* **DOM structure** – how nodes are related to one another and how to traverse from one to the next.
* **DOM manipulation** – how to add, remove, move, copy, create, and find nodes.
* **Events**- how to use them and the major differences between IE and the DOM event models.
* **XMLHttpRequest** – what it is, how to perform a complete GET request, how to detect errors.
* **Strict vs. quirks modes** – how to trigger each and why this matters.
* **The box model**- how margin, padding, and border are related and the difference between border-box (standards mode) and content-box (old Internet Explorer) sizing.
* **Block vs. inline elements** – how to manipulate using CSS, how they effect things around them and your ability to style them.
* **Floating elements** – how to use them, troubles with them, and how to work around the troubles.
* **HTML vs. XHTML** – how they’re different, why you might want to use one over the other.
* **JSON** – what it is, why you’d want to use it, how to actually use it, implementation details.

1. How do you optimize a website’s assets?
   * [There are a number of answers](http://www.sitepoint.com/web-site-optimization-steps/) to this question: File concatenation, file compression, CDN Hosting, offloading assets, re-organizing and refining code, etc. Have a few ready.
2. What are three ways to reduce page load time?
   * Again [there are many answers](http://blog.teamtreehouse.com/speeding-up-page-load-times) here: Reduce image sizes, remove unnecessary widgets, HTTP compression, put CSS at the top and script references at the bottom or in external files, reduce lookups, minimize redirects, caching, etc.
3. What kind of things must you be wary of when design or developing for multilingual sites?
   * Another problem with [many solutions](http://www.nomensa.com/blog/2010/7-tips-and-techniques-for-multi-lingual-website-accessibility/): setting the default language, using Unicode encoding, using the ‘lang’ attribute, being aware of standard font sizes and text direction, and language word length (may affect layout).
4. How many HTML tags are should be used for the most simple of web pages?
   * 8 total. 4 pairs of tags.

<HTML>

<HEAD>

<TITLE>Simplest page ever!</TITLE>

</HEAD>

<BODY>

Doesn’t get simpler than this.

</BODY>

</HTML>

1. How do you make comments without text being picked up by the browser?
   * Comments are used to explain and clarify code or to prevent code from being recognized by the browser. Comments start with “\*<!--” and end with ” -->“.

