Frontend Masters

Table of Contents

Front-End Developer Handbook 2018

Written by Cody Lindley sponsored by — Frontend Masters

Frontend Masters

This is a guide that anyone could use to learn about the practice of front-end development. It broadly outlines and discusses the practice of front-end engineering: how to learn it and what tools are used when practicing it in 2018.

It is specifically written with the intention of being a professional resource for potential and currently practicing front-end developers to equip themselves with learning materials and development tools. Secondarily, it can be used by managers, CTOs, instructors, and head hunters to gain insights into the practice of front-end development.

The content of the handbook favors web technologies (HTML, CSS, DOM, and JavaScript) and those solutions that are directly built on top of these open technologies. The materials referenced and discussed in the book are either best in class or the current offering to a problem.

The book should not be considered a comprehensive outline of all resources available to a front-end developer. The value of the book is tied up in a terse, focused, and timely curation of just enough categorical information so as not to overwhelm anyone on any one particular subject matter.

The intention is to release an update to the content yearly.

The handbook is divided into the following three parts:

Part I. The Front-End Practice

Part one broadly describes the practice of front-end engineering.

Part II: Learning Front-End Development

Part two identifies self-directed and direct resources for learning to become a front-end developer.

Part III: Front-End Development Tools

Part three briefly explains and identifies tools of the trade.

Download a .pdf, .epub, or .mobi file from:

- https://www.gitbook.com/book/frontendmasters/front-end-developer-handbook-2018/details

Contribute content, suggestions, and fixes on github:

- https://github.com/FrontendMasters/front-end-handbook-2018

What Is a Front-End Developer?

> Front-end web development, also known as client-side development is the practice of producing HTML, CSS and JavaScript for a website or Web Application so that a user can see and interact with them directly. The challenge associated with front end development is that the tools and techniques used to create the front end of a website change constantly and so the developer needs to constantly be aware of how the field is developing.
>
> The objective of designing a site is to ensure that when the users open up the site they see the information in a format that is easy to read and relevant. This is further complicated by the fact that users now use a large variety of devices with varying screen sizes and resolutions thus forcing the designer to take into consideration these aspects when designing the site. They need to ensure that their site comes up correctly in different browsers (cross-browser), different operating systems (cross-platform) and different devices (cross-device), which requires careful planning on the side of the developer.
>
> *https://en.wikipedia.org/wiki/Front-end_web_development*

HTML, CSS, & JavaScript:

A front-end developer architects and develops websites and applications using web technologies (i.e., HTML, CSS, DOM, and JavaScript), which run on the Open Web Platform or act as compilation input for non-web platform environments (i.e., React Native).

