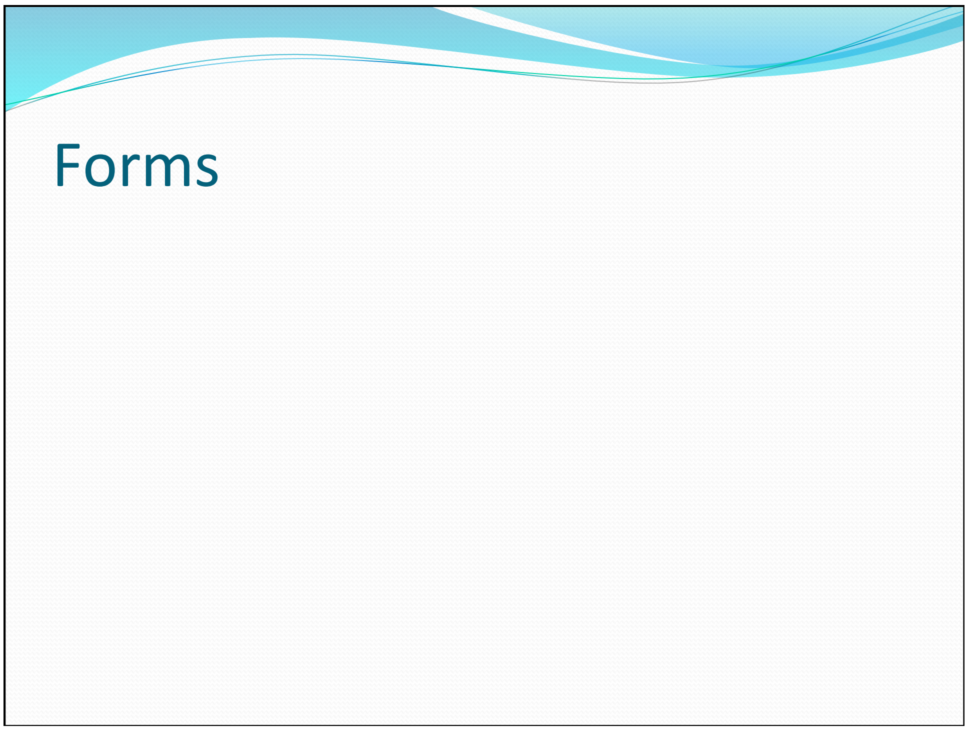
```
@media screen and (max-device-width:
  480px)
{
.class1
 {
   background:#000;
 }
}
```

CSS-Media01.htm
defStyle.css
media-queries.css

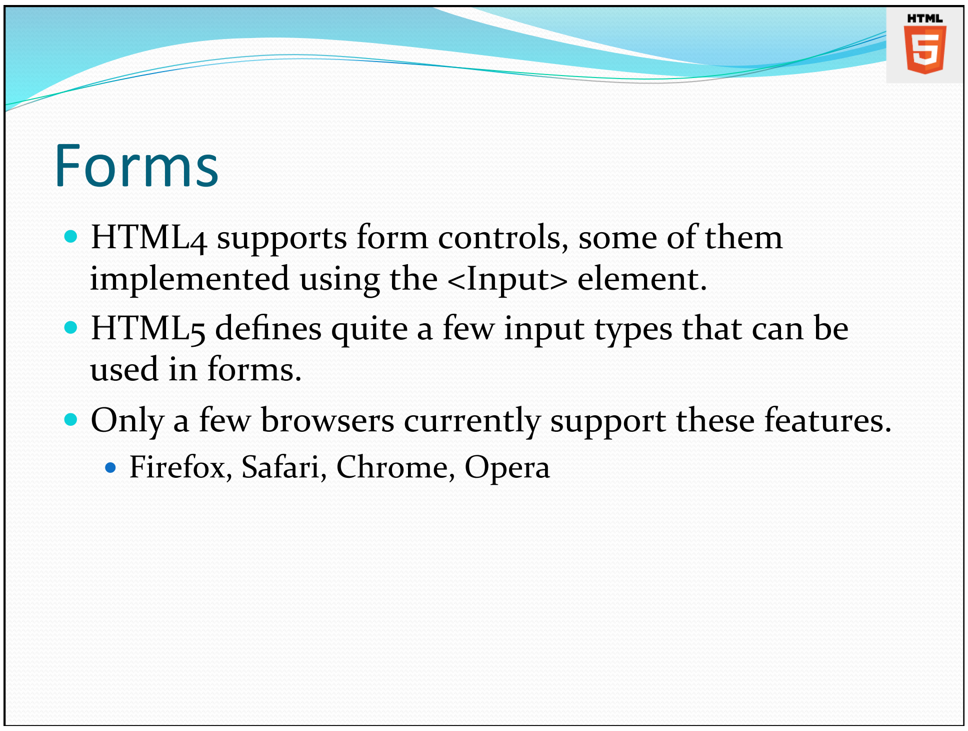
HTML5 Detection Library

- Modernizr (http://www.modernizr.com)
  - An open source, MIT-licensed JavaScript library
  - Detects support for many HTML5 and CSS3 features
  - Runs automatically
  - Creates a global object called **`Modernizr`** that contains a set of Boolean properties for each feature it can detect.

ModernizrDemo.htm



Forms



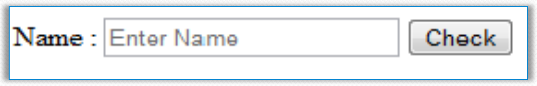
Forms

- HTML4 supports form controls, some of them implemented using the <Input> element.
- HTML5 defines quite a few input types that can be used in forms.
- Only a few browsers currently support these features.
  - Firefox, Safari, Chrome, Opera

Placeholder Text

- This provides the ability to set placeholder text in an input field.
- It is displayed in the field as long as the field is empty and not focused.

```
<form>
   <label for="txtName">Name : </label>
   <input id="txtName"
          placeholder="Enter Name"/>
       <input type="submit" value="Check"/>
</form>
```



HTMLForm01.htm

Autofocus Fields

- JavaScript has been the choice to focus an input field on a form.
- HTML5 introduces an autofocus attribute on all form controls.
- Unlike scripts this is markup and therefore will be consistent across all sites.

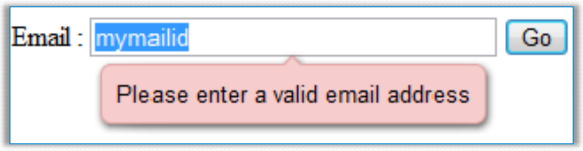
```
<form>
   <label for="txtName">Name : </label>
   <input id="txtName" type="text"

     autofocus/>
    <input type="submit" value="Check"/>
</form>
```

HTMLForm02.htm

Email addresses

- “email” is regarded as one the types of input in HTML5.

```
<form>
   <label for=“txtEMail”>EMail : </label>
   <input id=“txtEmail”
                         type=“email”/>
   <input type=“submit” value=“Go”/>
</form>
```


Email : mymailid Go

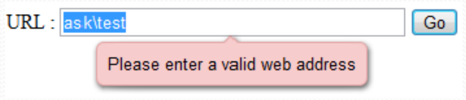
Please enter a valid email address

HTMLForm03.htm

Web Addresses

- The syntax of a web address is constrained by relevant Internet standards.
- “url” is one of the additions in HTML5.

```
<form>
        <label for=“txtURL”>URL : </label>
        <input name=“txtURL” type=“url”/>
        <input type=“submit” value=“Go”/>
</form>
```


URL : ask\test Go

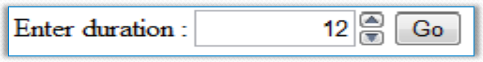
Please enter a valid web address

HTMLForm04.htm

Dealing with Numbers

- Numbers can be trickier than email or web addresses since we may need them in a range.
- We may need numbers of a certain kind in a range.
- HTML5 caters to these numbering needs!

```
<input type="number" min="0" max="10"
                   step="2" value="4"/>
```

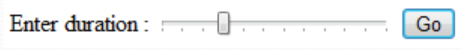


HTMLForm05.htm

Dealing with Numbers…

- Slider controls can be used in forms.
- The type of input is "range".
- The available attributes are the same as those for type="number".
- The difference is in the UI.

```
<input type="range" min="0" max="10"
              step="2" value="4"/>
```



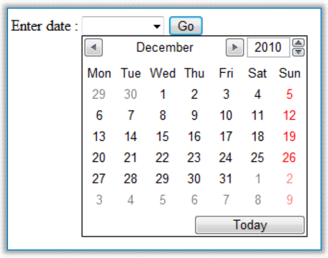
HTMLForm06.htm

Date Pickers

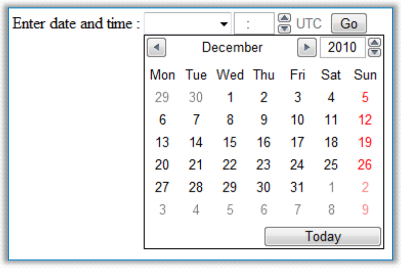
- Date picker control was sorely lacking in HTML4.
  - This was worked around with the help of JavaScript frameworks
- HTML5 defines a way to include a native date picker.
- Options include
  - date, month, week, time, date + time

Date Pickers…

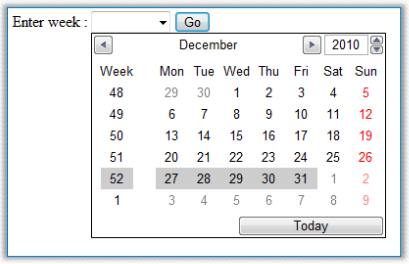
```
<input type="date"/>
<input type="datetime"/>
<input type="week"/>
<input type="time"/>
<input type="month"/>
```



Mon	Tue	Wed	Thu	Fri	Sat	Sun
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9



Mon	Tue	Wed	Thu	Fri	Sat	Sun
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9



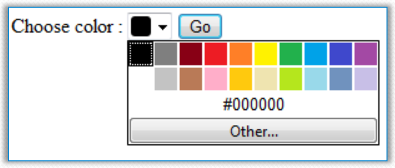
Week	Mon	Tue	Wed	Thu	Fri	Sat	Sun
48	29	30	1	2	3	4	5
49	6	7	8	9	10	11	12
50	13	14	15	16	17	18	19
51	20	21	22	23	24	25	26
52	27	28	29	30	31	1	2
1	3	4	5	6	7	8	9

HTMLForm07.htm

Color Picker

- HTML5 defines an <input> element for color.
- It helps pick a color and returns the hexadecimal representation of the chosen color.

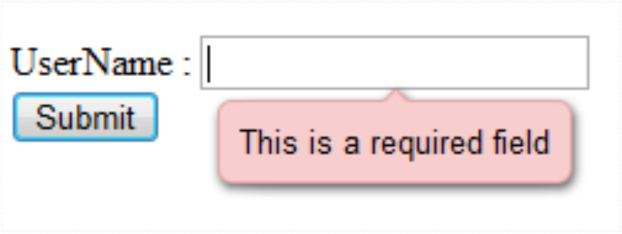
```
<input type="color"/>
```



HTMLForm08.htm

Required

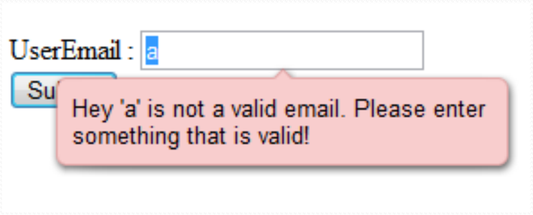
- "required" attribute in HTML5 makes a field mandatory.
- <input type="text" required>



HTMLForm10.htm

Custom Validation Messages

- HTML5 shows validation messages in line with the field which fails validation.
- Custom messages can be shown instead of standard messages using **constraint validation API**
- The validation message can be set using **`element.setCustomValidity(message)`**



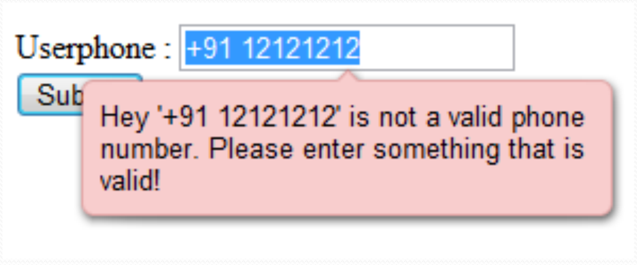
Hey 'a' is not a valid email. Please enter something that is valid!

HTMLForm11.htm

Pattern

- The **`Pattern`** attribute is used for field validation and accepts values if they match a specific format.

```
<input type="text" name="Tel"
  pattern="[+]\d{1,3} \d{3}-\d{8}" />
```



Hey '+91 12121212' is not a valid phone number. Please enter something that is valid!

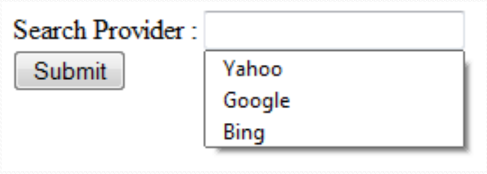
HTMLForm12.htm

DataList

- **`<datalist>`** specifies a list of pre-defined options for an **`<input>`** element.
- **`<datalist>`** can be used to provide a drop down from a text input. (auto-complete)
- The **`list`** attribute of the **`<input>`** element can be used to bind it with a **`<datalist>`** element.

DataList...

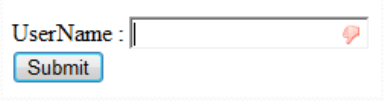
```
<input type="text" list="search_list"/>
<datalist id="search_list">
   <option value="http://www.yahoo.com"
  label="Yahoo"/>
   <option value="http://www.google.com"
  label="Google"/>
   <option value="http://www.bing.com" label="Bing"/
  >
</datalist>
```

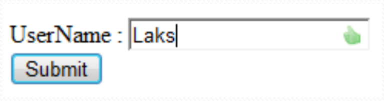


HTMLForm13.htm

Styling inputs using CSS

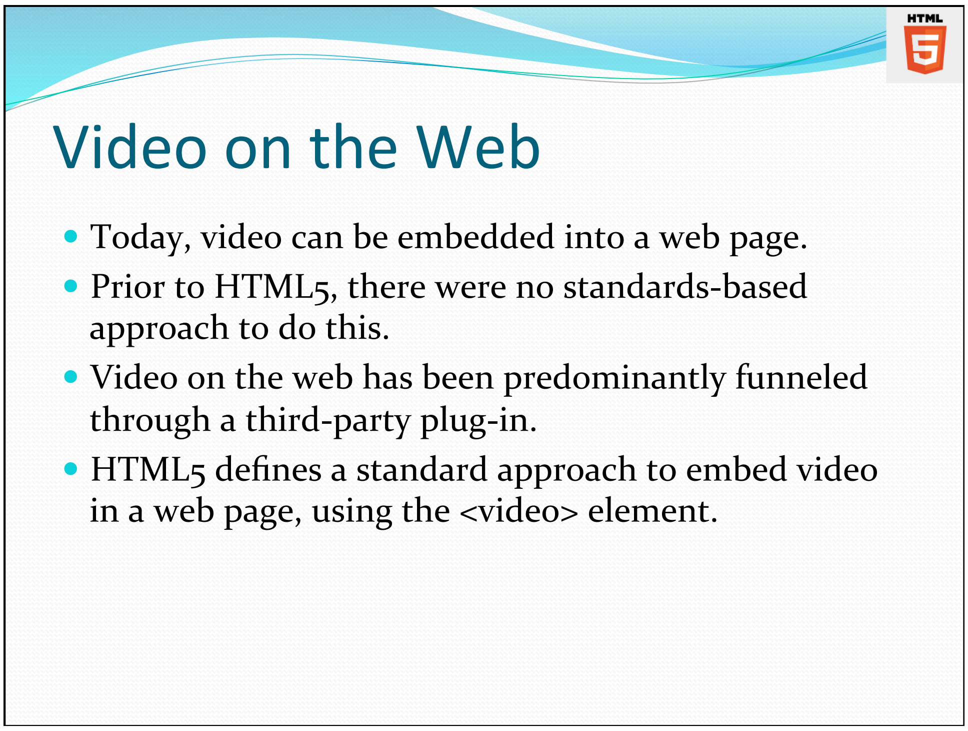
- `input:required:invalid, input:focus:invalid`
  - Apply to inputs that are required but empty or
  - To inputs that have a required format that has not yet been met
- `input:required:valid`
  - Apply to inputs that are both required and valid





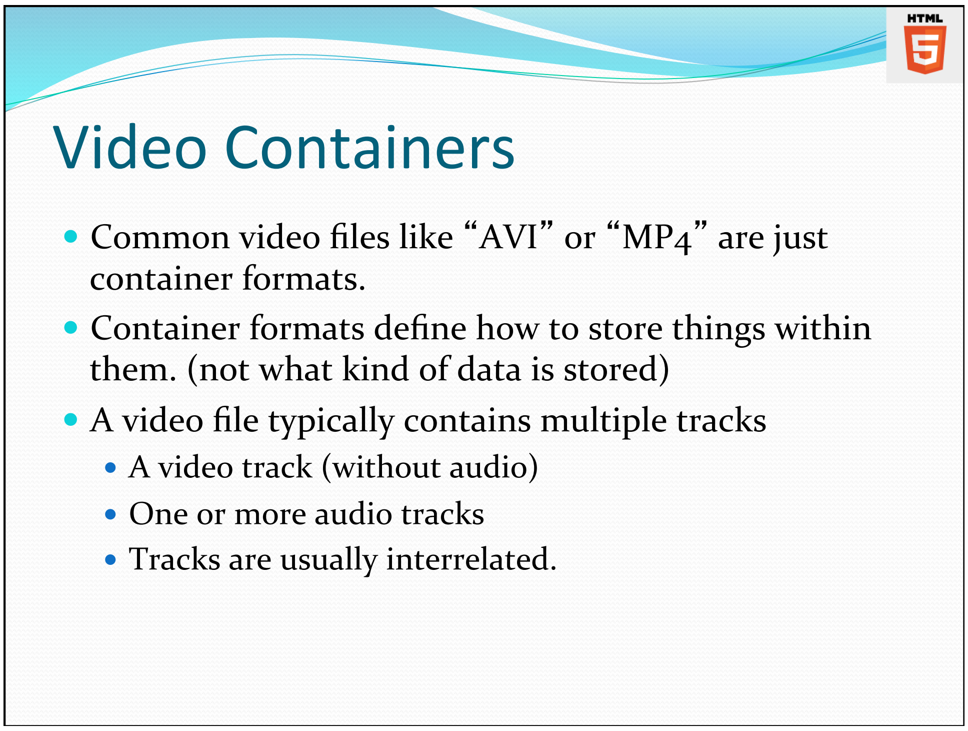
HTMLForm14.htm

Audio and Video



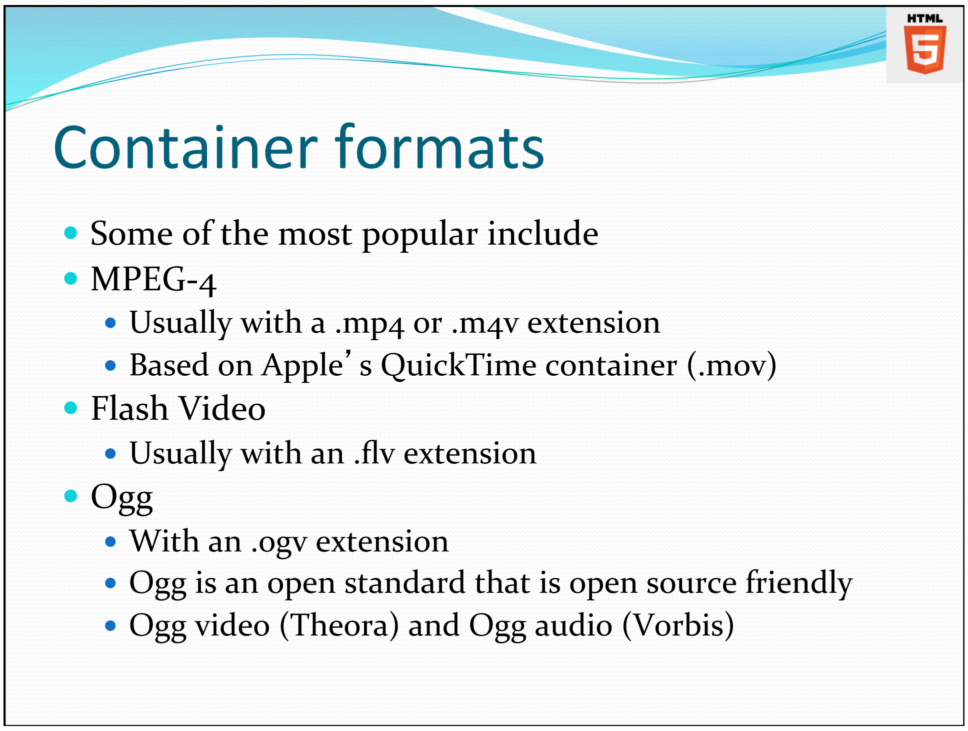
Video on the Web

- Today, video can be embedded into a web page.
- Prior to HTML5, there were no standards-based approach to do this.
- Video on the web has been predominantly funneled through a third-party plug-in.
- HTML5 defines a standard approach to embed video in a web page, using the <video> element.