<!-- Insert  comment here. -->

1. What is the syntax difference between a bulleted list and numbered list?
   * Bulleted lists use the <ul> tag, which stands for “unordered,” whereas <ol> is used to create an ordered list.
2. Ok, what’s the real difference between HTML and HTML5?
   * [There are many](http://www.html5rocks.com/en/). From a broader perspective, HTML was a simple language for laying out text and images on a webpage, whereas HTML5 can be viewed as an application development platform that does what HTML does that and more, including better support for audio, video, and interactive graphics. It has a number of new elements, supports offline data storage for applications, and has more robust exchange protocols. Thus, proprietary plug-in technologies like Adobe Flash, Microsoft Silverlight, Apache Pivot, and Sun JavaFX are no longer needed, because browsers can now process these elements without additional requirements.
3. What is the new DOCTYPE?
   * Instead of typing out a ridiculously long DOCTYPE statement to tell the browser how to render your webpage, this long line of code has been truncated to <!doctype html>.
4. What are some new HTML5 markup elements?
   * [There are many](http://dev.w3.org/html5/markup/elements.html): <article>, <aside>, <bdi>, <command>, <details>, <figure>, <figcaption>, <summary>, <header>, <footer>, <hgroup>, <mark>, <meter>, <nav>, <progress>, <ruby>, <rt>, <section>, <time>, and <wpr>.
5. What elements have disappeared?
   * As mentioned above, <frame> and <frameset> have been eliminated. Other elements that are no longer supported include: <noframe>, <applet>, <bigcenter> and <basefront>.
6. What are the new media-related elements in HTML5?
   * HTML5 has strong support for media. There are now special <audio> and <video> tags. There are additional A/V support tags as well: <embed> is a container for 3rd party applications. <track> is for adding text tracks to media. <source> is useful for A/V media from multiple sources.
7. What are the new image elements in HTML5?
   * Canvas and WebGL. <Canvas> is a new element that acts as a container for graphical elements like images and graphics. Coupled with JavaScript, it supports 2D graphics. [WebGL](http://www.khronos.org/webgl/wiki/Main_Page" \t "_blank) stands for Web Graphics Language, a free cross-platform API that is used for generating 3D graphics in web browsers.
8. What is the difference between SVG and <Canvas>?
   * <Canvas> is an element that manipulates two-dimensional (2D) pixels while [Scalable Vector Graphics](http://www.w3.org/TR/SVG/) works in 2D and three-dimensional (3D) vectors. Essentially, <Canvas> is to SVG as Photoshop is to Illustrator.
9. What are some new input attributes in HTML5?
   * There are many new form elements including: datalist, datetime, output, keygen, date, month, week, time, number, range, email, and url.
10. What are data- attributes good for?
    * The [HTML5 data- attribute](http://webdesign.tutsplus.com/tutorials/htmlcss-tutorials/all-you-need-to-know-about-the-html5-data-attribute/) is a new addition that assigns custom data to an element. It was built to store sensitive or private data that is exclusive to a page or application, for which there are no other matching attributes or elements.
11. What is the difference between HTML5 interaction in Sencha and Twitter/Bootstrap?
    * [Sencha](http://senchadevelopers.blogspot.com/2012/10/what-you-can-do-with-sencha.html) and [Twitter/Bootstrap](http://webdesign.tutsplus.com/tutorials/complete-websites/twitter-bootstrap-101-introduction/) are both HTML development frameworks that integrate HTML5, CSS3, and JavaScript. The major difference is that in Sencha, the three languages are all comingled together in code, whereas in Bootstrap, HTML and CSS and decoupled.
12. Describe the difference between cookies, sessionStorage, andlocalStorage.
    * Cookies are small text files that websites place in a browser for tracking or login purposes. Meanwhile, [localStorage and sessionStorage](https://www.inkling.com/read/javascript-definitive-guide-david-flanagan-6th/chapter-20/localstorage-and-sessionstorage" \t "_blank) are new objects, both of which are storage specifications but vary in scope and duration. Of the two, localStorage is permanent and website-specific whereas sessionStorage only lasts as long as the duration of the longest open tab.
13. What are some of the major new API’s that come standard with HTML5?
    * To name a few: Media API, Text Track API, Application Cache API, User Interaction, Data Transfer API, Command API, Constraint Validation API, and the History API.
14. What is the difference in caching between HTML5 and the old HTML?
    * An important feature of HTML5 is the Application Cache. It creates an offline version of a web application. and stores website files such as HTML files, CSS, images, and JavaScript, locally. It is a feature that speeds up site performance.

#### JS Questions:

* Explain event delegation
* Explain how this works in JavaScript
* Explain how prototypal inheritance works
* How do you go about testing your JavaScript?
* AMD vs. CommonJS?
* Explain why the following doesn't work as an IIFE: function foo(){ }();.
  + What needs to be changed to properly make it an IIFE?
* What's the difference between a variable that is: null, undefined or undeclared?
  + How would you go about checking for any of these states?
* What is a closure, and how/why would you use one?
* What's a typical use case for anonymous functions?
* How do you organize your code? (module pattern, classical inheritance?)
* What's the difference between host objects and native objects?
* Difference between: function Person(){}, var person = Person(), and var person = new Person()?
* What's the difference between .call and .apply?

The main difference is that apply lets you invoke the function with arguments as an array; callrequires the parameters be listed explicitly.

See [here](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Function/apply) and [here](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Function/call).

Pseudo syntax:

theFunction.apply(valueForThis, arrayOfArgs)

theFunction.call(valueForThis, arg1, arg2, ...)