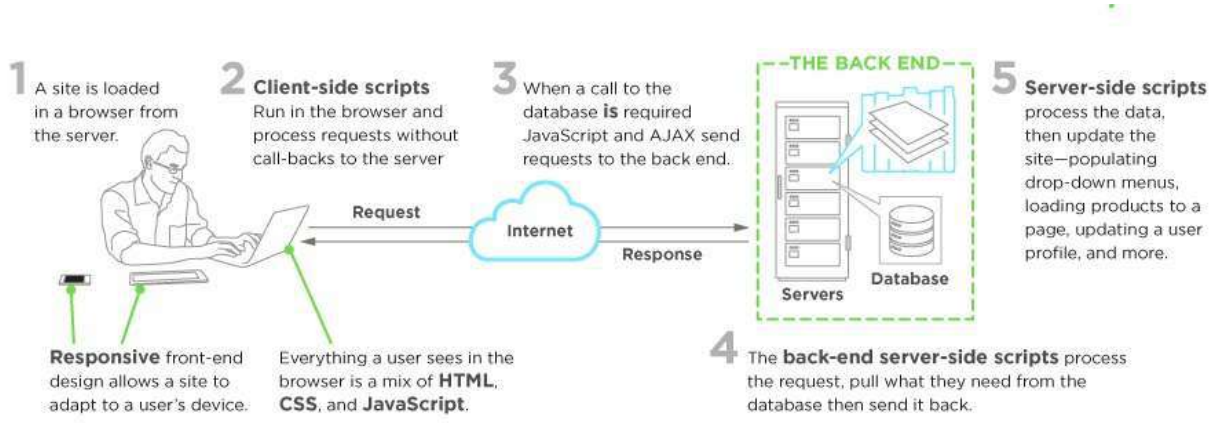


Image source: https://www.upwork.com/hiring/development/front-end-developer/

Typically, a person enters into the field of front-end development by learning to develop HTML, CSS, and JavaScript which commonly runs in web browser but can also run in a headless browser, WebView, or as compilation input for a native runtime environment.

These four run times scenarios are explained below.

Web Browsers (most common)

A web browser is software used to retrieve, present, and traverse information on the WWW. Typically, browsers run on a desktop or laptop computer, tablet, or phone, but as of late a browser can be found on just about anything (i.e, on a fridge, in cars, etc.).

The most common web browsers are (shown in order of most used first):

- Chrome
- Safari
- Internet Explorer (Note: not Edge, referring to IE 9 to IE 11)
- Firefox
- Edge

Headless Browsers

Headless browsers are a web browser **without** a graphical user interface that can be controlled from a command line interface programmatically for the purpose of web page automation (e.g., functional testing, scraping, unit testing, etc.). Think of headless browsers as a browser that you can run from the command line that can retrieve and traverse web pages.

The most common headless browsers are:

- Headless Chromium
- Zombie
- slimerjs

Webviews

Webviews are used by a native OS, in a native application, to run web pages. Think of a webview like an iframe or a single tab from a web browser that is embedded in a native application running on a device (e.g., iOS, android, windows).

The most common solutions for webview development are:

- Cordova (typically for native phone/tablet apps)
- NW.js (typically used for desktop apps)
- Electron (typically used for desktop apps)

Native from Web Tech

Eventually, what is learned from web browser development can be used by front-end developers to craft code for environments that are not fueled by a browser engine. As of late, development environments are being dreamed up that use web technologies (e.g., CSS and JavaScript), without web engines, to create native applications.

Some examples of these environments are:

- Flutter
- React Native

NOTES:

Make sure you are clear what exactly is meant by the "web platform". Read, "The Web platform: what it is" and read the, "Open Web Platform" Wikipedia page. Explore the many technologies that make up the web platform.

Recap of Front-end Development in 2017

- HTML 5.2 is done.
- It was a banner year for Vue.js in terms of adoption and popularity. No question about it.
- The great divide between a front-end HTML & CSS developer v.s. front-end application developer is realized/verbalized.
- Being a Front-end JavaScript developer who builds applications using web technologies continues to get better and worse.
- This year seemed fuller than most of app/framework solutions trying to contend with the mainstream JavaScript app tools (i.e. React, Angular, and Vue etc...) Let me list them for you. Moon, Marko, Hyperapp, Quasar Framework, POI, frint, BunnyJS, jsblocks, Sapper, Stimulus, Choo,
- This was the year that jsbin and jsfiddle evolved to things like StackBliz and codeSandbox. Making it dead simple to share a working app.
- React continues to be flattered by things like preact, inferno, nerv, dva, and rax.
- Cheatsheets got organized with devhints.io.
- We figured out that the correct pattern for an app boilerplate/cli tool is something very opinionated like React Create App with the ability to escape from it when needed.
- Most developers found that the combination of a really good code editor, eslint, and now prettier make writing code faster, easier, pleasurable.
- CSS Flexbox and Grid gain browser support and thus more developers are paying attention to both.
- We get, a headless chrome, finally.
- You no longer need Less or Sass to do amazing things with CSS.
- CSS revolutions/revolts are under way.
- JavaScript object explorer tools have arrived, JavaScript Array Explorer and JavaScript Object Explorer. This is a handy interface pattern for learning about JavaScript data types (e.g. Objects and Arrays) and their methods.
- The Chrome web browser dominates the market and people begin to fear the past might be repeating itself.
- Brave becomes the most pleasant and safest way to browser the internet.
- PhantomJS is no longer maintained, Headless Chrome and Puppeteer step in.
- Prettier comes from left field and becomes a staple for development.
- A whole lot of developers adopt static type checking for mostly subjective reasons or band wagon emotions. Some sell out completely to Typescript and the Microsoft way of doing things while others take on a slower approach with Flow. One thing is for sure, most developers don't need types, they are simply complicating already complex problems and solutions. Like most things, most of this trend is subjective dogma not

objective value.

