HTML5: BUILDING THE NEXT GENERATION OF WEB APP

Features  Performance  Tools  Compatibility

Eric Bidelman, Google
COSCUP / GNOME.Asia - Taipei, Taiwan
August 14, 2010
I’m going to cover a bunch of stuff today. It’s really difficult to give an HTML5 talk because there is SO much to cover. However, if you take anything away from this talk, it’s really meant to get you thinking about what is possible with some of HTML5’s features. I’ll cover some of the lesser known aspects of HTML5.

First, I’ll talk about some techniques and APIs to consider for creating faster, and more feature-rich applications.

Of course fast web apps are good, but it’s also important that they’re compatible across the different browsers. We’ll look at Google’s Closure Tools and Chrome Frame for that reason.
First up, quick tips and tricks for improving your web app’s performance
DON’T UNDERESTIMATE CSS!

- Rounded corners, box shadows, reflection, rotations, alpha, css masks
- CSS animations & transitions

```html
div.box {
  left: 40px;
  -webkit-transition: left 0.3s ease-out;
  -moz-transition: left 0.3s ease-out;
  -o-transition: left 0.3s ease-out;
}
div.box.totheleft { left: 0px; }
div.box.totheright { left: 80px; }
```

- 3D transforms trigger HW compositing in the GPU

```
-webkit-transform: translate3d(10px, 0, 0);
```

- pseudo-selectors are your friend ( :hover, :active, :valid, :invalid, :focus, :empty )
- web fonts

At the presentation level, we can gain some quick wins from css3. There’s large amount of low hanging fruit here that you can use for optimizing a UI. Before you start implementing things in JavaScript, I encourage you to try and find a CSS solution. It’s likely one exists!
HTML5 has a ton of flashy APIs. It’s easy to forget about some of the lesser known native browser enhancements.
DON’T FORGET ABOUT JAVASCRIPT 1.6+

• Array iterative methods: `map()`, `filter()`, `forEach()`, `every()`, `some()`

```javascript
[5, 6, 7, 8].map(function(value){ // [50, 60, 70, 80]
    return value * 10;
});

// Return a new array of all mathematical constants under 2
[3.14, 2.718, 1.618].filter(function(number){
    return number < 2;
});

// [1.618]

['html5', 'css3', 'webgl'].forEach(function(value){
    // use value
});
```

• Array item location methods: `indexOf('html5')`, `lastIndexOf('webgl')`

The browser vendors are also hard at work implementing new features of the JavaScript language itself. Many of EcmaScript's improvements are a direct result of the popularity of libraries like Jquery, dojo, and prototype. So why not bake that functionality directly into the browser if developers are going to use it. Again, use the native functionality if that option is available. In some cases it can be an was performance win.
HTML5 MARKUP FOR WEB APPS

...more than just semantics
REL ATtributes

- **rel="pingback"**
  - enables reverse linking
  - automatically notifies original blog when other sites link to it
  
  `<a rel="pingback" href="http://blog.blogspot.com">A Blog</a>`

- **rel="prefetch"**
  - hint for the browser that the resource is likely to be used
  
  `<link rel="prefetch" href="URL to top of search result"/>
  <a rel="prefetch" href="next_page.html">Next page &gt;</a>`

---

HTML5 has a bunch of new tag elements for describing the structure of a page. This like `<header>`, `<footer>`, `<section>`, `<nav>`, `<hgroup>`. However, today we’re at a place where web apps are becoming more and more apart of our everyday lives. Things like Gmail, facebook, twitter come to mind. The web is no longer just markup and static pages. Spec designers have kept this in mind when designing this stuff.