Sample code:

function theFunction(name, profession) {

alert("My name is " + name + " and I am a " + profession + ".");

}

theFunction("John", "fireman");

theFunction.apply(undefined, ["Susan", "school teacher"]);

theFunction.call(undefined, "Claude", "mathematician");

* explain Function.prototype.bind?
* When do you optimize your code?
* When would you use document.write()?
  + Most generated ads still utilize document.write() although its use is frowned upon
* What's the difference between feature detection, feature inference, and using the UA string
* Explain AJAX in as much detail as possible
* Explain how JSONP works (and how it's not really AJAX)
* Have you ever used JavaScript templating?
  + If so, what libraries have you used? (Mustache.js, Handlebars etc.)
* Explain "hoisting".
* Describe event bubbling.
* What's the difference between an "attribute" and a "property"?
* Why is extending built in JavaScript objects not a good idea?
* Difference between document load event and document ready event?
* What is the difference between == and ===?
* Explain the same-origin policy with regards to JavaScript.
* Make this work:

[1,2,3,4,5].duplicate(); *// [1,2,3,4,5,1,2,3,4,5]*

* Why is it called a Ternary expression, what does the word "Ternary" indicate?
* What is "use strict";? what are the advantages and disadvantages to using it?

**jQuery Questions:**

* Explain "chaining".
* Explain "deferreds".
* What are some jQuery specific optimizations you can implement?
* What does .end() do?
* Name 4 different values you can pass to the jQuery method.
  + Selector (string), HTML (string), Callback (function), HTMLElement, object, array, element array, jQuery Object etc.
* What is the difference between .get(), [], and .eq()?

<http://www.toptal.com/javascript/interview-questions>

<http://blog.sourcing.io/interview-questions>

**Security**

* It's a lot to digest but the [OWASP development guide](http://www.owasp.org/index.php/Category:OWASP_Guide_Project) covers Web Site security from top to bottom.
* Know about Injection especially [SQL injection](http://en.wikipedia.org/wiki/SQL_injection) and how to prevent it.
* Never trust user input, nor anything else that comes in the request (which includes cookies and hidden form field values!).
* Hash passwords using [salt](http://security.stackexchange.com/q/21263/396) and use different salts for your rows to prevent rainbow attacks. Use a slow hashing algorithm, such as bcrypt (time tested) or scrypt (even stronger, but newer) ([1](http://www.tarsnap.com/scrypt.html), [2](http://it.slashdot.org/comments.pl?sid=1987632&cid=35149842)), for storing passwords. ([How To Safely Store A Password](http://codahale.com/how-to-safely-store-a-password/)). The [NIST also approves of PBKDF2 to hash passwords](http://security.stackexchange.com/q/7689/396)", and it's [FIPS approved in .NET](http://security.stackexchange.com/a/2136/396) (more info [here](http://security.stackexchange.com/questions/211/how-to-securely-hash-passwords)). *Avoid* using MD5 or SHA family directly.
* [Don't try to come up with your own fancy authentication system](http://stackoverflow.com/questions/1581610/how-can-i-store-my-users-passwords-safely/1581919#1581919). It's such an easy thing to get wrong in subtle and untestable ways and you wouldn't even know it until *after* you're hacked.
* Know the [rules for processing credit cards](https://www.pcisecuritystandards.org/). ([See this question as well](http://stackoverflow.com/questions/51094/payment-processors-what-do-i-need-to-know-if-i-want-to-accept-credit-cards-on-m))
* Use [SSL](http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt)/[HTTPS](http://en.wikipedia.org/wiki/Https) for login and any pages where sensitive data is entered (like credit card info).
* [Prevent session hijacking](http://en.wikipedia.org/wiki/Session_hijacking#Prevention).
* Avoid [cross site scripting](http://en.wikipedia.org/wiki/Cross-site_scripting) (XSS).
* Avoid [cross site request forgeries](http://en.wikipedia.org/wiki/Cross-site_request_forgery) (CSRF).
* Avoid [Clickjacking](http://en.wikipedia.org/wiki/Clickjacking).
* Keep your system(s) up to date with the latest patches.
* Make sure your database connection information is secured.
* Keep yourself informed about the latest attack techniques and vulnerabilities affecting your platform.
* Read [The Google Browser Security Handbook](http://code.google.com/p/browsersec/wiki/Main).
* Read [The Web Application Hacker's Handbook](http://amzn.com/0470170778).
* Consider [The principal of least privilege](https://en.wikipedia.org/wiki/Principle_of_least_privilege). Try to run your app server [as non-root](http://security.stackexchange.com/questions/47576/do-simple-linux-servers-really-need-a-non-root-user-for-security-reasons). ([tomcat example](http://tomcat.apache.org/tomcat-8.0-doc/security-howto.html#Non-Tomcat_settings))