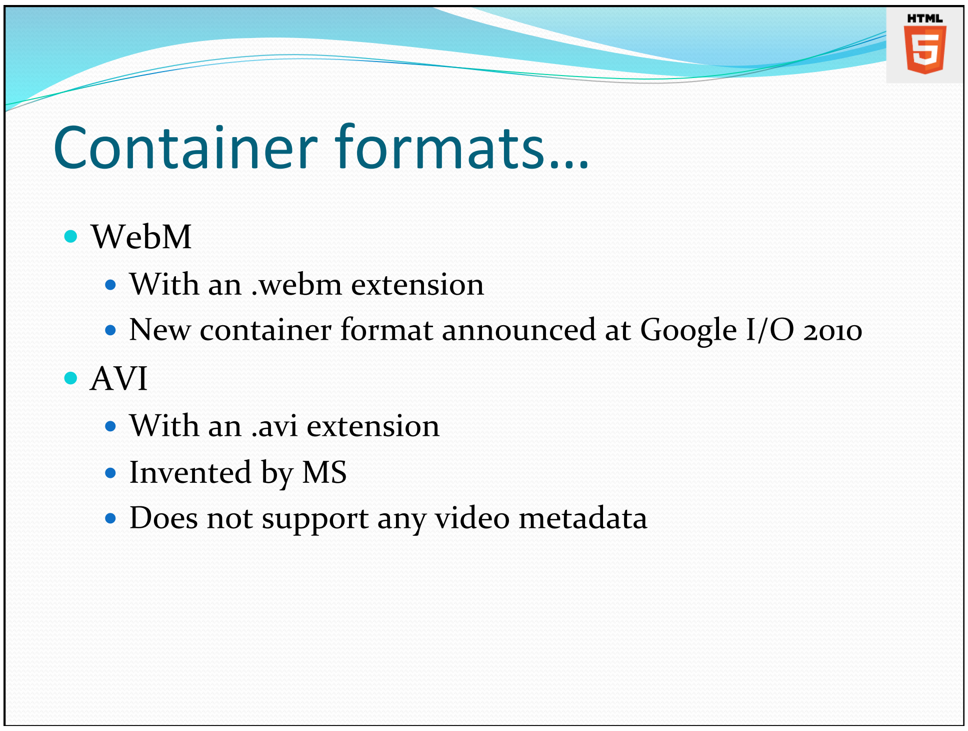
Video Containers

- Common video files like “AVI” or “MP4” are just container formats.
- Container formats define how to store things within them. (not what kind of data is stored)
- A video file typically contains multiple tracks
  - A video track (without audio)
  - One or more audio tracks
  - Tracks are usually interrelated.



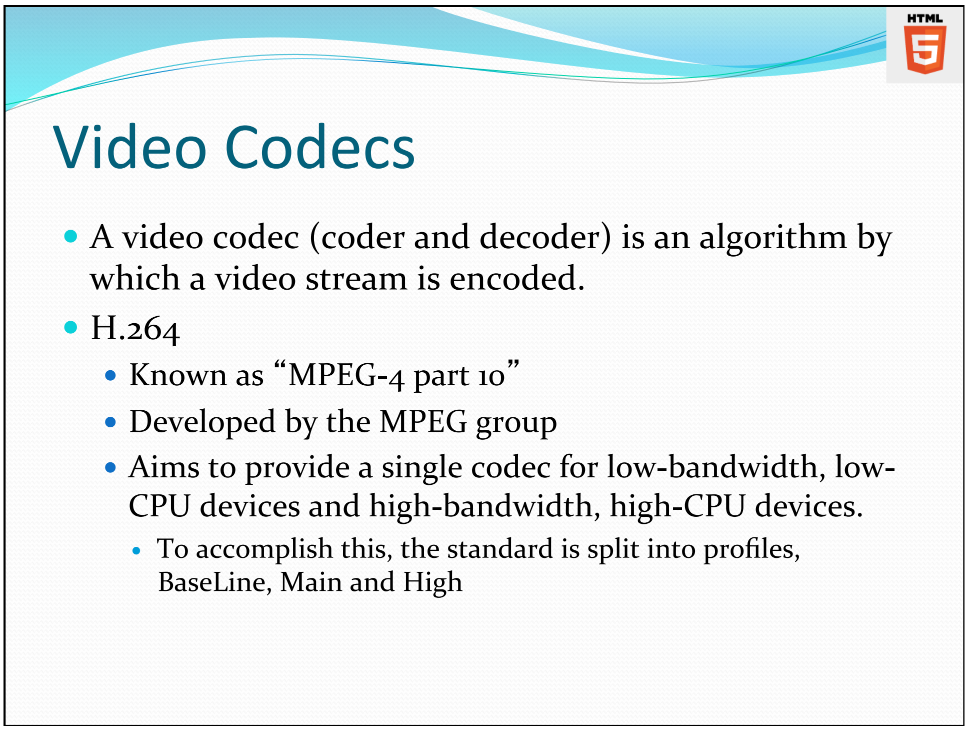
Container formats

- Some of the most popular include
- MPEG-4
  - Usually with a .mp4 or .m4v extension
  - Based on Apple' s QuickTime container (.mov)
- Flash Video
  - Usually with an .flv extension
- Ogg
  - With an .ogv extension
  - Ogg is an open standard that is open source friendly
  - Ogg video (Theora) and Ogg audio (Vorbis)



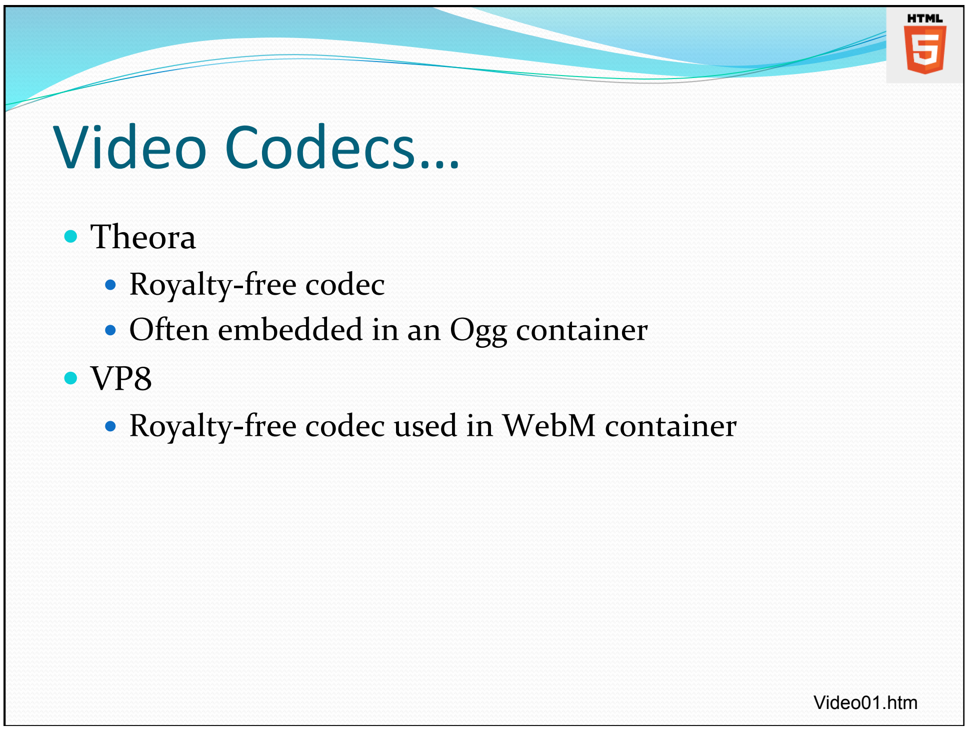
Container formats...

- WebM
  - With an .webm extension
  - New container format announced at Google I/O 2010
- AVI
  - With an .avi extension
  - Invented by MS
  - Does not support any video metadata



Video Codecs

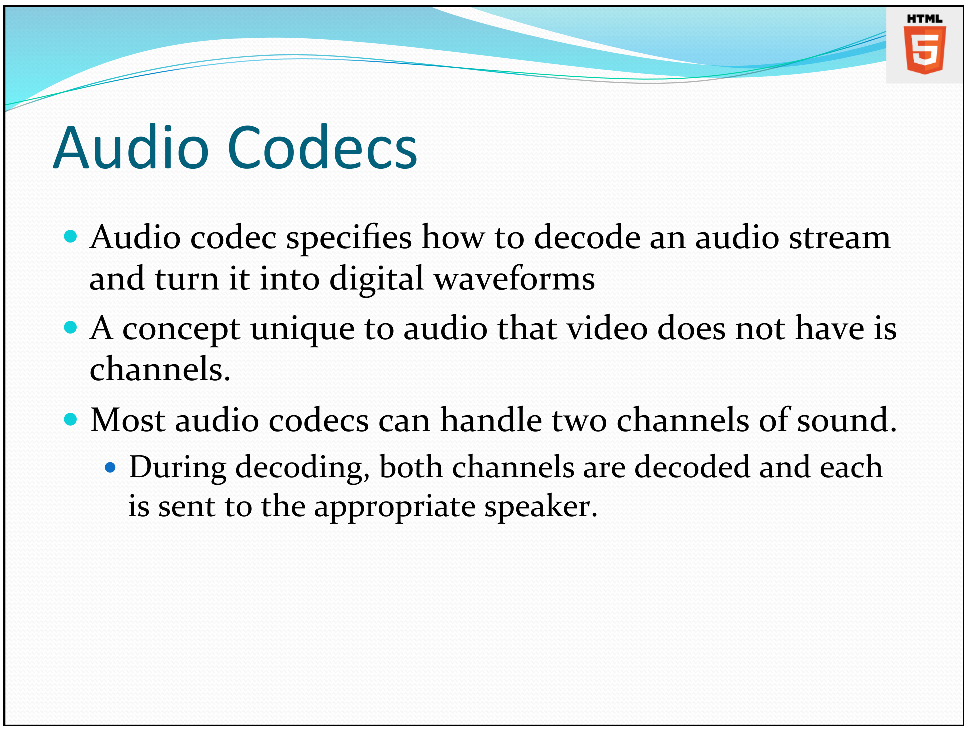
- A video codec (coder and decoder) is an algorithm by which a video stream is encoded.
- H.264
  - Known as “MPEG-4 part 10”
  - Developed by the MPEG group
  - Aims to provide a single codec for low-bandwidth, low-CPU devices and high-bandwidth, high-CPU devices.
    - To accomplish this, the standard is split into profiles, BaseLine, Main and High



Video Codecs…

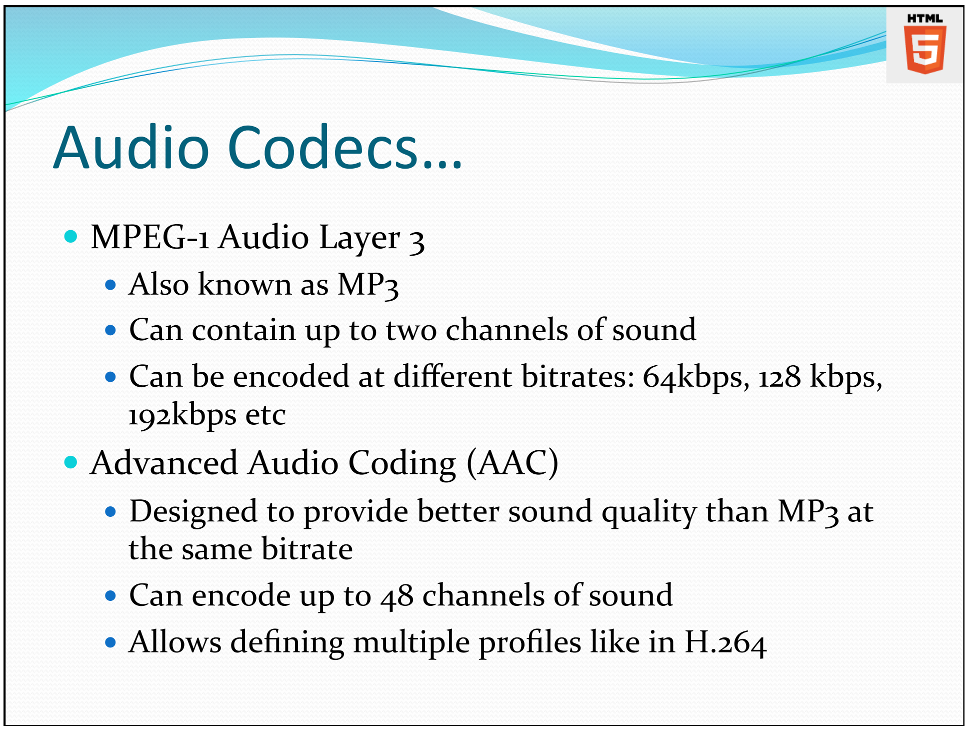
- Theora
  - Royalty-free codec
  - Often embedded in an Ogg container
- VP8
  - Royalty-free codec used in WebM container

Video01.htm



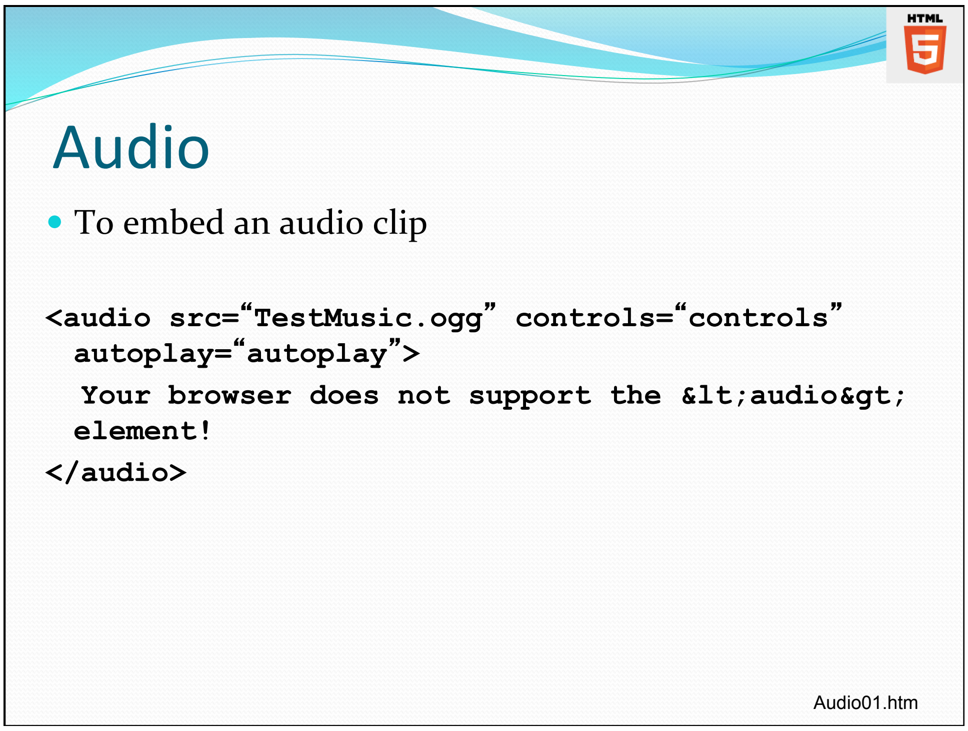
Audio Codecs

- Audio codec specifies how to decode an audio stream and turn it into digital waveforms
- A concept unique to audio that video does not have is channels.
- Most audio codecs can handle two channels of sound.
  - During decoding, both channels are decoded and each is sent to the appropriate speaker.



Audio Codecs…

- MPEG-1 Audio Layer 3
  - Also known as MP3
  - Can contain up to two channels of sound
  - Can be encoded at different bitrates: 64kbps, 128 kbps, 192kbps etc
- Advanced Audio Coding (AAC)
  - Designed to provide better sound quality than MP3 at the same bitrate
  - Can encode up to 48 channels of sound
  - Allows defining multiple profiles like in H.264

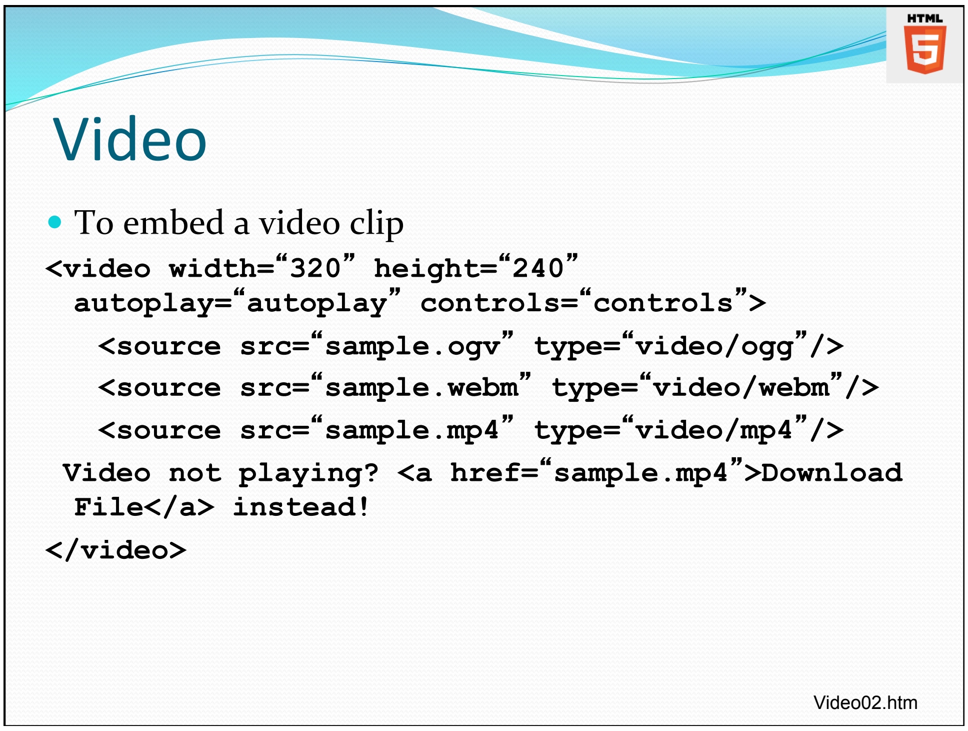


Audio

- To embed an audio clip

```
<audio src=“TestMusic.ogg” controls=“controls”
  autoplay=“autoplay”>
   Your browser does not support the &lt;audio&gt;
  element!
</audio>
```

Audio01.htm



Video

- To embed a video clip

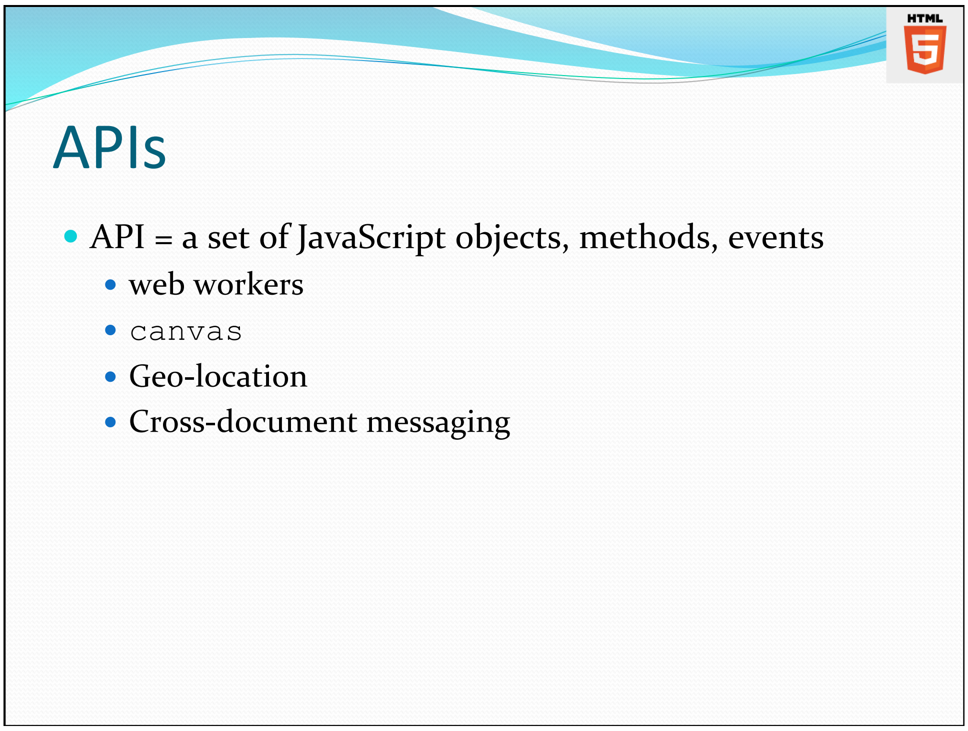
```
<video width=“320” height=“240”
  autoplay=“autoplay” controls=“controls”>
    <source src=“sample.ogv” type=“video/ogg”/>
    <source src=“sample.webm” type=“video/webm”/>
    <source src=“sample.mp4” type=“video/mp4”/>
 Video not playing? <a href=“sample.mp4”>Download
  File</a> instead!
</video>
```

