Internet Programming

HTML5 Extensions

Dan Brandon, Ph.D., PMP

History

- In 1998 the W3C decided not to continue HTML, but to follow XML standards
- HTML was frozen at version 4.01 and the specification for XHTML was born which was HTML within the XML syntax
- Several flavors of XHTML were created including XHTML Transitional which was intended to be a migration aid
- Next work began on the XHTML 2.0 standard which was not to be backward compatible with HTML

History (con't)

- Some "dissidents" (such as Ian Hickson originally at Opera and later Google) did not agree with that new direction, and they were "supported" to some degree by some industry heavyweights including Apple
- In 2004 they formed a group called WHATWG
  - Web Hypertext Application Technology Working Group (www.whatwg.org)
- And they started an effort called Web Applications 1.0 which was backward compatible with HTML

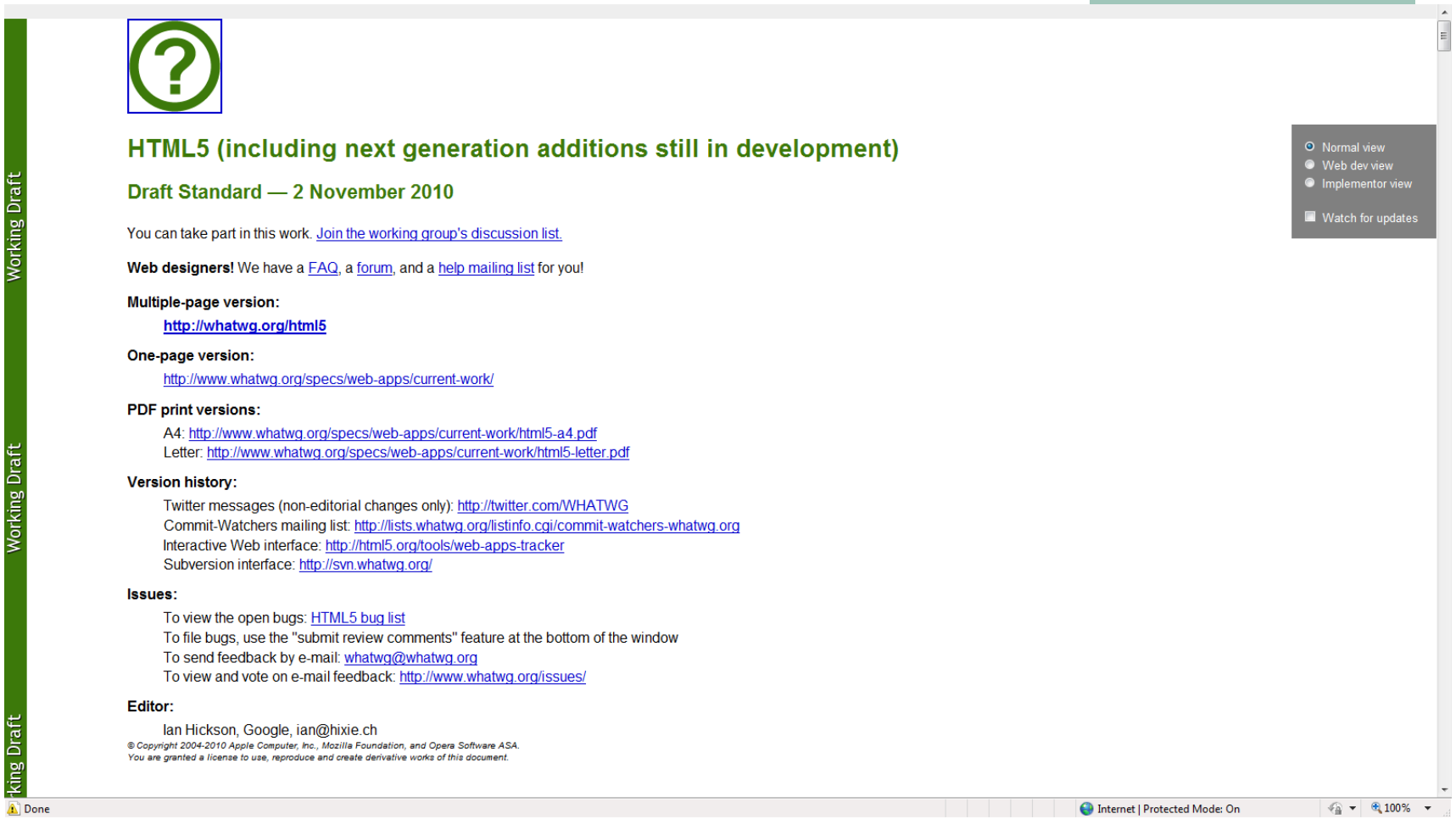
History (con't)

- In 2006 the W3C determined they had perhaps marched off in the wrong direction, and the world did not want to move to XML/XHTML and lose backward compatibility with HTML
- In 2007 W3C's restructured working group then voted to use the WHASTWG's Web Applications spec for the next version of HTML → HTML5
  - www.w3.org/TR/html5
- In 2009 the W3C stopped working on XHTML 2.0 and threw all its support behind HTML5

History (con't)

- The specification is an ongoing work, and is expected to remain so for many years, although parts of HTML5 are implemented in browsers before the whole specification reaches final Recommendation status
- On 14 February 2011, the W3C extended the charter of its HTML Working Group with clear milestones for HTML5
- In May 2011, the working group advanced HTML5 to "Last Call", an invitation to communities inside and outside W3C to confirm the technical soundness of the specification
- In 2012 work started in earnest and now the W3C has developed a comprehensive test suite to achieve broad interoperability for the full specification

WHATWG



Working Draft

Working Draft

king Draft

HTML5 (including next generation additions still in development)

Draft Standard — 2 November 2010

Normal view
Web dev view
Implementor view

Watch for updates

You can take part in this work. Join the working group's discussion list.

Web designers! We have a FAQ, a forum, and a help mailing list for you!

Multiple-page version:

http://whatwg.org/html5

One-page version:

http://www.whatwg.org/specs/web-apps/current-work/

PDF print versions:

A4: http://www.whatwg.org/specs/web-apps/current-work/html5-a4.pdf
Letter: http://www.whatwg.org/specs/web-apps/current-work/html5-letter.pdf

Version history:

Twitter messages (non-editorial changes only): http://twitter.com/WHATWG
Commit-Watchers mailing list: http://lists.whatwg.org/listinfo.cgi/commit-watchers-whatwg.org
Interactive Web interface: http://html5.org/tools/web-apps-tracker
Subversion interface: http://svn.whatwg.org/

Issues:

To view the open bugs: HTML5 bug list
To file bugs, use the "submit review comments" feature at the bottom of the window
To send feedback by e-mail: whatwg@whatwg.org
To view and vote on e-mail feedback: http://www.whatwg.org/issues/

Editor:

Ian Hickson, Google, ian@hixie.ch

Done

Internet | Protected Mode: On

100%

W3C HTML5

W3C Editor's Draft

HTML5

A vocabulary and associated APIs for HTML and XHTML

Editor's Draft 30 January 2012

Latest Published Version:
http://www.w3.org/TR/html5/
Latest Editor's Draft:
http://dev.w3.org/html5/spec/Overview.html
Previous Versions:
http://www.w3.org/TR/2011/WD-html5-20110525/
http://www.w3.org/TR/2011/WD-html5-20110405/
http://www.w3.org/TR/2011/WD-html5-20110113/
http://www.w3.org/TR/2010/WD-html5-20101019/
http://www.w3.org/TR/2010/WD-html5-20100624/
http://www.w3.org/TR/2010/WD-html5-20100304/
http://www.w3.org/TR/2009/WD-html5-20090825/
http://www.w3.org/TR/2009/WD-html5-20090423/
http://www.w3.org/TR/2009/WD-html5-20090212/
http://www.w3.org/TR/2008/WD-html5-20080610/
http://www.w3.org/TR/2008/WD-html5-20080122/
Editor:
Ian Hickson, Google, Inc.

This specification is available in the following formats: single page HTML, multipage HTML, web developer edition. This is revision $Revision: 1.5532 $.

Abstract

This specification defines the 5th major revision of the core language of the World Wide Web: the Hypertext Markup Language (HTML). In this version, new features are introduced to help Web application authors, new elements are introduced based on research into prevailing authoring practices, and special attention has been given to defining clear conformance criteria for user agents in an effort to improve interoperability.

SGML, XML & Compatibility

- The HTML5 syntax is no longer based on SGML despite the similarity of its markup
- It is not an XML language and must be served up as a MIME type of text/html
- However, it comes with a new introductory line that looks like an SGML document type declaration, <!DOCTYPE html>, which enables standards-compliant rendering in all browsers that use "DOCTYPE sniffing"
- HTML5 has been designed to be backward compatible with common parsing of older versions of HTML
- HTML5 also specs technologies such as AJAX, and specifically defines error handling

Differences from XHTML

- Forgiving syntax:
  - Upper/lowercase
  - Quoting
  - Omitting implied elements (head, body, etc.)
  - Omitting implied declarations (type = …)

Differences from HTML 4+

- Don’t use “center” (use CSS)
- No more frames (use CSS) - [iframe still OK]
- Don’t use tables for page layout (use CSS) – no more align, bgcolor, cellpadding, cellspacing, height, width, nowrap, rules, valign
- Deprecated elements will be dropped altogether (eventually): acronym, applet, basefont, big, center, dir, font, frame, frameset, isindex, noframes, s, strike, tt, u

Differences from HTML 4+

- New parsing rules: oriented towards flexible parsing and compatibility -- not based on SGML
- Ability to use inline SVG (Scalable Vector Graphics) and MathML in text/html
- New types of form controls: dates and times, email, url, search, color
- New attributes and global attributes (that can be applied for every element): id, tabindex, hidden, data-* (custom data attributes)
- Some deprecated elements from HTML 4.01 have been dropped, including purely presentational elements such as <font> and , whose effects are achieved using CSS (Cascading Style Sheets)

New Elements and Attributes

- HTML5 introduces a number of new elements and attributes that reflect typical usage on modern websites
  - New elements: article, aside, audio, canvas, command, datalist, details, embed, figcaption, figure, footer, header, hgroup, keygen, mark, meter, nav, output, progress, rp, rt, ruby, section, source, summary, time, video, wbr
- Some of them are semantic replacements for common uses of generic block ( ) and inline () elements, for example <nav> (website navigation block) and <footer> (usually referring to bottom of web page)
- Other elements provide new functionality through a standardized interface, such as the multimedia elements <audio> and <video>

API's

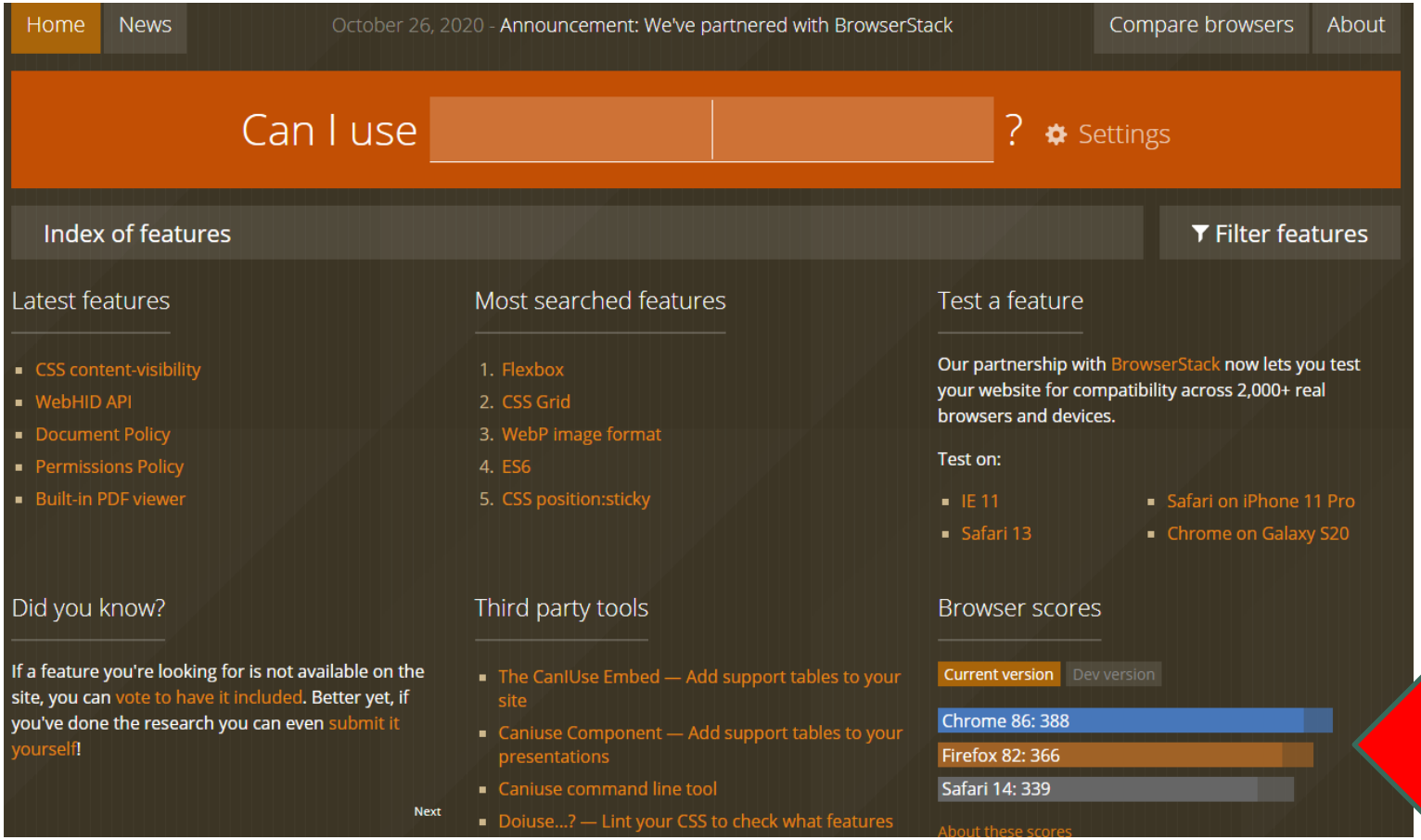
- In addition to specifying markup, HTML5 specifies scripting application programming interfaces (APIs)
- There is also a renewed emphasis on the importance of DOM scripting (e.g., JavaScript) in Web behavior; existing document object model (DOM) interfaces are extended and de facto features documented
- There are also new APIs, such as:
  - The canvas element for immediate mode 2D drawing
  - Timed media playback
  - Offline storage database (offline web applications). See Web Storage
  - Document editing
  - Drag-and-drop
  - Cross-document messaging
  - Browser history management
  - MIME type and protocol handler registration.
  - Microdata

