



HTML5 From the Front Lines

What to Embrace Today (and What to Avoid) | 2011.11.1



What We're Going to Talk About

- Introduction
- General Support Strategies
- Working With HTML5 Today



Sapiient

Introduction

Who?

- 12 Years of HTML CSS JavaScript
- Senior Specialist, Platform Sapient Global Markets
 - 1+ week on the job
 - Job #1: Build a world class front end engineering team
- Previously at Isobar, the Brand Experience and Cramer.
 - I've worked with companies like Adidas, Reebok, Motorola, Harvard Kennedy School and Boston's Museum of Science
- Mobile Web App Cookbook (Manning)
 - Due out in mid-2012
- Open Source: HTML5 Boilerplate , etc.
- Twitter: @robreact
- Blog @ HTML + CSS + JavaScript <http://htmlcssjavascript.com/>



Experience With Emerging Web Technologies

- (Like everyone) I've been experimenting with these technologies as soon as they show up in a browser
- Additionally I've also been implementing a laundry list of these technologies in large scale production sites for the last couple of years
 - Semantic elements
 - Geolocation
 - Canvas 2d API
 - Web Storage
 - Audio/Video
 - History
 - More...
- While I **am** biased about what technologies I prefer to use, I have to **ship** so my preferences take a back seat to solving problems.

Just So You Know Where I Stand: “HTML5” as a Blanket Marketing Term Doesn’t Bother Me

- "HTML5 is anything you want it to be as long as it's new and cool." – Peter Paul Koch
 - http://www.quirksmode.org/blog/archives/2010/01/html5_means_wha.html

HTML



Technology People Will Figure Out the Difference

- Non-technical people just need to be excited so that we can implement this new stuff. Whatever works.
- See: Saying “Ajax” for pure-browser animations... Back in my day we called that DHTML

HTML





Sapient

A Generic HTML5 Support Strategy



Before You Write a Line of Code

- Don't Back Yourself Into a Corner!
 - No Flash support in iOS doesn't mean can't use Flash at all. It just means you can't use Flash for iOS.



Set Reasonable Support Targets and Communicate Them Early and Often

- This is as important for legacy browsers as it is for the latest/greatest desktop or mobile browsers
- Saying “this could be a problem” or “we can’t support this 100% in browser x, we should come up with a different strategy if that browser matters” makes you look like you know what you’re doing. Saying, “Sorry boss, this won’t work in your mother’s favorite web browser” doesn’t.

On the Desktop

- “old IE” will never work as well as the latest Chrome or Firefox. Slower JS engine + polyfills (often in other technologies like VML)
- Some things just aren’t going to be worth the effort for wide support (many CSS3 modules fall into this category) Again. **Set expectations early**

On Mobile Devices

- Be very specific about the devices you're going to support
(just in case you missed it the first time)
- **Be very specific about the devices you're going to support**
- Have the budget in place and plan to buy the devices (you'd be surprised...)
- "Webkit" on mobile probably doesn't mean what you think it means so don't make any assumptions about what will and will not work because you tested it in Safari. (See the first two points again.)

REMEMBER

Your site or app doesn't have to be exactly the same in every browser. The more people you convince of this the happier you will be- the happier we'll all be...



(Really) Use Modernizr

- They say “Modernizr is the right micro-library to get you up and running with HTML5 & CSS3 today.”
 - Allows for scripting and styling of new HTML5 elements in older versions of IE
 - Tests for over 40 emerging web features
 - Creates Modernizr JS Object that contains the results of these tests as boolean properties (e.g. `if (Modernizr.geolocation) { ... }`)
 - Adds classes to the html element that expose what features are and are **not** supported
 - Adds a script loader
 - *Doesn't actually add any functionality. No matter what the name implies.*

(Really) Use Modernizr

- Some people include Modernizr and all they **actually** use it for is supporting HTML5 semantic element in legacy versions of Internet Explorer
 - If that's all you need, use this (HTML5Shiv/m):

```
/*@cc_on'abbr article aside audio canvas details figcaption figure footer header hgroup mark meter nav output progress section summary subtitle time video'.replace(/\w+/g,function(n){document.createElement(n)})@*/
```

Leave the rest of the code at home.
- The CSS classes and the Modernizr object (and optional loader) are much more important when using emerging technologies

The Final Piece: Cross Browser Polyfills

- This is where the modernizing happens
- The people who write these are heroes
- Here's the big list:
 - <https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills>



Sapiient

From the Front Lines

Part One: Select Core Technologies In Depth

About This Section

- Introduce the feature
- Support (thanks to caniuse.com)
- Verdict & Polyfills (warning: potential shades of gray)

For a full rundown on the components of the emerging technology landscape, see my article [HTML5, CSS3, and related technologies](http://www.ibm.com/developerworks/web/library/wa-webstandards/index.html) (<http://www.ibm.com/developerworks/web/library/wa-webstandards/index.html>)

