General Web Questions

What UI, Security, Performance, SEO, Maintainability or Technology considerations do you make while building a web application or site

- UI:
  - Constraint vs Freedom: How much do you let the user do (guided wizard vs adobe photoshop-like app)
  - Direct on-canvas vs form-based editing
  - Navigation and progress tracking (e.g. step 1 of 5 done)
  - links: 1, 2
- Security:
  - User authentication
  - Server authentication
  - Is RBAC (Role Based Access Controls) needed
  - Secure transmission (SSL)
  - Firewalls
  - Validate user input
  - What access does the user have to your data
  - links: 1, 2
- Performance
  - Nubmer/size of external resources
  - Optimize images
  - Optimize code
  - Placement of script blocks
  - Minimize cookie size, eliminate dead cookies
  - Low-bandwidth options
  - links 1, 2, 3, 4
- SEO
  - Content on your page relevant to your targeted keywords
  - Title, meta-description, meta-keywords, h1 - content rich
  - Pretty url's
  - Use of 301 redirects
  - Get rid of 404's, 500's,
  - links: 1, 2, 3, 4
- Maintainability
  - Very tight, issue specific commits
  - Helpful commenting
  - Informative use of Github tools - issues, PR's, milestones
  - links: 1, 2
- Technology
  - What browsers, devices, screen resolutions should you be targeting

If you have 5 different stylesheets, how would you best integrate them into the site?

- Use LESS and `@import` other modularzied stylesheets
- Concatenate them into a smaller number of files
- Remove duplicates/unused selectors
- Compress/minify the CSS
- link: 1, 2, 3, 4

Can you describe the difference between progressive enhancement and graceful degradation

Degrading Gracefully means looking back whereas **Enhancing Progressively** means looking forward.

- **Graceful Degradation** is the practice of building your web functionality so that it provides a certain level of user experience in more modern browsers, but it will also degrade gracefully to a lower level of user in experience in older browsers.
- **Progressive Enhancement** is similar, but it does things the other way round. You start by establishing a basic level of user experience that all browsers will be able to provide when rendering your web site, but you also build in more advanced functionality that will automatically be available to browsers that can use it.

In either case, you test for a feature that doesn't have complete support across all browsers/versions, and then apply it or an alternative.

Graceful Degradation starts from the status quo of complexity and tries to fix for the lesser experience whereas Progressive Enhancement starts from a very basic, working example and allows for constant extension for future environments.

Explain what "Semantic HTML" means

- Semantic HTML clearly describes the intention of the programmer (its meaning) to other developers and machines (e.g. the browser, Google) reading the code. E.g. Different content contained in the same `<section>` tag was intended to be thematically related.

How would you optimize a websites assets/resources

1. Reduce the number of requests
2. Reduce the overall size of the content
3. Promote parallelization (i.e. simultaneous download of assets)
4. Lazy load images - use blank png as base image, then use JS to swap image when other image is fully loaded
5. Use a framework that employs a virtual-dom (Elm, Mecury)

Code techniques:

- Concatenate files into a smaller group of files
- Remove duplicates and unused selectors/code
- Compress/minify code

Infrastructure Techniques * Consider using a CDN * Download files in parallel link * Leverage browser caching * Leverage proxy caching

Tech Specific

- HTML
  - HTML5 Lint
  - Separate style and functionality from structure
- CSS
  - Use LESS and `@import` other modularzied stylesheets
- Javascript
  - Adhere to JS Optimization rules
  - Parse/compress with UglifyJS
- Images
  - Use compression
  - Use CSS sprites

Traditionally, why has it been better to serve site assets from multiple domains

"Domain Sharding" is the practice of creating multiple sub-domains so that the number of resources downloaded in parallel is increased.

Browsers limit the number of downloads that occur in parallel by the number of host names that are involved. So spreading assets equally across multiple sub-domains uses only a single hostname and enables a larger number of assets to be d/l'd in parallel.

Because of the adoption of SPDY, domain sharding may hurt performance rather than help it.

Exceptions

- Flash loading external data (XML, etc)
- JavaScript manipulating or communicating with a page inside an iframe
- JavaScript fetching a file via XMLHttpRequest
- Other technologies such as Java applets or Silverlight fetching external data

Downsides with Mobile

- Network connections that breach the recommended limit come with a setup overhead - there's a `DNS Lookup`, a `TCP 3-way handshake` and a `TCP slow start`. This causes a higher latency with mobile users than desktop users. The setup overhead also uses more CPU, memory, and battery power - all important considerations with Mobile.
- Modern mobile browsers implement `HTTP pipelining` and don't observe `HTTP 1.1`

`connection` rules.

SDPY as an exception

SPDY supports concurrent requests (send all the request headers early) as well as request prioritization. Sharding across multiple domains diminishes these benefits. SPDY is supported by Chrome, Firefox, Opera, and IE 11. *If your traffic is dominated by those browsers, you might want to skip domain sharding*. On the other hand, IE 6&7 are still somewhat popular and only support 2 connections per hostname, so domain sharding is an even bigger win in those browsers.