Related Technologies

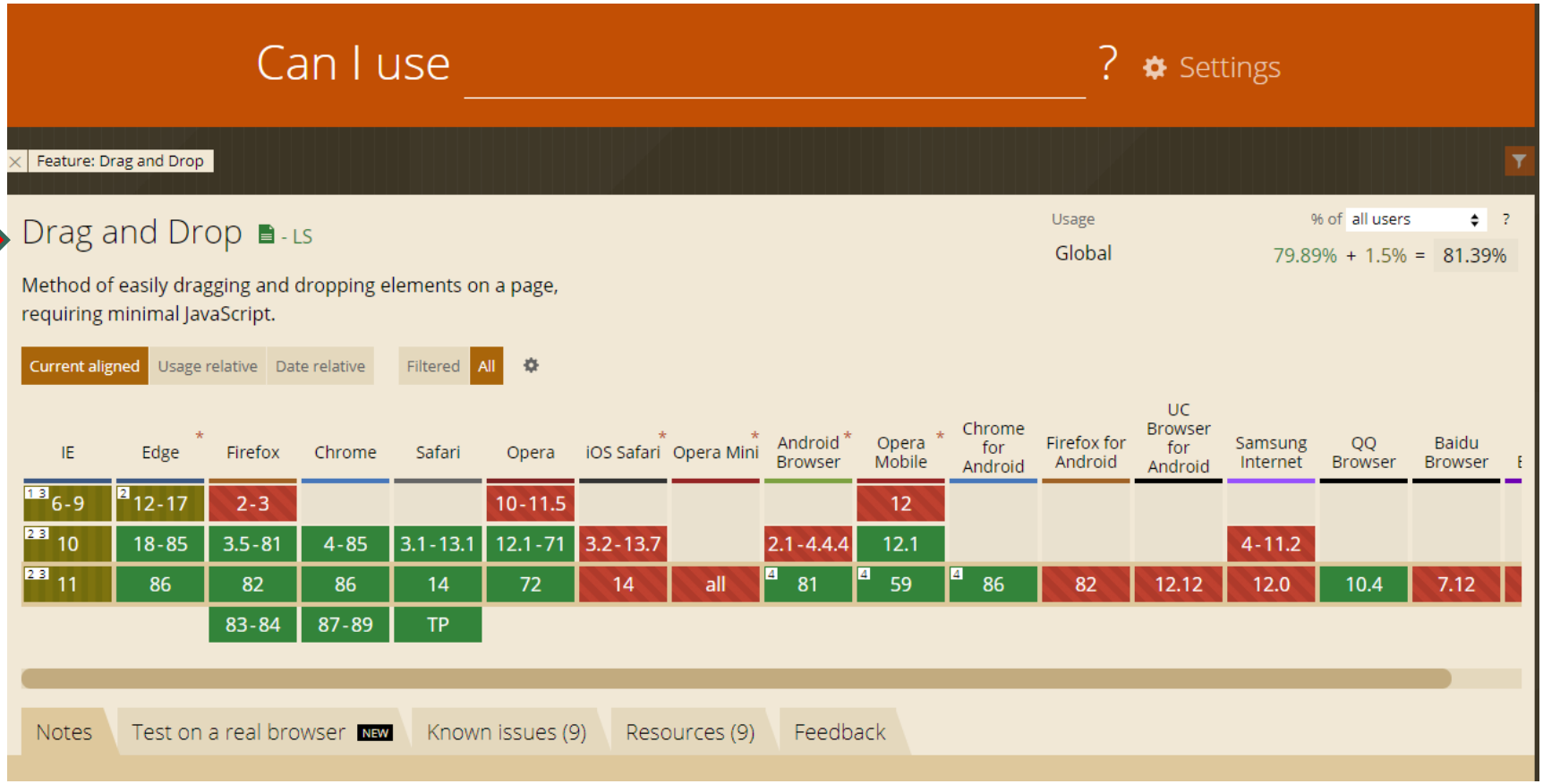
- Not all of the previous technologies are included in the W3C HTML5 specification, though they are in the WHATWG HTML specification
- Some related technologies, which are not part of either the W3C HTML5 or the WHATWG HTML specification, are
  - Geolocation
  - Web SQL Database, a local SQL Database
  - The Indexed Database API, a indexed hierarchical key-value store (formerly WebSimpleDB)
- The W3C publishes specifications for these separately

When Can I Use

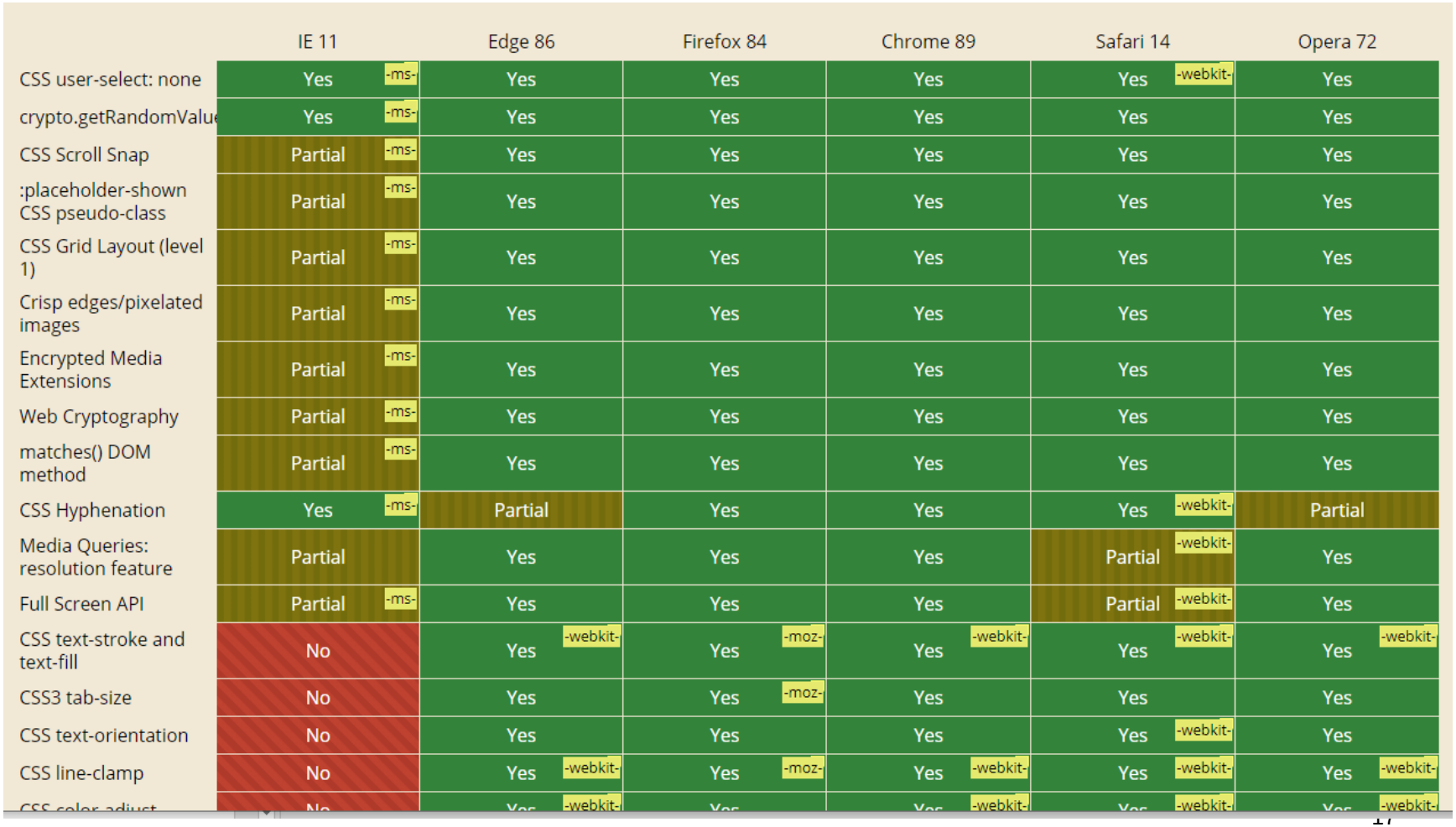
[http://caniuse.com/]



When Can I Use – Specific Features



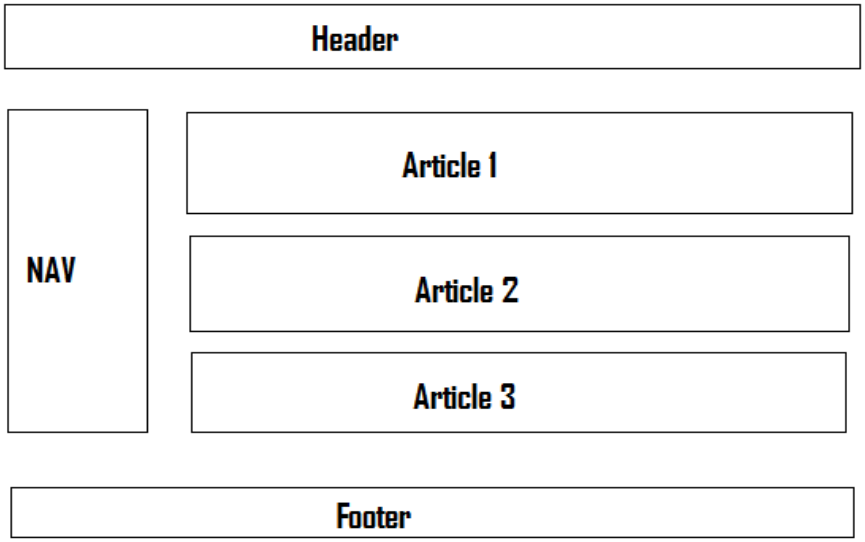
When Can I Use – Feature by Browser



	IE 11	Edge 86	Firefox 84	Chrome 89	Safari 14	Opera 72
CSS user-select: none	Yes -ms-	Yes	Yes	Yes	Yes -webkit-	Yes
crypto.getRandomValue	Yes -ms-	Yes	Yes	Yes	Yes	Yes
CSS Scroll Snap	Partial -ms-	Yes	Yes	Yes	Yes	Yes
:placeholder-shown CSS pseudo-class	Partial -ms-	Yes	Yes	Yes	Yes	Yes
CSS Grid Layout (level 1)	Partial -ms-	Yes	Yes	Yes	Yes	Yes
Crisp edges/pixelated images	Partial -ms-	Yes	Yes	Yes	Yes	Yes
Encrypted Media Extensions	Partial -ms-	Yes	Yes	Yes	Yes	Yes
Web Cryptography	Partial -ms-	Yes	Yes	Yes	Yes	Yes
matches() DOM method	Partial -ms-	Yes	Yes	Yes	Yes	Yes
CSS Hyphenation	Yes -ms-	Partial	Yes	Yes	Yes -webkit-	Partial
Media Queries: resolution feature	Partial	Yes	Yes	Yes	Partial -webkit-	Yes
Full Screen API	Partial -ms-	Yes	Yes	Yes	Partial -webkit-	Yes
CSS text-stroke and text-fill	No	Yes -webkit-	Yes -moz-	Yes -webkit-	Yes -webkit-	Yes -webkit-
CSS3 tab-size	No	Yes	Yes -moz-	Yes	Yes	Yes
CSS text-orientation	No	Yes	Yes	Yes	Yes -webkit-	Yes
CSS line-clamp	No	Yes -webkit-	Yes -moz-	Yes -webkit-	Yes -webkit-	Yes -webkit-
CSS color-adjust	No	Yes -webkit-	Yes	Yes -webkit-	Yes -webkit-	Yes -webkit-

Layout Control – Structural Elements

- HTML5 defines new structural elements that facilitate page layout that traditionally was accomplished with CSS classes and DIV's, and provides more semantics
  - Header
  - Nav
  - Article & Section
  - Footer



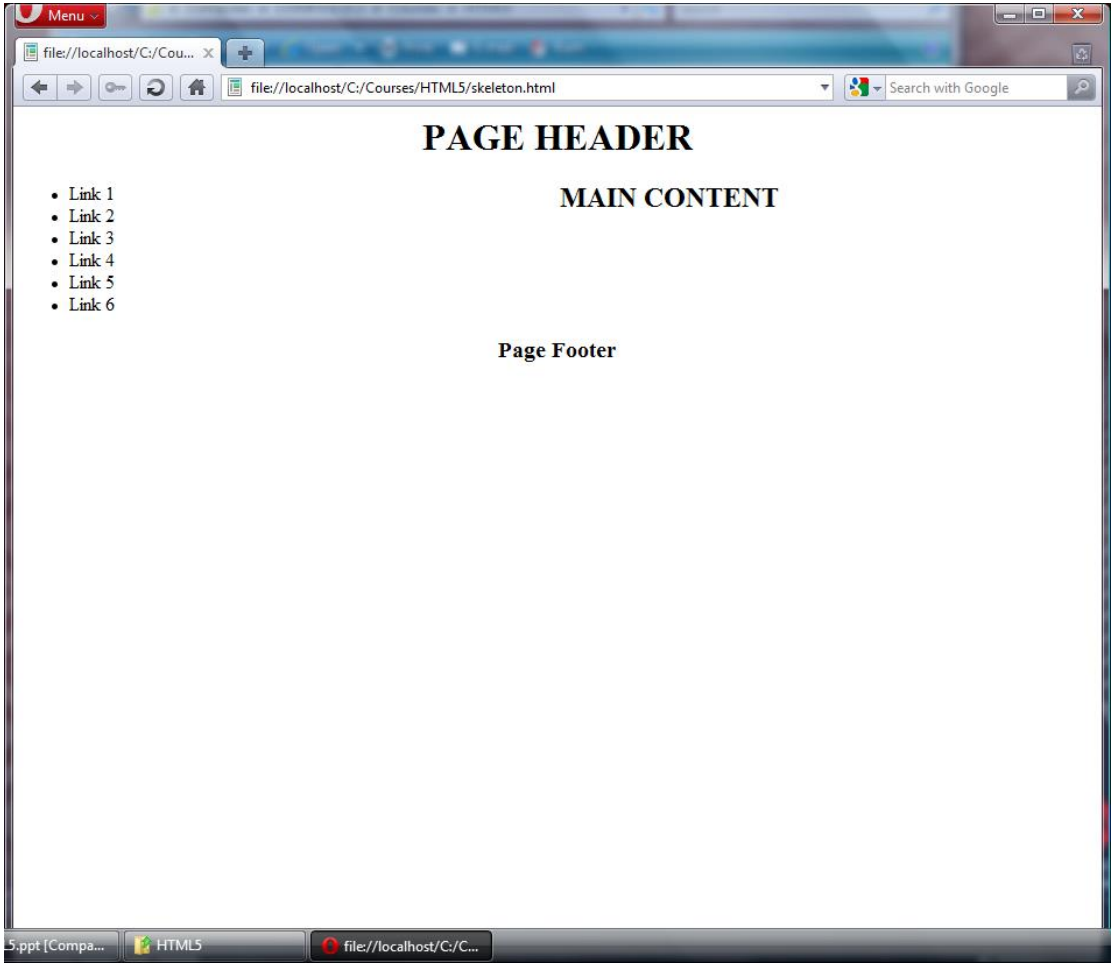
Structural Elements (con't)

- However (without custom css files) the current browsers still do not understand these new structural tags
- One can simulate the behavior of these new tags by defining them in CSS
- The next slide shows an example of this new navigation using the HML5 tags with an external CSS file to simulate the effect

Example HTML5 File

```
<html lang="en">
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=us-ascii">
        <link rel="stylesheet" type="text/css" href="html5_css.css">
</head>
<body>
        <header>
                <h1 align="center">PAGE HEADER</h1>
        </header>
        <nav>
                <ul>
                        <li>Link 1 </li>
                        <li>Link 2 </li>
                        <li>Link 3 </li>
                        <li>Link 4 </li>
                        <li>Link 5 </li>
                        <li>Link 6 </li>
                </ul>
        </nav>
        <article>
                <h2 align="center">MAIN CONTENT</h2>
        </article>
        <footer>
                <h3 align="center">Page Footer</h3>
        </footer>
</body>
</html>
```