- Static site generators & API CMS tools aka Headless CMS's are now on most developers radar.
- Web components still lurking and wait for significant traction by developers that might never come to be.
- JavaScript settled and CSS erupt and everyone will cry fatigue by this time next year.
- A lot of people stop doing CSS in CSS and move to CSS in JS when building application using component trees.
- Yarn seems to have filled a need, because a lot of people jump the npm ship. However, the real value of Yarn is the fact that it brings competition to NPM. Making npm better.
- A new video format for Interactive coding screencasts (recording of working in a live editor that you can edit too) becomes a real thing with Scrimba.
- Most people begin to see the correlation between component architectures and atomic design.
- And so it begins that ES modules will be part of the browser and if used a backup plan will be required (i.e. a bundle from something like webpack).
- MVC frameworks are on the outs.
- Developing and displaying React components outside of your applications is made popular by tools like Bluekit, Storybook, React Styleguidist, and bit.
- Getting a front-end job in 2017 is about experience, which is displayed from personal projects and a developers Github account.
- Preloading resources (CSS, JavaScript, Media etc..) from HTML documents arrives.
- Cypress arrives as a complete testing solution and hopefully testing will get better as end to end testing becomes the focus for app code.
- WebAssembly support now shipping in all major browsers
- Webpack dominates, and then competitors show up.
- React 16 aka fiber is released.
- React begins to rival jQuery in popularity in certain contexts.
- React clearly is the most used tool for building UI's with state.
- Facebook sheds its React BSD license for the MIT license (same for Jest, Flow, Immutable.js, and GraphQL)
- GraphQL got hot in 2017.
- Facebook continues to take charge in the development space with forthcoming tools like prepack.io.
- As expected ECMA-262 edition 8 is released.
- React Router finally stabilizes.
- All modern browsers pretty much now support ECMAScript 2015 (aka ES6).
- Async JavaScript functions start getting some serious attention and usage. Mostly because all modern browsers now support Async functions.
- Mobile development, still too hard. A strong rebellion advocating the web platform as a

solution to the pain gained momentum this year.

In 2018 expect...

- Nothing will change or slow the usage or popularity of React for many years to come.
- GraphQL will replace a lot of REST API's this year.
- The web will continue to become more native-like with offline capabilities and seamless mobile experiences.
- HTML 5.3 is coming.
- Keep an eye on turbo, a blazing fast NPM client.
- Expect to learn and use CSS transforms 3d, CSS transitions, CSS flexbox, CSS filters, CSS grid
- JavaScript usage will continue to grow with no slowdown in sight.
- Still waiting on Web Assembly to peak. This will likely require tooling.
- Universal/isomorphic JavaScript solutions continue to evolve e.g. next.js and Sapper.
- Web components still lurk and wait for significant traction from developers.
- I believe the end is in sight for CSS pre-processors as PostCSS, CSSnext, and CSS in JS take over.
- Older server centric application patterns show up again but with a new spin. The pendulum could start to swinging away from strick SPA applications. People will begin to pull back on the complexity of single page applications and return to things like pjax (A mix of SPA and Server-side Rendering. See https://stimulusjs.org).
- Progressive Web Applications hopefully will catch fire. If they don't, I fear they never will. At least not in there current form.
- "Chatbots created on the basis of artificial intelligence and neural networks will continue to evolve helping to increase communication online. I wonder what it will lead to, but this is unconditional web development trends 2018". Nods.
- Vue.js usage will likely overtake all Angular usage.
- AR/AV, AI, and chat bots will continue to evolve and find there sweet spot.
- JavaScript Symbol and Generators will likely go unnoticed by most front-end developers.
- More developers will divorce themselves from plain JavaScript and try to marry another. But, just like in marital divorce one always takes most of the same problems with them to the greener grass and little actually changes. Preferences and values just get re-prioritized and history will repeat itself.
- Webpack 4 will happen, and be better, due to competition!
- Continued exploration for the ideal CSS solution for a tree of UI components will not cease.
- State management gets a reset and people start to simplify. Hopefully, this will be the year for solutions like mobx to shine.