Here are two new rel attribute values that were proposed as a direct result of web apps performing these common tasks.
HTML5 FORMS

• New <input> types mean you don’t need bloated JS libraries!
  • tel, email, url, datetime, date, month, week, time, datetime-local, number, range, color

• Attributes: placeholder, required, autofocus, pattern, min, max, step
DEMOS

open
WEB STORAGE

Not Just For Offline
**WEB STORAGE APIs**

**localStorage**

- key/value pairs
- great for storing user preferences

```javascript
localStorage.dateOfBirth = '1984-07-22';
delete localStorage.dateOfBirth;

localStorage['user'] = JSON.stringify({username: john, id: 100});
var retrieved = JSON.parse(localStorage['user']);
```

**sessionStorage**

- non-persistent key/value pairs (e.g. sensitive data)

**Web SQL DB**

- 5MB of persistent storage
- reduces round trips to the server

---

As of right now, there’s really 3 option.

Remember cookie data is sent on every request. Save some overhead by using one of the web storage APIs

**sessionStorage** – great for sensitive data that should be cleared after a user session. Think a public computer, or a banking session when a user logs out.
var webdb = {};

webdb.open = function() {
    var dbSize = 5 * 1024 * 1024; // 5MB
    webdb.db = openDatabase('Todo', '1.0', 'todo manager', dbSize);
}

webdb.onError = function(tx, e) {
    alert('Something unexpected happened: ' + e.message);
}

webdb.onSuccess = function(tx, r) {
    // re-render all the data in the DOM
}

webdb.createTable = function() {
    webdb.db.transaction(function(tx) {
        tx.executeSql('CREATE TABLE IF NOT EXISTS todo(ID INTEGER PRIMARY KEY ASC, todo TEXT, added_on DATETIME)', []);
    });
}

webdb.addTodo = function(todoText) {
    webdb.db.transaction(function(tx){
        var addedOn = new Date();
        tx.executeSql('INSERT INTO todo(todo, added_on) VALUES (?,?)',
            [todoText, addedOn], webdb.onSuccess, webdb.onError);
    });
}
A 4TH STORAGE OPTION...

Indexed DB

- Hybrid of `localStorage/sessionStorage` APIs and Web SQL DB.
  - In-order retrieval
  - Faster search - Index on any keys
- Browser support is still sparse
  - Implemented in FF4
- landing in Chrome soon...
APPLICATION CACHE

• Caches entire web app locally

• Why?

  1. HTML, CSS, and JS stay fairly consistent

  2. Native browser caching is unreliable

  3. Caching resources creates speedier apps

  • Native iPhone & Android Gmail app uses AppCache

also not just for offline! A way to programmatically cache the assets of your web app. Best part about it, is that you have complete control over when and what is cached locally. There’s 3 big reasons to use app cache.
What is cached is determined in the cache manifest file.

tips:
- don’t cache your cache manifest file!
- create manifest using a script that walks the tree of your site
- include a version number in the manifest when you change a resource so browser will re-cache site.
Debugging appache is painful. If any part of the manifest fails to download, the entire cache update fails. But using the JS api, you have event information for just about everything that the browser is doing behind the scenes.
Chrome Dev Tools just added console logging to give you better insight into what the browser is doing.
DEMO

http://3.ly/timer
Webworkers is a bit of a shift in paradigm. As many of you may know, JS runs single-threaded in the rendering process of the browser. However, the idea behind webworkers is enable the ability to spawn multiple threads in JS. That means you can finally take advantage of a user’s multi-core CPU in your web app.

So what would you use a webworker for? Here are some possible uses cases.
<output id="result"></output>

<script>
    var worker = new Worker('task.js');
    worker.addEventListener('message', function(e) {
        document.getElementById('result').textContent = JSON.stringify(e.data);
    }, false);
    worker.postMessage({'cmd': 'start', 'msg': 'Hi'});
</script>

// task.js
self.addEventListener('message', function(e) {
    var data = e.data;
    switch (data.cmd) {
    case 'start':
        self.postMessage('WORKER STARTED: ' + data.msg);
        break;
    case 'stop':
        self.close(); // Terminates the worker.
    }
}, false);

based off of the CORS (cross origin resource sharing) spec, which is a way to pass messages between origins using a postMessage() API.
WEB SOCKETS
REALTIME