**Performance**

* Implement caching if necessary, understand and use [HTTP caching](http://www.mnot.net/cache_docs/) properly as well as [HTML5 Manifest](http://www.w3.org/TR/2011/WD-html5-20110525/offline.html).
* Optimize images - don't use a 20 KB image for a repeating background.
* Learn how to [gzip/deflate content](http://developer.yahoo.com/performance/rules.html" \l "gzip) ([deflate is better](http://stackoverflow.com/questions/1574168/gzip-vs-deflate-zlib-revisited)).
* Combine/concatenate multiple stylesheets or multiple script files to reduce number of browser connections and improve gzip ability to compress duplications between files.
* Take a look at the [Yahoo Exceptional Performance](http://developer.yahoo.com/performance/) site, lots of great guidelines, including improving front-end performance and their [YSlow](http://developer.yahoo.com/yslow/) tool (requires Firefox, Safari, Chrome or Opera). Also, [Google page speed](https://developers.google.com/speed/docs/best-practices/rules_intro) (use with [browser extension](https://developers.google.com/speed/pagespeed/insights_extensions)) is another tool for performance profiling, and it optimizes your images too.
* Use [CSS Image Sprites](http://alistapart.com/articles/sprites) for small related images like toolbars (see the "minimize HTTP requests" point)
* Busy web sites should consider [splitting components across domains](http://developer.yahoo.com/performance/rules.html#split). Specifically...
* Static content (i.e. images, CSS, JavaScript, and generally content that doesn't need access to cookies) should go in a separate domain [*that does not use cookies*](http://blog.stackoverflow.com/2009/08/a-few-speed-improvements/), because all cookies for a domain and its subdomains are sent with every request to the domain and its subdomains. One good option here is to use a Content Delivery Network (CDN), but consider the case where that CDN may fail by including alternative CDNs, or local copies that can be served instead.
* Minimize the total number of HTTP requests required for a browser to render the page.
* Utilize [Google Closure Compiler](http://developers.google.com/closure/compiler/) for JavaScript and [other minification tools](http://developer.yahoo.com/yui/compressor/).
* Make sure there’s a favicon.ico file in the root of the site, i.e. /favicon.ico. [Browsers will automatically request it](http://mathiasbynens.be/notes/rel-shortcut-icon), even if the icon isn’t mentioned in the HTML at all. If you don’t have a /favicon.ico, this will result in a lot of 404s, draining your server’s bandwidth.

**SEO (Search Engine Optimization)**

* Use "search engine friendly" URLs, i.e. use example.com/pages/45-article-title instead of example.com/index.php?page=45
* When using # for dynamic content change the # to #! and then on the server $\_REQUEST["\_escaped\_fragment\_"] is what googlebot uses instead of #!. In other words, ./#!page=1 becomes ./?\_escaped\_fragments\_=page=1. Also, for users that may be using FF.b4 or Chromium, history.pushState({"foo":"bar"}, "About", "./?page=1"); Is a great command. So even though the address bar has changed the page does not reload. This allows you to use ? instead of #! to keep dynamic content and also tell the server when you email the link that we are after this page, and the AJAX does not need to make another extra request.
* Don't use links that say ["click here"](http://ux.stackexchange.com/questions/12100/why-shouldnt-we-use-the-word-here-in-a-textlink). You're wasting an SEO opportunity and it makes things harder for people with screen readers.
* Have an [XML sitemap](http://www.sitemaps.org/), preferably in the default location /sitemap.xml.
* Use [<link rel="canonical" ... />](http://googlewebmastercentral.blogspot.com/2009/02/specify-your-canonical.html) when you have multiple URLs that point to the same content, this issue can also be addressed from [Google Webmaster Tools](http://www.google.com/webmasters/).
* Use [Google Webmaster Tools](http://www.google.com/webmasters/) and [Bing Webmaster Tools](http://www.bing.com/toolbox/webmaster).
* Install [Google Analytics](http://www.google.com/analytics/) right at the start (or an open source analysis tool like [Piwik](http://piwik.org/)).
* Know how [robots.txt](http://en.wikipedia.org/wiki/Robots_exclusion_standard) and search engine spiders work.
* Redirect requests (using 301 Moved Permanently) asking for www.example.com to example.com (or the other way round) to prevent splitting the google ranking between both sites.
* Know that there can be badly-behaved spiders out there.
* If you have non-text content look into Google's sitemap extensions for video etc. There is some good information about this in [Tim Farley's answer](http://stackoverflow.com/questions/72394/what-should-a-developer-know-before-building-a-public-web-site#167608).