Example HTML5 File Rendered



CSS Simulation File

- The CSS file to simulate HTML5 styling is:
  - header, nav, footer, article {display:block}
  - nav {float:left; width:20%}
  - article {float:right; width:79%}
  - footer {clear:both}

IE and CSS Elements

- The previous CSS file to simulate new HTML5 styling will work in all browsers except earlier IE
- For IE before version 9, the following JavaScript will also need to be added:
  - document.createElement('header');
  - document.createElement('nav');
  - document.createElement('article');
  - document.createElement('footer');
- And JavaScript will need to enabled

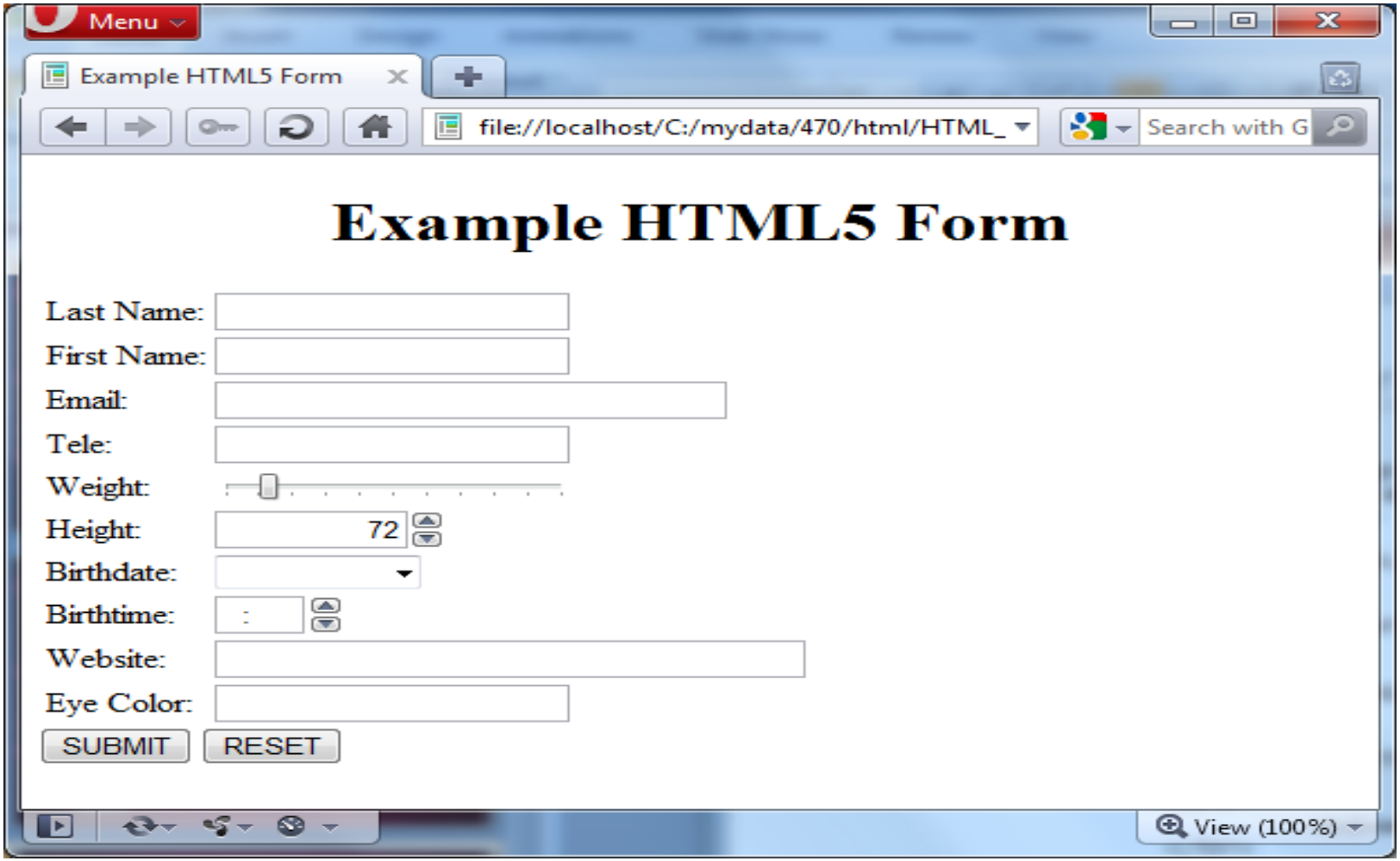
Other Structural Elements

- Ol
- Dl
- Cite
- Address
- Em, i
- Strong, b
- Hr
- hgroup
- aside
- details
- figure
- mark

HTML 5 Extensions for Forms

- HTML 5 provides new types of form controls which may provide both error checking and/or entry aids such as “pickers”:
  - range (exact number not important)
  - number
  - telephone number
  - color
  - date, time, datetime (UTC), datetime-local, month, week
  - email
  - image
  - search
  - url
  - Password
- New attributes and functions are also available

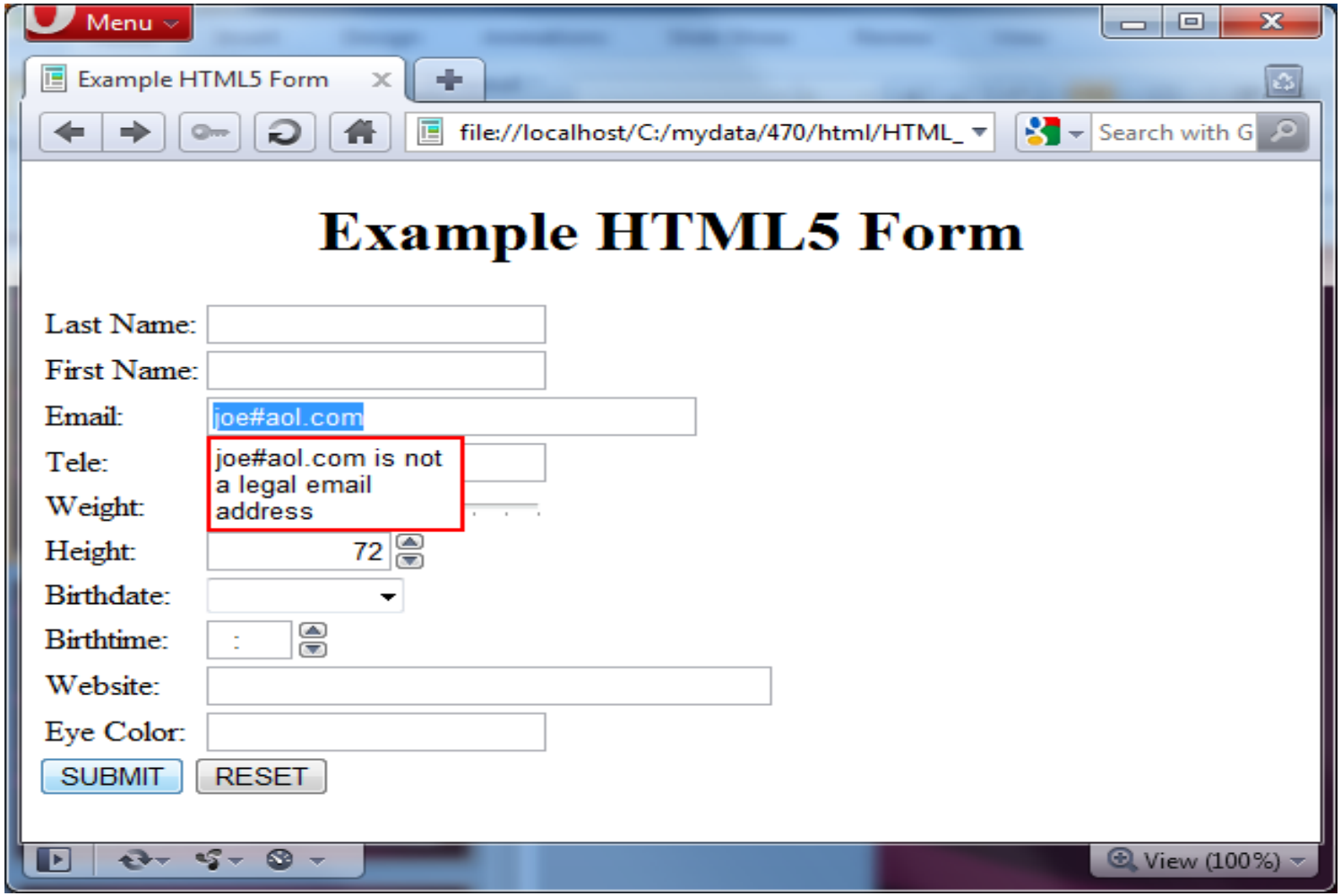
Example Form



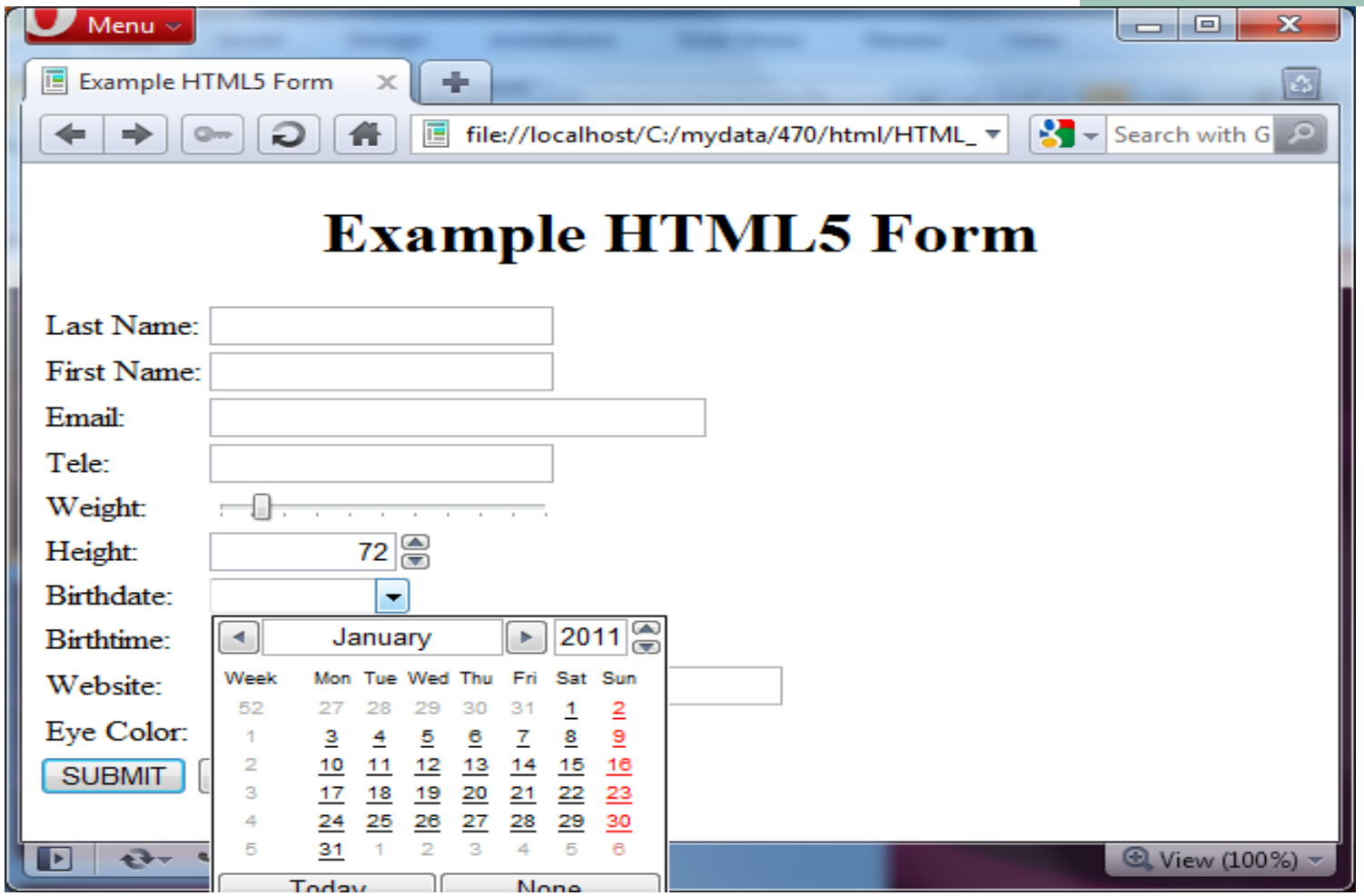
Example HTML 5 Form Code

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Example HTML5 Form</title>
	</head>
	<body>
		<h1 style="text-align:center">Example HTML5 Form</h1>
		<form method="post" action="echo.php">
		<table align="left">
			<tr>
				<td>Last Name:</td><td><input name="lname' placeholder="Enter your last name" autofocus></td>
			</tr>
			<tr>
				<td>First Name:</td><td><input name="fname' placeholder="Enter your first name"></td>
			</tr>
			<tr>
				<td>Email:</td><td><input type="email" name="email' placeholder="Enter your email address"></td>
			</tr>
			<tr>
				<td>Tele:</td><td><input type="tel" name="tele' placeholder="Enter your telephone number"></td>
			</tr>
			<tr>
				<td>Weight:</td><td><input type="range" min="50" max="1000" step="10" value="170" name="weight" placeholder="Enter your weight in
pounds"></td>
			</tr>
			<tr>
				<td>Height:</td><td><input type="number" min="36" max="96" step="2" value="72" name="height" placeholder="Enter your weight in
pounds"></td>
			</tr>
			<tr>
				<td>Birthdate:</td><td><input type="date" name="bdate' placeholder="Enter your birthdate"></td>
			</tr>
			<tr>
				<td>Birthtime:</td><td><input type="time" name="btime' placeholder="Enter your birth time (24 hour)"></td>
			</tr>
			<tr>
				<td>Website:</td><td><input type="url" name="web' placeholder="Enter your web site"></td>
			</tr>
			<tr>
				<td>Eye Color:</td><td><input type="color" name="ecolor' placeholder="Enter your eye color"></td>
			</tr>
		</table>
		<br clear="all">
		<input type="submit" value="SUBMIT">
		<input type="reset" value ="RESET">
		</form>
	</body>
</html>
```

Example Form (con't)



Example Form (con't)



Fieldset & Legend

Fieldset.html - Notepad

File Edit Format View Help

```
<HTML>
<BODY>
        <FORM>
                <FIELDSET>
                        <LEGEND>Address</LEGEND>
                                <LABEL>Street: <INPUT type="text" name="street" id="street" size=40></LABEL><BR><BR>
                                <LABEL>City: <INPUT type="text" name="city" id="city" size=30></LABEL>
                                <LABEL>State:<INPUT type="text" name="state" id="state" size=3></LABEL>
                                <LABEL>Zip:<INPUT type="text" name="zip" id="zip" size=10></LABEL><BR><BR>
                </FIELDSET>
        </FORM>
</BODY>
</HTML>
```

Address

Street:

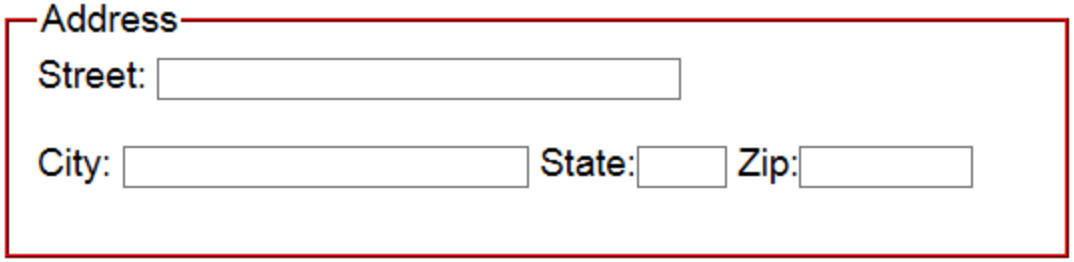
City: State: Zip:

Fieldset & Legend (adding style)

Fieldset.html - Notepad

File Edit Format View Help