Video02.htm

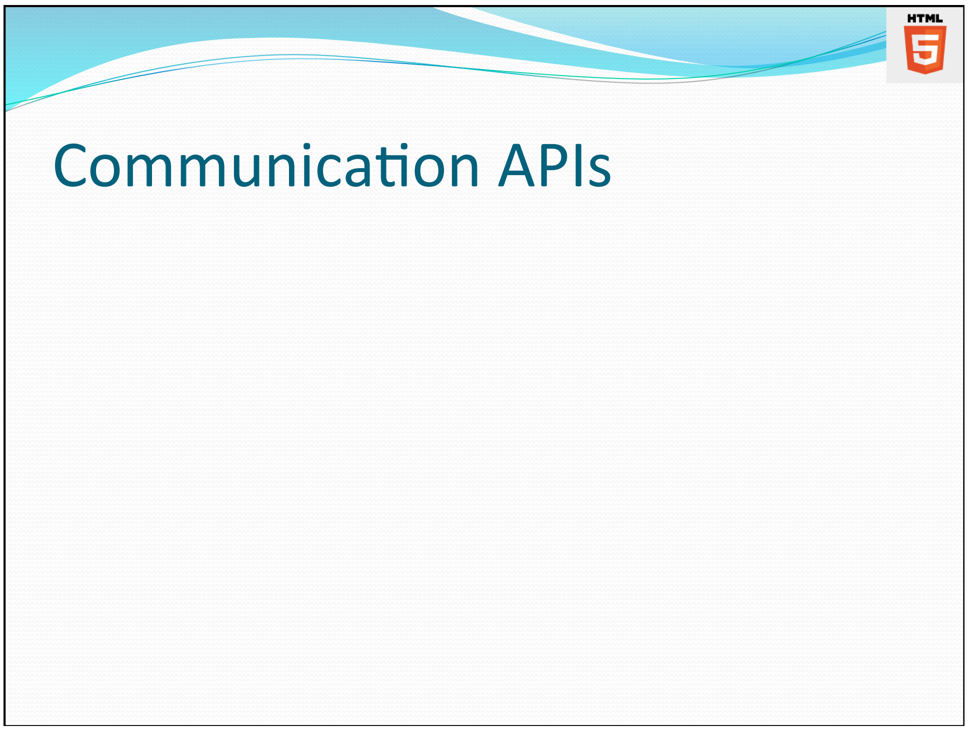
Programming HTML5

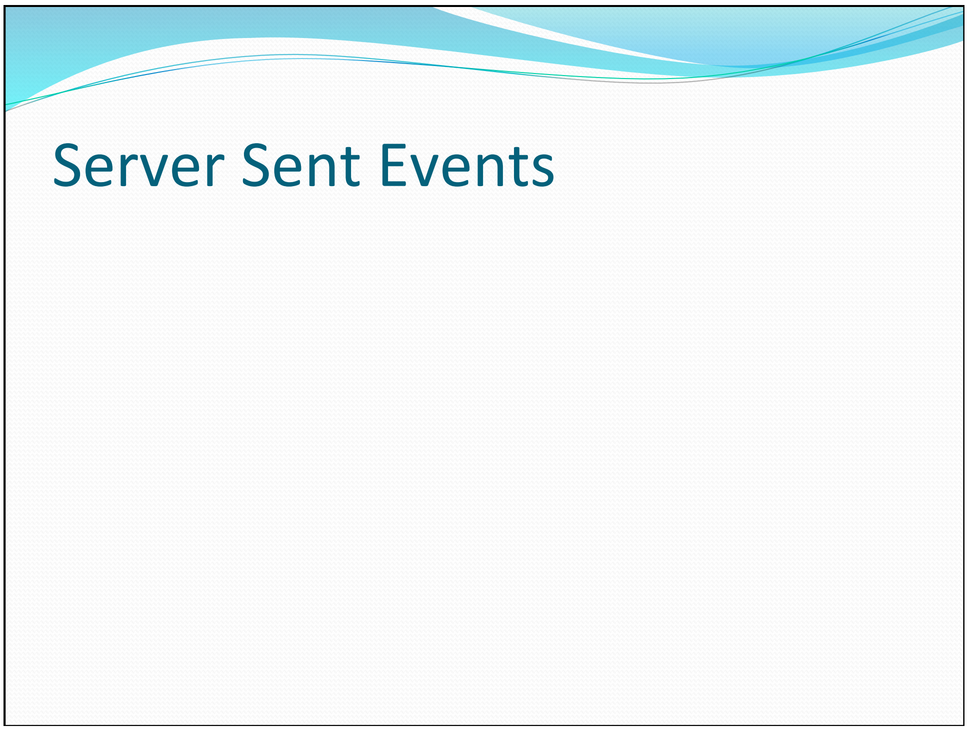
Overview of HTML5 APIs

- In the past there has been a clear distinction between HTML and DOM API.
- Now, DOM specification is part of HTML5.
- HTML5 provides "more power to the browser as a programming platform"
- With the growing demand for interactive content on web pages, HTML5 provides several APIs

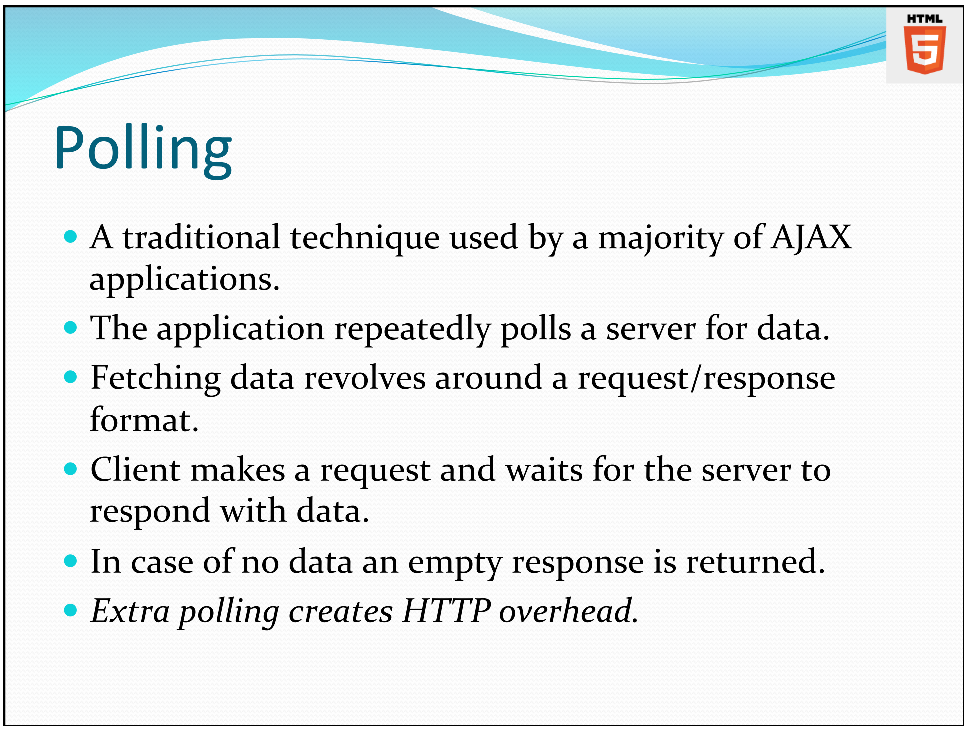

APIs

- API = a set of JavaScript objects, methods, events
  - web workers
  - `canvas`
  - Geo-location
  - Cross-document messaging


Communication APIs

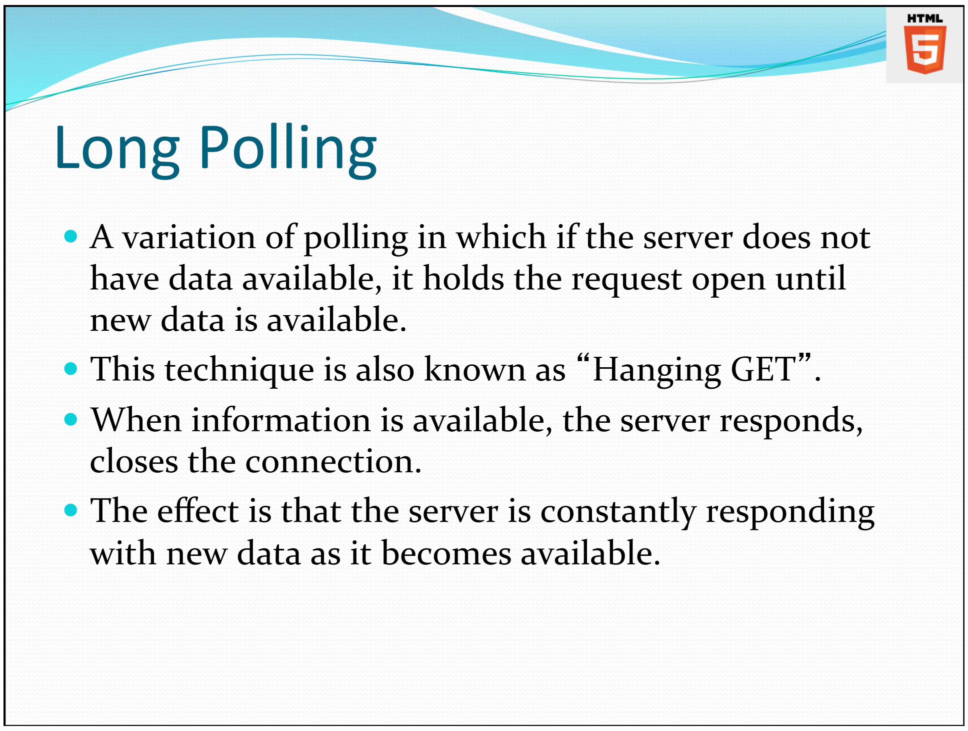


Server Sent Events



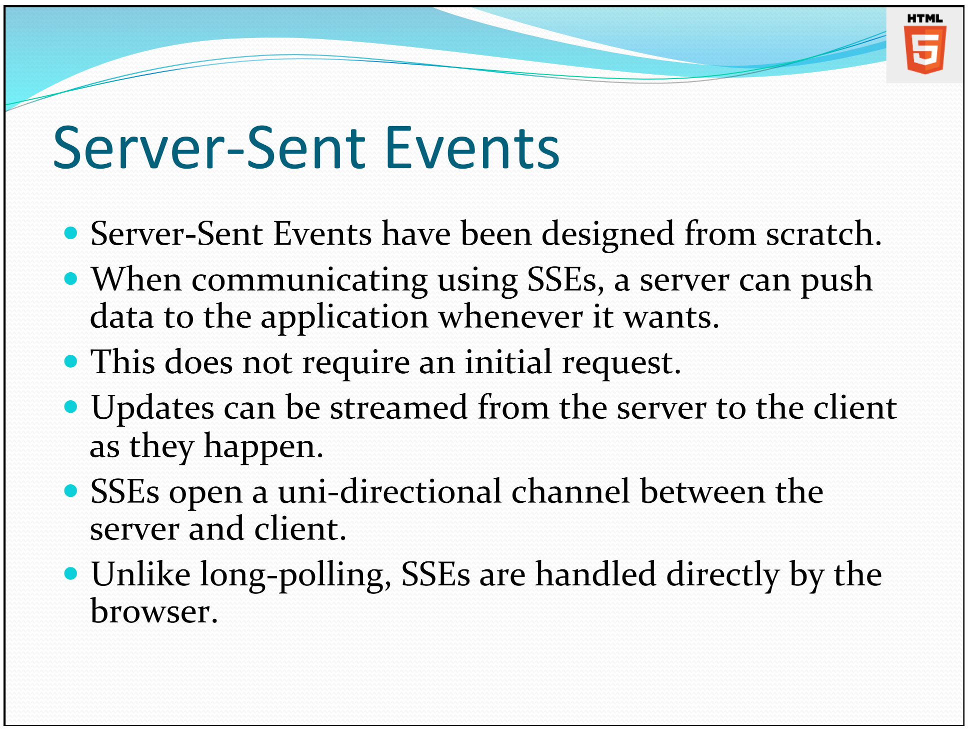
Polling

- A traditional technique used by a majority of AJAX applications.
- The application repeatedly polls a server for data.
- Fetching data revolves around a request/response format.
- Client makes a request and waits for the server to respond with data.
- In case of no data an empty response is returned.
- *Extra polling creates HTTP overhead.*



Long Polling

- A variation of polling in which if the server does not have data available, it holds the request open until new data is available.
- This technique is also known as “Hanging GET”.
- When information is available, the server responds, closes the connection.
- The effect is that the server is constantly responding with new data as it becomes available.



Server-Sent Events

- Server-Sent Events have been designed from scratch.
- When communicating using SSEs, a server can push data to the application whenever it wants.
- This does not require an initial request.
- Updates can be streamed from the server to the client as they happen.
- SSEs open a uni-directional channel between the server and client.
- Unlike long-polling, SSEs are handled directly by the browser.

The API

- To subscribe to a new event stream, start by creating a new EventSource object and pass in the entrypoint

```
var source = new
  EventSource(“myEvents.php”);
```

- The referenced URL must be on the same origin (scheme, domain and port) as the page in which the object is created.

The API…

- The **`EventSource`** instance has a **`readyState`** property with values
  - 0 : indicates it is connecting to the server
  - 1 : indicates an open connection
  - 2 : indicates a closed connection
- Three events are associated with the **`EventSource`**
  - **`open`** : fired when the connection is established
  - **`message`** : fired when a new event is received from the server
  - **`error`** : fired when no connection can be made

The API…

```
source.addEventListener(“message”,getData,false);

 function getData(e)
 {
    var data = e.data;
 }
```

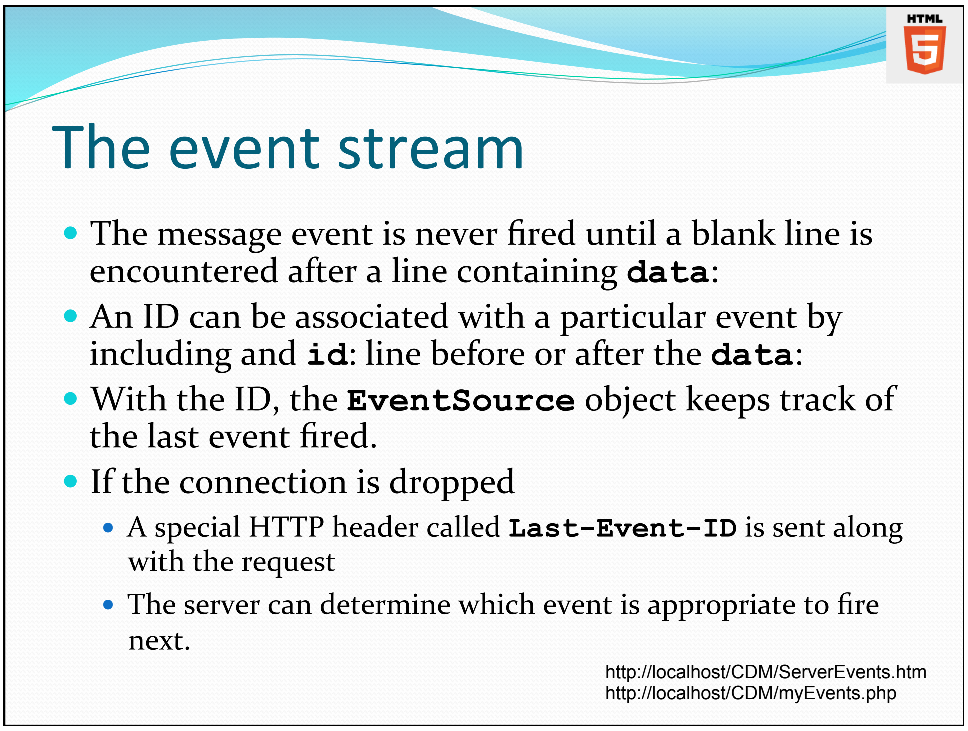
- Information sent back from the server is returned via **`event.data`** as a string.
- The **`EventSource`** object will attempt to keep the connection alive with the server.
- The object can be forced to disconnect immediately by calling the **`close`**() method.

```
source.close();
```

The event stream

- Server events are sent along a long-lasting HTTP request with a MIME type of **`text/event-stream`**
- The format of the response is plain text.
- It is made up of the prefix **`data`**: followed by text.
- When there are two or more consecutive lines beginning with data:
  - it is interpreted as a multiline piece of data
  - The values are concatenated with a newline character.

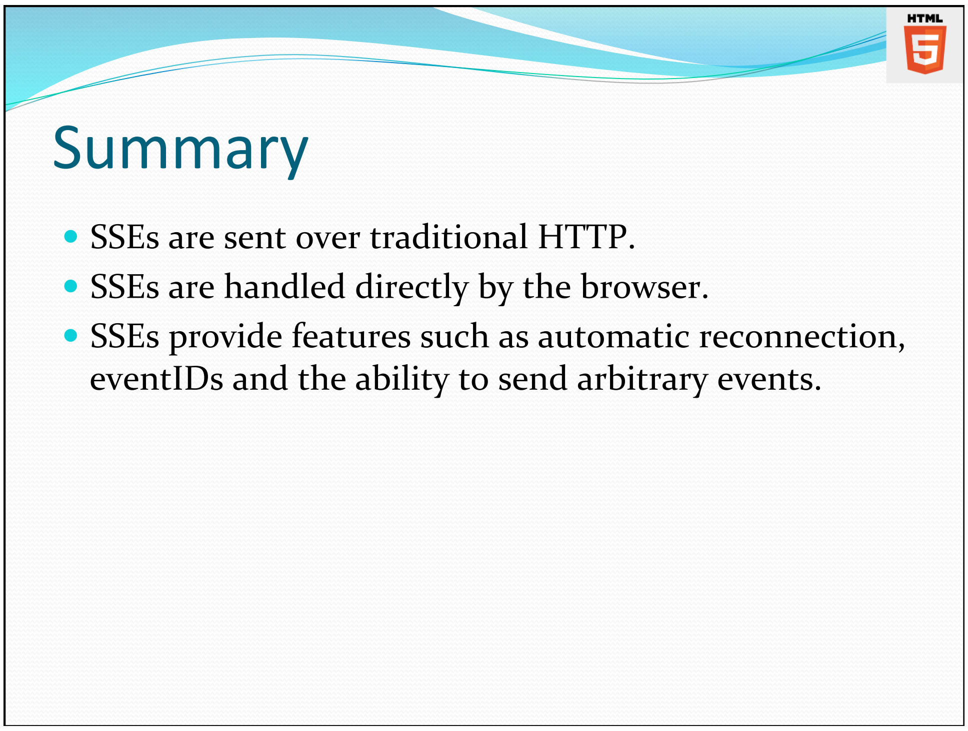
```
data: sometext
data: somemoretext
```



The event stream

- The message event is never fired until a blank line is encountered after a line containing **`data`**:
- An ID can be associated with a particular event by including and **`id`**: line before or after the **`data`**:
- With the ID, the **`EventSource`** object keeps track of the last event fired.
- If the connection is dropped
  - A special HTTP header called **`Last-Event-ID`** is sent along with the request
  - The server can determine which event is appropriate to fire next.

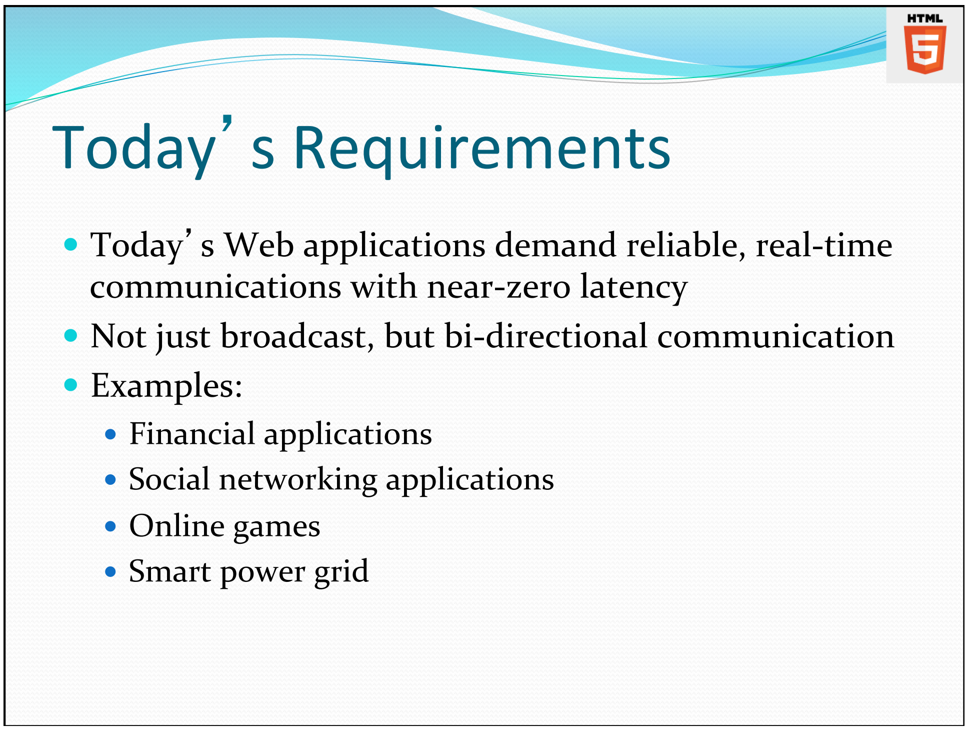
http://localhost/CDM/ServerEvents.htm
http://localhost/CDM/myEvents.php



Summary

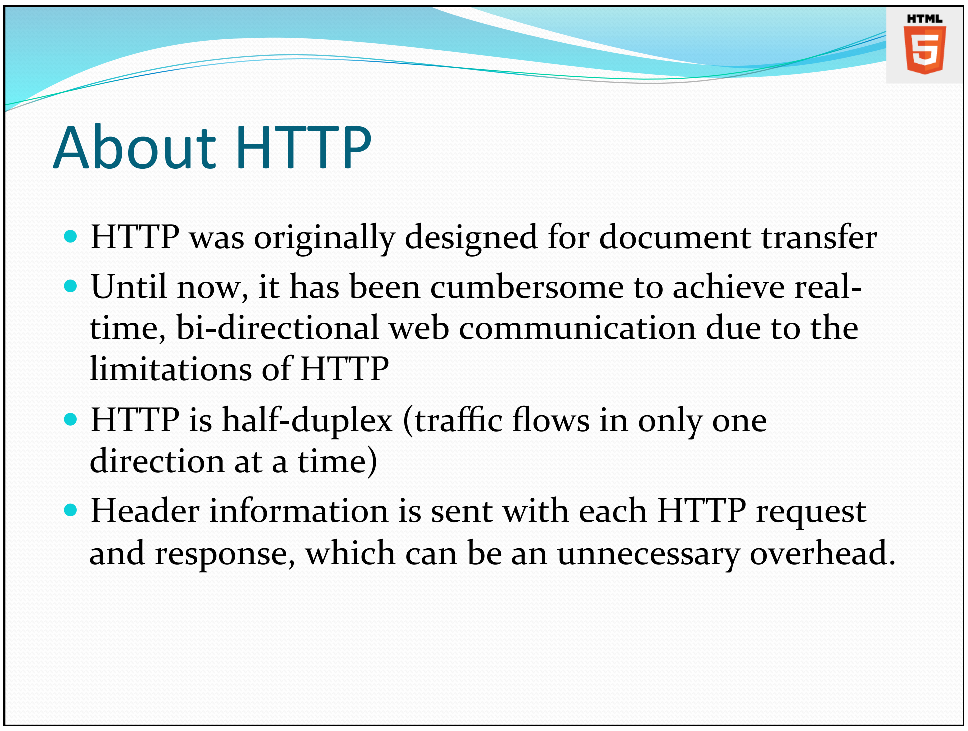
- SSEs are sent over traditional HTTP.
- SSEs are handled directly by the browser.
- SSEs provide features such as automatic reconnection, eventIDs and the ability to send arbitrary events.