Part I. The Front-End Practice

Part one broadly describes the practice of front-end engineering.

Front-End Jobs Titles

Below is a list and description of various front-end job titles. The common, or most used (i.e., generic), title for a front-end developer is, "front-end developer" or "front-end engineer". Note that any job that contains the word "front-end", "client-side", "web UI", "HTML", "CSS", or "JavaScript" typically infers that a person has some degree of HTML, CSS, DOM, and JavaScript professional know how.

Front-End Developer

The generic job title that describes a developer who is skilled to some degree at HTML, CSS, DOM, and JavaScript and implementing these technologies on the web platform.

Front-End Engineer (aka JavaScript Developer or Full-stack JavaScript Developer)

The job title given to a developer who comes from a computer science, engineering, background and is using these skills to work with front-end technologies. This role typically requires a computer science degree and years of software development experience. When the word "JavaScript Application" is included in the job title, this will denote that the developer should be an advanced JavaScript developer possessing advanced programming, software development, and application development skills (i.e has years of experience building front-end applications).

CSS/HTML Developer

The front-end job title that describes a developer who is skilled at HTML and CSS, excluding JavaScript and Application know how.

Front-End Web Designer

When the word "Designer" is included in the job title, this will denote that the designer will posses front-end skills (i.e., HTML & CSS) but also professional design (Visual Design and Interaction Design) skills.

Web/Front-End User Interface (aka UI) Developer/Engineer

When the word "Interface" or "UI" is included in the job title, this will denote that the developer should posses interaction design skills in addition to front-end developer skills or front-end engineering skills.

Mobile/Tablet Front-End Developer

When the word "Mobile" or "Tablet" is included in the job title, this will denote that the developer has experience developing front-ends that run on mobile or tablet devices (either natively or on the web platform, i.e., in a browser).

Front-End SEO Expert

When the word "SEO" is included in the job title, this will denote that the developer has extensive experience crafting front-end technologies towards an SEO strategy.

Front-End Accessibility Expert

When the word "Accessibility" is included in the job title, this will denote that the developer has extensive experience crafting front-end technologies that support accessibility requirements and standards.

Front-End Dev. Ops

When the word "DevOps" is included in the job title, this will denote that the developer has extensive experience with software development practices pertaining to collaboration, integration, deployment, automation, and measurement.

Front-End Testing/QA

When the word "Testing" or "QA" is included in the job title, this will denote that the developer has extensive experience testing and managing software that involves unit testing, functional testing, user testing, and A/B testing.

NOTES:

If you come across the "Full Stack" or the generic "Web Developer" terms in job titles these words may be used by an employer to describe a role that is responsible for all aspects of web/app development, i.e., both front-end (potentially including design) and back-end.

Web Technologies Employed by Front-End Developers

Image source: http://www.2n2media.com/compare-front-end-development-and-back-end-development

The following core web technologies are employed by front-end developers (consider learning them in this order):

1. Hyper Text Markup Language (aka HTML)
2. Cascading Style Sheets (aka CSS)
3. Uniform Resource Locators (aka URLs)
4. Hypertext Transfer Protocol (aka HTTP)
5. JavaScript Programming Language (aka ECMAScript 262)
6. JavaScript Object Notation (aka JSON)
7. Document Object Model (aka DOM)
8. Web APIs (aka HTML5 and friends or Browser APIs)
9. Web Content Accessibility Guidelines (aka WCAG) & Accessible Rich Internet Applications (aka ARIA)

For a comprehensive list of all web related specifications have a look at platform.html5.org.