• Bi-directional communication
• Eliminates need for XHR polling!
• Close as we can get to TCP/IP socket connections in JS
• Port 80 with scheme ws://, wss://

• Use cases:
  • chat rooms
  • white boarding
  • games

web has constraints
This will look familiar to the webworkers api. Again, it’s based on the CORS model.
DEMO

http://mrdoob.com/projects/multiuserpad/
How about browser compatibility? It’s a big issue, especially since many HTML5 features are still being spec’d out.
I briefly mentioned web fonts early. I think one of the more exciting features of HTML5 are webfonts. This is an area the web has been lacking in for quite some time. Some browser vendors have tried to push web fonts in the past, but it’s never stuck. Finally, we have a solution to having beautiful (and native) typography on the web.
Here’s a news site that’s not using webfonts. With HTML5, we can transform this news article into something beautiful. This sample is taking advantage of webfonts, html5 markup for columns and css for rounded corners and shadows. The beauty of this is that the everything on this page is selectable. That means search engines can crawl and index this site, more easy, and more effectively, making your content discoverable.
Here’s a news site that’s not using webfonts. With HTML5, we can transform this news article into something beautiful. This sample is taking advantage of webfonts, html5 markup for columns and css for rounded corners and shadows. The beauty of this is that the **everything on this page is selectable.** That means **search engines can crawl and index this site,** more easy, and more effectively, making your content discoverable.
The Fonts API is dead simple to use! There’s also a JS font loader API available in the common Google Ajax APIs loader. You can hook into things like onload events. An example use case is mobile. You could display a lower quality font or native browser font when the page first loads. Then, when your webfont has been downloaded, switch over to it by changing the font-family style of the your <body> tag.
Let’s see it in action.
So I mentioned Google’s Closure Library has reusable UI widgets. That’s great for cross-browser compatibility, but that only gets us so far. What if my new web app relies on newer HTML5 features like canvas, audio, or video? Sure, you could write wrapper libraries to fake of this behavior and fallback to things like Flash, but what if I need something webworkers or websockets? That’s not so easy. The answer is, we can do better.
Here is a current breakdown of the browser share for July of this year. I wanted to show you this chart as a reminder of where we’re at today regarding browsers. We’ve been talking a lot about innovation and moving the web forward, but the sad reality is that a majority of users are still use outdated, insecure browsers. Chrome Frame is all about bringing older browsers up to speed.
http://acid3.acidtests.org/

To pass the test, a browser must use its default settings, the animation has to be smooth, the score has to end on 100/100, and the final page has to look exactly, pixel for pixel, like this reference rendering.

layout, colors, css selectors animations
http://acid3.acidtests.org/

Acid3

20/100

LINKTEST FAILED
To pass the test, a browser must use its default settings, the animation has to be smooth, the score has to end on 100/100, and the final page has to look exactly, pixel for pixel, like this reference rendering.
What developers end up having to do is code around older browsers. Doing so is often nontrivial and costly. You lose functionality, speed, and security. For example, recent sites like Google Wave has chosen to stop supporting IE (at least older versions of IE) because it limits the cool features they can crank out.
So what is Chrome Frame? Well, the simplest explanation is that it is a browser plugin that gives you the benefits of the latest improvements to webkit, chrome, and the super fast v8 js engine. As a web developer you’ve probably told yourself, “The HTML5 features that Jeremy showed us earlier will be great to use...someday”. Well, that’s basically where Chrome Frame steps in. It lets us use HTML5, today! What this means is that you no long need to target your site with different versions for each particular browser. This is ideal for something like the Chrome Web Store.
There are 2 ways to enable GCF on a site. The simplest is to add a single meta tag to your site. However, the best way is to add the X-UA-Compatible extension header to your server's responses. Sometimes it isn't practical to add a header to every page of your site. It's worth no

If you're familiar with this extension header, you'll recognize that flag as the way to trigger what version of the IE rendering engine to use for a particular page

You can also use server-side detection. navigator.userAgent will return Google Chrome's user agent string.
There are a couple of things going on here. First, you’ll notice that the entire GCF section is wrapped in a if IE conditional. That means this markup and code will only ever been rendered by an IE browser. Other browsers will just ignore this section, saving us a HTTP trip for the library, unused markup, and attaching the event listener to the page.