HTML5 Web Sockets



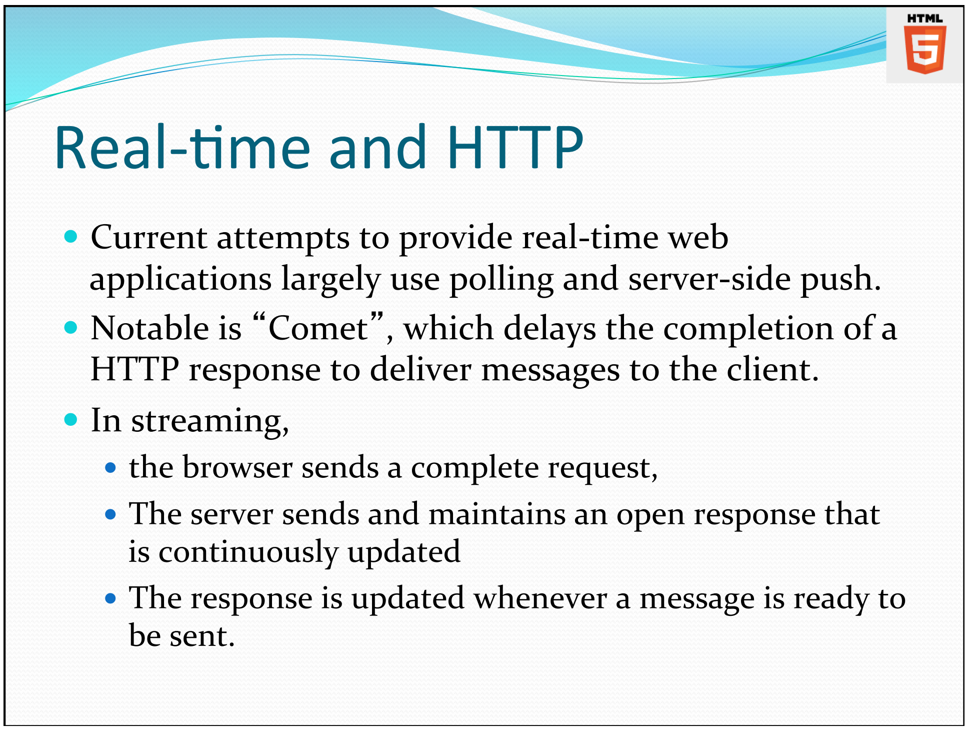
Today’s Requirements

- Today’s Web applications demand reliable, real-time communications with near-zero latency
- Not just broadcast, but bi-directional communication
- Examples:
  - Financial applications
  - Social networking applications
  - Online games
  - Smart power grid



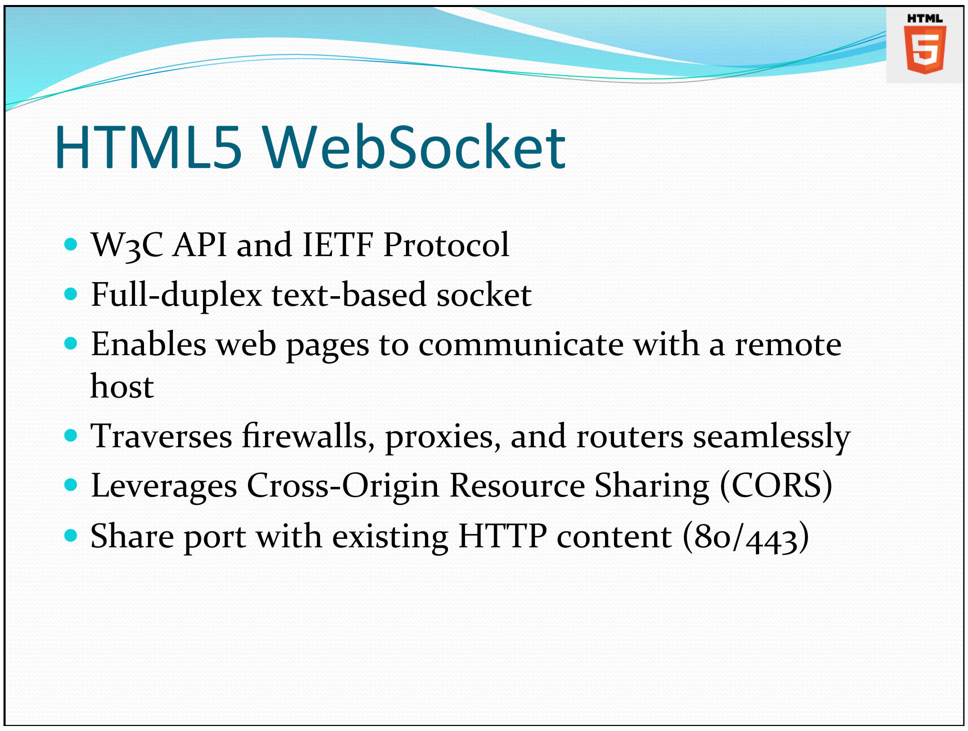
About HTTP

- HTTP was originally designed for document transfer
- Until now, it has been cumbersome to achieve real-time, bi-directional web communication due to the limitations of HTTP
- HTTP is half-duplex (traffic flows in only one direction at a time)
- Header information is sent with each HTTP request and response, which can be an unnecessary overhead.



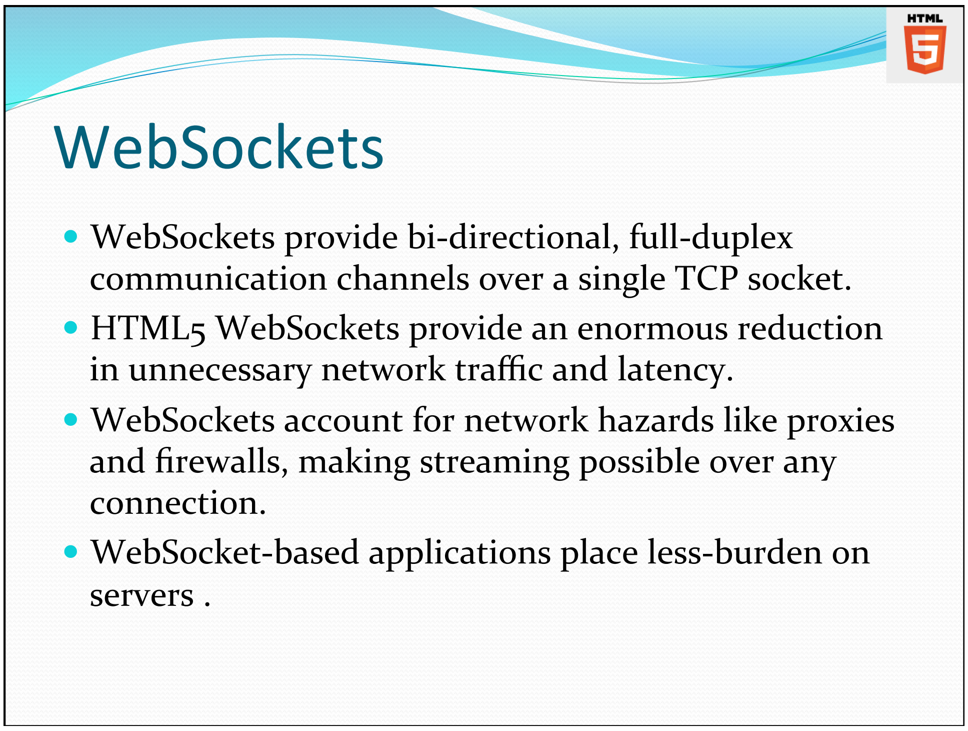
Real-time and HTTP

- Current attempts to provide real-time web applications largely use polling and server-side push.
- Notable is "Comet", which delays the completion of a HTTP response to deliver messages to the client.
- In streaming,
  - the browser sends a complete request,
  - The server sends and maintains an open response that is continuously updated
  - The response is updated whenever a message is ready to be sent.



HTML5 WebSocket

- W3C API and IETF Protocol
- Full-duplex text-based socket
- Enables web pages to communicate with a remote host
- Traverses firewalls, proxies, and routers seamlessly
- Leverages Cross-Origin Resource Sharing (CORS)
- Share port with existing HTTP content (80/443)



WebSockets

- WebSockets provide bi-directional, full-duplex communication channels over a single TCP socket.
- HTML5 WebSockets provide an enormous reduction in unnecessary network traffic and latency.
- WebSockets account for network hazards like proxies and firewalls, making streaming possible over any connection.
- WebSocket-based applications place less-burden on servers .

Basic WebSocket-based architecture

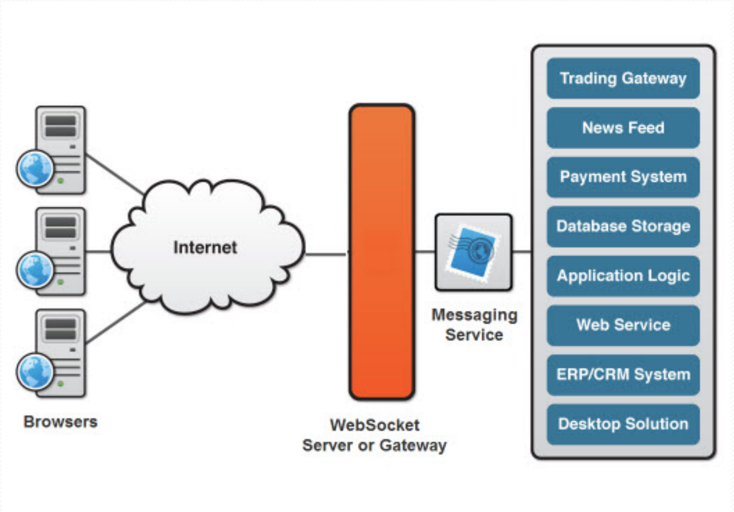
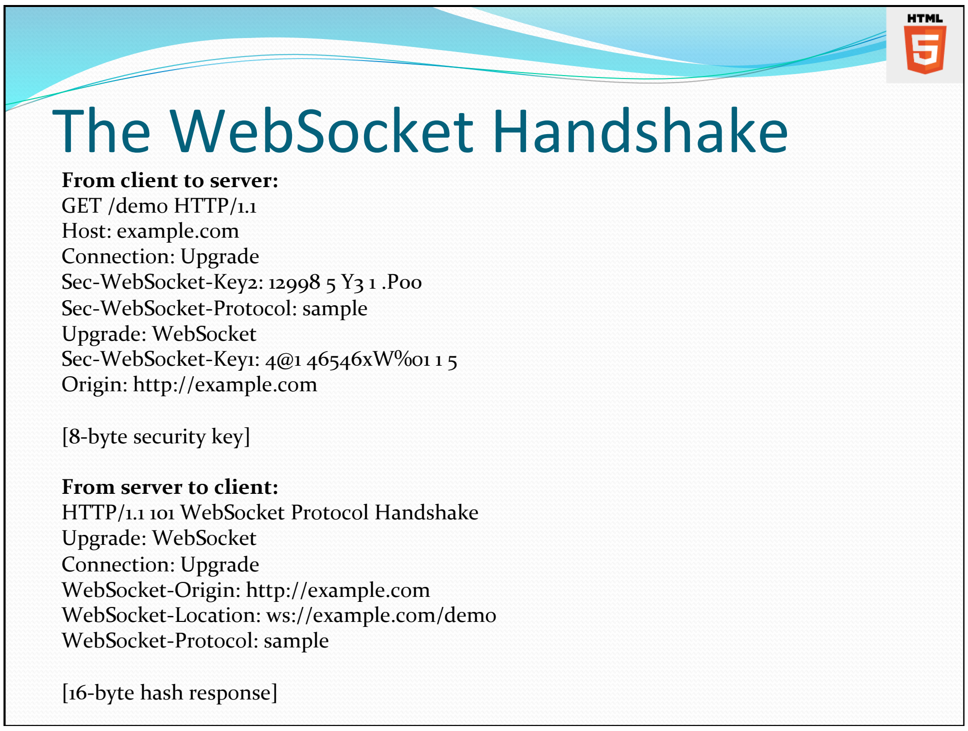


Image Courtesy : http://websocket.org/

The WebSocket Handshake

- To establish a WebSocket connection, the client and the server upgrade from HTTP to the WebSocket protocol during their initial handshake.
- This process automatically sets up a tunnel through to the server
- Once established, the WebSocket is a full duplex channel between the client and the server.



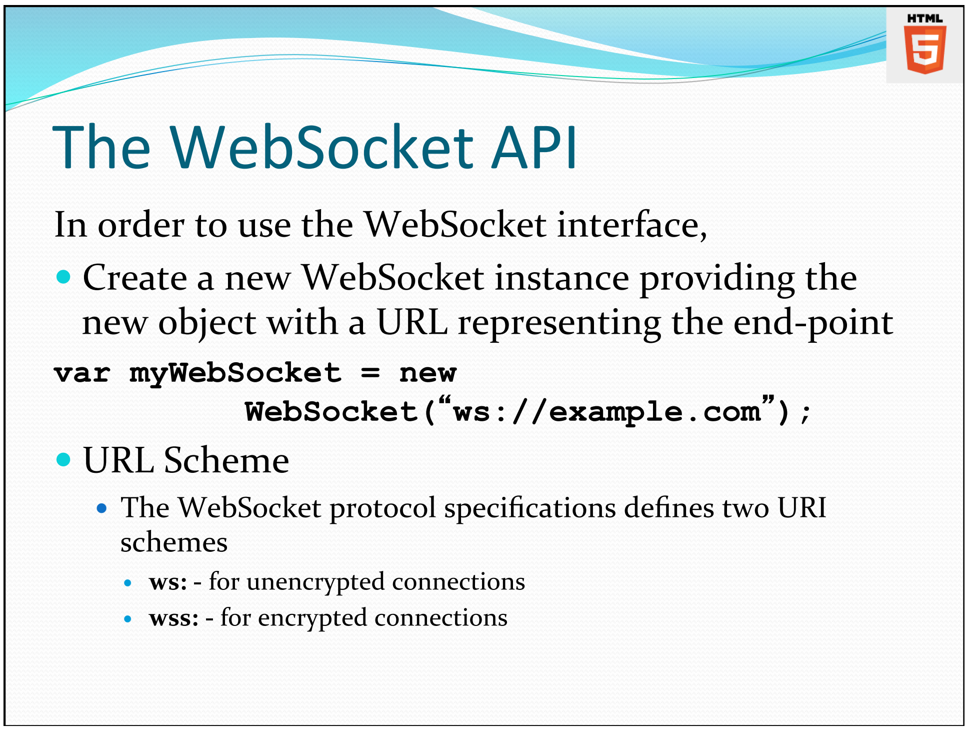
The WebSocket Handshake

From client to server:
GET /demo HTTP/1.1
Host: example.com
Connection: Upgrade
Sec-WebSocket-Key2: 12998 5 Y3 1 .P00
Sec-WebSocket-Protocol: sample
Upgrade: WebSocket
Sec-WebSocket-Key1: 4@1 46546xW%0l 1 5
Origin: http://example.com

[8-byte security key]

From server to client:
HTTP/1.1 101 WebSocket Protocol Handshake
Upgrade: WebSocket
Connection: Upgrade
WebSocket-Origin: http://example.com
WebSocket-Location: ws://example.com/demo
WebSocket-Protocol: sample

[16-byte hash response]



The WebSocket API

In order to use the WebSocket interface,

- Create a new WebSocket instance providing the new object with a URL representing the end-point

```
var myWebSocket = new
          WebSocket("ws://example.com");
```

- URL Scheme
  - The WebSocket protocol specifications defines two URI schemes
    - **ws:** - for unencrypted connections
    - **wss:** - for encrypted connections

Check for Browser Support

- In order to use the HTML5 WebSocket API, browser support needs to be ensured.

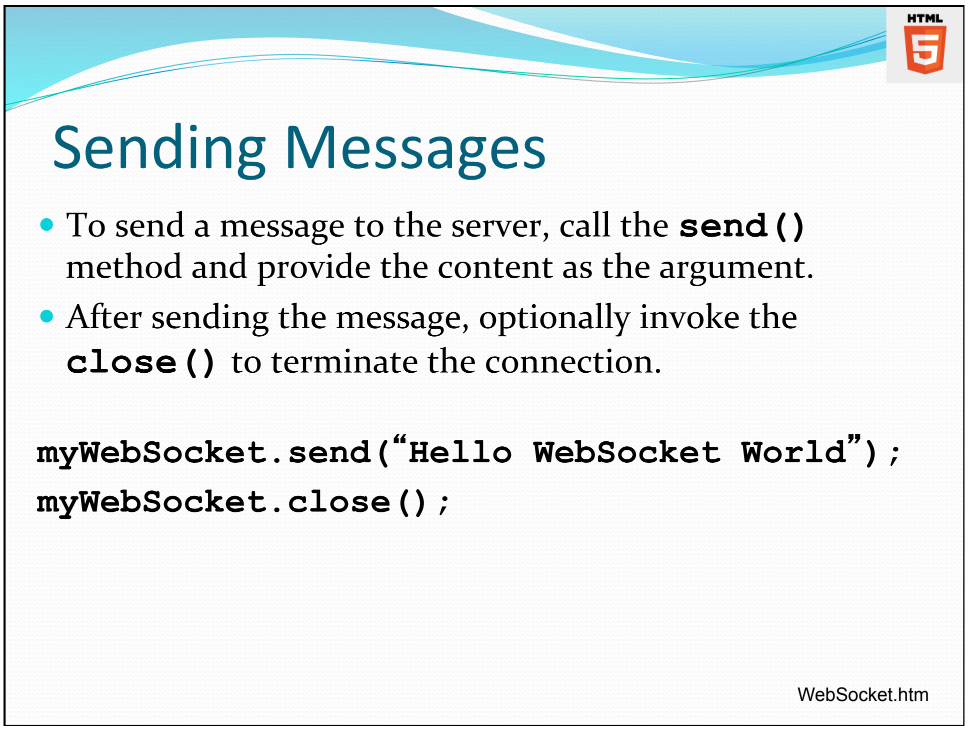
```
if(window.WebSocket)
{
“supported”
}
else
{
“Not Supported”
}
```

CheckWebSocket-Support.htm

WebSocket API…

- Once a connection is established WebSocket data frames can be sent back and forth between the client and server.
- Before connecting to an end-point and sending a message, event listeners can be associated to handle each phase

```
myWebSocket.addEventListener(“open”,openConn,false);
myWebSocket.addEventListener(“message”,getData,false);
myWebSocket.addEventListener(“close”,closeConn,false);
```

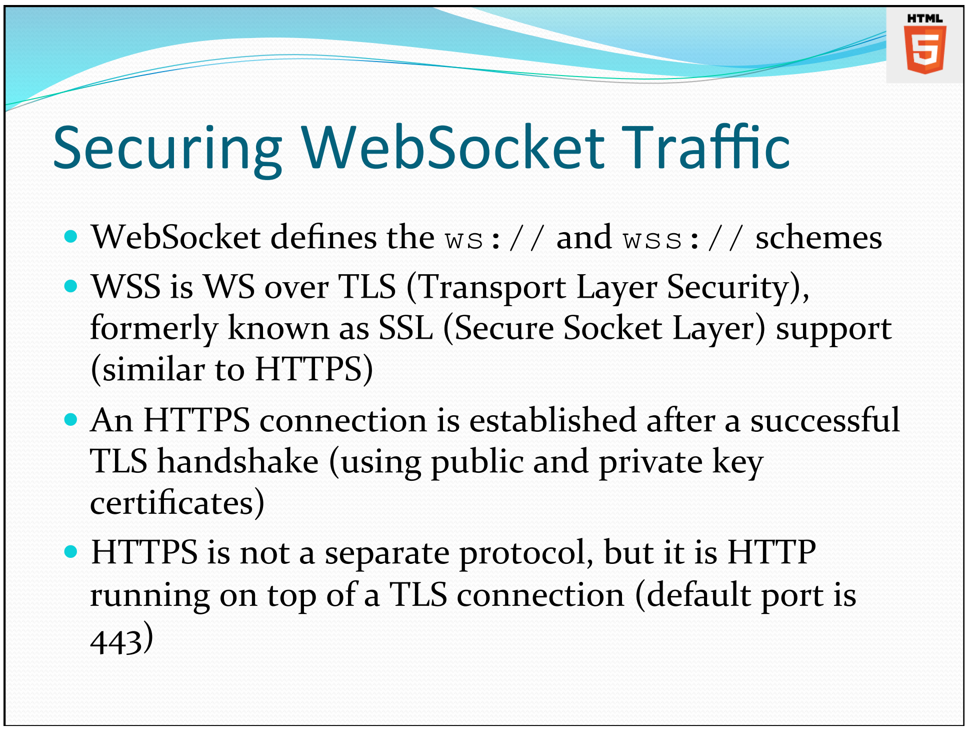


Sending Messages

- To send a message to the server, call the **`send()`** method and provide the content as the argument.
- After sending the message, optionally invoke the **`close()`** to terminate the connection.