The nine technologies just mentioned are defined below along with a link to the relevant documentation and specification for each technology.

Hyper Text Markup Language (aka HTML)

> HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.
>
> — *Wikipedia*

Most relevant specifications / documentation:

- All W3C HTML Spec
- The elements of HTML from the Living Standard
- Global attributes
- HTML 5.2 from W3C
- HTML attribute reference
- HTML element reference
- The HTML Syntax from the Living Standard

Cascading Style Sheets (aka CSS)

> Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. Although most often used to change the style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.
>
> — *Wikipedia*

Most relevant specifications / documentation:

- All W3C CSS Specifications
- Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification
- CSS reference
- Selectors Level 3

Hypertext Transfer Protocol (aka HTTP)

> The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.
>
> — *Wikipedia*

Most relevant specifications:

- Hypertext Transfer Protocol -- HTTP/1.1
- HTTP/2

Uniform Resource Locators (aka URL)

> A uniform resource locator (URL) (also called a web address) is a reference to a resource that specifies the location of the resource on a computer network and a mechanism for retrieving it. A URL is a specific type of uniform resource identifier (URI), although many people use the two terms interchangeably. A URL implies the means to access an indicated resource, which is not true of every URI. URLs occur most commonly to reference web pages (http), but are also used for file transfer (ftp), email (mailto), database access (JDBC), and many other applications.
>
> — *Wikipedia*

Most relevant specifications:

- Uniform Resource Locators (URL)
- URL Living Standard

Document Object Model (aka DOM)

> The Document Object Model (DOM) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML, and XML documents. The nodes of every document are organized in a tree structure, called the DOM tree. Objects in the DOM tree may be addressed and manipulated by using methods on the objects. The public interface of a DOM is specified in its application programming interface (API).
>
> — *Wikipedia*

Most relevant specifications / documentation:

- Document Object Model (DOM) Level 3 Events Specification
- DOM Living Standard
- W3C DOM4

JavaScript Programming Language (aka ECMAScript 262)

> JavaScript is a high level, dynamic, untyped, and interpreted programming language. It has been standardized in the ECMAScript language specification. Alongside HTML and CSS, it is one of the three essential technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern web browsers without plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage or graphics facilities, relying for these upon the host environment in which it is embedded.
>
> — *Wikipedia*

Most relevant specifications / documentation:

- ECMAScript® 2017 Language Specification

Web APIs (aka HTML5 and friends)

> When writing code for the Web using JavaScript, there are a great many APIs available. Below is a list of all the interfaces (that is, types of objects) that you may be able to use while developing your Web app or site.
>
> — *Mozilla*

Most relevant documentation:

- Web API Interfaces

JavaScript Object Notation (aka JSON)

> c It is the primary data format used for asynchronous browser/server communication (AJAJ), largely replacing XML (used by AJAX). Although originally derived from the JavaScript scripting language, JSON is a language-independent data format. Code for parsing and generating JSON data is readily available in many programming languages. The JSON format was originally specified by Douglas Crockford. It is currently described by two competing standards, RFC 7159 and ECMA-404. The ECMA standard is minimal, describing only the allowed grammar syntax, whereas the RFC also provides some semantic and security considerations. The official Internet media type for JSON is application/json. The JSON filename extension is .json.
>
> — *Wikipedia*

Most relevant specifications:

- Introducing JSON

- JSON API
- The JSON Data Interchange Format

Web Content Accessibility Guidelines (aka WCAG) & Accessible Rich Internet Applications (aka ARIA)

> Accessibility refers to the design of products, devices, services, or environments for people with disabilities. The concept of accessible design ensures both “direct access” (i.e., unassisted) and "indirect access" meaning compatibility with a person's assistive technology (for example, computer screen readers).
>
> — *Wikipedia*

- Accessible Rich Internet Applications (WAI-ARIA) Current Status
- Web Accessibility Initiative (WAI)
- Web Content Accessibility Guidelines (WCAG) Current Status

Front-End Dev Skills

Image source: http://blog.naustud.io/2015/06/baseline-for-modern-front-end-developers.html

Basic to advanced HTML, CSS, DOM, JavaScript, HTTP/URL, and browser skills are assumed for any type of front-end developer.