```
<HTML>
<HEAD>
<STYLE>
        FIELDSET {width: 500px; border: 2px ridge #ff0000; font-family: Arial, sans-serif;  padding 10px}
</STYLE>
</HEAD>
<BODY>
        <FORM>
                <FIELDSET>
                        <LEGEND>Address</LEGEND>
                                <LABEL>Street: <INPUT type="text" name="street" id="street" size=40></LABEL><BR><BR>
                                <LABEL>City: <INPUT type="text" name="city" id="city" size=30></LABEL>
                                <LABEL>State:<INPUT type="text" name="state" id="state" size=3></LABEL>
                                <LABEL>Zip:<INPUT type="text" name="zip" id="zip" size=10></LABEL><BR><BR>
                </FIELDSET>
```



Browser Awarness

- Browsers will now know what type of data is to be entered in each type of form field, and can provide “accommodations”
- For example, the IPhone pop up keyboard is customized for each form element type
  - The keyboard for an email field would have big buttons for the “.” and “@” symbols

Pattern Attribute

- If none of these new input types suits your needs, HTML 5 provides the pattern attribute for input elements with type="text“
- The value of the pattern attribute is a regular expression, as defined in ECMAScript and used in JavaScript
- For example, if one wanted to match a five-digit or nine-digit US ZIP code or a six-character Canadian postal code, one could use this pattern:
  - ([0-9]{5}(-[0-9]{4})?)|([A-Z][0-9][A-Z]\s+[0-9][A-Z][0-9])

Pattern Attribute (con't)

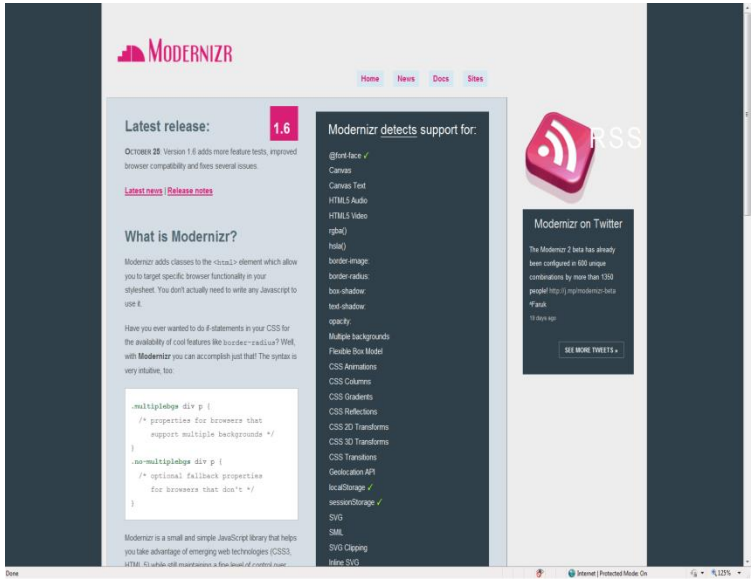
- <input type="text" name="postCode" required="required"
- pattern="([0-9]{5}(-[0-9]{4})?)|([0-9][A-Z][0-9]\s+[A-Z][0-9][A-Z])"
- title="US: 99999-1234; Canadian: 0A1 B2C" />

Drag & Drop (DnD)

- The HTML5 spec defines an event-based mechanism, JavaScript API, and additional markup for declaring that just about any type of element be “draggable” on a page
- Many apps that utilize DnD would have a poor experience without it, for example, imagine a chess game pieces that don't move
- Currently supported in latest versions of mose browsers
- Determining if a browser implements DnD is important for providing a solution that degrades nicely →

Drag & Drop (con't)

- Via Modernizr (which sets a boolean property for each feature it tests), checking for DnD is a one-liner:
  - if (Modernizr.draganddrop) { // Browser supports HTML5 DnD}
  - else { // Fallback to a another solution}



http://www.modernizr.com/

Draggable Property

- Making an object draggable is simple - set the draggable=true attribute on the element you want to make moveable
- Just about anything can be drag-enabled, including images, links, files, or other DOM nodes:
  - …
- In most browsers, text selections, img elements, and anchor elements with an href attribute are draggable by default
  - Most browsers support dragging an image which can be dropped in the address bar, a <input type="file" /> element, or even the desktop; if you want to enable other types of content to be draggable, you'll need to use HTML5 DnD methods

Dragging Events

- ondragstart, ondrop, ondragend are used to signal the start end and end of the drag process and the actual drop (release)
- ondragenter, ondragover, and ondragleave event handlers are used to provide additional visual cues during the drag process
- Example:
  - …

dataTransfer Object

- The dataTransfer object is a part of the event object
- It has a property named effectAllowed that allows one to specify what DnD operation is allowed
- It also has functions named setData() and getData() that allows one to specify what data is to be dragged with the object, and also a function named setDragImage() that allows one to specify the image of the item being dragged
- For example:
  - e.dragTransfer.effectAllowed='move';
  - e.dragTransfer.setData("Text", e.target.getAttribute('id'));
  - e.dragTransfer.setDragImage(e.target, 0, 0);

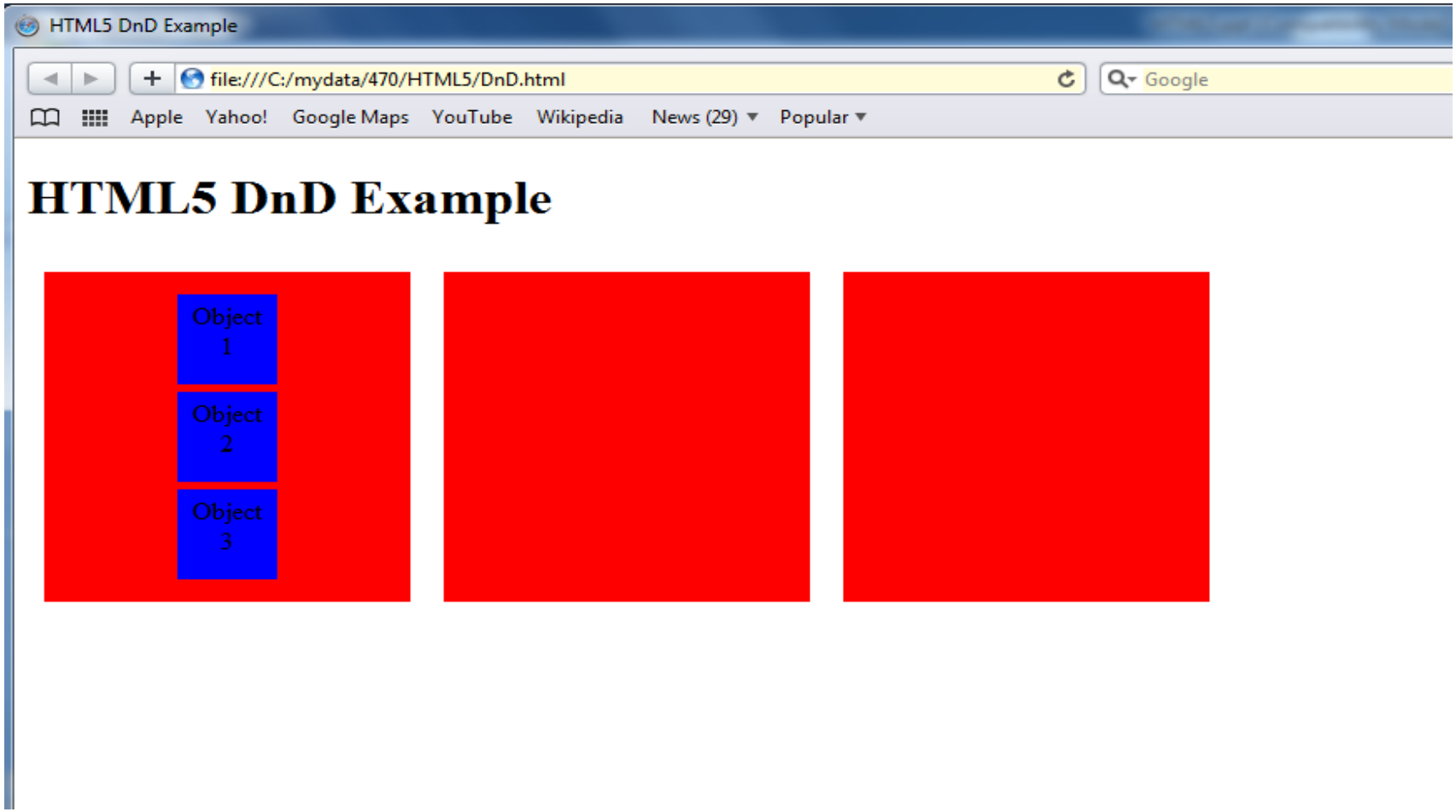
Drag & Drop (con't)

Events
- dragstart
- dragend

Attributes:
- draggable="true"

Events
- dragenter
- dragover
- dragleave
- drop

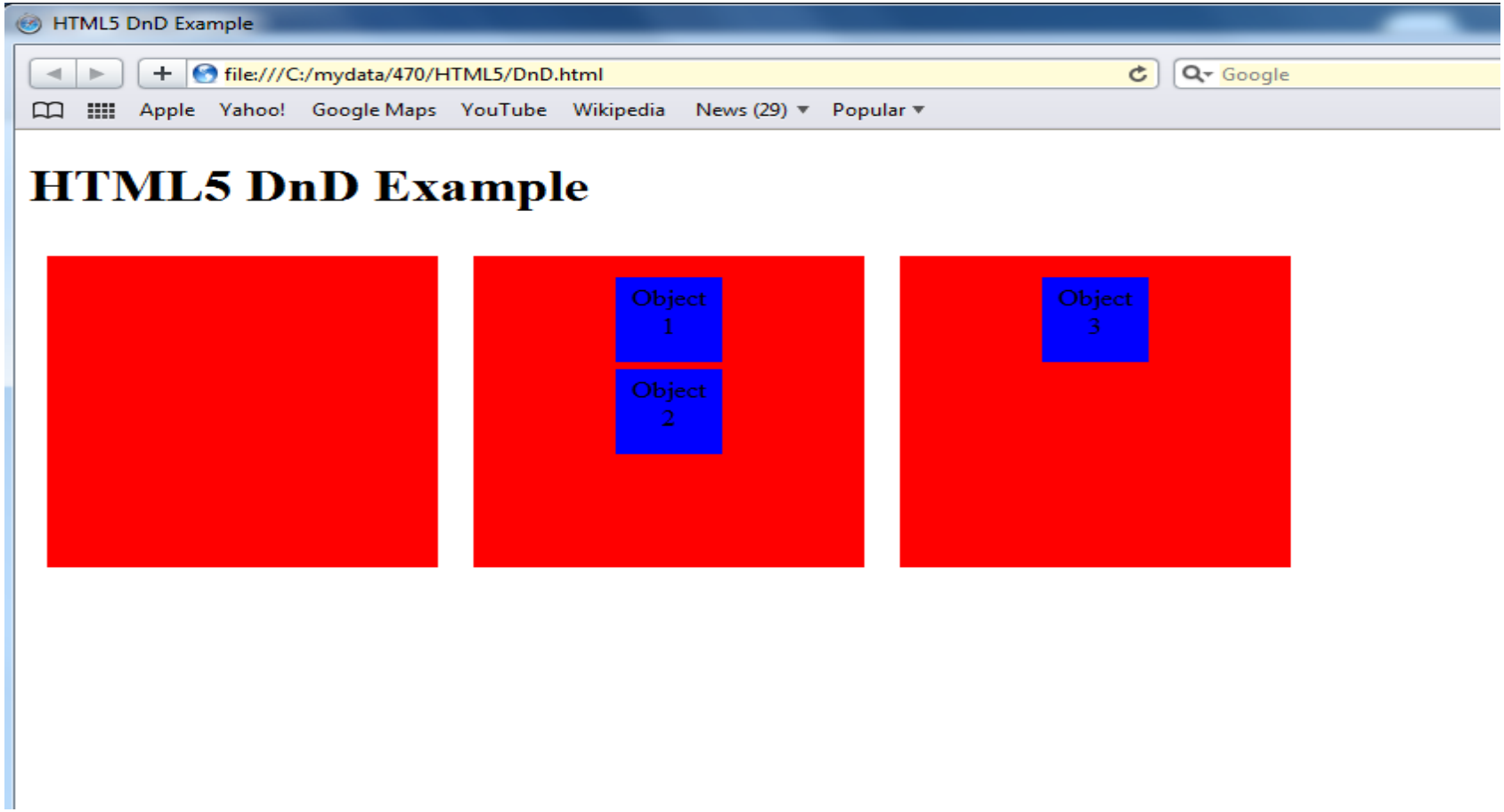
DnD Example - Before



DnD Example - Code

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>HTML5 DnD Example</title>
		<script type="text/javascript">
			function start(e) {
				e.dataTransfer.effectAllowed='move';
				var idz = e.target.getAttribute("id");
				e.dataTransfer.setData("Text",idz);
				e.dataTransfer.setDragImage(e.target, 0, 0); /* offset of 0,0 */
				return true;
			}
			function enter(e) {return true;}
			function over(e) {
				/* if a value of true is returned, the dragged object may not be dropped; if false, it can be dropped */
				var id = e.target.getAttribute('id');
				if (id == 'target1') return false; /* any item may be dropped on target1 */
				var idx = e.dataTransfer.getData("Text");
				/* draggable object 3 may only be dropped on target 3 */
				if (id == 'target3' && idx == 'dragobject3') return false;
				if (id == 'target2' && (idx == 'dragobject1' || idx == 'dragobject2')) return false;
				return true;
			}
			function drop(e) {
				var idx = e.dataTransfer.getData("Text");
				e.target.appendChild (document.getElementById(idx)); /* append draggable <div> to target <div> */
				e.stopPropagation(); /* stop bubble up of event */
				return false;
			}
			function end(e) {
				e.dataTransfer.clearData("Text"); /* clear data in dataTransfer object */
				return true;
			}
		</script>
		<style type="text/css">
			/* set visual properties of target areas */
			#target1, #target2, #target3 {background-color:red; float:left; width:200px; height:200px; padding:10px; margin:10px;}
			/* set visual properties of draggable objects */
			#dragobject1, #dragobject2, #dragobject3 {text-align:center; margin-left:auto; margin-right:auto; background-color:blue; width:50px; height:50px; padding:5px; margin-
top:5px;}
		</style>
	</head>
	<body>
		<h1>HTML5 DnD Example</h1>
		<div id="target1" ondragenter="return enter(event)" ondragover="return over(event)" ondrop="return drop(event)">
			<div id="dragobject1" draggable="true" ondragstart="return start(event)" ondragend="return end(event)">Object 1</div>
			<div id="dragobject2" draggable="true" ondragstart="return start(event)" ondragend="return end(event)">Object 2</div>
			<div id="dragobject3" draggable="true" ondragstart="return start(event)" ondragend="return end(event)">Object 3</div>
		</div>
		<div id="target2" ondragenter="return enter(event)" ondragover="return over(event)" ondrop="return drop(event)"></div>
		<div id="target3" ondragenter="return enter(event)" ondragover="return over(event)" ondrop="return drop(event)"></div>
	</body>
</html>
```

