Website Performance: Kyle Simpson

(Video: 0_Introduction.mp4): **Introduction**

- 00:00:00-00:07:50: An introduction and a discussion about how developers need to change their mindset to think about web performance from the start. Kyle also presents a few initial resources including a discount code for the O'Reilly HTML5 Cookbook and some projects like LABjs, grips, and h5ive.
    - http://getify.me
    - HTML5 Cookbook O'Reilly discount code: AUTHD
    - http://speakerdeck.com/getify

Part 1: Make Performance a Priority (00:07:51-01:00:36)

(Video: 1_A.mp4): **Why Web Performance?**

- 00:07:51-00:09:56: While this presentation will focus greatly on optimization, performance tuning is only one piece of the development puzzle. Weaving these practices into your natural workflow will keep your website's performance a priority and not an afterthought.
- 00:09:57-00:13:00: Steve Souders once said, "Eighty to ninety percent of the end-user response time is spent on the front end". Website optimization should start there.
    - http://www.stevesouders.com/blog/2012/02/10/the-performance-golden-rule/

(Video: 1_B.mp4): **Performance and User Experience**

- 00:13:01-00:19:06: Great performance can lead to great user experience. The opposite is true too. A website's performance influences a user's first impression. Comparing performance data from one site to another is a great way to start gauging areas of poor optimization.
    - http://webpagetest.org
- 00:19:07-00:28:22: The document ready state is the point when a user is able to meaningfully interact with a webpage. Optimizing the document ready state can increase the perceived performance of the page.
    - http://whichloadsfaster.com

(Video: 1_C.mp4): **How we get Web performance**

- 00:28:23-00:31:02: A well performing site can create a higher conversion rate. Every second counts when qualified leads have reached your content. Optimized code is quality code. Reusing quality code can save time and money on future development projects.
- 00:31:03-00:37:28: Measurement and perception define web performance. Web professionals must measure the optimization techniques to determine if higher performance has been achieved.

(Video: 1_D.mp4): **Focus on the Critical Path**

- 00:37:29-00:44:23: Worrying about the speed of the non-critical parts of a website can lead to wasted time. Start with the critical parts.

- 00:44:24-00:51:04: Most users are not able to distinguish the differences when response times are less than 100 milliseconds. The threshold for response time is typically 1000 milliseconds, or 1 second. Anything over 1 second will exceed the tolerance of a user.
    - http://msdn.microsoft.com/en-us/magazine/gg622887.aspx
    - http://www.useit.com/papers/responsetime.html

(Video: 1_E.mp4): **The Total Cost of Ownership**

- 00:51:05-00:55:24: Owning poor code is expensive. Deciding to write poor code now and optimize it later is unrealistic and causes the code to become systemic.
- 00:55:25-01:00:36: The dirty little secret of front-end performance is that most of "front-end" optimization actually happens in the "middle-end". This workshop will start by focusing on optimizing the "middle-end". Then it will move to front-end performance enhancements.
    - http://github.com/getify/web-performance-exercises.git
    - http://nodejs.org

Part 2: The Middle-End: YSlow (01:00:37-01:38:08)

(Video 2_A.mp4): **YSlow Rules**

- 01:00:37-01:05:57: YSlow is one of the first browser plugins that allowed developers to test the performance of a website. YSlow uses a set of rules to gauge the performance. The first rule is fewer HTTP requests.
    - http://addons.mozilla.org/en-US/firefox/addon/yslow
    - https://chrome.google.com/webstore/detail/yslow/ninejjcohidippngpapiilnmkgllmakh
- 01:05:58-01:08:30: The next YSlow rule is to use a CDN. CDN's have multiple locations all over the world and can take advantage of shared caching.
- 01:08:31-01:11:42: Using *expires* and *cache-control* headers give you the ability to control the loading policy around stable content. Gzipping can compress your content anywhere from 50%-70% decreasing loading time.

(Video: 2_B.mp4): **YSlow Rules, Continued**

- 01:11:43-01:14:14: Another rule of YSlow is to include your stylesheets at the top and scripts at the bottom. With scripts, remember that including them at the bottom simply gives priority to the rest of the markup. You may need to include your scripts at the top in situations where they become more important.
- 01:14:15-01:17:33: Avoid CSS expressions because they will slow down the UI thread greatly. Also, externalize your JavaScript and CSS so they will take advantages of performance features like caching. Fewer DNS lookups will also increase performance by decreasing the latency of the page.
- 01:17:34-01:21:39: Minifying JavaScript and CSS should be higher on the YSlow list because it's an easy way to reduce the size of a web application. Avoiding redirects is another YSlow rule. An example is a website redirecting to the "www" version of the domain. When you are using the same JavaScript framework across a website, ensure you do not have duplicate script requests.