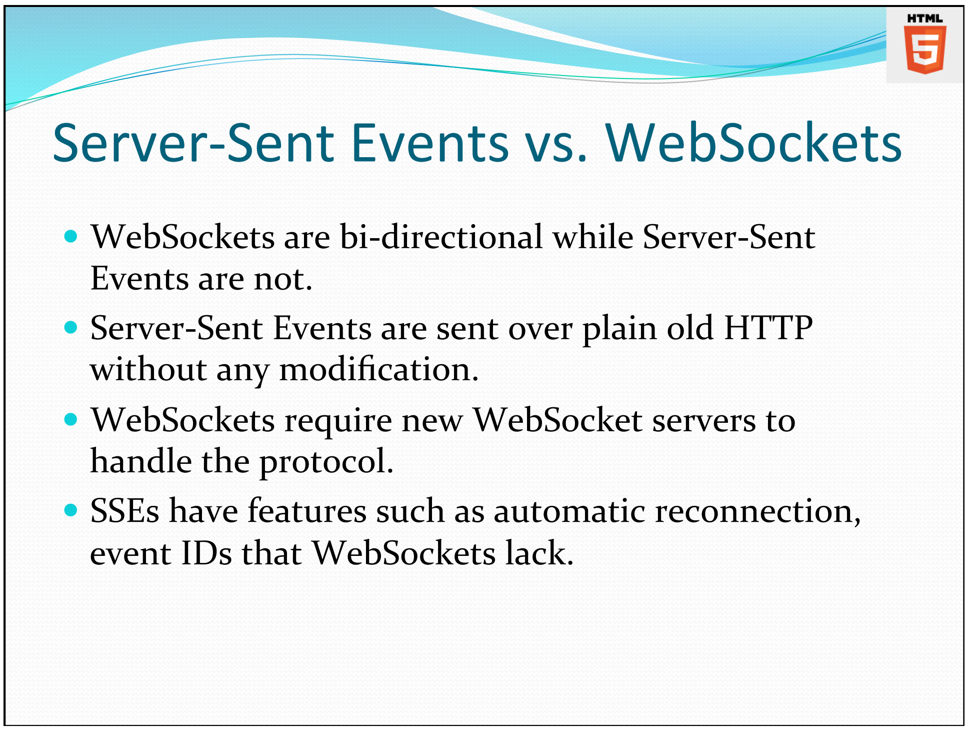
```
myWebSocket.send("Hello WebSocket World");
myWebSocket.close();
```

WebSocket.htm



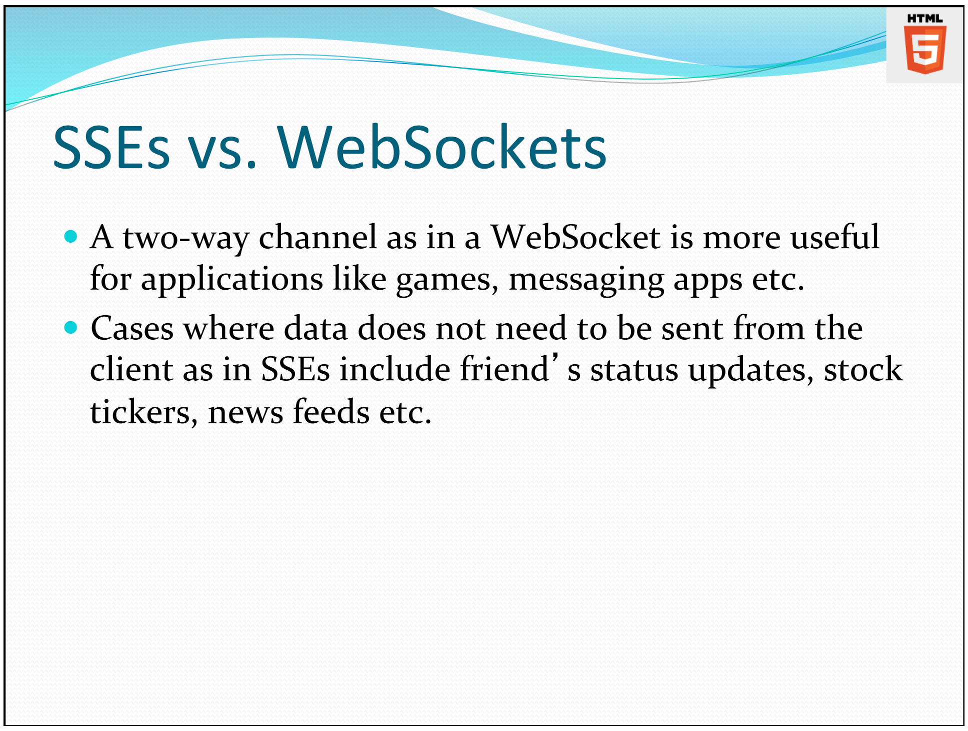
Securing WebSocket Traffic

- WebSocket defines the `ws://` and `wss://` schemes
- WSS is WS over TLS (Transport Layer Security), formerly known as SSL (Secure Socket Layer) support (similar to HTTPS)
- An HTTPS connection is established after a successful TLS handshake (using public and private key certificates)
- HTTPS is not a separate protocol, but it is HTTP running on top of a TLS connection (default port is 443)



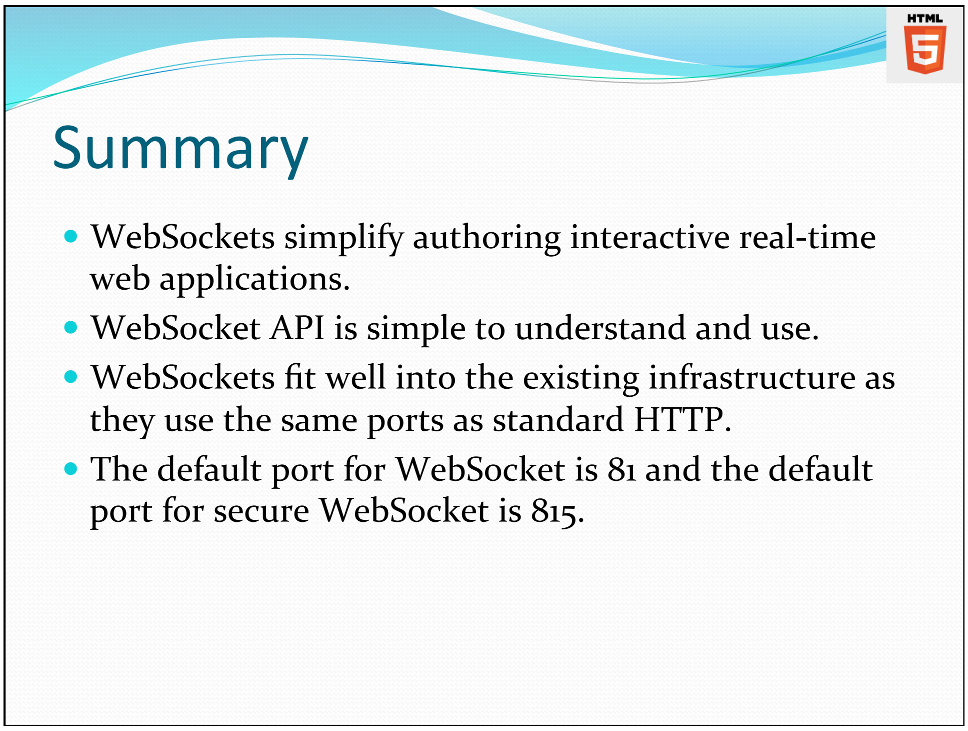
Server-Sent Events vs. WebSockets

- WebSockets are bi-directional while Server-Sent Events are not.
- Server-Sent Events are sent over plain old HTTP without any modification.
- WebSockets require new WebSocket servers to handle the protocol.
- SSEs have features such as automatic reconnection, event IDs that WebSockets lack.



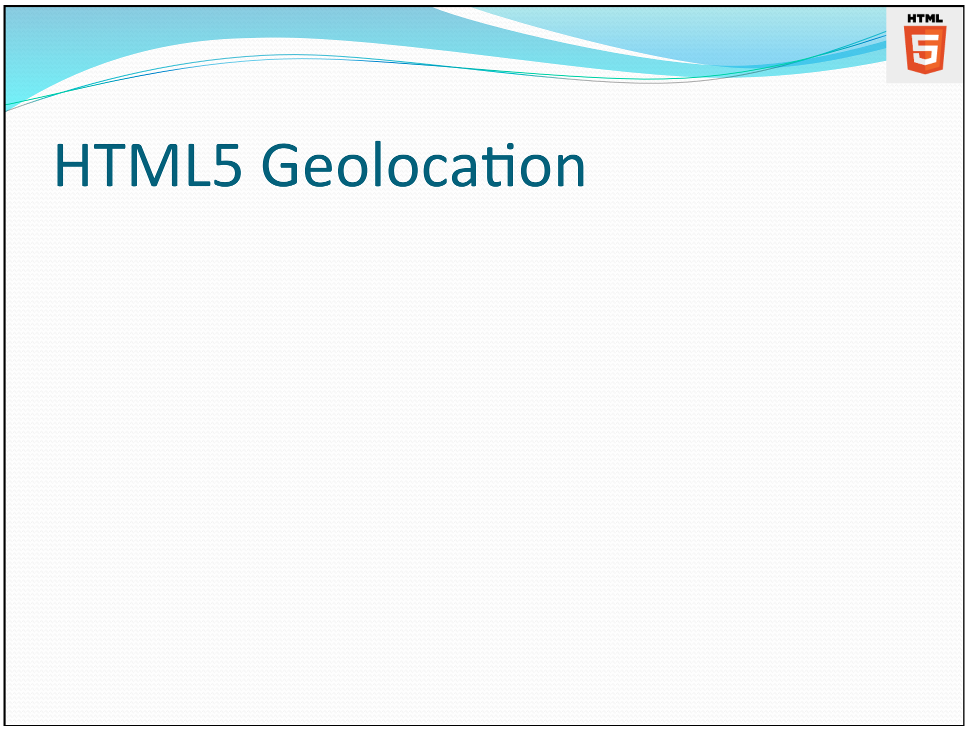
SSEs vs. WebSockets

- A two-way channel as in a WebSocket is more useful for applications like games, messaging apps etc.
- Cases where data does not need to be sent from the client as in SSEs include friend' s status updates, stock tickers, news feeds etc.

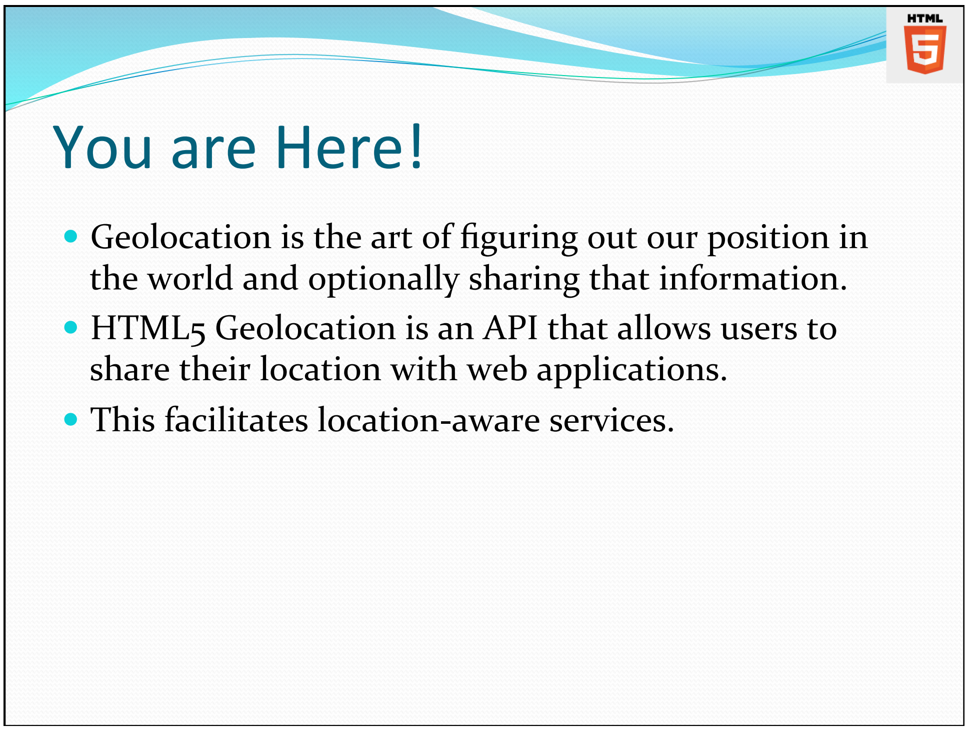


Summary

- WebSockets simplify authoring interactive real-time web applications.
- WebSocket API is simple to understand and use.
- WebSockets fit well into the existing infrastructure as they use the same ports as standard HTTP.
- The default port for WebSocket is 81 and the default port for secure WebSocket is 815.

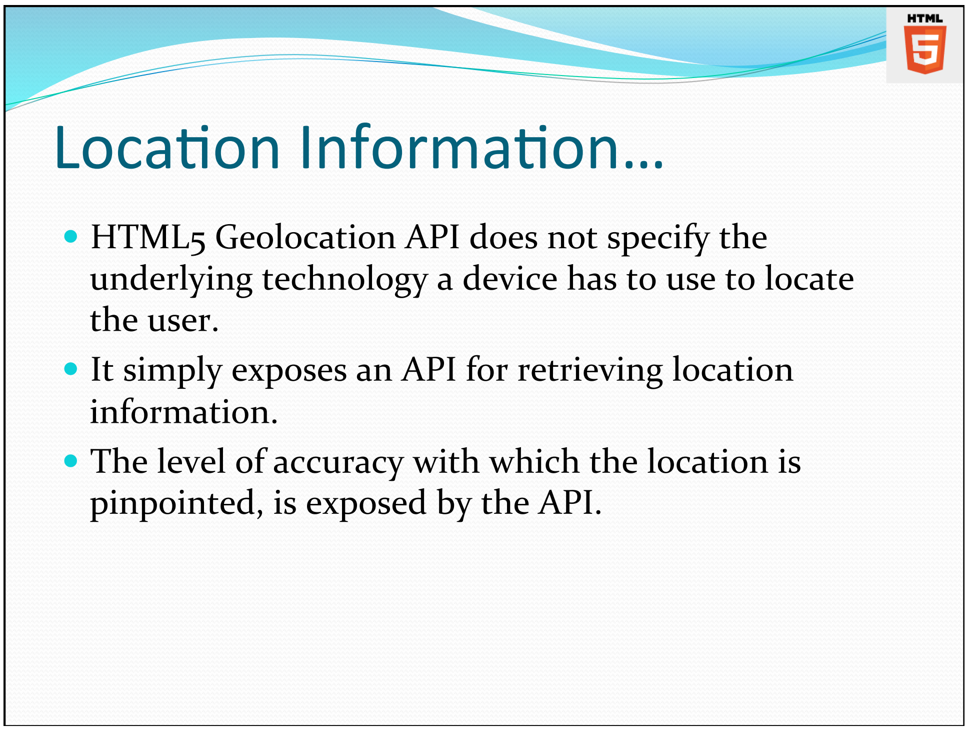


HTML5 Geolocation



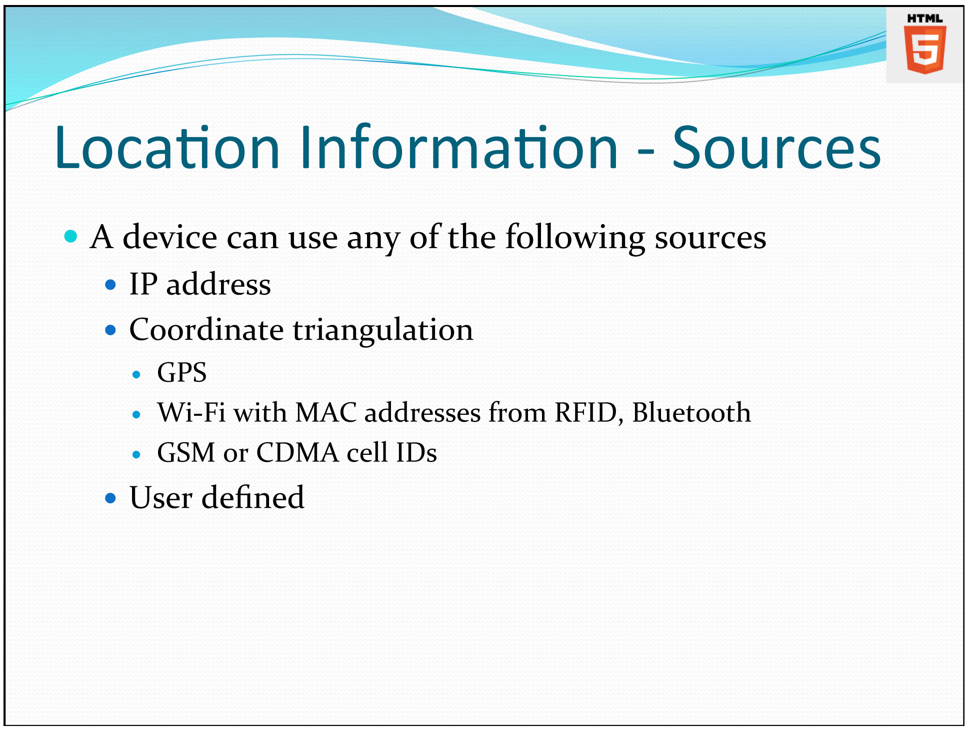
You are Here!

- Geolocation is the art of figuring out our position in the world and optionally sharing that information.
- HTML5 Geolocation is an API that allows users to share their location with web applications.
- This facilitates location-aware services.



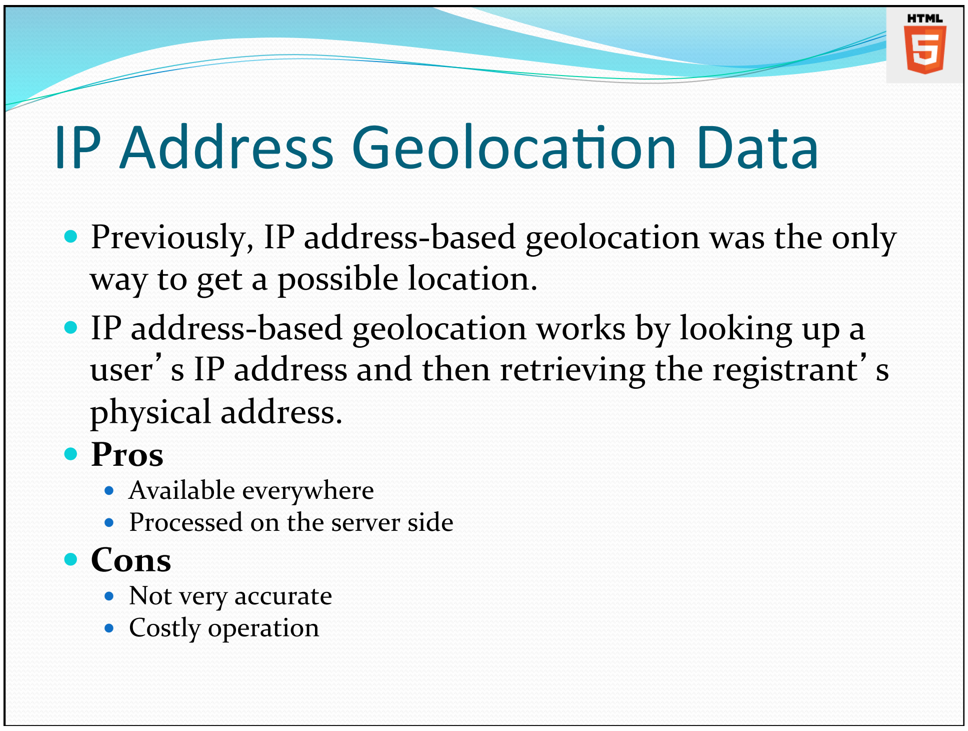
Location Information…

- HTML5 Geolocation API does not specify the underlying technology a device has to use to locate the user.
- It simply exposes an API for retrieving location information.
- The level of accuracy with which the location is pinpointed, is exposed by the API.