Beyond HTML, CSS, DOM, JavaScript, HTTP/URL, and browser development know-how, a front-end developer could be skilled in one or more of the following:

- Content Management Systems (aka CMS)
- Node.js
- Cross-Browser Testing
- Cross-Platform Testing
- Unit Testing
- Cross-Device Testing
- Accessibility / WAI-ARIA
- Search Engine Optimization (aka SEO)
- Interaction or User Interface Design
- User Experience
- Usability
- E-commerce Systems
- Portal Systems
- Wireframing
- CSS Layout / Grids
- DOM Manipulation (e.g., jQuery)

- Mobile Web Performance
- Load Testing
- Performance Testing
- Progressive Enhancement / Graceful Degradation
- Version Control (e.g., GIT)
- MVC / MVVM / MV*
- Functional Programming
- Data Formats (e.g., JSON, XML)
- Data APIs (e.g Restful API)
- Web Font Embedding
- Scalable Vector Graphics (aka SVG)
- Regular Expressions
- Content Strategy
- Microdata / Microformats
- Task Runners, Build Tools, Process Automation Tools
- Responsive Web Design
- Object-Oriented Programming
- Application Architecture
- Modules
- Dependency Managers
- Package Managers
- JavaScript Animation
- CSS Animation
- Charts / Graphs
- UI Widgets
- Code Quality Testing
- Code Coverage Testing
- Code Complexity Analysis
- Integration Testing
- Command Line / CLI
- Templating Strategies
- Templating Engines
- Single Page Applications
- XHR Requests (aka AJAX)
- Web/Browser Security
- HTML Semantics
- Browser Developer Tools

Front-End Developers Develop For...

A front-end developer crafts HTML, CSS, and JS that typically runs on the web platform (e.g. a web browser) delivered from one of the following operating systems (aka OSs):

- Android
- Chromium
- iOS
- OS X
- Ubuntu (or some flavor of Linux)
- Windows Phone
- Windows

These operating systems typically run on one or more of the following devices:

- Desktop computer
- Laptop / netbook computer
- Mobile phone
- Tablet
- TV
- Watch
- Things (i.e., anything you can imagine, car, refrigerator, lights, thermostat, etc.)