After DnD


HTML5 DnD Example
file:///C:/mydata/470/HTML5/DnD.html
Google
Apple
Yahoo!
Google Maps
YouTube
Wikipedia
News (29)
Popular
HTML5 DnD Example
Object 1
Object 2
Object 3

HTML5 Inline Editing

- Make elements editable
- Make an entire document editable
- Formatting and spell check (currently Firefox only)
- Currently supported in: Chrome, Firefox, IE 9+, Safari, Opera
- Enables web developers to build rich text editors

Editing Attributes

- contenteditable
  - true
    - <div id="editME“ contenteditable="true“ >…
  - false
  - inherit
- designmode
  - on
  - off
- spellcheck
  - true
  - false

Document Interaction API

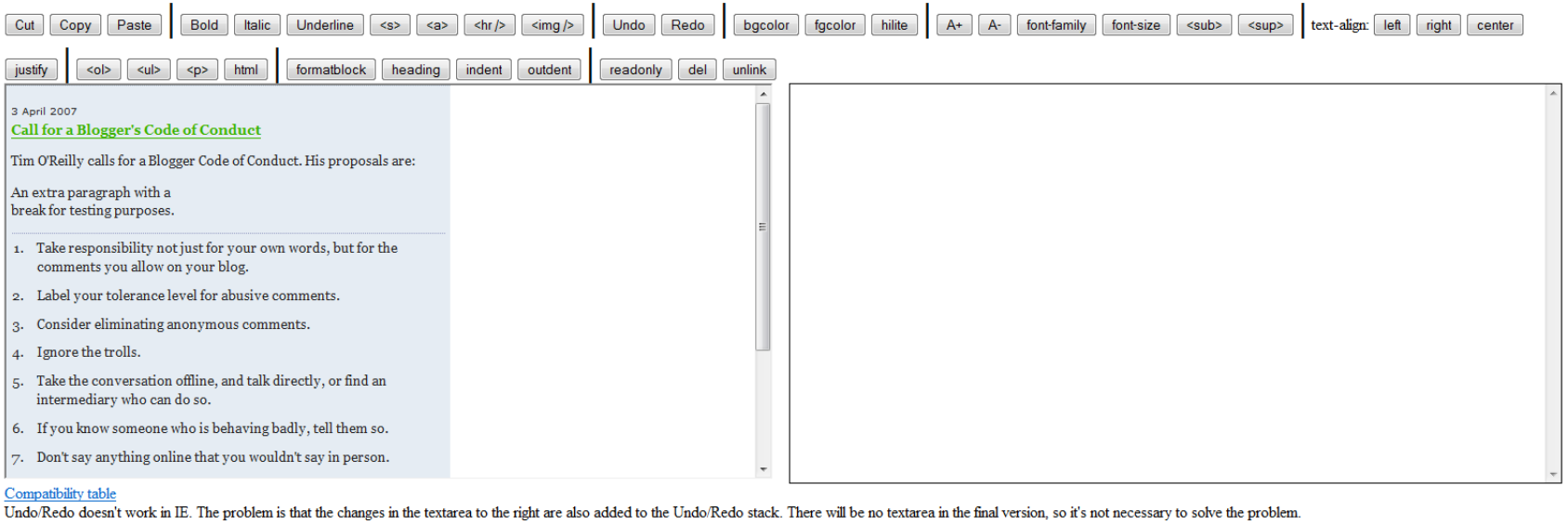
- The API for interacting with the document is:
  - document.execCommand - executes the given command
  - document.queryCommandEnabled - determines whether the given command can be executed on the document in its current state
  - document.queryCommandIndeterm - determines whether the current selection is in an undetermined state
  - document.queryCommandState - determines whether the given command has been executed on the current selection
  - document.queryCommandValue - determines the current value of the document, range, or current selection for the given command

Formatting

- To change the format of selected text:
  - Object.execCommand(CMD, INTF, VALUE)
    - CMD – command to execute
    - INTF – show user interface (true or false) [default false]
    - VALUE – value to assign [optional]
  - onclick="document.execCommand('bold', false, null);”
- Only basic stuff like bold, italic, creating links, and changing colors is well-supported across browsers; after that, there are compatibility issues

Test Page of Commands

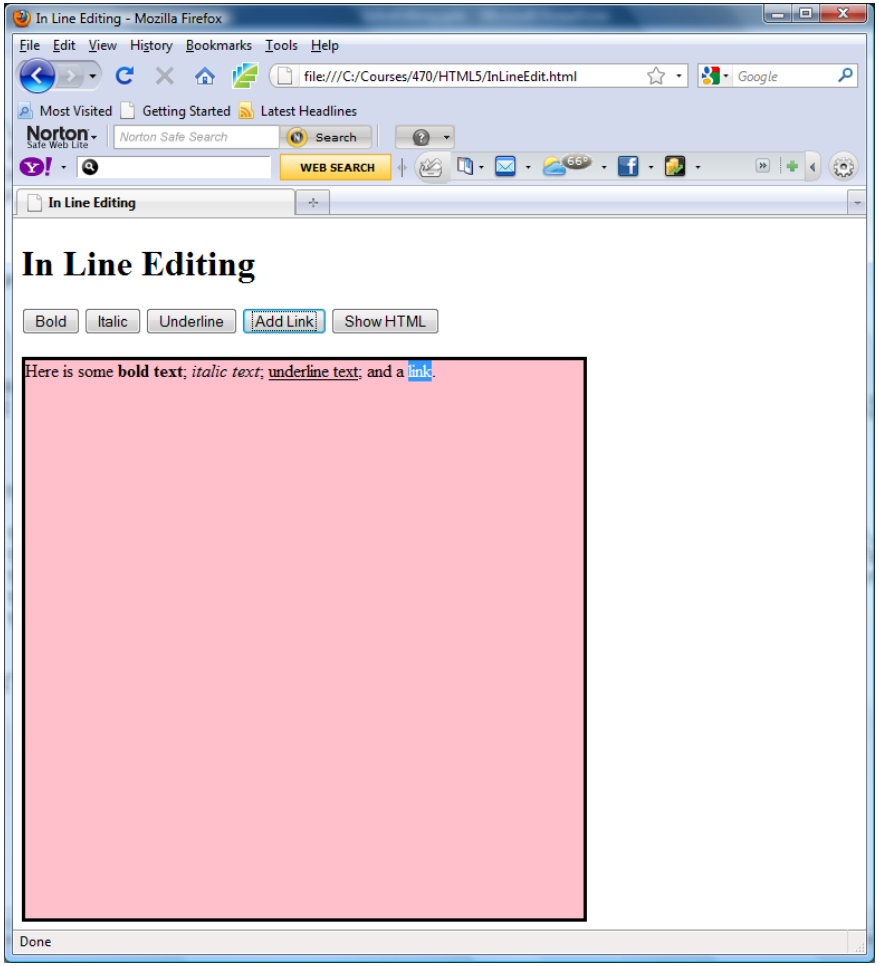
[http://www.quirksmode.org/dom/execCommand/]



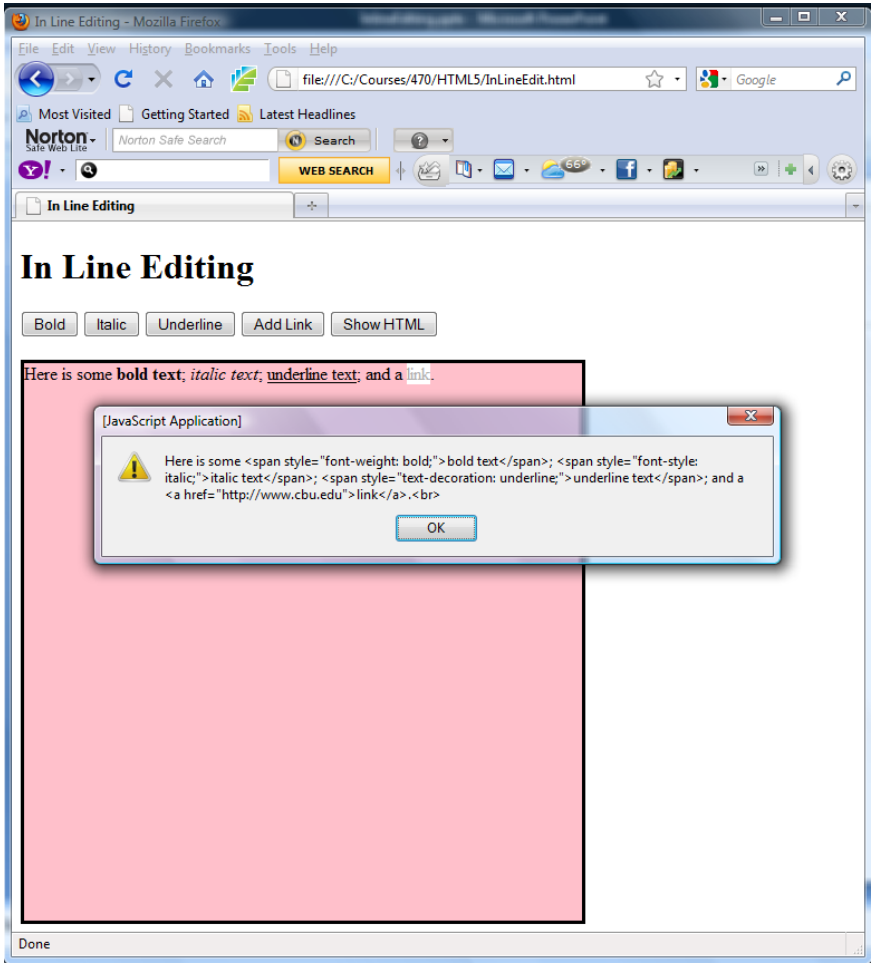
Compatibility table

Undo/Redo doesn't work in IE. The problem is that the changes in the textarea to the right are also added to the Undo/Redo stack. There will be no textarea in the final version, so it's not necessary to solve the problem.

Inline Editing Example



Hitting “Show HTML” Button



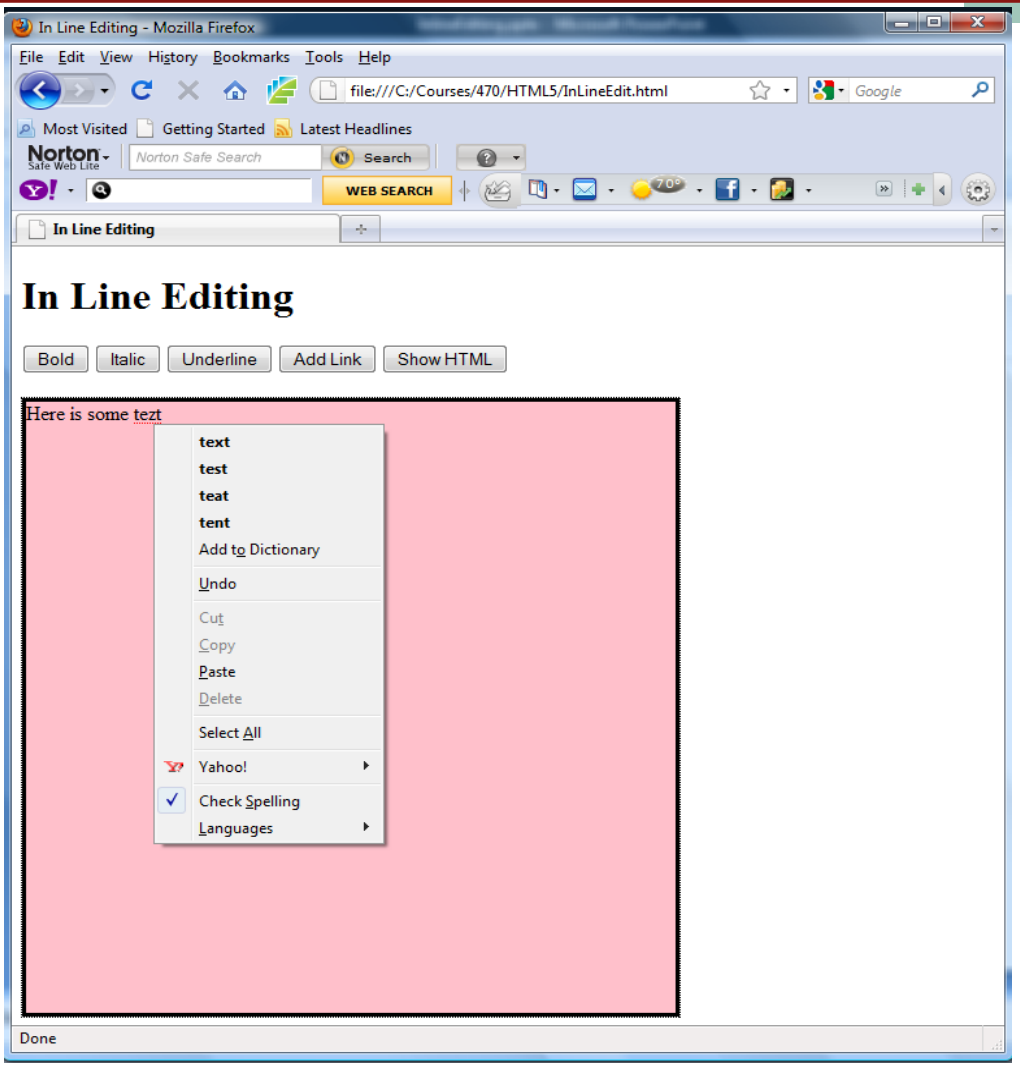
Inline Editing Example (con't)

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>In Line Editing</title>
		<script type="text/javascript">
			function createLink() {
				var u = prompt("Enter URL:", "http://");
				if (u) document.execCommand ("createlink", false, u);
			}
			function showHTML() {
				var x = document.getElementById("editME").innerHTML;
				alert(x);
			}
		</script>
		<style type="text/css">
			#editME {border:solid black; background-color:pink; height:500px; width:500px}
		</style>
	</head>
	<body>
		<h1>In Line Editing</h1>
		<div>
			<input type="button" value="Bold" onclick="document.execCommand('bold', false, null);">
			<input type="button" value="Italic" onclick="document.execCommand('italic', false, null);">
			<input type="button" value="Underline" onclick="document.execCommand('underline', false, null);">
			<input type="button" value="Add Link" onclick="createLink();">
			<input type="button" value="Show HTML" onclick="showHTML();">
		</div>
		<br>
		<div id="editME" contenteditable="true"></div>

