IGME-330

Rich Media Web Application Development I
Week 1

Developing Rich Media Apps

Today’s topics

- Tools we’ll use – what’s the IDE we’ll be using? (hint: none)
- This class is about “Rich Media” – we’ll need a “Rich client” – what’s that?
- Rich media Plug-ins v. Native browser support for rich media
- Who’s in charge of the HTML5 browser API? (hint: no one!)
- Where did HTML5 come from?
- What are the capabilities of an HTML5 browser?
- Browser layout engines
- JavaScript Engines

Tools we'll use

- Browsers:
  - Google Chrome - assignments will be graded on Chrome
  - Safari
  - Firefox
- Text Editor of your choice – no IDE necessary
  - Notepad++ on Windows
  - BBEdit or TextWrangler on Mac
  - Gedit on Linux
  - Brackets, if you wish - http://brackets.io
- Documentation:
  - Mozilla Developer Network Canvas Tutorial
  - Mozilla Developer Network DOM Docs
  - Mozilla Developer Network JavaScript Guide

What is a “Rich Client”

- A traditional “web 1.0” application needs to refresh the entire page if there is even the smallest change to it.
- A “rich client” application can update just part of the page without having to reload the entire page. This makes it act like a desktop application - see Gmail, Flickr, Facebook, ...

Rich Client programming in a web browser

- Two choices:
  - Use a plug-in like Flash, Silverlight, Java, or ActiveX
  - Use the built-in JavaScript functionality of modern web browsers to access the native DOM (Document Object Model) of HTML5 compliant web browsers.

Browser plug-in

(Flash or Silverlight or Java or ActiveX)

- Advantages:
  - Rapid Application Development with helpful IDEs, APIs, and libraries
  - Source code hidden from users
  - Executable runs on multiple platforms and older browsers
  - Cross platform! Developers target the plug-in, not individual browsers

Browser plug-in

(Flash or Silverlight or Java)

- Disadvantages:
  - Tied to single vendor, IDEs often expensive!
  - Performance worse than native apps and now generally slower than web apps
  - Plug-ins tend to be memory and battery hogs, graphics not hardware accelerated
  - Plug-ins not usually supported on mobile devices.
  - Declining support for the above plug-ins on all platforms

We're doing this one, obviously.

Native Browser DOM

(part of HTML5)

- Advantages:
  - "Browser DOM" means native browser "Document Object Model" API
  - RAD development with scripting languages and JS libraries
  - performance has improved dramatically, graphics becoming hardware accelerated
  - easy to deliver application - just upload it to the web
  - runs on multiple platforms.
  - Loads of new native APIs: Web Storage, location, device orientation, accelerometer, voice recognition, and more.

We're doing this one, obviously.

Browser DOM

- Disadvantages:
  - slower than true native applications, lacking full access to hardware
  - JavaScript source code is visible to user (but we can minify and obfuscate)
  - browser inconsistencies still a problem (but JS libraries help us with this).

What is HTML5?

Colloquially, not just the markup language, but a whole bunch of new browser capabilities:

- Web Fonts
- Geolocation
- SVG (vector drawing)
- Canvas (bitmap drawing)
- Client-side data storage (localStorage API)
- Drag and Drop
- File API
- WebAudio
- Web Workers (multi-threaded JavaScript)
- Web Sockets

You might be wondering:

Is everything a browser can do written down in one place?

Is there a *html5apps.org* with official docs?

No one group is in charge of web standards!

- The World Wide Web Consortium (W3C) was once considered to be the sole authority on browser web standards, but they became perceived as being too rigid.

No one group is in charge of web standards

- The W3C XHTML2 standard would have required “draconian error handling” - even a single mistake in a tag would have resulted in a page that would not display.
- They advocated plug-ins for any browser capability beyond text and static images.
- W3C advocated “dumb browsers” and “smart plug-ins”

Bye bye XHML2

- The WHAT Working Group (Web Hypertext Application Technology) formed in 2004 (as an alternative to the W3C) and advocated the position that browsers should have better native capabilities and not be dependent on plug-ins.

https://whatwg.org/news/start

Nope

The genesis of HTML5

- Between 2004-2006 Apple, Google, and Mozilla (and others), put together the HTML5 specification. (Turns out that HTML5 was initially meant for their then secret smart phone programs)
- A history is here:

http://diveintohtml5.info/past.html

HTML Layout Engines

- **Webkit** layout engine (open source) - Chrome (until recently), Safari, Netflix, Opera, Nintendo 3DS, others
http://www.webkit.org

- **Blink** layout engine (a fork of Webkit, open source) - Chrome (23+) and Opera (15+)
http://www.chromium.org/blink
http://dev.opera.com

- **Gecko** layout engine (open source) - Firefox, Thunderbird
https://developer.mozilla.org/en-US/docs/Mozilla/Gecko

- **Trident** layout engine (proprietary)- used in Microsoft Internet Explorer

HTML Layout Engines

• **EdgeHTML** layout engine - a fork of Trident, used in the Microsoft Edge browser.

In practice, each browser’s layout engine renders HTML differently, but the differences have become slight.

In this class, we’ll be targeting Chrome, and not worry about edge cases with the other browsers too much. (*But on a job or coop, you will usually have to worry about supporting old versions of IE, for example!*)

Sources for HTML5 Spec

There are multiple sources of the HTML specification (Mozilla, W3C, WHATWG and more) – see this wikipedia article for info and links:

https://en.wikipedia.org/wiki/HTML5

HTML5 Right Now

- Browsers adopt proposed new HTML5 features (for example Web RTC, Web Workers, Web Sockets) on different schedules.
- No browser implements every feature in the HTML Spec at W3C and WHATWG.
- We will use feature detection techniques to avoid leaving older browsers behind.

One more thing - JavaScript Engines

- JavaScript engines (virtual machines) also vary from browser to browser.
- ECMAScript 5 (ES5) is the current standard that is supported by all current browsers.
- Where engines differ is in raw performance, and in their support for new ES6 features like classes.
- JS Engines have names like Rhino, Spidermonkey, V8, Nitro – see the wikipedia article: https://en.wikipedia.org/wiki/JavaScript_engine#JavaScript_engines
- This is the Squirrelfish (another name for Nitro) logo:

Which newer HTML5 Features are covered in this course?

- Canvas
- Web Fonts
- localStorage (*view source* of the Project-1 checklist page to see in action)
- Web Audio
- Geolocation

Let’s get going on canvas!