New Semantic Elements

- Examples: Header, Footer, Section, and Aside
 - Based on common usage patterns found during a web census conducted by editor Ian Hickson.
 - `<div id="header"></div><div id="footer"></div>`, etc.
- Others like `hgroup`, `mark`, `time`, and `figure` were logical additions



New Semantic Elements : Support

- Supported in all browsers except IE6,7,8 owing to the way unknown HTML elements are handled

New Semantic Elements : Verdict and Polyfills

It's worth it now. Every site I've built over the past two+ years has used the new semantic/structural elements. While there's no native support in older versions of IE, the solutions for support are solid and relatively painless. The one piece that's come up as a problem is the creation of dynamic html5 elements in IE, but even that is solved- especially now with jQuery 1.7+ baking a solution into the library

- Modernizr
- HTML5 Shim(v)
- InnerShiv for dynamic content in IE <http://jdbartlett.com/innershiv/>
- jQuery 1.7 wraps an optimized dynamic solution into the core

- Also, test against the new outlining algorithm:
 - <http://gsnedders.html5.org/outliner/>

An Outline Generated From HTML5 Markup

Document uses sectioning elements and multiple H1s

- Front-end Code Standards & Best Practices
 - Overview
 - General Guidelines
 - Pillars of Front-end Development
 - General Practices
 - Indentation
 - Readability vs Compression
 - Markup
 - HTML5
 - Template
 - Doctype
 - Character Encoding
 - General Markup Guidelines
 - Quoting Attributes
 - CSS
 - General Coding Principles
 - CSS Box Model
 - CSS Validation
 - CSS Formatting
 - Alphabetize
 - Classes vs. IDs
 - Naming Conventions for Selectors
 - [snipped outline from <http://na.isobar.com/standards/>)

Canvas 2D API

- The Canvas 2D context provides a scriptable interface for drawing two-dimensional images and bitmaps in the browser.
- One of the earliest stars of the HTML5 era

There are some excellent resources available already. One I've been knee deep in recently:

- JavaScript InfoVis Toolkit <http://thejit.org/>

Canvas 2D API : Support

- Supported in all major browsers except IE 6, 7, 8

Canvas 2D API: Verdict and Polyfills

With the understanding that there will be issues with more dynamic Canvas animations (in IE, in legacy versions of supporting browsers, on mobile and similarly underpowered machines), you can use Canvas right now.

- Flash Canvas <http://flashcanvas.net/>
 - This is one of the best reasons to keep the door open for Flash. It's faster than excanvas and supports 70% of the Canvas spec, which is competitive with the support levels in modern browsers
- Excanvas <http://code.google.com/p/explorercanvas/>
 - The original



Geolocation

- The Geolocation API is a standard interface for retrieving the geographical location of a device. It provides a `window.geolocation` object that in turn provides methods that figure out a device's location through the use of location information servers. Location information is pulled from a variety of sources, including IP address, device GPS, Wi-Fi and Bluetooth MAC address, radio-frequency ID (RFID), and Wi-Fi connection location.



Geolocation: Support

- Supported in all major browsers except IE6 , 7, 8 and Safari 3.2 and 4.0

Geolocation: Verdict and Polyfills

Between the ability to design to it (e.g. fall back to a simple text field labeled “location”) and the available polyfill techniques geolocation can be used today.

Polyfills aren’t as good as, say, device GPS, but they can do the job.

- From WebShims
 - <https://github.com/aFarkas/webshim/blob/master/src/shims/geolocation.js>
- Also: yqlgeo.js
 - <https://github.com/codepo8/YQL-Geo-Library/blob/master/yqlgeo.js>

Audio/Video

Playing audio and video in the browser is such a common event that it's easy to forget that, for most of the history of the web, there was no native method for doing so.

Enter the new HTML5 audio and video elements.

From a specification perspective, the inclusion of browser-native APIs for playing audio and video is straightforward. Anyone familiar with the way the replaced elements like IMG work will understand how to embed video and audio.

There's a tag, a source and some attributes.

With methods like `play()` and `pause()` the basics of the API are pretty easy to pick up

Audio/Video:

<-- the HTML -->

```
<video src="_assets/video/sample.webm" controls autoplay width="400" height="300" id="video-sample" data-description="sample web video"> your browser does not support the video tag
```

```
</video>
```

```
< button id="toggle"></button>
```

//the JavaScript

```
var video = document.getElementById("video-sample"),
    toggle = document.getElementById("video-toggle");
```

```
toggle.onclick = function() {
  if (video.paused) {
    video.play();
    toggle.className="playing"
  } else {
    video.pause();
    toggle.className="paused"
  }
};
```

Audio/Video: Support

- ~~Supported in all major browsers except IE6,7,8 and Safari 3.2~~
- Without taking into account the questions of video format (containers and codecs) , the above is actually true.
- With containers and codecs?
 - Unfortunately, getting to the point where video works in all major browsers with a single video source is still a long way off. With browser vendors facing off on two sides of the codec divide, it's far more complicated than it should be. Until we move away from Apple and Microsoft standing firmly behind the patent-encumbered h.264, versus Google, Opera, and Mozilla backing free, open, and royalty-free video formats like WebM, video on the web will remain more, rather than less, complicated than it was in the Flash-only era.