	</body>
</html>
```

SpellCheck

- On by default in Firefox
- Suspected spelling errors are underlined – right click to see list of possible corrections
- <div id="editME“ contenteditable="true“ spellcheck=“true” >

SpellCheck (con't)



Design Mode

- The designMode attribute governs the entire document (i.e. it makes the entire document editable, like a dedicated HTML editor)
- The contentEditable attribute governs just the element on which it appears, and that element's children
- Enabling designMode causes scripts in general to be disabled and the document to become editable
- When the Document has designMode enabled, event listeners registered on the document or any elements owned by the document must do nothing

HTML5 Messaging

- Cross window or cross domain
- One window (or iframe) can send messages to another
- Current browser support: Opera, Safari
- Messaging API:
  - postMessage() function:
    - window.postMessage(message, target)
  - onMessage event
  - Event attributes:
    - event.data, event.origin, event.source

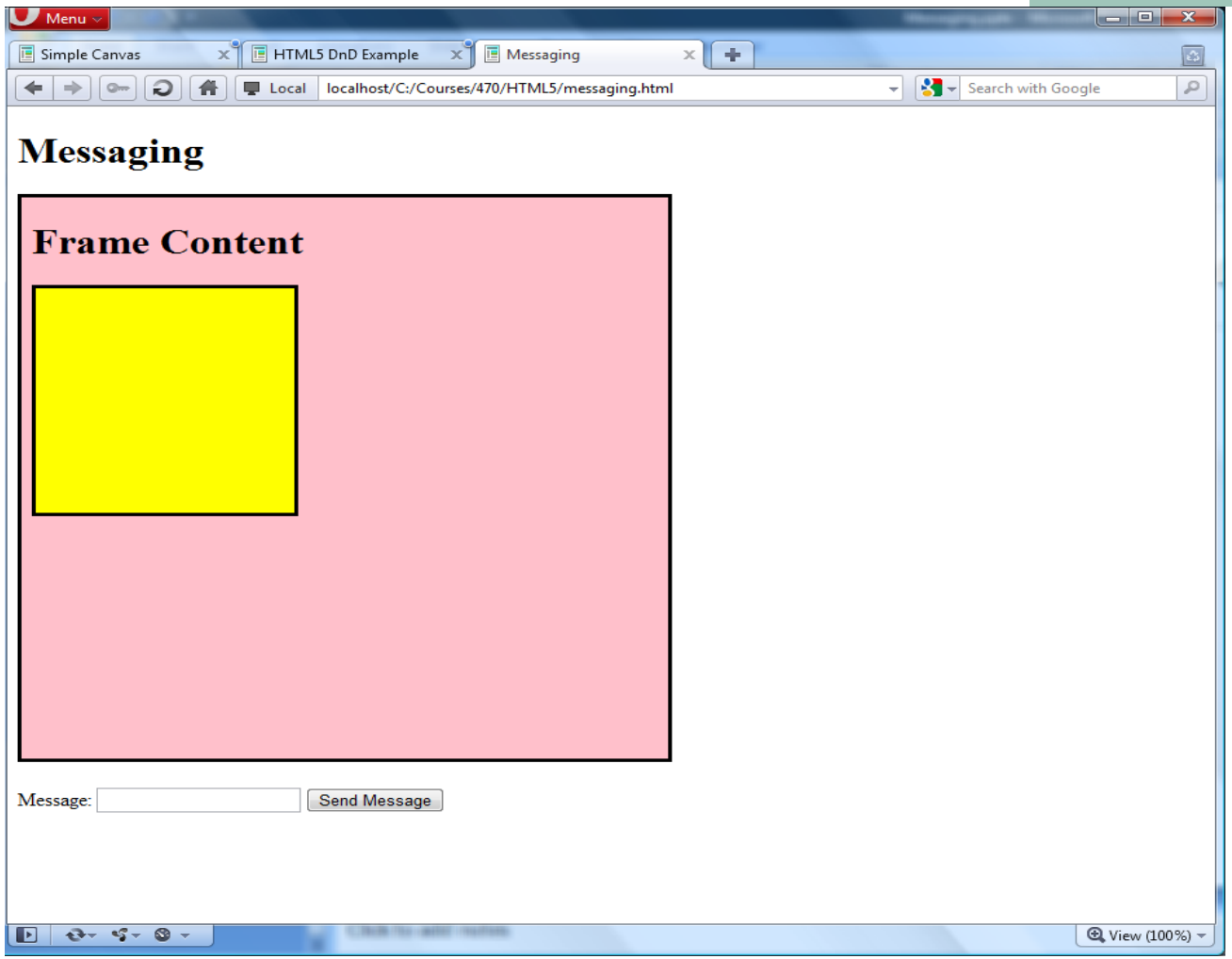
Messaging Example

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title>Messaging</title>
                <script type="text/javascript">
                        function send() {
                                var m = document.getElementById("msgText").value;   // message
                                var l = window.location;
                                var p = l.protocol;
                                var h = l.host;
                                var t = p + "//" + h; // target
                                document.getElementById("frame1").contentWindow.postMessage(m,t)
                        }

                </script>
                <style type="text/css">
                        #frame1 {border:solid black; background-color:pink; height:500px; width:500px}
                </style>
        </head>
        <body>
                <h1>Messaging</h1>
                <iframe id="frame1" src="frameContent.html"></iframe>
                <br><br>
                <form>
                        Message: <input id="msgText" type="text">
                        <input type="button" value = "Send Message" onclick="send();">
                </form>
        </body>
</html>
```

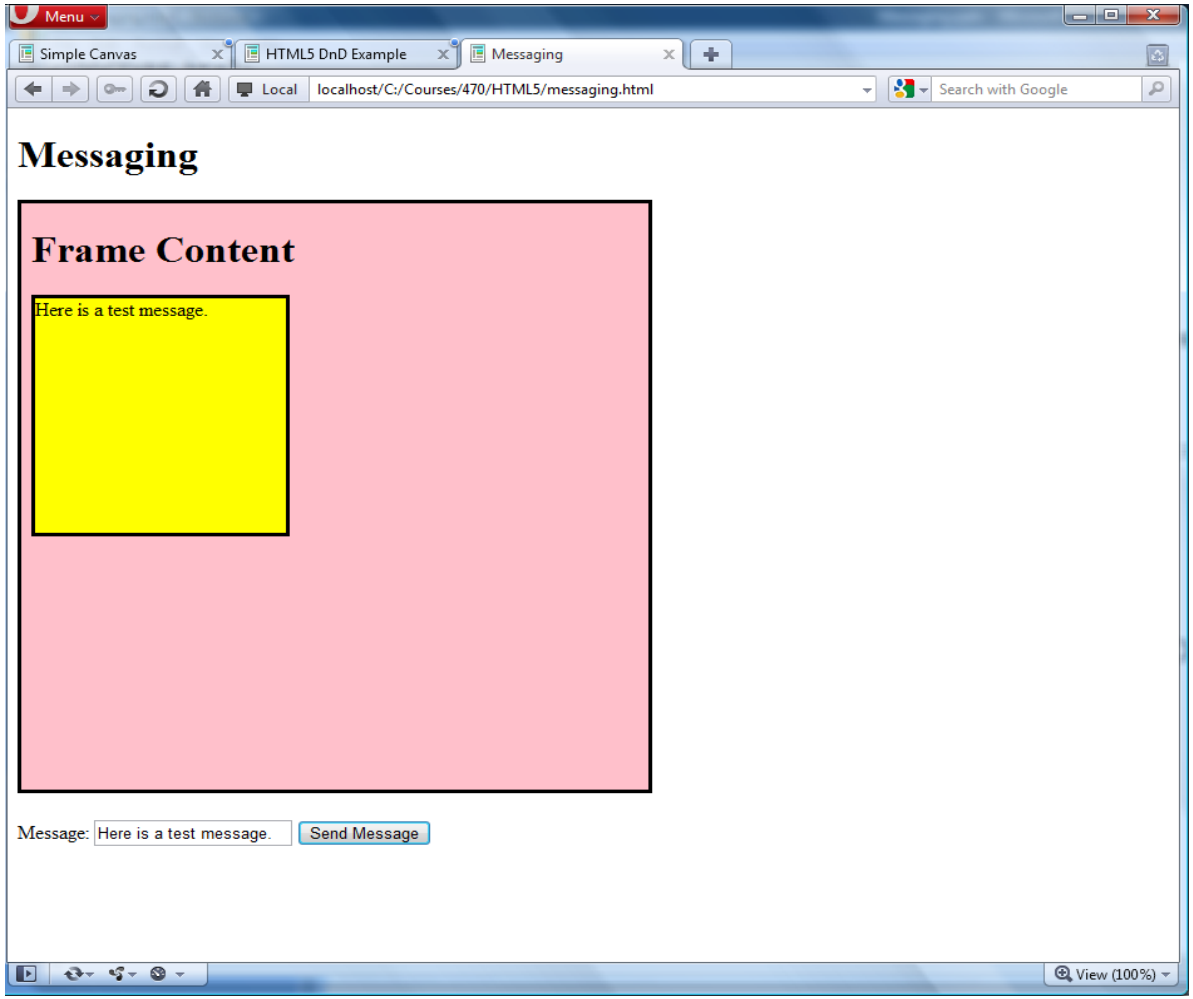
Frame Content (HTML)

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title>Frame Content</title>
                <script type="text/javascript">
                        window.addEventListener("message", loadMsg, false)
                        function loadMsg(e) {
                                document.getElementById("msgTarget").innerHTML = e.data;
                        }
                </script>
                <style type="text/css">
                        #msgTarget {border:solid black; background-color:yellow; height:200px;
width:200px}
                </style>
        </head>
        <body>
                <h1>Frame Content</h1>
                <div id="msgTarget"></div>

        </body>
</html>
```

Before Sending Message



After entering message, and hitting send button…



Offline Storage

- Web Storage (key/value pairs, currently values must be strings [can serialize via JSON API])
  - Session Storage
  - Local Storage
- File API
- Web SQL Database (based on SQLite with transaction support)
- IndexedDB (with transactions)

JSON API for Javascript

JSON in JavaScript

JavaScript is a general purpose programming language that was introduced as the page scripting language for Netscape Navigator. It is still widely believed to be a subset of Java, but it is not. It is a Scheme-like language with C-like syntax and soft objects. JavaScript was standardized in the ECMAScript Language Specification, Third Edition.

JSON is a subset of the object literal notation of JavaScript. Since JSON is a subset of JavaScript, it can be used in the language with no muss or fuss.

```
var myJSONObject = {"bindings": [
        {"ircEvent": "PRIVMSG", "method": "newURI", "regex": "^http://.*"},
        {"ircEvent": "PRIVMSG", "method": "deleteURI", "regex": "^delete.*"},
        {"ircEvent": "PRIVMSG", "method": "randomURI", "regex": "^random.*"}
    ]
};
```

In this example, an object is created containing a single member "bindings", which contains an array containing three objects, each containing "ircEvent", "method", and "regex" members.

Members can be retrieved using dot or subscript operators.

```
myJSONObject.bindings[0].method    // "newURI"
```

To convert a JSON text into an object, you can use the eval() function. eval() invokes the JavaScript compiler. Since JSON is a proper subset of JavaScript, the compiler will correctly parse the text and produce an object structure. The text must be wrapped in parens to avoid tripping on an ambiguity in JavaScript's syntax.

```
var myObject = eval('(' + myJSONtext + ')');
```

The eval function is very fast. However, it can compile and execute any JavaScript program, so there can be security issues. The use of eval is indicated when the source is trusted and competent. It is much safer to use a JSON parser. In web applications over XMLHttpRequest, communication is permitted only to the same origin that provide that page, so it is trusted. But it might not be competent. If the server is not rigorous in its JSON encoding, or if it does not scrupulously validate all of its inputs, then it could deliver invalid JSON text that could be carrying dangerous script. The eval function would execute the script, unleashing its malice.

To defend against this, a JSON parser should be used. A JSON parser will recognize only JSON text, rejecting all scripts. In browsers that provide native JSON support, JSON parsers are also much faster than eval. It is expected that native JSON support will be included in the next ECMAScript standard.

```
var myObject = JSON.parse(myJSONtext, reviver);
```

The optional reviver parameter is a function that will be called for every key and value at every level of the final result. Each value will be replaced by the result of the reviver function. This can be used to reform generic objects into instances of pseudoclasses, or to transform date strings into Date objects.

```
myData = JSON.parse(text, function (key, value) {
    var type;
    if (value && typeof value === 'object') {
        type = value.type;
        if (typeof type === 'string' && typeof window[type] === 'function') {
            return new (window[type])(value);
        }
    }
    return value;
});
```

Done Internet | Protected Mode: On 100%

Session Storage

- When a browser connects to a server, a new “session” is started
- In earlier versions of HTML, when the user closed the browser window or set focus to another window, the session was closed
- In order to save data from a session, that data needed to be stored locally in cookies or up on the server (in the server’s session memory or on server files) via server programming
- With HTML5 and JavaScript, the browser can store data in a server session as long as there is not 15 minutes of inactivity (since window closed)

sessionStorage Object

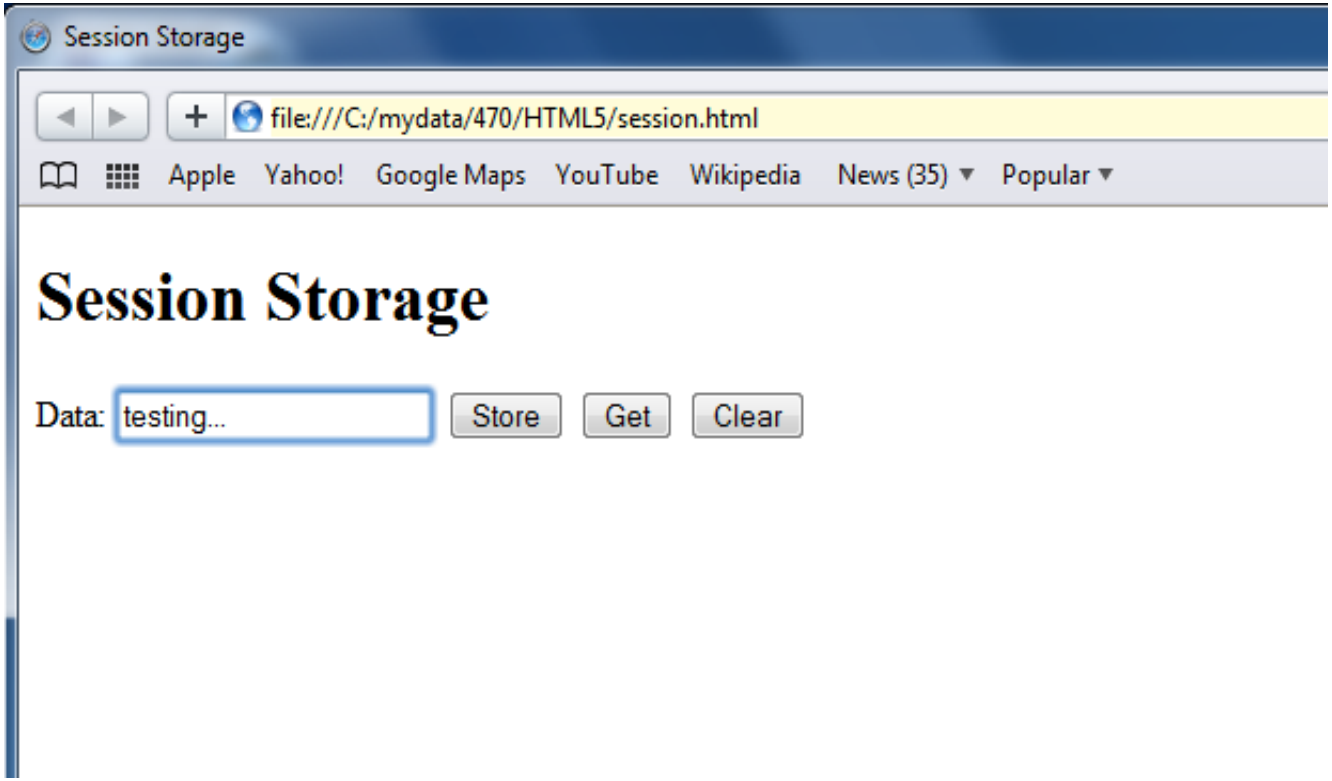
- Attribute:
  - length – number of key/value pairs
- Methods:
  - key(n) – return the name of the n’th key
  - getItem(key) – return value for key
  - setItem(key, value) – set value for key
  - removeItem(key) – remove a key/value
  - clear() – clear all session data
- Currently supported in: Firefox, Safari

Example Session Storage

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title>Session Storage</title>
                <script type="text/javascript">
                        function sStore() {
                                var x = document.getElementById("data").value;
                                sessionStorage.setItem("Data", x);

                        }
                        function sGet() {
                                var x = sessionStorage.getItem("Data");
                                document.getElementById ("data").value = x;
                        }
                        function sClear() {
                                sessionStorage.removeItem("Data");
                                document.getElementById("data").value = "";
                        }
                </script>
        </head>
        <body>
                <h1>Session Storage</h1>
                <form>
                        Data: <input id="data" type="text">
                        <input type="button" Value="Store" onclick="sStore();">
                        <input type="button" Value="Get" onclick="sGet();">
                        <input type="button" Value="Clear" onclick="sClear();">
                </form>
        </body>
</html>
```