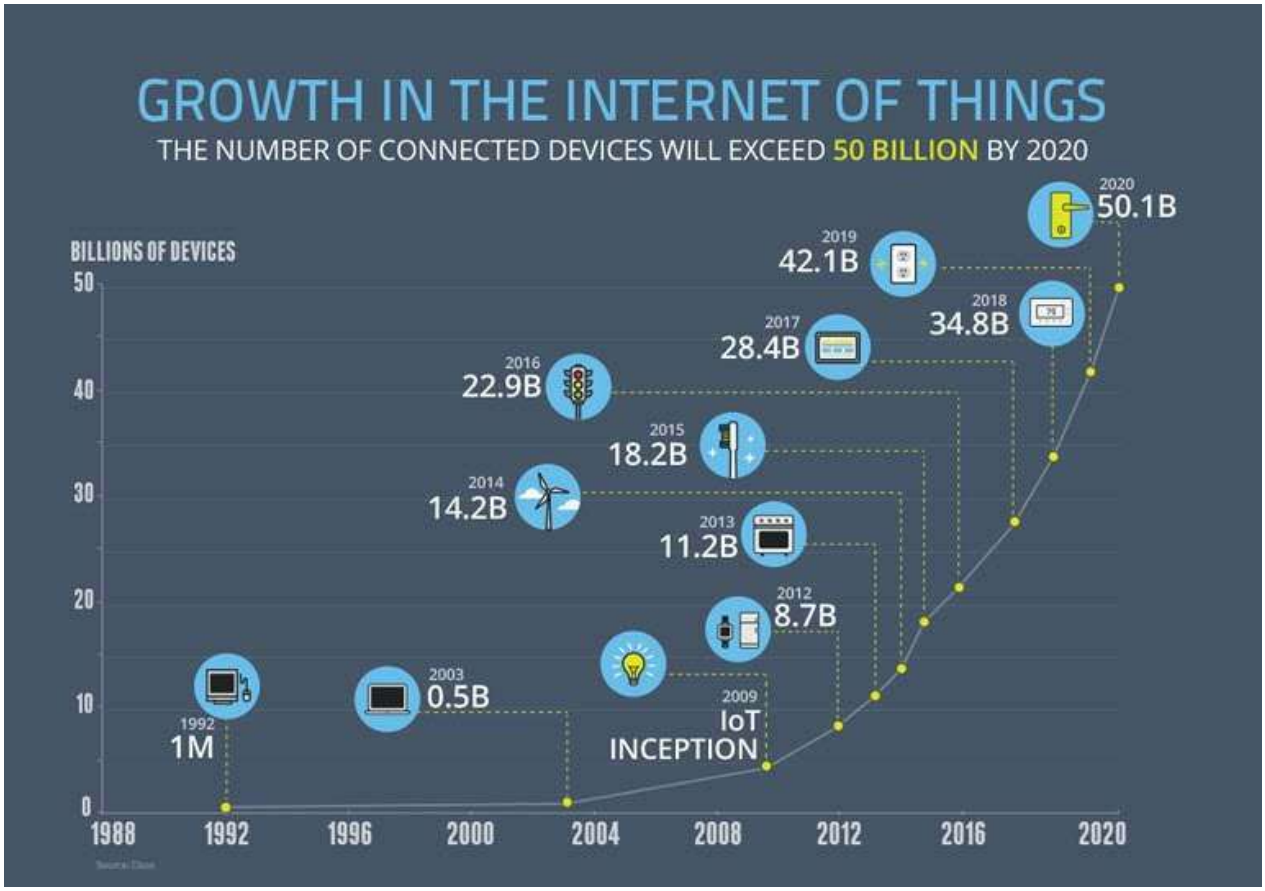


Image source: https://www.enterpriseirregulars.com/104084/roundup-internet-things-forecasts-market-estimates-2015/

Generally speaking, front-end technologies can run on the aforementioned operating systems and devices using the following run time web platform scenarios:

- A web browser (examples: Chrome, IE, Safari, Firefox).
- A headless browser (examples: phantomJS).
- A WebView/browser tab (think iframe) embedded within a native application as a runtime with bridge to native APIs. WebView applications typically contain a UI constructed from web technologies. (i.e., HTML, CSS, and JS). (examples: Apache Cordova, NW.js, Electron)
- A native application built from web tech that is interpreted at runtime with a bridge to native APIs. The UI will make use of native UI parts (e.g., iOS native controls) not web technologies. (examples: NativeScript, React Native)

Front-End on a Team

A front-end developer is typically only one player on a team that designs and develops web sites, web applications, or native applications running from web technologies.

A bare bones development team for building ***professional*** web sites or software for the web platform will typically, minimally, contain the following roles.

- Visual Designer (i.e., fonts, colors, spacing, emotion, visuals concepts & themes)
- UI/Interaction Designer/Information Architect (i.e., wireframes, specifying all user interactions and UI functionality, structuring information)
- Front-End Developer (i.e., writes code that runs in client/on device)
- Back-End Developer (i.e., writes code that runs on server)

The roles are ordered according to overlapping skills. A front-end developer will typically have a good handle on UI/Interaction design as well as back-end development. It is not uncommon for team members to fill more than one role by taking on the responsibilities of an over-lapping role.

It is assumed that the team mentioned above is being directed by a project lead or some kind of product owner (i.e., stakeholder, project manager, project lead, etc.)

A larger web team might include the following roles not shown above:

- SEO Strategists
- DevOps Engineers
- Code Quality Engineers
- Performance Engineers
- API Developers
- Database Administrators
- QA Engineers / Testers

NOTES:

A small trend seems to be occurring where a, "full-stack developer" takes on the responsibilities of both a front-end and back-end developer.