Is Audio Any Better?

- No. In the audio space OGG Vorbis plays the role of WebM and MP3 plays the role of MPEG4/h.264 *with the same factions*

Audio/Video : Verdict and Polyfills

The good news is, the people that serve video for a living have put together solutions that will work on any device seamlessly. Use them.

There are also nice solutions if you want to go it alone

- Video for Everybody
 - A no-javascript solution that uses a series of clever fallbacks to present the best video solution in different browsers and devices. Doesn't work in Android < 2.3
- MediaElement.js <http://mediaelementjs.com/>
 - <video>, flash, silverlight
- Video.js <http://videojs.com/>
 - Built on Video for Everybody, adds more device support with JavaScript

Audio follows a similar support strategy (Flash fallback)

- SoundManager2 <http://www.schillmania.com/projects/soundmanager2/>
- jPlayer <http://github.com/happyworm/jPlayer>

Web Storage

- The Web Storage specification defines an API for persistent data storage of key-value pairs in web browsers. This specification is similar to, but greatly improves upon, the functionality currently offered by cookies.
- Cookies
 - Cookies are limited to 4k
 - We try to avoid them for performance. Web storage removes the cookie overhead on every request
 - Cookies are annoying to code. I've been doing this for a long time and I tolerate them with gritted teeth.
- Storage takes two forms: `sessionStorage` and `localStorage`. Each provides similar methods for managing items (`setItem()`, `removeItem()`, and `getItem()`) and for clearing the entire storage (`clear()`). Session storage is designed to hold information for just the current browsing session. Local storage is meant for longer-term storage of site preferences or other user data. There's also a storage event that can be listened to, for purposes of monitoring and reacting to storage activity.



Web Storage: Support

- Supported in all major browsers except IE6,7

Web Storage : Verdict and Polyfills

The support landscape is good and there are multiple polyfill options

storage polyfill <https://gist.github.com/350433>

Amplify.js <http://amplifyjs.com/api/store/>

PersistJS <http://pablotron.org/?cid=1557>

Also:

Lawnchair <http://westcoastlogic.com/lawnchair/>

“A Lawnchair is sorta like a couch except smaller and outside. Perfect for HTML5 mobile apps that need a lightweight, adaptive, simple and elegant persistence solution. “

Keep in mind- data is store unencrypted on disk

<http://www.nczonline.net/blog/2010/04/13/towards-more-secure-client-side-data-storage/>



Sapiient

From the Front lines

Part Two: The Lightning Round

Lighting Round: Form elements and input types

- HTML5 provides several new form elements to better reflect common input tasks. Formats like Email and URL can now be indicated to the browser in a meaningful way.
 - These are safe to use now even if support is limited since browsers just default to type="text"
 - <http://www.miketaylr.com/code/input-type-attr.html>
- Completely new form inputs include range (for a slider), date (for a date picker), and color (for a color picker).
 - Support here is miserable
 - <http://www.miketaylr.com/code/html5-forms-ui-support.html>

Lightning Round: WebGL

The Web-based Graphics Library (WebGL) enhances JavaScript with the ability to create interactive, three-dimensional graphics in the browser. WebGL is a context of the canvas HTML element. The specification went to Version 1.0 on 3 March 2011 and is managed by the nonprofit Khronos Group.

- WebGL is very exciting, BUT...
- There's no hint of support in Internet Explorer
 - For older versions the polyfill path translates the 3d context of WebGL to Canvas 2D API and then would use Flash Canvas or Excanvas to render
 - There's also a commercial plugin for Internet Explorer <http://iewebgl.com/Default.aspx>
- Even supporting browsers require up-to-date drivers and decent hardware
 - No mobile support
 - <http://blog.mozilla.com/bjacob/2011/03/28/do-users-actually-get-hardware-acceleration/>

Lightning Round: History API

- One of the consistent usability issues of the Ajax era is the way Ajax-heavy applications break the standard history stack. The History API adds the ability to add entries to the browser history and respond properly when the user clicks the back button.
 - No current IE support (it's coming in 10,) buggy in Safari
 - Polyfill with History.js
 - `history.pushState` , `history.replaceState`, and the `popstate` event

Lightning Round: SVG

- Probably the strangest technology to be roped into the HTML5 catch-all is Scalable Vector Graphics (SVG.) SVG is a vector graphics grammar defined in XML. The SVG specification has been under development by the W3C since 1999, so including it as either "new" or part of HTML5 is a stretch. But yet... people do.
- Still, newfound excitement for SVG is justified as there's now some real traction for the standard. There's some level of support now available in the latest versions of all the major browsers and an API for older Internet Explorer versions presented by Raphael.js.
 - Aside: SVG Objects are DOM Objects. That makes them easier to script.
- <http://raphaeljs.com/>

Also...I'm a fan of HighCharts

<http://www.highcharts.com/>



Thank You!