Location Information - Sources

- A device can use any of the following sources
  - IP address
  - Coordinate triangulation
    - GPS
    - Wi-Fi with MAC addresses from RFID, Bluetooth
    - GSM or CDMA cell IDs
  - User defined



IP Address Geolocation Data

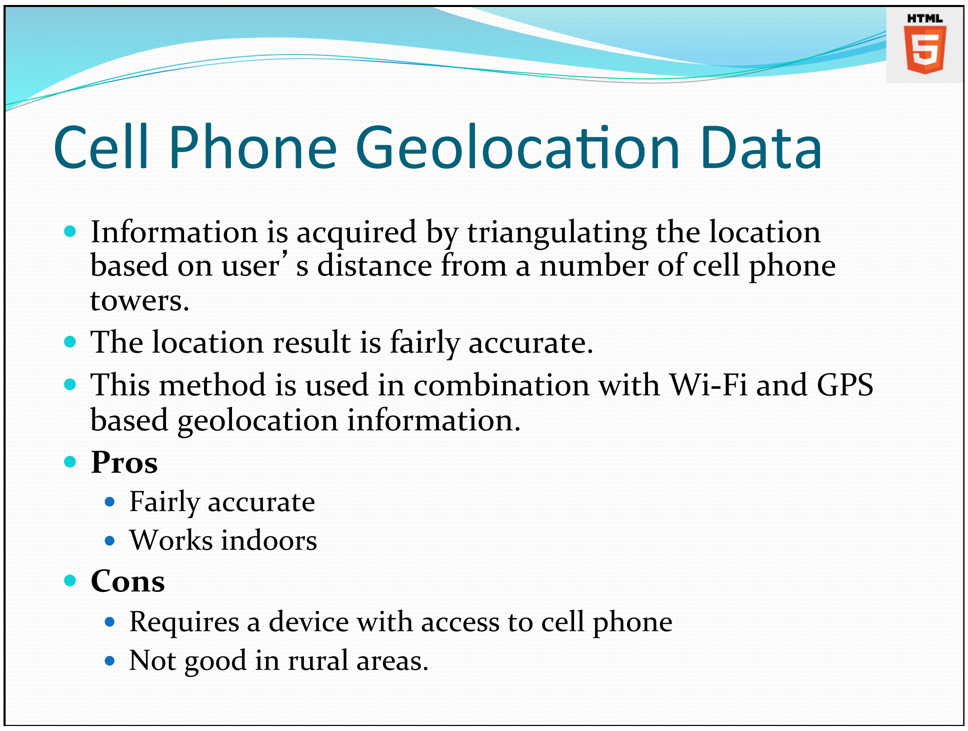
- Previously, IP address-based geolocation was the only way to get a possible location.
- IP address-based geolocation works by looking up a user's IP address and then retrieving the registrant's physical address.
- **Pros**
  - Available everywhere
  - Processed on the server side
- **Cons**
  - Not very accurate
  - Costly operation

GPS Geolocation Data

- GPS provides accurate location as long as there is line of sight with the satellites.
- A GPS fix is acquired by acquiring the signal from multiple GPS satellites.
- It can take a while to get a fix, therefore this task can be asynchronous
- **Pros**
  - Very accurate
- **Cons**
  - Takes a while and consumes power
  - Does not work well indoors
  - Additional hardware

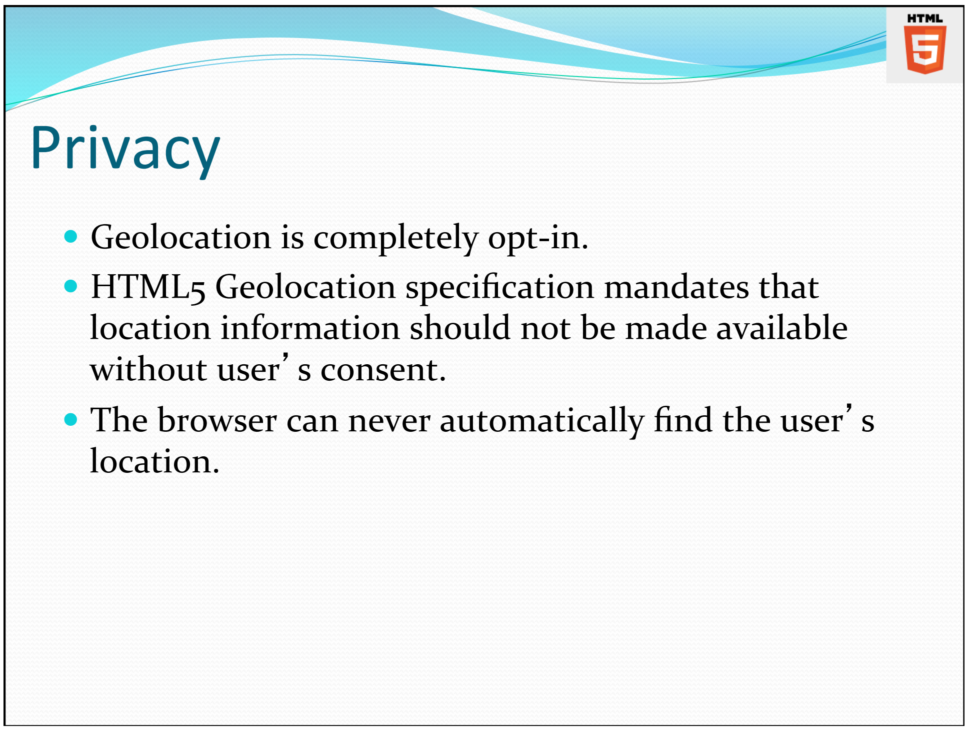
Wi-Fi Geolocation Data

- The information is acquired by triangulating the location based on the user' s distance from a number of known Wi-Fi access points.
- **Pros**
  - Accurate
  - Works indoors
  - Can get a fix quickly
- **Cons**
  - Not good in rural areas



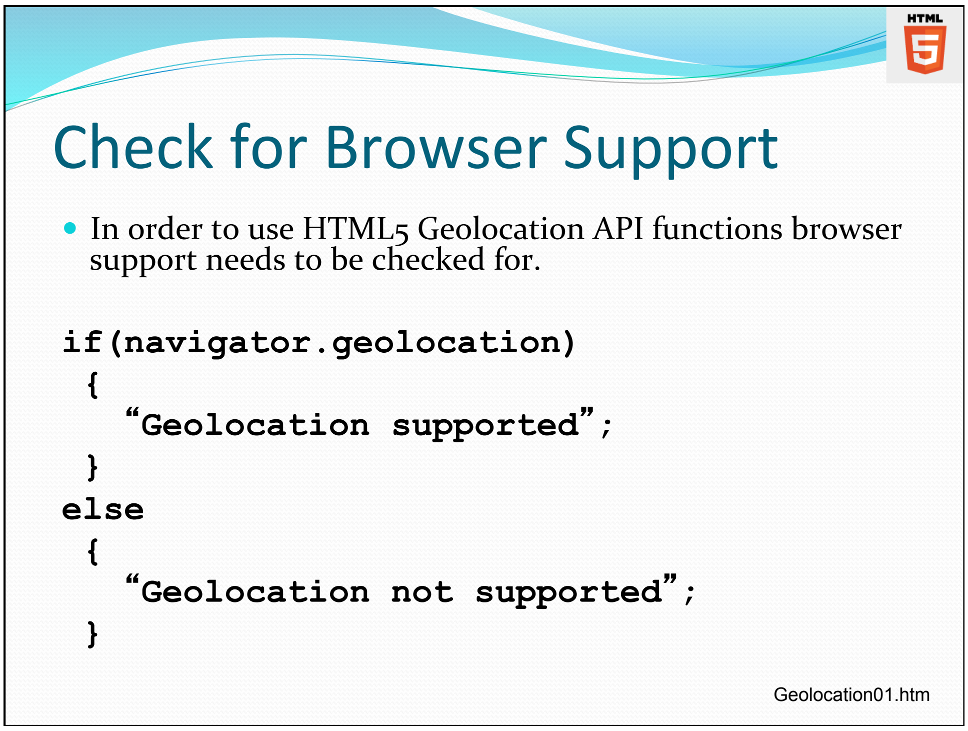
Cell Phone Geolocation Data

- Information is acquired by triangulating the location based on user' s distance from a number of cell phone towers.
- The location result is fairly accurate.
- This method is used in combination with Wi-Fi and GPS based geolocation information.
- **Pros**
  - Fairly accurate
  - Works indoors
- **Cons**
  - Requires a device with access to cell phone
  - Not good in rural areas.



Privacy

- Geolocation is completely opt-in.
- HTML5 Geolocation specification mandates that location information should not be made available without user' s consent.
- The browser can never automatically find the user' s location.

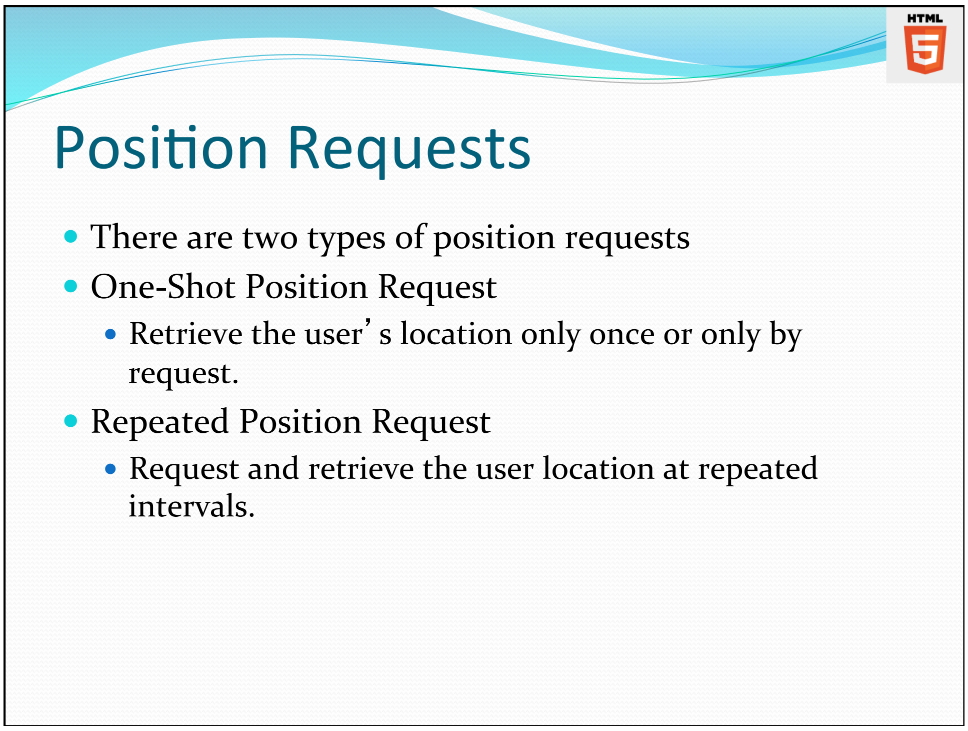


Check for Browser Support

- In order to use HTML5 Geolocation API functions browser support needs to be checked for.

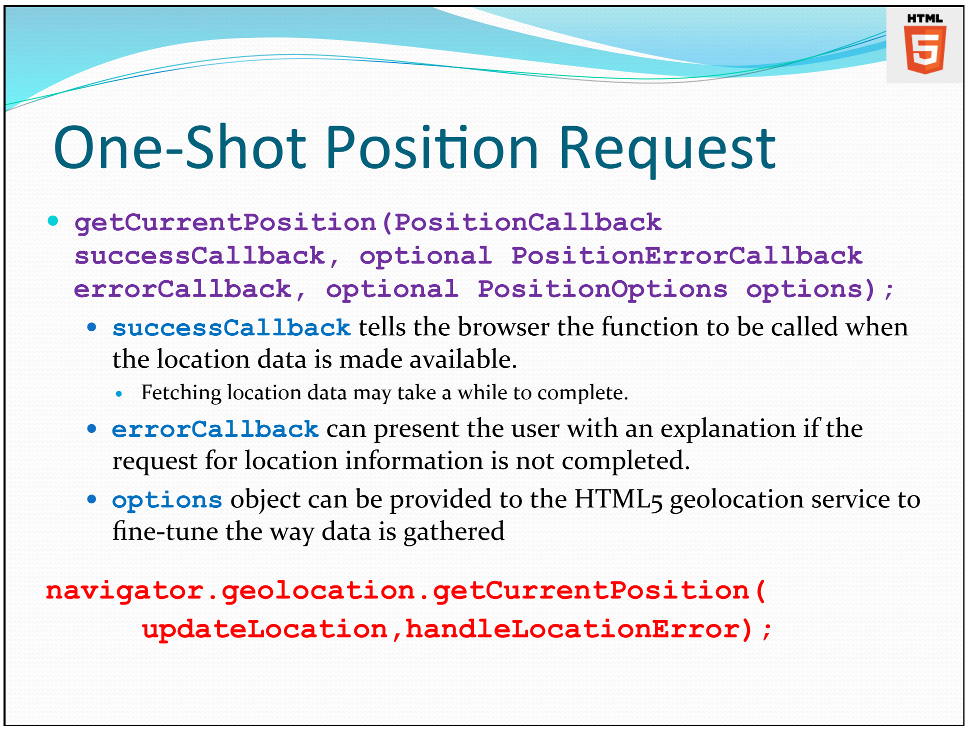
```
if(navigator.geolocation)
 {
   “Geolocation supported”;
 }
else
 {
   “Geolocation not supported”;
 }
```

Geolocation01.htm



Position Requests

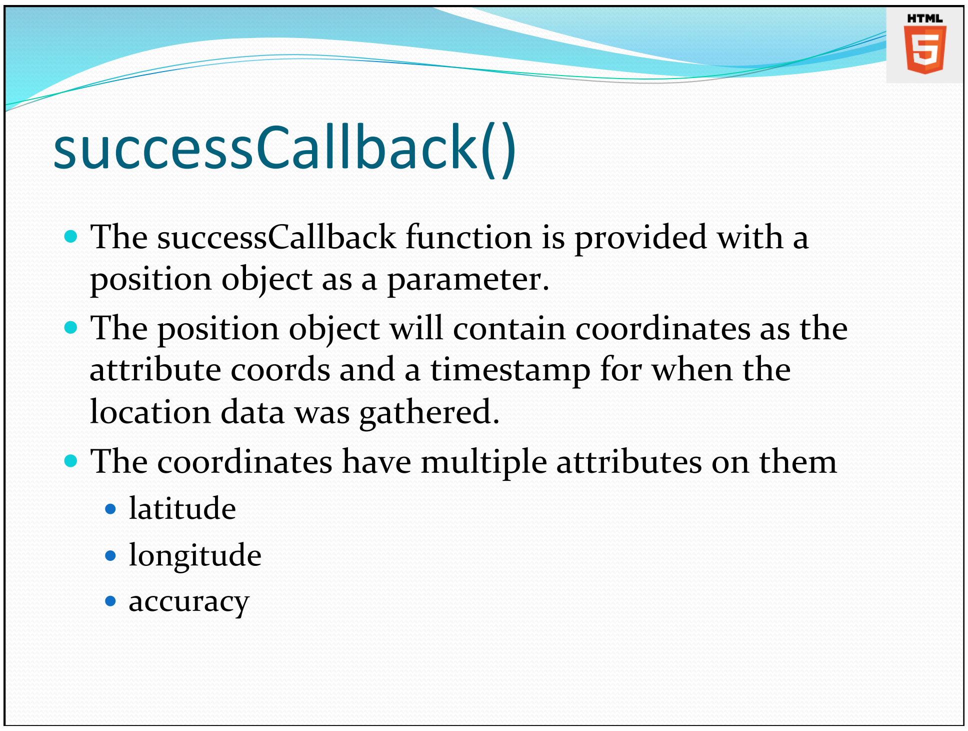
- There are two types of position requests
- One-Shot Position Request
  - Retrieve the user’s location only once or only by request.
- Repeated Position Request
  - Request and retrieve the user location at repeated intervals.



One-Shot Position Request

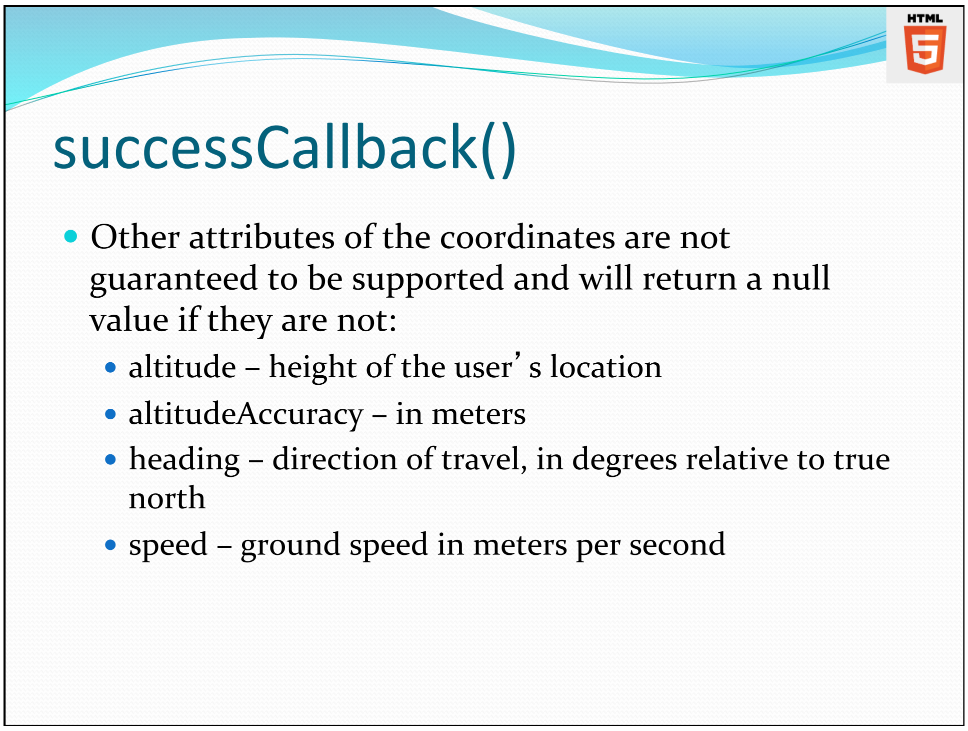
- `getCurrentPosition(PositionCallback successCallback, optional PositionErrorCallback errorCallback, optional PositionOptions options);`
  - `successCallback` tells the browser the function to be called when the location data is made available.
    - Fetching location data may take a while to complete.
  - `errorCallback` can present the user with an explanation if the request for location information is not completed.
  - `options` object can be provided to the HTML5 geolocation service to fine-tune the way data is gathered

```
navigator.geolocation.getCurrentPosition(
      updateLocation,handleLocationError);
```



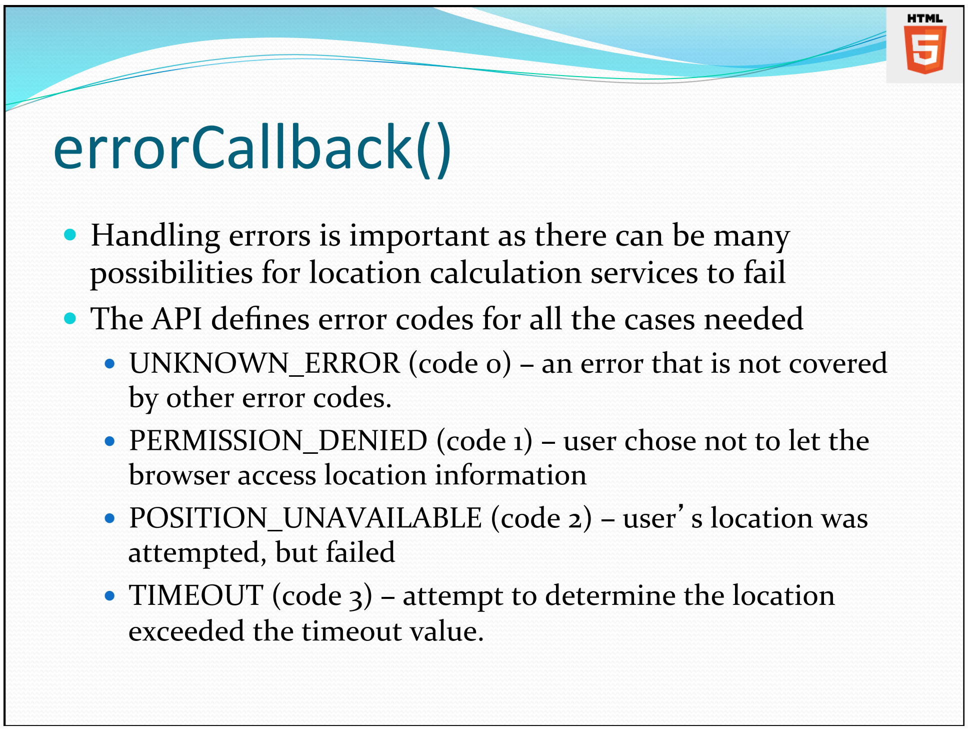
successCallback()

- The successCallback function is provided with a position object as a parameter.
- The position object will contain coordinates as the attribute coords and a timestamp for when the location data was gathered.
- The coordinates have multiple attributes on them
  - latitude
  - longitude
  - accuracy



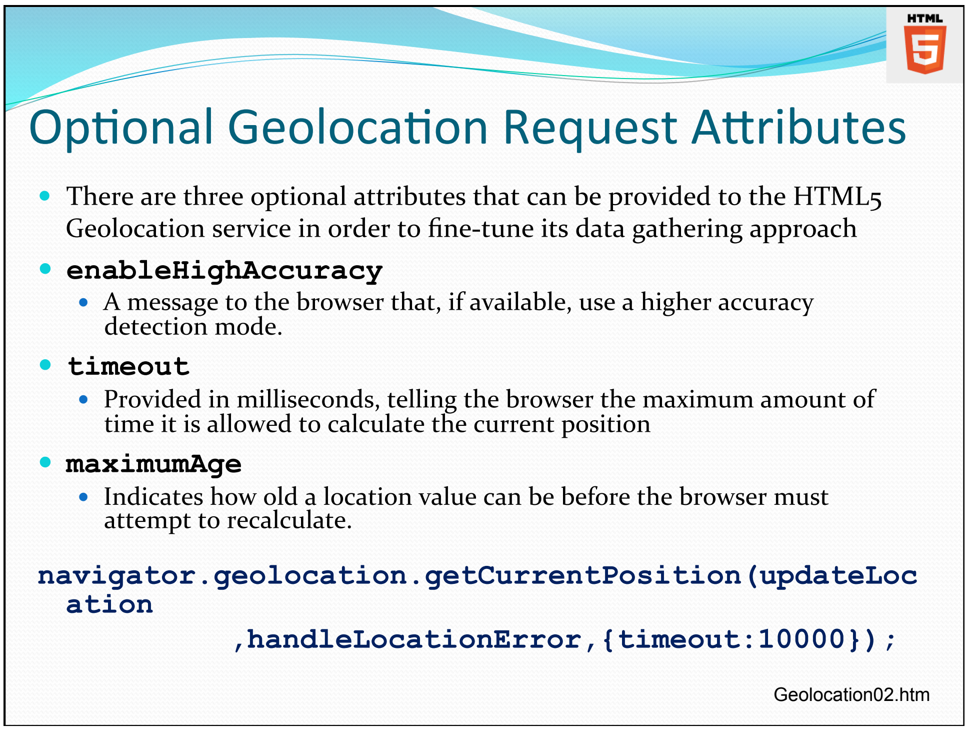
successCallback()

- Other attributes of the coordinates are not guaranteed to be supported and will return a null value if they are not:
  - altitude – height of the user’ s location
  - altitudeAccuracy – in meters
  - heading – direction of travel, in degrees relative to true north
  - speed – ground speed in meters per second



errorCallback()

- Handling errors is important as there can be many possibilities for location calculation services to fail
- The API defines error codes for all the cases needed
  - UNKNOWN_ERROR (code 0) – an error that is not covered by other error codes.
  - PERMISSION_DENIED (code 1) – user chose not to let the browser access location information
  - POSITION_UNAVAILABLE (code 2) – user’ s location was attempted, but failed
  - TIMEOUT (code 3) – attempt to determine the location exceeded the timeout value.

