A new baseline for front-end development

In 2012 Rebeca Murphey wrote A baseline for front-end developers as a minimum of what people should know to work in front-end development back in the day.

I went back to the article and was surprised at how well it holds up 7 years after it was written; the tools may have changed but we need to keep the discussion going. What does it take to do front-end development in 2019? What tools we need to do it?.

Javascript

I won't advocate specific frameworks or technologies. I believe we should all start from a basic understanding of the language in question so we can build basic interactivity and move to advanced concepts and frameworks at a later time.

If you need to work through some of the basics these are some good resources

Eloquent Javascript: A wonderful book (also available as a PDF, an EPUB and a print purchase) that takes you back to JavaScript basics without being overbearing.

While Eloquent Javascript provides a strong foundation it doesn't cover all the changes brought into the language in newer versions. JavaScript for impatient programmers provides a more recent introductory Javascript text, covering the language until its more recent update (ES2019). It also skips portions of the language that only work in the browser.

If you feel like you have a good background in the lnaguage, Exploring ES6 and the updates Exploring ES2016 and 2017 and Exploring ES2018 and ES2019 provide information only on the new feaatures released in those version of the language specification.

Babel

Front end developers should know what Babel is, how it works and how to use it to create bundles based on what the browser supports using the `babel-env` plugin.

Anothe important thing to know when working with Babel is how to build a Babel configuration file to convert modern ES2015+ so it will run in current browsers.

For a refresher I suggest A short and simple guide to Babel

CSS and pre-processors

CSS Preprocessors like SASS and LESS, Compass (and their associated libraries like Bourbon Neat, Bitters, and others) and post processors like PostCSS have greatly improved the way we work with CSS but whatever tool we use, we need to remember that we still need to know our CSS basics... none of the tools will build CSS for us, they will only enhance what's already there.

Sites like MDN's Learn to style HTML using CSS are a good starting point for learning the what and the how about CSS.

The last thing in this area is to know about vendor prefixes, what they are, how they work and how to implement them with tools like Autoprefixer.

Modularity

The way we create modular content has changed. From AMD and CommonJS we're moving to CommonJS in Node.js and ES native modules and, some time in the not so distant future, to native ES Modules alone.

As front end developers we should be aware of the differences between Common JS and ES Modules, where and when we would use each of them.

We should learn how to optimize loading of production code using differential loading.

Understand what the following snippet does

```
<script type="module" src="myModules.js"></script>
<script nomodule src="myScript.js" defer></script>
```

Git and Github

Most development happens in Git repositories, either private, hosted on Gitlab or hosted on Github (who has lost users since the Microsoft acquisition1).

Git has become essential to development, either front-end or backend so we need to know the basics of how the software works.

Some suggested starting points.

- Create repositories in your local machine and on your Git host
- Set your local git repostory to sync with a remote Git server like Github, Gitlab or another Git host
- Create and use a branching strategy for your team to collaborate in a project

There are plenty of resources for working with Git.

- The book Pro Git is available under a Creative Commons license
- Github Resources
    - Git Handbook
    - Git Cheatsheet

Build and Process Automation

Building our web content has gotten more complicated both in terms of what we're requiring our build systems to do and the number of build systems available on the market.

Pick a system and learn it well enough so you can use it to build the tooling that you need for your projects.

Understand the differences and similarities between build systems like Gulp, Grunt and others versus bundlers like WebPack and RollUp and when would you use one or the other.

Mulyana's Gulp 4 tutorial provides both a good overview of how to implement plugins and an example of the varied things you can do with the tool

The Web Performance Optimization with webpack series gives you a good

starting point for how to use Webpack.

Browser Dev Tools

The developer tools built into modern browsers give you a lot of power to inspect your site/app. Learn what DevTools can do and how to best use them... for example, learn how to use DevTools to debug a web application on a device.

Different browsers DevTools have different areas where they excel so it pays to learn what the differences are and what areas where each of them excels (if any).

Chrome has a particularly good PWA debugging and give you an easier way to clean up the content of your caches when you're testing your application.

Firefox CSS Grid Inspector and Shapes Editor make it easier for you to work with the respective items.

Here's some information for each browser's DevTools

- Chrome DevTools 101
- Firefox DevTools
- Safari browser for developers

Your web server matters

The web server we use to serve our content may also have an impact on how we buildand package our content. Be aware of the differences between HTTP1.1 and http2 in terms of performance and how it changes the way we package and deliver our front end content.

See Getting Ready For HTTP2: A Guide For Web Designers And Developers, and HTTP2 is here, let's optimize for starting points in understanding the differences and how they affect front end practices.