

HTML 5 - This Ain't Yo Momma's HTML

Robert Kieffer, BendTech.com

"This specification should be read like all other specifications. First, it should be read cover-to-cover, multiple times. Then, it should be read backwards at least once. Then it should be read by picking random sections from the contents list and following all the cross-references."

HTML5 Draft, §1.8.1 - How to read this specification, page 14 of 325

*"This specification evolves HTML and its related APIs to ease the authoring of **Web-based applications**"*

HTML5 Draft, Abstract, page 1 of 325

History of Web App Standards

- 1991 - Tim Berners Lee invents HTML
- 1995 - HTML 2 spec, JavaScript
- 1996 - DOM level 0 in IE & NS, CSS 1, HTML 3.2, **HTML 4.0**
- 1998 - DOM level 1, **CSS 2, EcmaScript 3**
- 2000 - HTML 4.0 approved by ISO, DOM Level 2
- 2000-2004 (*web apps become mainstream*)
- 2004 - **DOM Level 3**
- 2004 - **WHATWG starts work on HTML 5**

Present day

- HTML 5 - "last call for comments"
- CSS 3 - "in development"
- EcmaScript 5 - Published 10/2009

11:45:25

Top 10 Ways to Torture Your Web Developer

1. **"This needs to work offline"** - Working Offline
2. **"What happens if that changes on the server?"** - Pushing data from the server
3. **"Save that as a preference, please"** - Storing data on the user's computer
4. **"It needs to work with a screenreader"** - Creating accessible apps
5. **"Users should be able to make the text bold. And red."** - Editing rich text (with valid markup)
6. **"The 'Back' button doesn't work right"** - Playing nice with browser services
7. **"Put that in a pie chart"** - 2D graphics
8. **"We'll need a custom slider control"** - Sophisticated user interfaces
9. **"That field should only allow email addresses"** - User input validation
10. **"Separate content and style"** - Separation of content and presentation

Graphics & Video (i.e. "Replacing Flash")

SVG element

Scalable Vector Graphics (the PNG?)

VIDEO & DEVICE

Video streaming & Access to cameras, microphones

CANVAS element

```
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");

ctx.fillStyle = "rgb(200,0,0)";
ctx.fillRect (10, 10, 55, 50);

ctx.fillStyle = "rgba(0, 0, 200, 0.5)";
ctx.fillRect (30, 30, 55, 50);
```

[Canvas tutorial](#) [Sphere](#) [Dynamic content injection](#)

11:45:25

Separating Content And Presentation

Generally accepted design philosophy.

(and a damn good idea if you want to support more than one kind of user)

Element house-keeping

Out: BASEFONT, BIG, CENTER, FONT, S, STRIKE, TT, U, ACRONYM, ISINDEX, DIR

In: SECTION, ARTICLE, ASIDE, HGROUP, HEADER, FOOTER, NAV, DIALOG, FIGURE

Attribute house-keeping

Out: align, alink, and, background, bgcolor, border, cellpadding, cellspacing, char, charoff, clear, compact, frame, frameborder, height, hspace, link, marginheight, marginwidth, noshade, nowrap, rules, scrolling, size, text, type, valign, vlink, vspace, width

Sophisticated user interfaces

New input types

- 'color' (colorwell / color picker!)
- 'number' (text or number spinner)
- 'range' (slider)
- 'search' (platform-consistent search field)
- 'time,' DATETIME, DATE, MONTH, WEEK

Toolbar and menu support

- MENU elements have 'type' attribute: "context" or "toolbar"
- COMMAND elements have 'action' attribute

'contextmenu' Attribute

- Value refers to a MENU element elsewhere in the document
- Browser uses this element as the context menu

(No browsers support these features yet)

11:45:25

Form Validation

HTML5 introduces "constraints" for form elements, and new constrained input types.

New types

- 'tel'(ephone)
- 'email'
- 'url'

'validity' API for elements

- *element.willValidate*
- *element.checkValidity()*
- *element.setCustomValidity(message)*
- *element.validity.(error-flag)*

E.g:

```
var ZIP_SYNTAX = /^\\d{5}(-\\d{5})+$/;

myForm.zipcode.checkValidity = function() {
  var valid = ZIP_SYNTAX.test(this.value);
  this.setCustomValidity(valid ? '' : 'Not a valid zip code');
  return valid;
}
```

Browser Services

History management

```
history.pushState(data,title[,url])
history.replaceState(data,title[,url])
```

MIME-type protocol handling

```
navigator.registerProtocolHandler(scheme,url,title)
navigator.registerContentHandler(mimeType,url,title)
```

Cross-document messaging

11:45:25

```
window.postMessage  
window.onmessage
```

Editing Content

Document editing

'contentEditable' attribute can be used to make [almost] any part of a document editable.

Spell check

- 'spellcheck' attribute

UndoManager

Not yet supported, but in the spec.

```
undoManager.position undoManager.add(data,title) undoManager.remove(index)  
undoManager.clearUndo() undoManager.clearRedo()
```

Drag & Drop

- Hooks for native via *drag/drop* events
- Native pasteboard access via *event.dataTransfer* API
- Better drag-image via *draggable* attribute

[Outline demo](#)

Decorating the DOM - Data, Microdata, and Accessibility

Data attributes

- Data attributes attributes ("data-*")

Microdata (Microformats)

11:45:25

- item* attributes, items API. vCard & vEvent microformats.