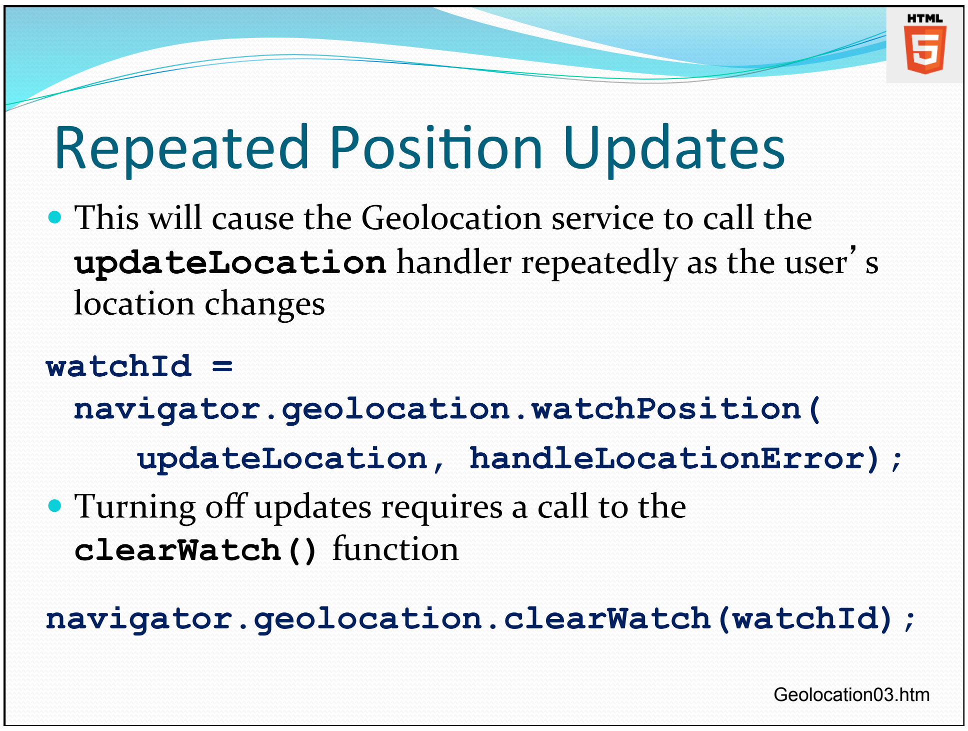


Optional Geolocation Request Attributes

- There are three optional attributes that can be provided to the HTML5 Geolocation service in order to fine-tune its data gathering approach
- **`enableHighAccuracy`**
  - A message to the browser that, if available, use a higher accuracy detection mode.
- **`timeout`**
  - Provided in milliseconds, telling the browser the maximum amount of time it is allowed to calculate the current position
- **`maximumAge`**
  - Indicates how old a location value can be before the browser must attempt to recalculate.

```
navigator.geolocation.getCurrentPosition(updateLoc
  ation
             ,handleLocationError,{timeout:10000});
```

Geolocation02.htm



Repeated Position Updates

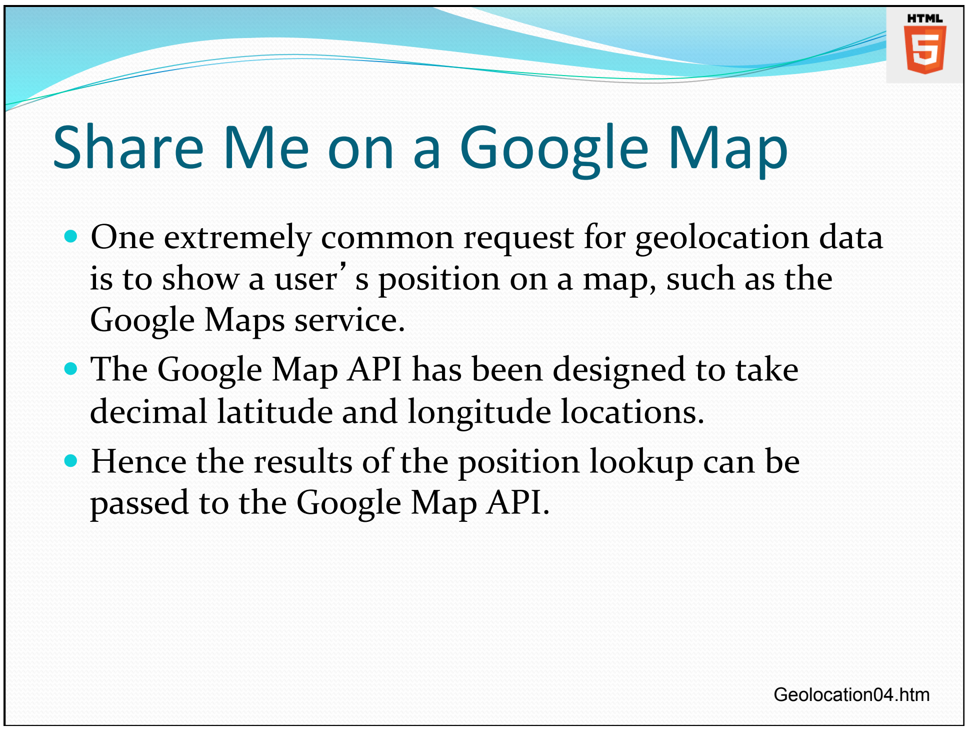
- This will cause the Geolocation service to call the **`updateLocation`** handler repeatedly as the user’ s location changes

```
watchId =
  navigator.geolocation.watchPosition(
      updateLocation, handleLocationError);
```

- Turning off updates requires a call to the **`clearWatch()`** function

```
navigator.geolocation.clearWatch(watchId);
```

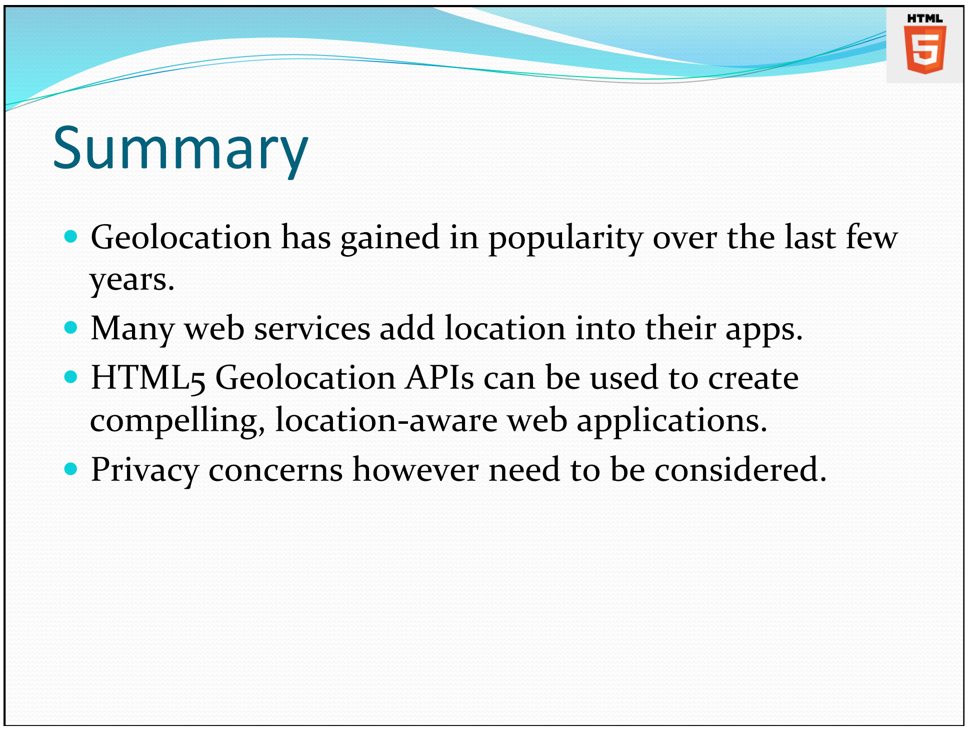
Geolocation03.htm



Share Me on a Google Map

- One extremely common request for geolocation data is to show a user's position on a map, such as the Google Maps service.
- The Google Map API has been designed to take decimal latitude and longitude locations.
- Hence the results of the position lookup can be passed to the Google Map API.

Geolocation04.htm



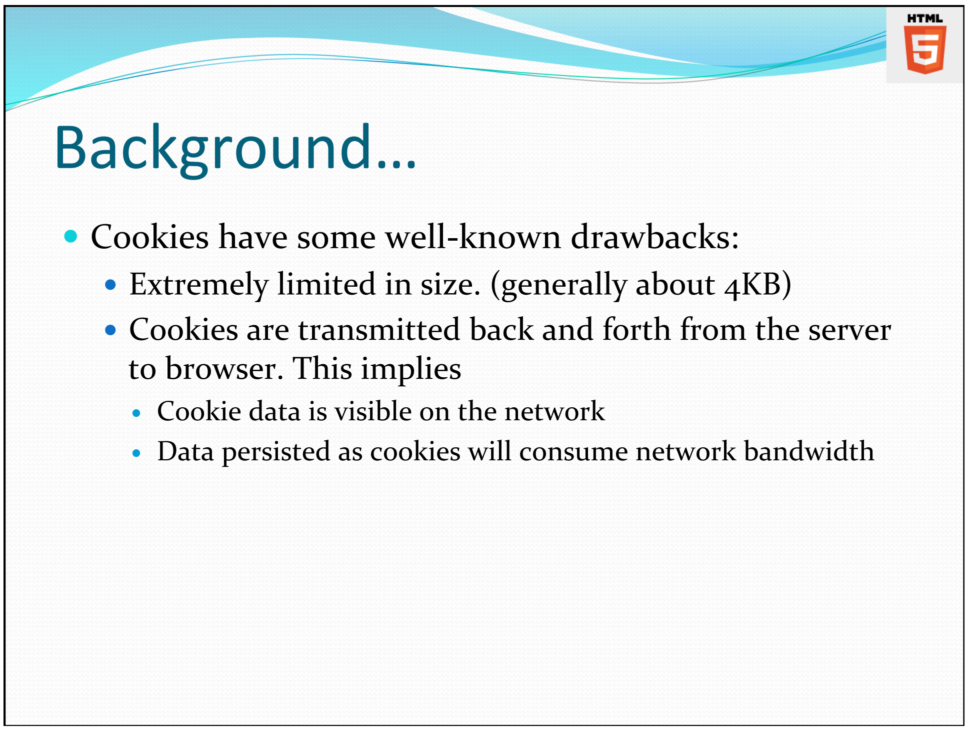
Summary

- Geolocation has gained in popularity over the last few years.
- Many web services add location into their apps.
- HTML5 Geolocation APIs can be used to create compelling, location-aware web applications.
- Privacy concerns however need to be considered.

HTML5 Web Storage

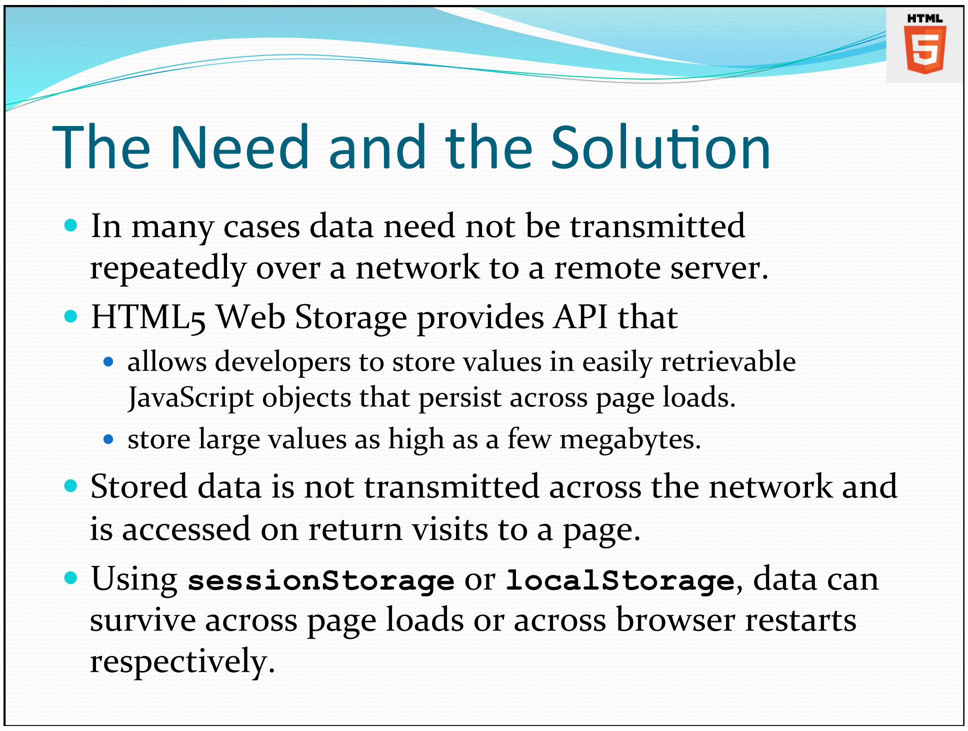
Background

- Browser cookies have been a way of sending text values back and forth from server to client.
- Servers can use the values in these cookies to track user information across web pages.
- Cookie values are transmitted back and forth every time a user visits a domain.
- Cookies can also be used for targeted advertising.



Background…

- Cookies have some well-known drawbacks:
  - Extremely limited in size. (generally about 4KB)
  - Cookies are transmitted back and forth from the server to browser. This implies
    - Cookie data is visible on the network
    - Data persisted as cookies will consume network bandwidth



The Need and the Solution

- In many cases data need not be transmitted repeatedly over a network to a remote server.
- HTML5 Web Storage provides API that
  - allows developers to store values in easily retrievable JavaScript objects that persist across page loads.
  - store large values as high as a few megabytes.
- Stored data is not transmitted across the network and is accessed on return visits to a page.
- Using **`sessionStorage`** or **`localStorage`**, data can survive across page loads or across browser restarts respectively.

Check for Browser Support

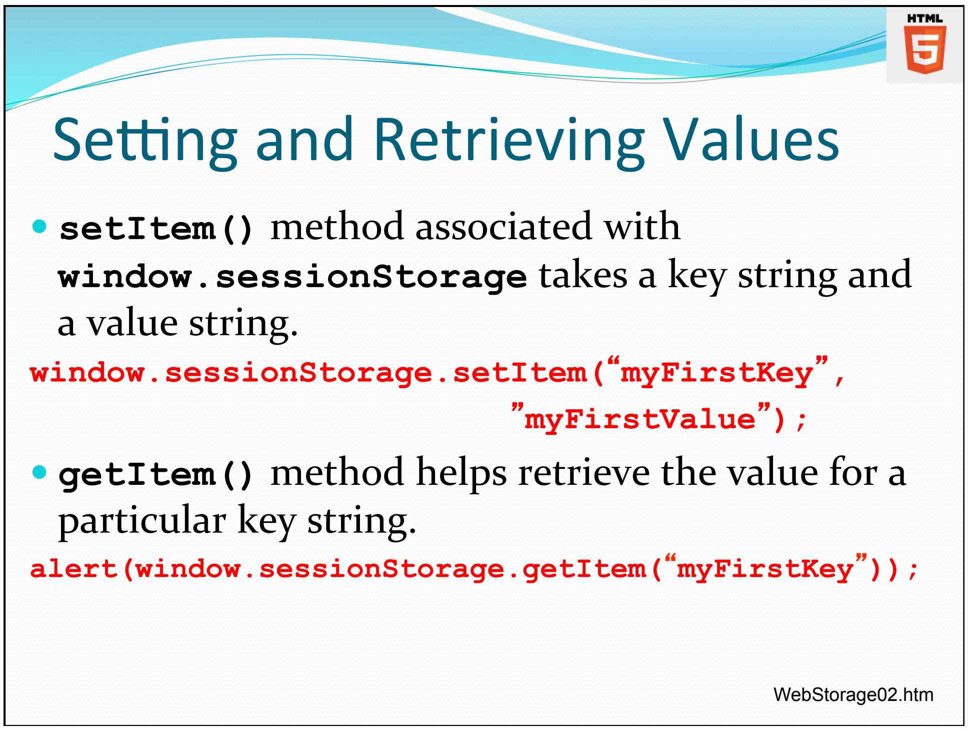
- The storage database for a given domain is accessed directly from the window object.

```
function checkStorageSupport()
 {
    if(window.sessionStorage)
    {
       "Browser supports sessionStorage"
    }
    else
    {
       "Browser does not support sessionStorage"
    }
 }
```

Check for Browser Support

```
    if(window.localStorage)
    {
       "Browser supports localStorage"
    }
    else
    {
       "Browser does not support
localStorage"
    }
```

WebStorage01.htm



Setting and Retrieving Values

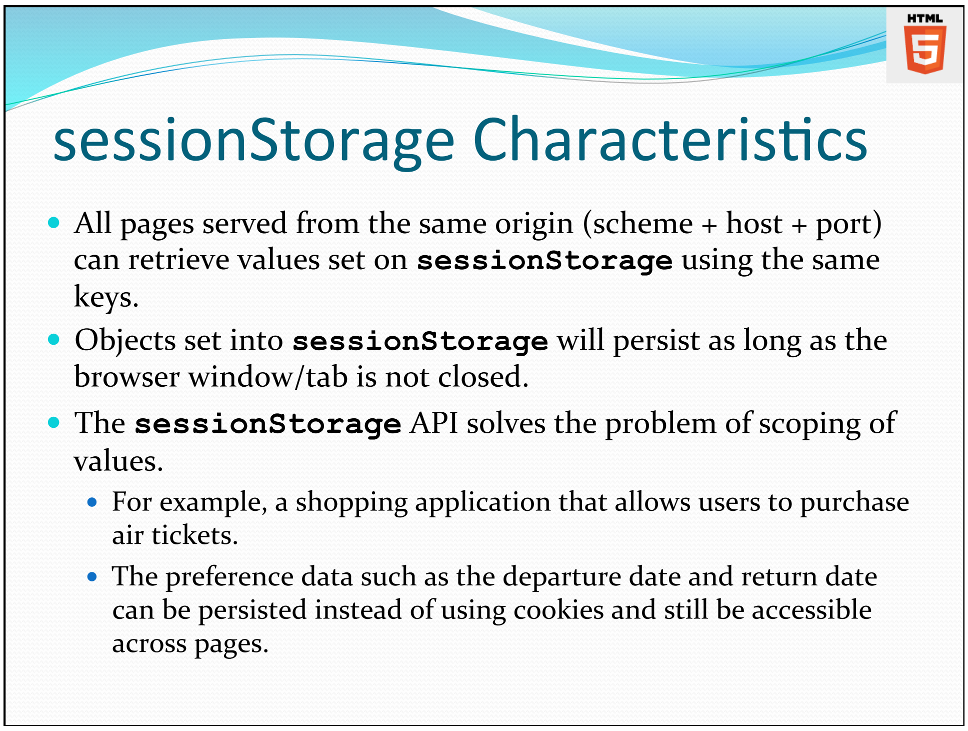
- **`setItem()`** method associated with **`window.sessionStorage`** takes a key string and a value string.

```
window.sessionStorage.setItem("myFirstKey",
                              "myFirstValue");
```

- **`getItem()`** method helps retrieve the value for a particular key string.

