Essential Skills in Web Development

PV219, spring 2019

Interface and User Experience

- Browsers implement standards inconsistently, make sure your site works reasonably well across all major browsers.
- At a minimum test against a recent Gecko engine (Firefox), a WebKit engine (Safari and some mobile browsers), Chrome (= Opera), and your supported IE browsers.
- Also consider how browsers render your site in different operating systems.

Interface and User Experience

- Consider how people might use the site other than from the major browsers: *cell phones*, *screen readers* and *search engines*, for example.
- Some accessibility info: WAI and Section508.
- It should be a legal requirement. Utilize: WAI-ARIA and WCAG 2 .

Interface and User Experience

- Don't display unfriendly errors directly to the user.
- Add the attribute *rel="nofollow"* to user-generated links to avoid spam.
- Build well-considered limits into your site - This also belongs under Security.

Interface and User Experience

- Learn how to do progressive enhancement or graceful degradation.
- Redirect after a POST if that POST was successful, to prevent a refresh from submitting again.
- Don't make me think

Security

- It's a lot to digest but the OWASP development guide covers Web Site security from top to bottom.
- Know about Injection especially 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!).

Security

- Hash passwords using salt 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, 2), for storing passwords.
- *Avoid* using MD5 or SHA family directly.

Security

- Don't try to come up with your own fancy authentication system. 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.
- Use SSL/HTTPS for login and any pages where sensitive data is entered (like credit card info).
- EU General Data Protection Regulation (GDPR)

Security

- Prevent session hijacking.
- Avoid cross site scripting (XSS).
- Avoid cross site request forgeries (CSRF).
- Avoid Clickjacking.

- Read The Google Browser Security Handbook.
- Read The Web Application Hacker's Handbook.

Security

- Consider The principal of least/minimal privilege. Try to run your app server as non-root.
- Keep your system(s) up to date with the latest patches.
- Make sure your database connection information is secured.

Performance

- Implement caching if necessary, understand and use HTTP caching properly as well as HTML5 Manifest.
- Optimize images – i.e. don't use a 20 Kb image for a repeating background.
- Learn how to gzip/deflate content (deflate is better).

Performance

- Combine/concatenate multiple stylesheets or multiple script files to reduce number of browser connections and improve gzip ability to compress duplications between files.
- Use CSS Image Sprites for small related images like toolbars (because of next point)
- Minimize the total number of HTTP requests required for a browser to render the page.

Performance

- Yahoo Exceptional Performance – lots of great guidelines, including improving front-end performance and their YSlow tool (requires Firefox, Safari, Chrome or Opera).
- Google page speed (use with browser extension) – a tool for performance profiling, and it optimizes your images too.

Performance

- Utilize Google Closure Compiler for JavaScript and other minification tools.
- Make sure there's a *favicon.ico* file in the root of the site, i.e. /favicon.ico. Browsers will automatically request it, 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.

Technology

- Understand HTTP and things like GET, POST, sessions, cookies, and what it means to be "stateless".
- Write your XHTML/HTML and CSS according to the W3C specifications and make sure they validate.
- Understand how JavaScript is processed in the browser.

Technology

- Understand how the JavaScript sandbox works, especially if you intend to use iframes.
- JavaScript can and will be disabled, and that AJAX is therefore an extension, not a baseline.
- NoScript is becoming more popular, mobile devices may not work as expected, and Google won't run most of your JavaScript when indexing the site.

Technology

- Learn the difference between 301 and 302 redirects (this is also an SEO issue).
- Consider using a Reset Style Sheet or normalize.css.
- Consider using a service such as the Google Libraries API to load frameworks.

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.

Bug fixing

- 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.

Bug fixing

- Use a version control system to store your files, such as Subversion, Mercurial or Git.
- Don't forget to do your Acceptance Testing.
- Frameworks like Selenium can help.

Bug fixing

- Make sure you have sufficient logging in place using frameworks such as log4j, log4net or log4r.
- 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/analyze the log output, as it'll show you where the key issues are in your site.