Accessible Rich Internet Applications (ARIA)

- aria-* attributes

Geolocation API

How it works

- Web app ask for location information
- Browser asks user if it is okay to share their location
- Web app callback is invoked with Coordinate info

"navigator.geolocation" API

```
navigator.geolocation.getCurrentPosition()  
navigator.geolocation.watchPosition()  
navigator.geolocation.clearWatch()
```

Coordinate data includes lat, lon, altitude, accuracy, heading, and speed

[Geolocation demo](#)

The Paradigm Shifters ...

And now for the *interesting* stuff ...

Offline Storage (related specifications)

localStorage and sessionStorage

A replacement for cookies

```
window.sessionStorage // session storage  
// ... or ...  
window.localStorage // persistent storage
```

11:45:25

```
store.length
store.key(index)
store.getItem(key)
store.setItem(key, data)
store.removeItem(key)
store.clear()
```

Application Cache

- Provides complete control over offline resource caching
- Cache manifest file
- Cache events
- Cache API

Example:

clock.html

```
<html manifest="clock.manifest">
<head>
  <script src="clock.js"></script>
  <link rel="stylesheet" href="clock.css">
</head>
<body>
  <p>The time is: <span id="clock"></span></p>
</body>
</html>
```

clock.js

```
setTimeout(function () {
  document.getElementById('clock').value = new Date();
}, 1000);
```

clock.css

```
#clock {font: 2em sans-serif;}
```

clock.manifest: *(must be served as "text/cache-manifest" MIME-type)*

```
CACHE MANIFEST
clock.html
clock.css
```

11:45:25

clock.js

Browser SQL

```
var db = openDatabase("notes", "", "The Example Notes App!", 1048576);

db.transaction(function(tx) {
  tx.executeSql('CREATE TABLE IF NOT EXISTS Notes(title TEXT, body TEXT)', []);

  tx.executeSql(â€˜SELECT * FROM Notesâ€™, [],
  function(tx, result) {
    for(var i = 0; i < result.rows.length; i++) renderNote(rs.rows[i]);
  },
  function(tx, error) {
    reportError('sql', error.message);
  });
});
```

(Standards status is unclear, but available in non-IE browsers today)

Web Sockets (related specification)

Enable Web applications to maintain bidirectional communications with server-side processes

```
var socket = new WebSocket('ws://game.example.com:12010/updates');

socket.onopen = function () {
  setInterval(function() {
    if (socket.bufferedAmount == 0)
      socket.send(getUpdateData());
  }, 50);
};
```

Web Workers (related specification)

An API for running scripts in the background independently of any user

11:45:25

interface scripts

About

- Multi-threading for web apps
- "Workers" (threads) are sandboxed in separate JS environment
- Communication done via document-messaging API

Use Cases

- On-the-fly syntax highlighting (parsing)
- Number crunching on the client [NPRRoadTrip](#)
- Asynchronous client-server synchronization

Example:

```
var worker = new Worker('worker.js');

worker.onmessage = function (event) {
  // Respond to message from worker
};
```

worker.js

```
// Calculate primes ad-infinitum
while (true) {
  n += 1;
  for (var i = 2; i <= Math.sqrt(n); i += 1)
    if (n % i == 0) continue search;
  postMessage(n); // found a prime!
}
```

Browser Support

What browsers support (or will soon support) what?

	Firefox	IE	WebKit	Opera
misc' elements	-	-	Y	-
Canvas	Y	?	Y	Y
Audio	Y	-	Y	Y
Video	Y	-	Y	Y
SVG	Y	?	Y	Y

11:45:25

CSS3	94%	20%	100%	100%
Drag and Drop	Y	?	?	-
Offline caching	Y	-	Y	-
SQL Storage	?	-	Y	-
Web Storage	Y	Y	Y	Y
Web Workers	Y	-	Y	-
Web Sockets	-	-	Y	-
Geolocation	Y	-	Y	Y

The New Paradigm: Unplugging the Server

- Storage and SQL are viable ways of saving documents on the client
- Offline support is no longer a pipedream
- Mobile platforms are the new bleeding edge of browser development
- **The browser wars are back, and web app technologies are the new battlefield!**
 - (*... and Microsoft lost!*)

Look for ...

- Server interaction that is less about function, and more about synchronization
 - Javascript ORM frameworks
 - Javascript Synchronization frameworks
 - (... paired with server frameworks; Most likely Java and Ruby.)
- Better scalability/performance from web apps (as server load is reduced)
- Some bitchin' new types of apps features

iPhone App Store will close by Q4, 2012. ('kidding!)

More Info

Web Hypertext Application Technology Working Group (WHATWG)

- <http://whatwg.org>
- HTML5 specification - <http://www.whatwg.org/specs/web-apps/current-work/multipage/>
- Web Workers specification - <http://www.whatwg.org/specs/web-workers/current-work/>

World Wide Web Consortium (W3C)

11:45:25

- <http://www.w3.org>
- Web Storage - <http://www.w3.org/TR/webstorage>
- Web Sockets - <http://dev.w3.org/html5/websockets>
- Offline Apps - <http://www.w3.org/TR/offline-webapps>

Miscellaneous

- [Comparison of layout engines \(HTML5\)](#)