As a developer, you have full control over styling of the GCF install prompt.

Lastly, we’re using IE’s window.attachEvent to add an onload event handler to the page. We know attachEvent is available to use because again, this section is wrapped in an if IE conditional.
So far, I’ve said that Chrome Frame is a little rendering engine that sits inside of IE. But the integration actually goes deeper than that.

We use IE’s network layer. If you have special proxies or certs installed, we’ll use those. That means that Chrome Frame’s network behavior is a bit different than that of the Chrome browser. For example, IE7 only allows 2 network connections open to any particular host. Chrome allows for 6. The result is that we’ll get that 2 connection limit enforced even though Chrome can do more.

We use the same cookies. When you request cookies in JS, you see the same cookies as seen in IE for an earlier session. Also, when you make a request to a document that came across the wire earlier, you’ll get that document in the cache. It’s IE’s cache, not Chrome’s. That means that when you clear the cache or cookies we’re not leaving any trails behind.

In-Private browsing results in Chrome’s incognito mode
MODERNIZR LIBRARY

• **BAD**: checking `navigator.userAgent`

  • Unreliable - some users change this to get around poorly designed sites that otherwise block certain UAs.

• **BETTER**: Feature detection!

  • Tests 20+ HTML5/CSS3 features by:
    1. `document.createElement('nav')`
    2. set a style
    3. retrieve that style

  • Creates a global `Modernizr` object to store the test results

  • Adds classes to the `<html>` element that explain precisely what features are (and are not) natively supported

What Modernizr does is, very simple. It tells you whether the current browser has a feature natively implemented or not. Unlike GCF (which ‘adds’ new HTML5 and rendering capabilities to older browsers, Modernizr simple tells you what is available natively in the browser.
<!-- In your HTML: -->
<div id="audio">
  <audio>
    <source src="mySong.ogg" />
    <source src="mySong.mp3" />
  </audio>
  <button id="play">Play</button>
  <button id="pause">Pause</button>
</div>

/* In your CSS: */
.no-audio #audio {
  display: none; /* Don't show Audio options */
}
.audio #audio button {
  /* Style the Play and Pause buttons nicely */
}

// In your JavaScript:
if (Modernizr.audio) {
  // Hook up functionality to Play and Pause buttons
}
Here’s a site that I highly recommend, caniuse.com. It gives an up-to-date look at what HTML5 features are supported by which browsers. In this example here, I searched for css transitions which you can see are now supported in most browsers.
There is also the developer tools inside of Chrome/Webkit browsers. These tools are really fantastic and the community is actively contributing and improving them. They deserve an entire hour-long talk by themselves.
Perhaps the most important part of our commitment is to developers. We just launched a new site dedicated to all that is HTML5, html5rocks.com. It contains tutorials, an interactive playground where you can experiment with code, and additional resources that should help you get up and running with these technologies. I talk with a lot of developers and the learning curve is steep for some of this stuff. There are so many new APIs to wrap your head around and the last thing we want is for anyone to be left behind.
THANKS!

• Tools
  • Google Fonts API: code.google.com/apis/webfonts/
  • html5rocks.com

• Compatibility:
  • Google Chrome Frame: http://code.google.com/chrome/chromeframe/
  • caniuse.com
  • modernizr.com

• File bugs against Chrome: crbug.com

• Stay in Touch!
  • Twitter: @ChromiumDev
  • groups.google.com/a/chromium.org/group/chromium-html5/