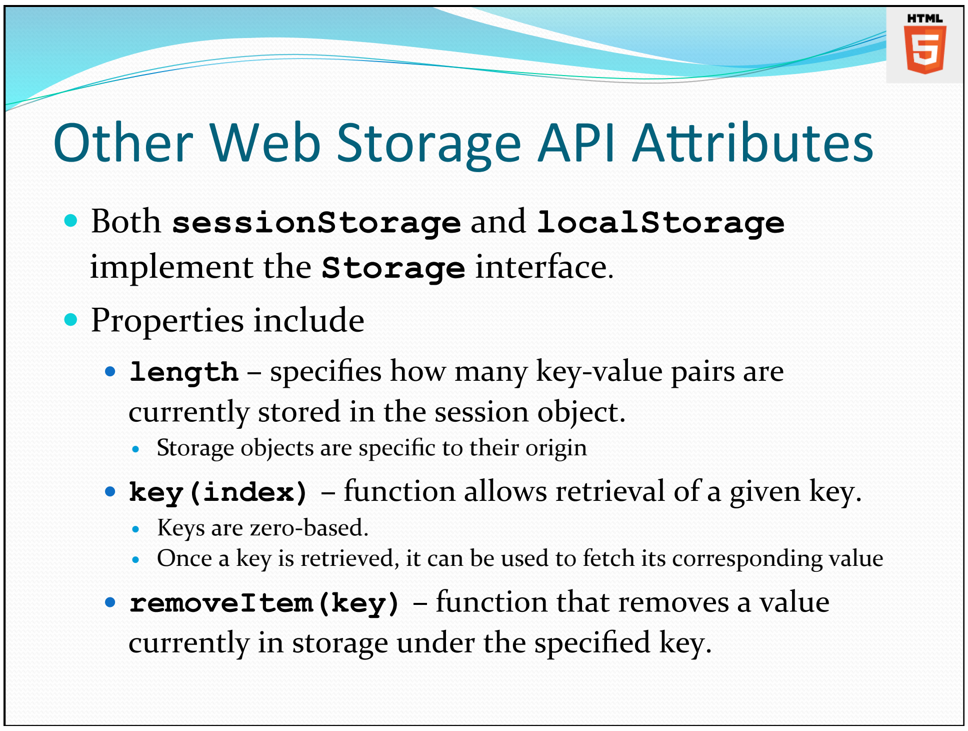
```
alert(window.sessionStorage.getItem("myFirstKey"));
```

WebStorage02.htm



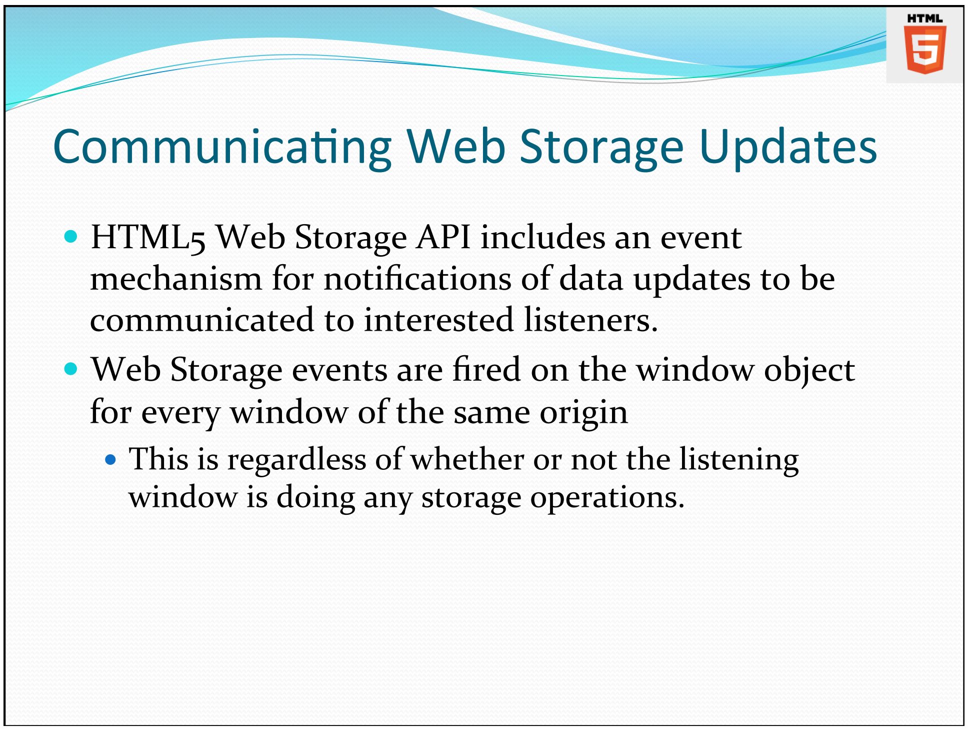
sessionStorage Characteristics

- All pages served from the same origin (scheme + host + port) can retrieve values set on **`sessionStorage`** using the same keys.
- Objects set into **`sessionStorage`** will persist as long as the browser window/tab is not closed.
- The **`sessionStorage`** API solves the problem of scoping of values.
  - For example, a shopping application that allows users to purchase air tickets.
  - The preference data such as the departure date and return date can be persisted instead of using cookies and still be accessible across pages.



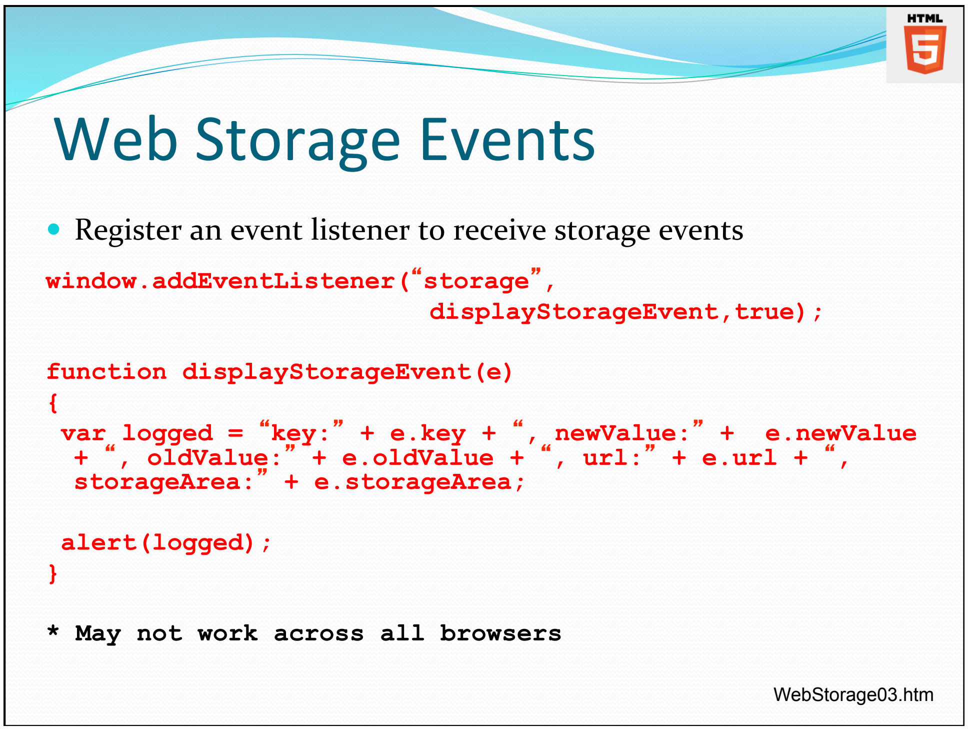
Other Web Storage API Attributes

- Both **`sessionStorage`** and **`localStorage`** implement the **`Storage`** interface.
- Properties include
  - **`length`** – specifies how many key-value pairs are currently stored in the session object.
    - Storage objects are specific to their origin
  - **`key(index)`** – function allows retrieval of a given key.
    - Keys are zero-based.
    - Once a key is retrieved, it can be used to fetch its corresponding value
  - **`removeItem(key)`** – function that removes a value currently in storage under the specified key.



Communicating Web Storage Updates

- HTML5 Web Storage API includes an event mechanism for notifications of data updates to be communicated to interested listeners.
- Web Storage events are fired on the window object for every window of the same origin
  - This is regardless of whether or not the listening window is doing any storage operations.

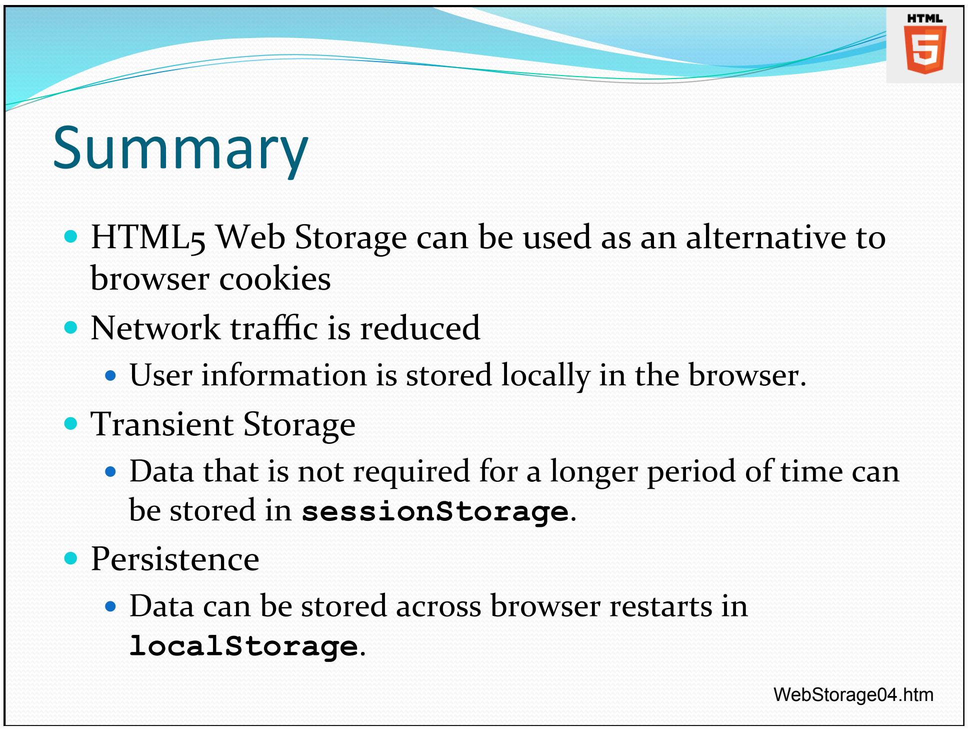


Web Storage Events

- Register an event listener to receive storage events

```
window.addEventListener(“storage”,
                          displayStorageEvent,true);

function displayStorageEvent(e)
{
 var logged = “key:” + e.key + “, newValue:” +  e.newValue
  + “, oldValue:” + e.oldValue + “, url:” + e.url + “,
  storageArea:” + e.storageArea;

 alert(logged);
}

* May not work across all browsers
```

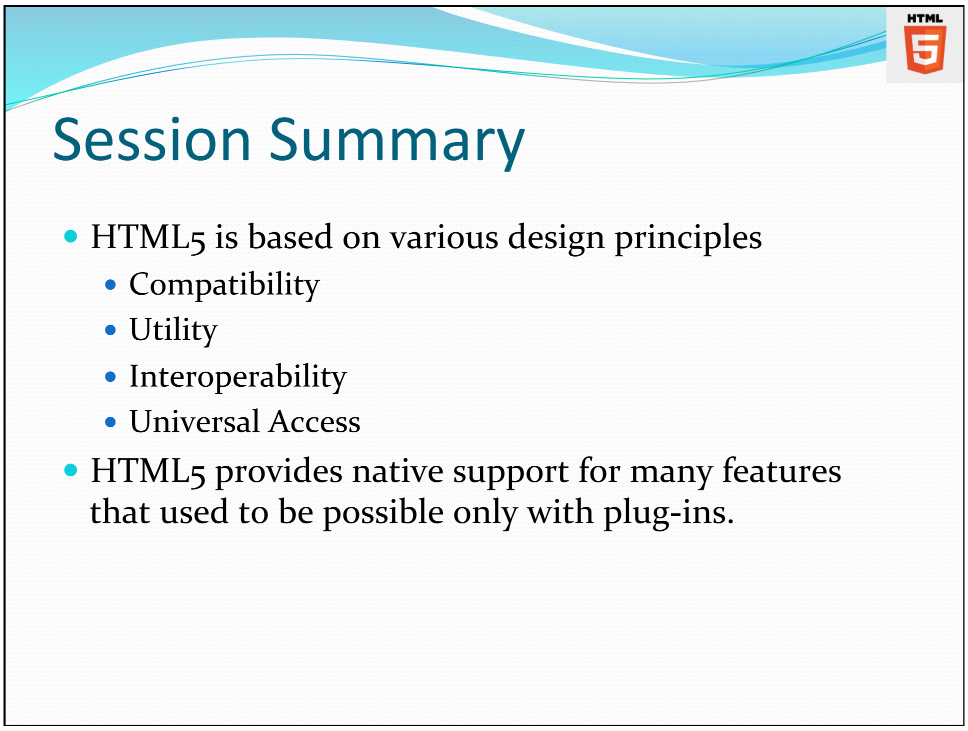
WebStorage03.htm



Summary

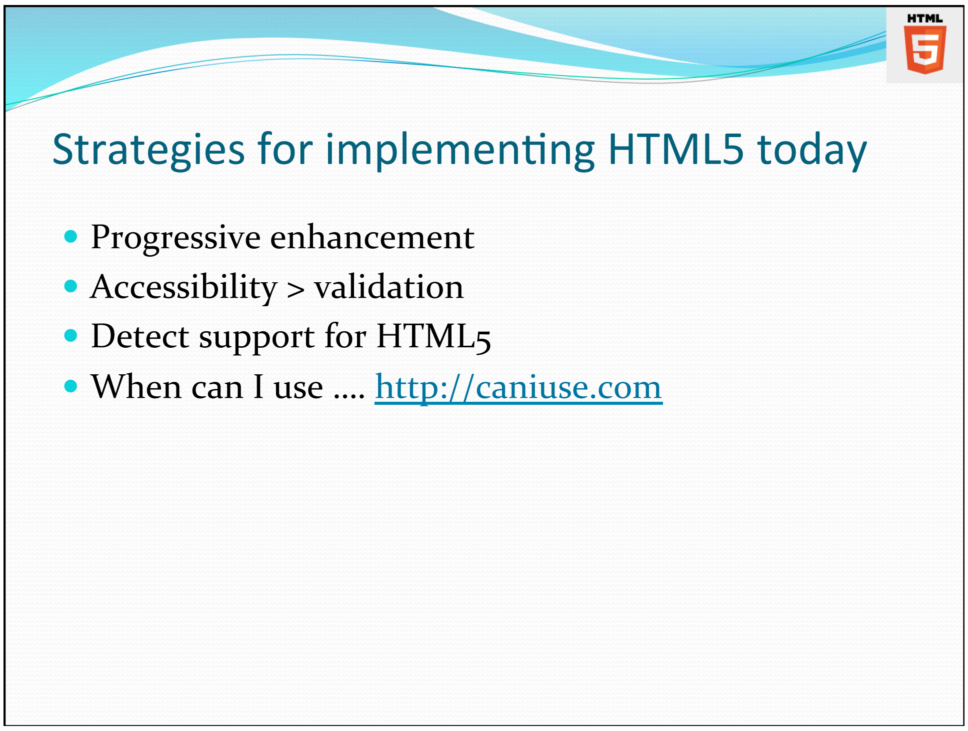
- HTML5 Web Storage can be used as an alternative to browser cookies
- Network traffic is reduced
  - User information is stored locally in the browser.
- Transient Storage
  - Data that is not required for a longer period of time can be stored in `sessionStorage`.
- Persistence
  - Data can be stored across browser restarts in `localStorage`.

WebStorage04.htm



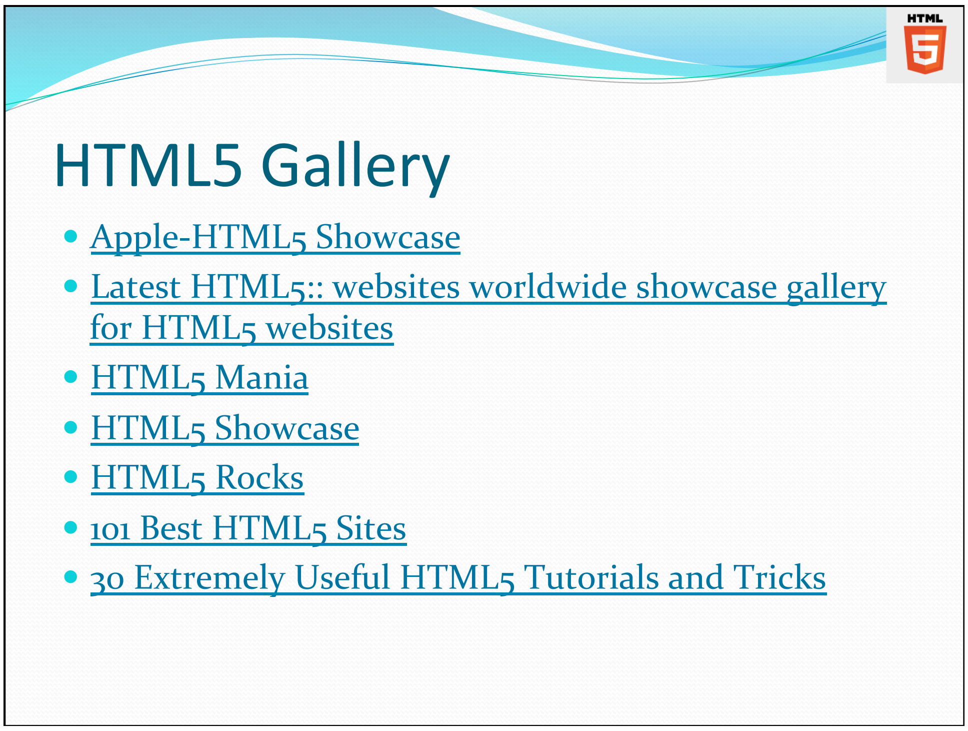
Session Summary

- HTML5 is based on various design principles
  - Compatibility
  - Utility
  - Interoperability
  - Universal Access
- HTML5 provides native support for many features that used to be possible only with plug-ins.



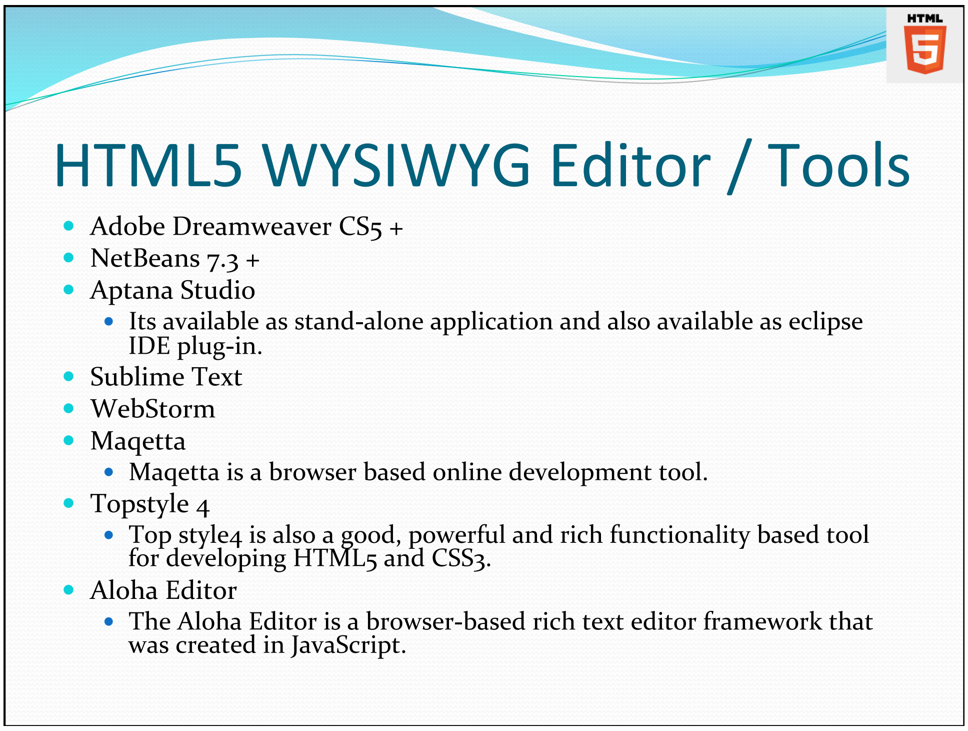
Strategies for implementing HTML5 today

- Progressive enhancement
- Accessibility > validation
- Detect support for HTML5
- When can I use .... http://caniuse.com



HTML5 Gallery

- Apple-HTML5 Showcase
- Latest HTML5:: websites worldwide showcase gallery for HTML5 websites
- HTML5 Mania
- HTML5 Showcase
- HTML5 Rocks
- 101 Best HTML5 Sites
- 30 Extremely Useful HTML5 Tutorials and Tricks



HTML5 WYSIWYG Editor / Tools

- Adobe Dreamweaver CS5 +
- NetBeans 7.3 +
- Aptana Studio
  - Its available as stand-alone application and also available as eclipse IDE plug-in.
- Sublime Text
- WebStorm
- Maqetta
  - Maqetta is a browser based online development tool.
- Topstyle 4
  - Top style4 is also a good, powerful and rich functionality based tool for developing HTML5 and CSS3.
- Aloha Editor
  - The Aloha Editor is a browser-based rich text editor framework that was created in JavaScript.