- links: 1, 2, 3, 4

What tools do you use to test your code's performance

- Profiler, JSPerf, Dromaeo, Jasmine, Karma

What are the differences between Long-Polling, Websockets and SSE

- **Regular AJAX Polling:** Javascript requests a file from the server at regular intervals (e.g. 1 second apart).
- **AJAX Long Polling:** Javascript requests a file from the server, but the server does not respond until there is new information. As soon as the server responds, the client immediately sends another request to the server, restarting the request.
- **HTML5 Server Sent Events (SSE)/EventSource:** Javascript opens a connection to the server. The server sends an event to the client when there is new information available. There is real-time traffic from the server to the client, but it's not possible to connect to the server with a server from another domain.
- **HTML5 Websockets:** Javascript opens a connection to the server. The server and client can send each other messages when new data (on either side) is available. With WebSockets it is possible to connect with a server from another domain. It's also possible to use a third party hosted websocket server.
- **Comet:** Comet is a collection of techniques prior to HTML5 which use streaming and long-polling to achieve real time applications.

Explain the importance of standards and standards bodies

The goal of creating web standards, such as the HTML standard, was to eliminate the differences in feature support across browsers and formalize de facto standards, enabling developers to create sites that work similarily across all browsers. Standards bodies such as the World Wide Web Consortium (W3C) were created as forums to establish agreement across the industry and among vendors.

What is FOUC? How do you avoid FOUC

A **Flash Of Unstyled Content (FOUC)** is an instance where a web page appears briefly with the browser's default styles prior to loading an external CSS stylesheet, due to the web browser engine rendering the page before all information is retrieved. The page corrects itself as soon as the style rules are loaded and applied.

To avoid FOUC, place all scripts at the bottom of the page. Possibly hide all content on the page until the styles have loaded.

Do your best to describe the process from the time you type in a website's URL to it finishing loading on your screen

Assuming the simplest possible HTTP request, no proxies and IPv4:

1. Browser checks cache; if requested object is in cache and is fresh, skip to #9
2. Browser asks OS for server's IP address
3. OS makes a DNS lookup and replies the IP address to the browser
4. Browser opens a TCP connection to server (this step is much more complex with HTTPS)
5. Browser sends the HTTP request through TCP connection
6. Browser receives HTTP response and may close the TCP connection, or reuse it for another request
7. Browser checks if the response is a redirect (3xx result status codes), authorization request (401), error (4xx and 5xx), etc.; these are handled differently from normal responses (2xx)
8. If cacheable, response is stored in cache
9. Browser decodes response (e.g. if it's gzipped)
10. Browser determines what to do with response (e.g. is it a HTML page, is it an image, is it a sound clip?)
11. Browser renders response, or offers a download dialog for unrecognized types

Cross Browser Questions

How do you ensure a front-end looks the same across multiple browsers

1. Set benchmarks of how you want your site to respond
    - e.g. load time under 5 seconds, hover states < 100ms, etc
2. Use PageTest to check
3. Set the expectation that not all broswers will have the exact same experience. Some browsers will perform better than others.
4. In some browsers <IE8, you may need to turn off some features
5. Develop according to best practices
6. Use plugins and libraries that are performance optimized
7. Keep DOM manipulation to a minimum when possible
8. Write styles that avoid visual changes that affect the page as it loads

9. Use feature detection with Modernizr
10. Perform regular testing and QA for performance as well as visual issues

Cross-Browswer Automated Testing Tools

- Functionality: Selenium
- Appearance: Browsera

HTML5

What is the Doctype and what does it do

A Doctype triggers standards mode (or Quirks Mode) in the browser. The HTML5 doctype is very streamlined.

What is the difference between session storage, local storage and cookies

Local Storage, **Session Storage** and **Cookies** are all client storage solutions.

Session data is held on the server where it remains under your direct control. **Session Storage** (as the name persists) is only available for the duration of the browser session (and is deleted when the window is closed) - it does however survive page reloads.

Cookies are slow, insecure (sent over the server), and limited to about 4KB of data.

Local Storage, or **Web Storage** stores key name pairs locally within the browser, and is faster, more secure (not sent over the server) and has a limit of 5MB of data.

There is also **WebSQL** and **IndexedDB** - local browser databases. WebSQL is no longer supported, and IndexedDB is supported in most browsers, but still a contentious issue.

What is the difference between XHTML and HTML5

- XHTML does not have good browser support
- HTML5 uses more semantic code
- IE and other user agents cannot parse XHTML as XML
- HTML5 has offline storage

CSS

What is the difference between 'hidden' and 'display: none'

`display: none;` - tag will not appear on page at all. No space will be allocated between other

tags.

`visibility: hidden;` - tag is not visible, but space is allocated for it on the page.

What is a CSS pre-processor and how does it work

A CSS preprocessor allows you to use more complex logic like variables, extends, mixins and even loops. The pre-processor then compiles the code into valid css, and may also remove duplicates / dead code, and minify it.

What are the dangers of using a CSS pre-processor

- Mixins and extends can hurt maintainability of code, since you have to search for the original styles.
- Handing the project to a developer that uses raw CSS - they may not know LESS/SASS

What are floats, and what are the considerations in using them

Make content appear side by side, LTR or RTL.

- Parents of floated elements may collapse
- Need to apply special css with `:after`
- Problematic with email

JAVASCRIPT

What is event bubbling

- When the event of a child element is triggered even though only the parent is clicked/activated

diff between properties and attributes

- Attributes are defined by HTML. Properties are defined by DOM.

diff between == and ===

`==` compares equality but `===` compares types as well as equality.

```
"abc" == new String('abc');     // true
"abc" === new String('abc');    //false
```

Explain the dom

Explain the css box model

Explain ajax

What types of data does ajax transfer

what is chaining in jQuery

Explain the event process in JS

Explain clojures

Explain inheritance in JS (prototypes)

Why is it important to use the var keyword in JS - if you don't, it assumes a global scope, even if used inside of functions)

What aggregation tools have you used before

What are the concerns with using Global variables and when should you use them