Example Session Storage (con't)

[to create a true session, one needs to load this from a server]



Type in test info, hit "Store", erase data, hit "Get", hit "Clear"

Local Storage

- With HTML5, the browser can also store data locally until the user closes the browser (the user can navigate away from the browser window, and return later to get the data)
- HTML5 provides a mechanism to detect if your’re online:
  - Navigator.onLine()
- You can also “catch events”
  - document.body.addEventListner(“online”, function() {…})
  - document.body.addEventListner(“offline”, function() {…})

localStorage Object

- Attribute:
  - length – number of key/value pairs
- Methods:
  - key(n) – return the name of the n’th key
  - getItem(key) – return value for key
  - setItem(key, value) – set value for key
  - removeItem(key) – remove a key/value
  - clear() – clear all local data
- Currently supported in: Firefox, Safari, Chrome

Local Storage Example

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title>Local Storage</title>
                <script type="text/javascript">
                        function lStore() {
                                var x = document.getElementById("data").value;
                                localStorage.setItem("Data", x);

                        }
                        function lGet() {
                                var x = localStorage.getItem("Data");
                                document.getElementById ("data").value = x;
                        }
                        function lClear() {
                                localStorage.removeItem("Data");
                                document.getElementById("data").value = "";
                        }
                </script>
        </head>
        <body>
                <h1>Local Storage</h1>
                <form>
                        Data: <input id="data" type="text">
                        <input type="button" Value="Store" onclick="lStore();">
                        <input type="button" Value="Get" onclick="lGet();">
                        <input type="button" Value="Clear" onclick="lClear();">
                </form>
        </body>
</html>
```

Local Storage Example (con't)

Type in test info, hit "Store", erase data, hit "Get", hit "Clear"

FileSystem API

- With HTML5 browsers can read and write files and directories
- With the FileSystem API, a web app can create, read, navigate, and write to a sandboxed section of the user's local file system
- The API is broken up into various themes:
  - Reading and manipulating files: File/Blob, FileList, FileReader
  - Creating and writing: BlobBuilder, FileWriter
  - Directories and file system access: DirectoryReader, FileEntry/DirectoryEntry, LocalFileSystem

Browser Support

- Currently Google Chrome 9+ has the only working implementation of the FileSystem API
- Since a dedicated browser UI does not yet exist for file/quota management, the API cannot be used without running Chrome with the --unlimited-quota-for-files flag
- This means there is currently no storage cap in place for apps, but that will change, and users will eventually receive a permission dialog to grant, deny, or increase storage for an app
- You may need the --allow-file-access-from-files flag if you're debugging your app from file://
  - Not using these flags will result in a SECURITY_ERR or QUOTA_EXCEEDED_ERR FileError

HTML5 Offline Applications

- An offline web application uses a list of resources for which it keeps a local copy
- When connection is established, the resources are updated; and when disconnected the local (last downloaded) version is used
- The resource list is kept in a “manifest” file for each web page, which is a text file starting with “CACHE MANIFEST”; for example:
  - CACHE MANIFEST
    - abc.css
    - def.js
    - xyz.jpg

HTML5 Offline Applications (con't)

- The offline capable web page has a declaration of its manifest in the HTML tag:
  - <!DOCTYPE HTML>
  - <html manifest="/abc.manifest">
- Manifest files must be served as content type text/cache-manifest; in Apache, place this directive in the .htaccess file at the root of the web directory:
  - Addtype text/cache-manifest .manifest
- The manifest file can also have sections for "fallback" files (alternatives) and "network" files (never cached) as well as "cache"

Google Docs & Offline/Local Storage

- What if you lose your data or WiFi connection?
- Do you lose all of your precious work?
- Nope. Thankfully, the Google Docs mobile Web app makes smart use of HTML5's offline storage feature
- If you're without a signal, the application still works; it just can't save your work
- Thus you can continue to type away and make changes, and the app will sync and save your document the next time the device is online
  - Just be careful about trying to load any other new pages or data in the meantime, as there's a chance doing so could cause you to lose your changes

Geolocation

- Determining where you are, and possibly sharing that info
- Where you are ?
  - GPS latitude and longitude
  - IP address
  - Wireless connection network
  - Which cell tower you are using
- Use your location to find other users, business, or things that may be near to you
- Browser support: Firefox, Safari, Chrome

Position Object

- navigator.geolocation.getCurrentPosition (callBack, handleError), such as:
  - function get_loc () {
    - if (modernizr.geolocation) {navigator.geolocation.getCurrentPosition (callBack);} else {
      - // no native support, use another API }
  - }
- For security purposes, must be approved by the browser user in some way such as with pop-ups

Call Back Functions

- function callBack (pos) {
  - var tsp = pos.timestamp;
  - var lat = pos.coords.latitude;
  - var lon = pos.coords.longitude;
  - var alt = pos.coords.altitude;
  - // do something with these variables
- }
- function handleError (err) {
  - If (err == 1) {// user denied}
  - else {// position not available}
- }

Methods for Location Determination

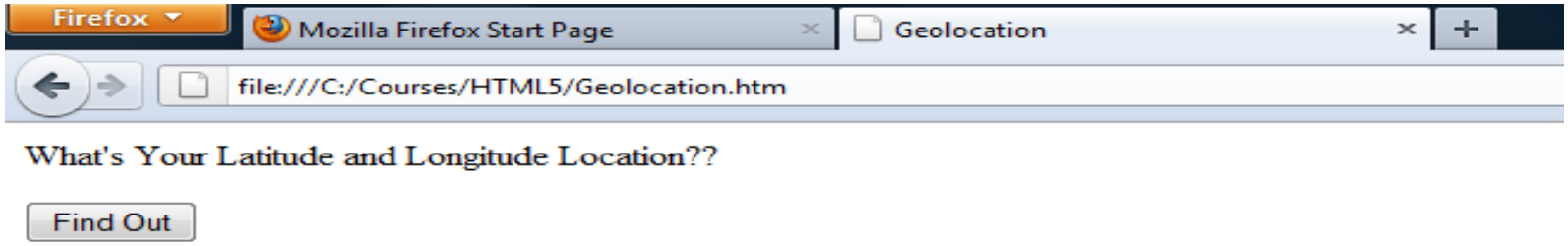
- Smart phones (such as iPhone and Androids) have two methods to determine position:
  - GPS
    - Needs GPS hardware in the device
    - Needs initialization time
    - Uses more power
  - Cell tower triangularization
    - Fast
    - Position is approximated
- The getCurrentPosition function has an optional third argument that can specify which options to use and timeouts

Geolocation Example

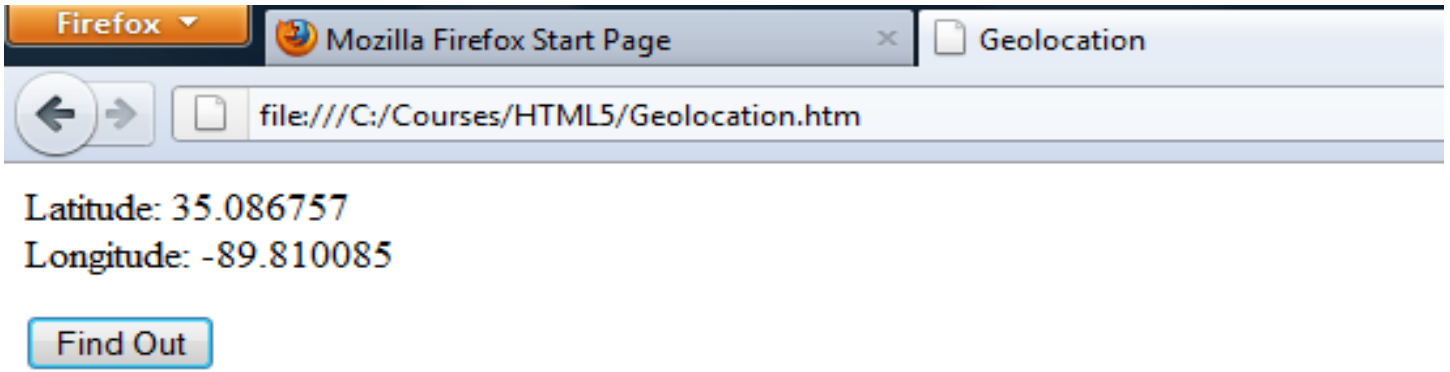
- <HTML><HEAD><TITLE>Geolocation</TITLE></HEAD>
- <BODY>
- What's Your Latitude and Longitude Location??
- <button onclick="getLocation()">Find Out</button>
  - <script>
  - var x=document.getElementById("geography");
  - function getLocation() {
  - if (navigator.geolocation) { navigator.geolocation.getCurrentPosition(showPosition); } else{x.innerHTML="Geolocation is not supported by this browser.";} }
  - function showPosition(position) { x.innerHTML="Latitude: " + position.coords.latitude + "
Longitude: " + position.coords.longitude; }
  - </script>
- <BODY></BODY></HTML>

Geolocation Example (con't)

- This code works locally, but geolocation is no longer working from the server unless you're on an HTTPS secured domain (not working From CBUstudent server)



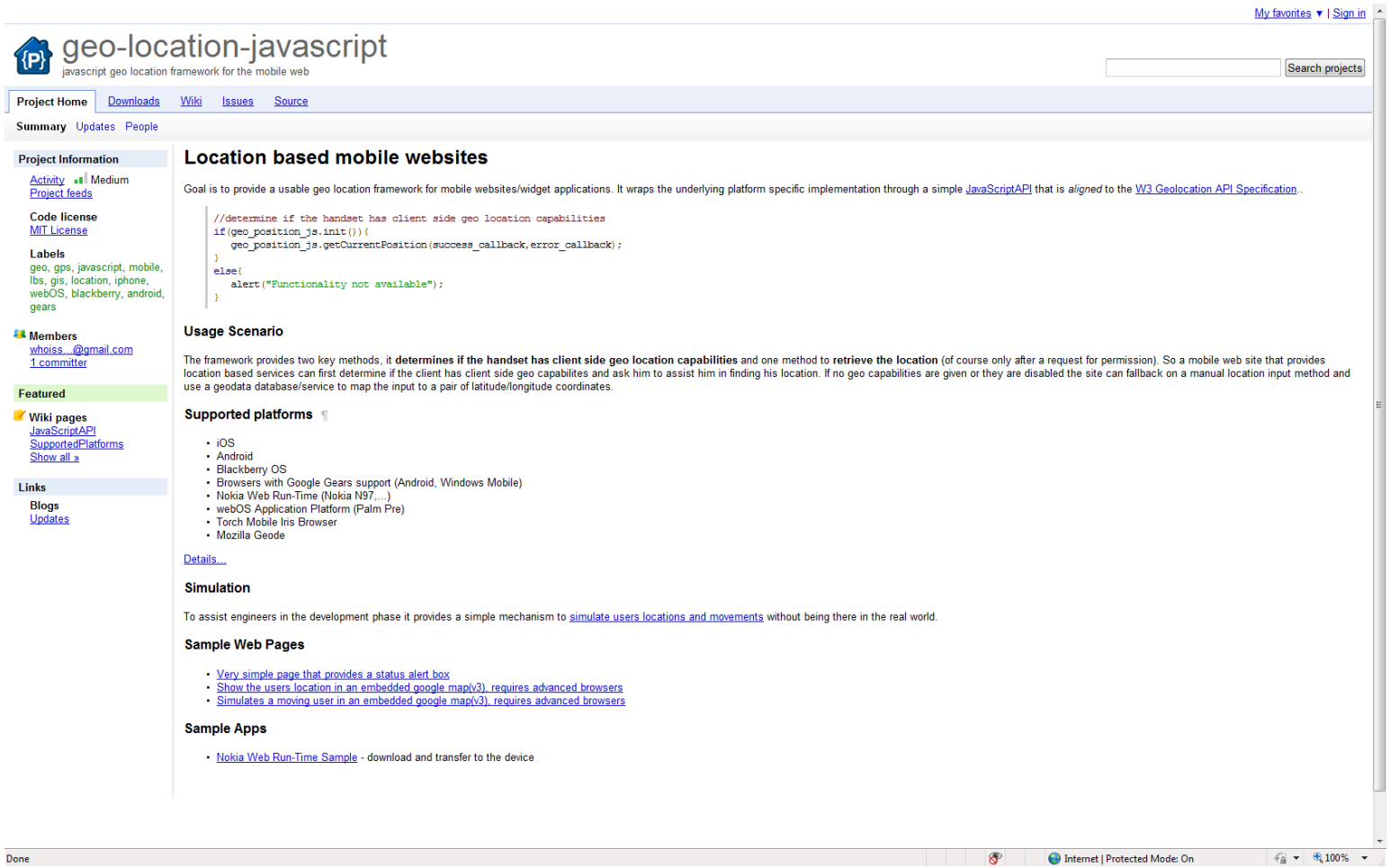
Geolocation Example (con't)



Other Geolocation API's

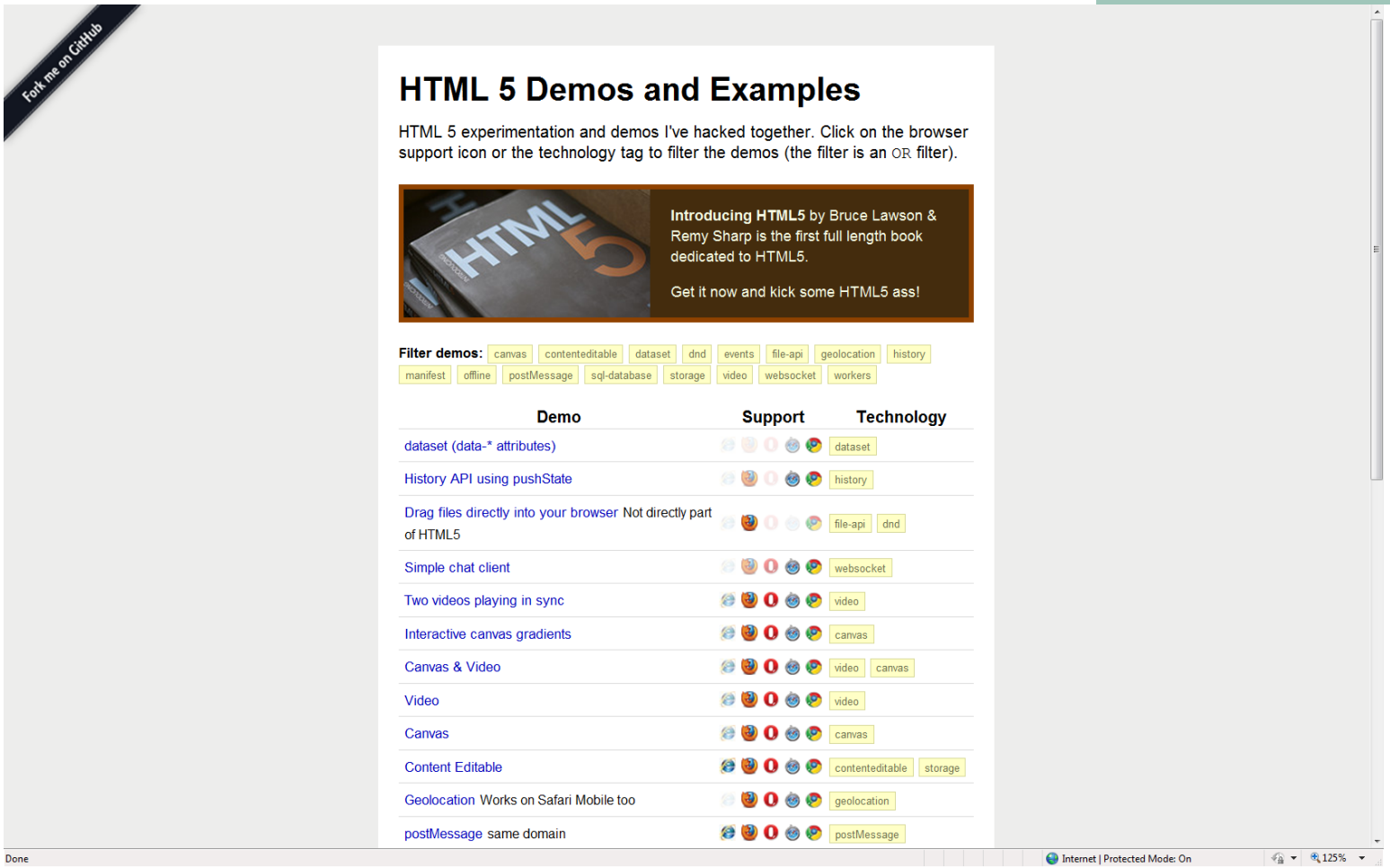
- Google Gears – works for IE also
- Device specific API's
  - Blackberry
  - Nokia
  - Palm
  - Etc.
- geo.js is open source and available to resolve differences between W3C geolocation, Gears, and other device API's

geo.js


My favorites ▼ | Sign in
geo-location-javascript
javascript geo location framework for the mobile web
Search projects
Project Home
Downloads
Wiki
Issues
Source
Summary
Updates
People
Project Information
Activity
Medium
Project feeds
Code license
MIT License
Labels
geo, gps, javascript, mobile, lbs, gis, location, iphone, webOS, blackberry, android, gears
Members
whoiss...@gmail.com
1 committer
Featured
Wiki pages
JavaScriptAPI
SupportedPlatforms
Show all »
Links
Blogs
Updates
Location based mobile websites
Goal is to provide a usable geo location framework for mobile websites/widget applications. It wraps the underlying platform specific implementation through a simple JavaScriptAPI that is aligned to the W3 Geolocation API Specification..
//determine if the handset has client side geo location capabilities
if(geo_position_js.init()){
geo_position_js.getCurrentPosition(success_callback,error_callback);
}
else{
alert("Functionality not available");
}
Usage Scenario
The framework provides two key methods, it determines if the handset has client side geo location capabilities and one method to retrieve the location (of course only after a request for permission). So a mobile web site that provides location based services can first determine if the client has client side geo capabilites and ask him to assist him in finding his location. If no geo capabilities are given or they are disabled the site can fallback on a manual location input method and use a geodata database/service to map the input to a pair of latitude/longitude coordinates.