- 01:21:40-01:25:53: ETags are hexadecimal codes generated for a particular resource. They function like a finger print to enable the conditional loading of that resource. Cacheable Ajax adds an additional layer of expiration on top any external resources you are loading. While these 14 YSlow rules can occasionally contradict each other, they are a basis for proper website performance.

(Video: 2_C.mp4): **Beyond the Big 14**

- 01:25:54-01:29:59: There are many ways to incorporate the 14 YSlow rules into a website. Kyle outlines a few techniques of implementing fewer HTTP requests, Gzip, and conditional requests.
    - http://www.gidnetwork.com/tools/gzip-test.php
- 01:30:00-01:36:40: Yahoo has published a number of additional performance rules. They include reducing the cookie size and optimizing sprites. The YSlow plugin gives you a letter grade and a numerical grade for your website. Remember to look through the report carefully and consider additional optimizations even in categories with higher scores.
    - http://developer.yahoo.com/performance/rules.html

(Video: 2_D.mp4): **Exercise 1**

- 01:36:41-01:38:08: In this exercise, test three websites with the YSlow plugin. Select the site with the lowest grade and examine some of the ways the performance could be improved based on the YSlow reports.

Part 3: The Middle-End: Resources (01:38:09-02:11:20)

(Video: 3_A.mp4): **Images**

- 01:38:09-01:46:41: Between November 2010 and November 2011 the average web page size has grown from 625kB to 784kB. This is largely due to the increase in JavaScript usage and image size. Using an image optimizer can help trim down images size.
    - http://www.smushit.com/ysmush.it/
    - http://imageoptimizer.net/
    - http://pmt.sourceforge.net/pngcrush/
- 01:46:42-01:50:05: Image sprites combine multiple image resources into a single image and utilize CSS background positioning to one display a single *sprite* within that image. Use a tool like SpriteMe to create sprites of your images assets. Inline images reduce the amount of external requests however the ability for the image to be cached is now tied to the CSS file.
    - http://spriteme.org
    - http://spritegen.website-performance.org
    - http://www.floweringmind.com/sprite-creator/
    - http://dopiaza.org/tools/datauri/
    - http://dataurl.net/#dataurlmaker
    - http://dataurl.net/#cssoptimizer

(Video: 3_B.mp4): **Minification**

- 01:50:06-01:56:00: There are many options to minify JavaScript and CSS code. Some perform better than other. Concatenating code can also help reduce download time and HTTP request. A helpful online tool for identifying scripts you can concatenate is zBugs.
    - http://compressorater.thruhere.net
    - http://marijnhaverbeke.nl/uglifyjs
    - http://www.minifycss.com/css-compressor/
    - http://jsbeautifier.org
    - http://cssbeautify.org
    - http://zbugs.com
    - http://robertnyman.com/2010/01/19/tools-for-concatenating-and-minifying-css-and-javascript-files-in-different-development-environments/

(Video: 3_C.mp4): **Exercise 2**

- 01:56:01-02:11:20: Use YSlow to obtain a benchmark for performance of the supplied website. Then analyze the report and begin optimizing the website with the tools shown so far. After some time, Kyle dives into the solutions and discusses some of the ways the site could be optimized.

Part 4: The Middle-End: Architecture & Communication (02:11:21-02:48:12)

(Video: 4_A.mp4): **Understanding the Middle-end**

- 02:11:21-02:21:43: The middle-end is a layer that exists between what's happening in the browser and what's happening in a database. Developing the correct architecture is the key to tying together the front, middle, and back ends.

(Video: 4_B.mp4): **Introduction to Architecture**

- 02:21:44-02:27:50: Architecture starts with the markup of a webpage. Think about a single-page website design. Can you architect a page that is able to load all mark-up once so any dynamic content is simply a duplication of what's already loaded?

(Video: 4_C.mp4): **Data Validation**

- 02:27:51-02:37:07: Often times data validation rules are written twice: Once on the client, and once on the server. A performance goal would be to have these rules written once. One way to achieve this would be with a middle-end written in server-side JavaScript.

(Video: 4_D.mp4): **JSON, Ajax, & Web Sockets**

- 02:37:08-02:41:45: While JSON can be faster than other data formats like XML, remember it's easy to add bloat to your data and get lazy with how it's structured. Laziness can lead to latency and a slower web application.

- 02:41:46-02:48:12: Ajax requests have been used for years and tend to be the de facto means for asynchronous communication. Web sockets, however can reduce the amount of data sent and open the door for two-way communication.
    - http://shortie.me

Part 5: The Front-End: Resource Loading (02:48:13-03:14:56)

(Video: 5_A.mp4): **Preloading Images**

- 02:48:13-02:58:19: The first technique that can be used for preloading is the *prefetch* relationship attribute in the *link* tag. This method does not guarantee preloading. It simply suggests to the browser to load the resources.