**Technology**

* Understand [HTTP](http://www.ietf.org/rfc/rfc2616.txt) and things like GET, POST, sessions, cookies, and what it means to be "stateless".
* Write your [XHTML](http://www.w3.org/TR/xhtml1/)/[HTML](http://www.w3.org/TR/REC-html40/) and [CSS](http://www.w3.org/TR/CSS2/) according to the [W3C specifications](http://www.w3.org/TR/) and make sure they [validate](http://validator.w3.org/). The goal here is to avoid browser quirks modes and as a bonus make it much easier to work with non-traditional browsers like screen readers and mobile devices.
* Understand how JavaScript is processed in the browser.
* Understand how JavaScript, style sheets, and other resources used by your page are loaded and consider their impact on *perceived* performance. It is now widely regarded as appropriate to [move scripts to the bottom](http://developer.yahoo.com/blogs/ydn/posts/2007/07/high_performanc_5/) of your pages with exceptions typically being things like analytics apps or HTML5 shims.
* Understand how the JavaScript sandbox works, especially if you intend to use iframes.
* Be aware that JavaScript can and will be disabled, and that AJAX is therefore an extension, not a baseline. Even if most normal users leave it on now, remember that [NoScript](http://noscript.net/) is becoming more popular, mobile devices may not work as expected, and Google won't run most of your JavaScript when indexing the site.
* Learn the [difference between 301 and 302 redirects](http://www.bigoakinc.com/blog/when-to-use-a-301-vs-302-redirect/) (this is also an SEO issue).
* Learn as much as you possibly can about your deployment platform.
* Consider using a [Reset Style Sheet](http://stackoverflow.com/questions/167531/is-it-ok-to-use-a-css-reset-stylesheet) or [normalize.css](http://necolas.github.com/normalize.css/).
* Consider JavaScript frameworks (such as [jQuery](http://jquery.com/), [MooTools](http://mootools.net/), [Prototype](http://www.prototypejs.org/), [Dojo](http://dojotoolkit.org/) or [YUI 3](http://developer.yahoo.com/yui/3/)), which will hide a lot of the browser differences when using JavaScript for DOM manipulation.
* Taking perceived performance and JS frameworks together, consider using a service such as the[Google Libraries API](http://developers.google.com/speed/libraries/devguide) to load frameworks so that a browser can use a copy of the framework it has already cached rather than downloading a duplicate copy from your site.
* Don't reinvent the wheel. Before doing ANYTHING search for a component or example on how to do it. There is a 99% chance that someone has done it and released an OSS version of the code.
* On the flipside of that, don't start with 20 libraries before you've even decided what your needs are. Particularly on the client-side web where it's almost always ultimately more important to keep things lightweight, fast, and flexible.

**Bug fixing**

* Understand you'll spend 20% of your time coding and 80% of it maintaining, so code accordingly.
* Set up a good error reporting solution.
* Have a system for people to contact you with suggestions and criticisms.
* Document how the application works for future support staff and people performing maintenance.
* Make frequent backups! (And make sure those backups are functional) Have a restore strategy, not just a backup strategy.
* Use a version control system to store your files, such as [Subversion](http://subversion.apache.org/), [Mercurial](http://mercurial.selenic.com/) or [Git](http://git-scm.org/).
* Don't forget to do your Acceptance Testing. Frameworks like [Selenium](http://seleniumhq.org/) can help.
* Make sure you have sufficient logging in place using frameworks such as [log4j](http://logging.apache.org/log4j/), [log4net](http://logging.apache.org/log4net/) or [log4r](http://log4r.rubyforge.org/). If something goes wrong on your live site, you'll need a way of finding out what.
* When logging make sure you capture both handled exceptions, and unhandled exceptions. Report/analyse the log output, as it'll show you where the key issues are in your site.