Supported platforms
iOS
Android
Blackberry OS
Browsers with Google Gears support (Android, Windows Mobile)
Nokia Web Run-Time (Nokia N97,...)
webOS Application Platform (Palm Pre)
Torch Mobile Iris Browser
Mozilla Geode
Details...
Simulation
To assist engineers in the development phase it provides a simple mechanism to simulate users locations and movements without being there in the real world.
Sample Web Pages
Very simple page that provides a status alert box
Show the users location in an embedded google map(v3), requires advanced browsers
Simulates a moving user in an embedded google map(v3), requires advanced browsers
Sample Apps
Nokia Web Run-Time Sample - download and transfer to the device
Done
Internet | Protected Mode: On
100%

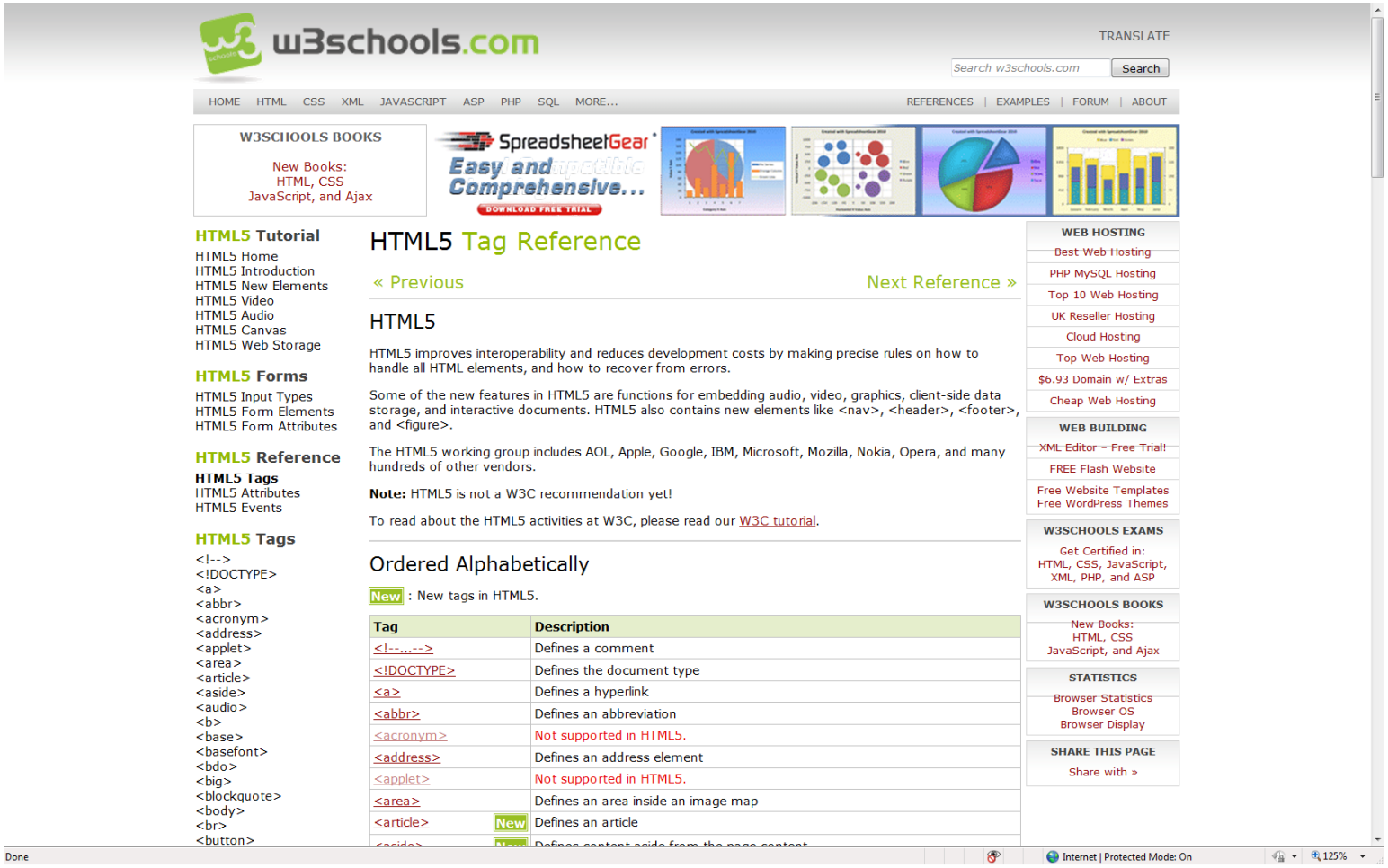
HTML5 Demos/Examples

[http://html5demos.com/]



HTML5 Tag Reference

[http://www.w3schools.com/html5/html5_reference.asp]

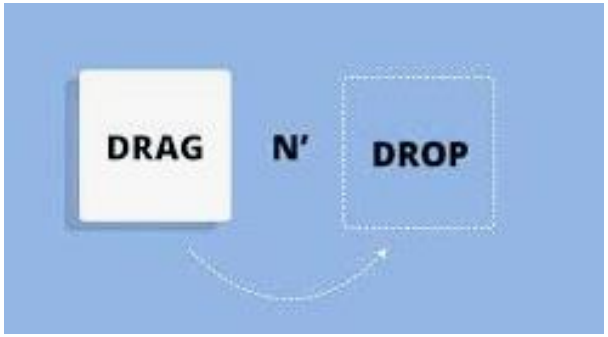


References

- Pilgrim, M. HTML5 Up and Running, O'Reilly/Google Press, Sebastopol, CA
- Lawson, L. & Sharp, R. *Introducing HTML5*. Berkley, CA, New Riders
- Celik, T. HTML5 Now, Berkely, CA, New Riders
- Holzner, S. SAMS Teach Yourself HTML5, Pearson
- Html5-Doctor – Helping You Implement HTML5 Today (n.d.) Retrieved from http://html5doctor.com/
- W3C HTML5 – A Vocabulary and Associated API for HTML and XHTML (n.d.). Retrieved from httpd://dev.w3.org/html5/spec/Overview.html
- W3C HTML 5 Differences from HTML 4 (n.d.). Retrieved from httpd://dev.w3.org/TR/2008/WD-html5-diff020080122
- W3C HTML 5 Reference – A Web Developers Guide to HTML 5 (n.d.). Retrieved from httpd://dev.w3.org/html5/html-author
- Web Hypertext Application Technology Working Group (n.d.). Retrieved from http://www.whatwg.org/specs/

Homework

- Optional (extra credit): Use one of the new HTML5 capabilities in a web page
  - Drop and Drag
  - Inline Editing
  - Messaging
  - Offline Storage
  - Geolocation
- See appendices →



New HTML5 Elements

- Article - The article element is a new sectional element and is used to represent a self-contained entry in a document, page, application or site.
- Aside - The aside element is used to represent content which is tangentially related to the content which is around aside element, sometimes, represented as sidebars.
- audio - The audio element is a type of media element used to represent an audio stream.
- canvas - The canvas element is used for rendering graphs, game graphics, art images or other visual images using a resolution dependent bitmap canvas.
- command - The command element represents a command which can be invoked by a user. It can be part of a menu element (explicitly part of a context menu or toolbar); alternately it can be placed anywhere else on the page to define a keyboard shortcut or to define other commands.
- datalist - The datalist element is used to represent a set of option elements which can act as options of other controls. datalist elements are wired to input elements using the list attribute on the input element.
- details - The details element is used to represent an area where users can go to obtain additional information. The details element in an interactive element represented as a widget.
- embed - The embed element is used to represent an integration point for a non-HTML application or interactive content.
- summary - The summary element is an interactive element used to represent a summary or legend for the rest of the content.
- Figure - The figure element is a grouping element is used to group content that is self-contained and is typically referenced as a single unit from the main flow of the document.
- Figcaption - The figcaption element represents the caption for a figure element.
- Footer - The footer element is used to represent a footer of the preceding sectional content.
- Header - The header element is used to represent a header for the succeeding sectional content.
- keygen - The keygen element is used to represent a key pair generator control which is used to store the private key in the local keystore and send the public key to the server when the form is submitted. The keygen element is a form element.
- mark - The mark element is used to highlight a range of text in a document for reference purpose. It is equivalent of using a highlighter to highlight a bunch of text.
- meter - The meter element is used to represent a scalar measurement within a known range, for example, how many respondents were male with kids. It should not be used to indicate progress.
- nav - The nav element is a section with navigation links and represents a section of page that links to other pages.
- output - The output element is used to represent the result of a calculation or user action. It supports representation of explicit relationship between itself and the elements that represent the values that went into the calculation of the output value.
- progress - The progress element is used to represent the progress of a task. It supports both indeterminate situations as well as determinate situations.
- Ruby - The ruby element is used to support one or more spans of content to be marked with ruby annotations.
- rt - The rt element is used to mark the ruby text component for a ruby annotation.
- rp - The rp element is used to provide parenthesis around a ruby text component of a ruby annotation.
- Section - The section element is used to represent a generic section of a document or application, usually containing a heading. E.g. chapters in a book.
- Source - The source element is used to specify to alternate media resources for media elements. It is not supposed to be dynamically modified, as that will have no effect.
- Summary- The summary element is an interactive element which is used to represent a summary, or caption for a details element.
- Time - The time element represents datetime attribute contents in a machine readable form (limited to various kinds of dates, times, time-zone offsets, and durations).
- Video - The video element is a type of media element used to represent a video stream.
- wbr - The wbr element is used to represent a line break in a web page.

HTML5 New JavaScript API's

- **Contacts** - The HTML5 specification mentions that the Contacts API allows to have a common contacts repository in the browser which can be access by any web application.
- **Selection** - The selection API supports selecting items in DOM (supports CSS3 type of selectors), to be used along with JQUERY.
- **Offline apps** - This API allows marking pages to be available in Offline mode. This is useful if a resource requires dynamic processing.
- **Indexed database** - This API is meant for a database of records holding simple values (including hierarchical objects). Every record has a key and a value. An indexed database is supposed to be implemented using b-trees. Web SQL DB is no longer being pursued as part of HTML5 specification.
- **Web workers** - This API is meant to be invoked by web application to spawn background workers to execute scripts which run in parallel to UI page. The concept of web works is similar to worker threads which get spawned for tasks which need to invoked separate from the UI thread.
- **Web storage** - This specification defines an API for persistent data storage of key-value pair data in Web clients.
- **Web sockets** - This API used for persisting data storage of data in a key-value pair format for Web clients.
- **Server-Sent Events** - This API is used for opening an HTTP connection to receive push notifications from a server. These events are received as DOM events. This API is supposed to be used with Push SMS.
- **XMLHttpRequest2** - This API is used to provide scripted client functionality to transfer data between a server and a client.
- **Geolocation** - This API is used to provide web applications with scripted access to geographical location information of the hosting device.
- **Canvas 2D Context** - This API provides objects, methods and properties to draw and manipulate graphics on a canvas drawing surface.
- **HTML Microdata** - This API is used to annotate content with specific machine-readable labels, e.g. to allow generic scripts to provide services that are customized to a page. Microdata allows nested groups of name-value pairs to be added to documents.
- **Media Capture** - This API is used to facilitate user access to a device's media capture mechanism (camera, microphone, file upload control, etc.). This only coves a subset of media capture functionality of the web platform.
- **Web Messaging** - This API is used for communications between browsing contexts in HTML documents.
- **Forms** - The Forms API can be used with the new data types supported with HTML5.
- **File API** - The File APIs are used by the browser to provide secure access to the file system.