(Video: 5_B.mp4): **Lazy Loading**

- 02:58:20-03:02:53: Lazy loading is often referred to as on-demand loading or post loading. This technique waits until it's absolutely sure the resource is needed. Lazy loading is on the other end of the spectrum from preloading but both methods are in a developer's arsenal of tools for performance tuning.
- 03:02:54-03:05:25: A simple coded example of a JavaScript file being lazy loaded on a web page.

(Video: 5_C.mp4): **Parallel Loading**

- 03:05:26-03:14:56: The focus of parallel loading is to preserve the execution order. Using a script loader like LabJS will help the parallel loading of resources. The big difference between a script loader and relying on a browser's native parallel loading technique is when the document's ready event fires.
    - http://labjs.com

Part 6: The Front-End: Abstractions & Animation (03:14:57-04:02:33)

(Video: 6_A.mp4): **OO is Slower**

- 03:14:57-03:21:52: Object oriented code in the browser does not work the same as object oriented code in a compiled language because, in the browser, it's a live interpreted link of inheritance. It can lead to more abstraction than necessary.

(Video: 6_B.mp4): **Exercise 4**

- 03:21:52-03:34:33: The point of this exercise is to look at some object oriented JavaScript code and apply some performance testing. Identify the classes that have composition relationships and which classes have substantive object oriented behavior. Can you modify the code to remove inheritance and abstraction to improve performance?

(Video: 6_C.mp4): **JavaScript Animations**

- 03:34:33-03:42:03: Offloading animations from JavaScript into CSS can drastically improve the performance of your website. Using the *setTimeout* function in JavaScript can lead to inconsistent animation behavior and be unreliable. Same is true with *setInterval*.

- 03:42:04-03:48:10: HTML5 added a better interface for animation called *requestAnimationFrame*. The purpose of this API is to tell the browser to run some code the next time it's going to repaint the screen. This API is a better way to animate in JavaScript.

(Video: 6_D.mp4): **CSS Transition vs. CSS Animation**

- 03:48:11-03:52:24: CSS animations are a powerful way to keyframe animations with CSS. CSS transitions, on the other hand, are pre-calculated animations based on a starting and ending property.

(Video: 6_E.mp4): **Exercise 5**

- 03:52:25-04:02:33: Compare the differences between the JavaScript animations and CSS transitions.

Part 7: The Front-End: UI Thread, Garbage Collection & jsPerf (04:02:34-05:01:55)

(Video: 7_A.mp4): **The Single Threaded Browser**

- 04:02:34-04:08:33: Threading allows more than one operation at the same moment in time. Threaded programming can be very complex to implement. JavaScript, however is single threaded. Don't confuse asynchronous programming with parallel programming. While JavaScript has asynchronous code execution, the event loop is single threaded.
- 04:08:34-04:12:23: The UI Thread in the browser is single threaded. This means the CSS rendering engine, DOM, JavaScript engine, etc. are all on the same thread.

(Video: 7_B.mp4): **Threaded JavaScript**

- 04:12:24-04:23:51: *Web workers* allow you ask a specific JavaScript file to run in its own thread. This means the execution of that JavaScript file will not affect the performance of the overall web application.

(Video: 7_C.mp4): **Dynamic Memory Allocation**

- 04:23:52-04:29:22: JavaScript allows you to create variable and elements in the DOM, and then remove the references to those objects. This flags the object for garbage collection and it will be eventually removed. The time the object is removed, however, is arbitrary.

(Video: 7_D.mp4): **Exercise 6**

- 04:29:23-04:34:45: Explore two different files: One focusing on the UI thread and another focusing on garbage collection.

(Video: 7_E.mp4): **Introduction jsPerf**

- 04:34:46-04:40:07: jsPerf is a web-based tool for running head to head test of JavaScript snippets and testing their performance.
    - http://jsperf.com

(Video: 7_F.mp4): **JavaScript Performance Mythbusters**

- 04:40:08-04:55:57: Kyle looks at a number of the common JavaScript performance misconceptions and demonstrates how to interpret the results in jsPerf.
    - http://docs.google.com/present/view?id=dq52zrp_22f2rx5zjz&pli=1

(Video: 7_F.mp4): **Exercise 7**

- 04:55:58-04:58:30: Visit an already created jsPerf. Test it out a few times and analyze the results.

(Video: 8_A.mp4): **Conclusion**

- 04:58:31-05:01:55: Conclusion