Generalist/Full-Stack Myth

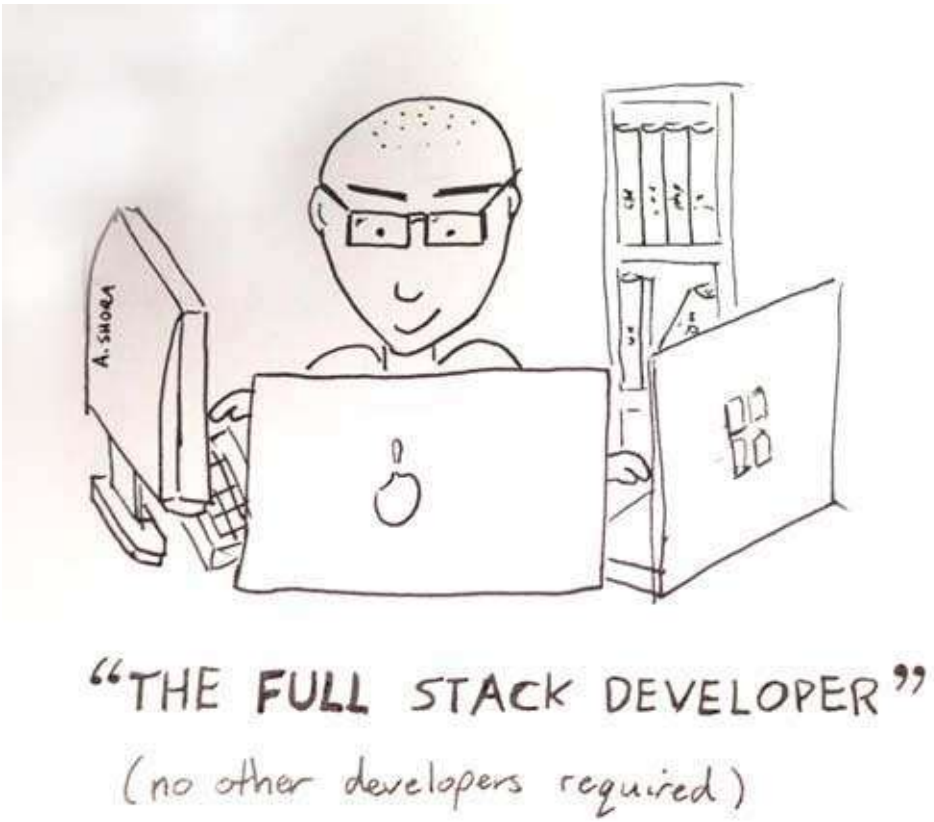


Image source: http://andyshora.com/full-stack-developers.html

The roles required to design and develop a web solution require a deep skill set and vast experience in the area of visual design, UI/interaction design, front-end development, and back-end development. Any person who can fill one or more of these 4 roles at a professional level is an extremely rare commodity.

Pragmatically, you should seek to be, or seek to hire, an expert in one of these roles (i.e. Visual Design, Interaction Design/IA, Front-end Dev, Back-end Dev). Those who claim to operate at an expert level at one or more of these roles are exceptionally rare and more than likely mythical.

However, given that JavaScript has infiltrated all layers of a technology stack (e.g. React, node.js, express, couchDB, gulp.js etc...) finding a full-stack JS developer who can code the front-end and back-end is becoming less mythical. Typically, these full stack developers only deal with JavaScript. A developer who can code the front-end, back-end, API, and database isn't as absurd as it once was (excluding visual design, interaction design, and CSS). Still mythical in my opinion, but not as uncommon as it once was. Thus, I wouldn't recommend a

developer set out to become a "full stack" developer. In rare situations it can work. But, as a general concept for building a career as a Front-end Developer, I'd focus on front-end technologies.

NOTES:

The term "Full-Stack" developer has come to take on several meanings. So many, that not one meaning is clear when the term is used. Just consider the results from the two surveys shown below. These results would lead one to believe that the majority of developers are full-stack developers. But, in my almost 20 years of experience, this is anything but the case.

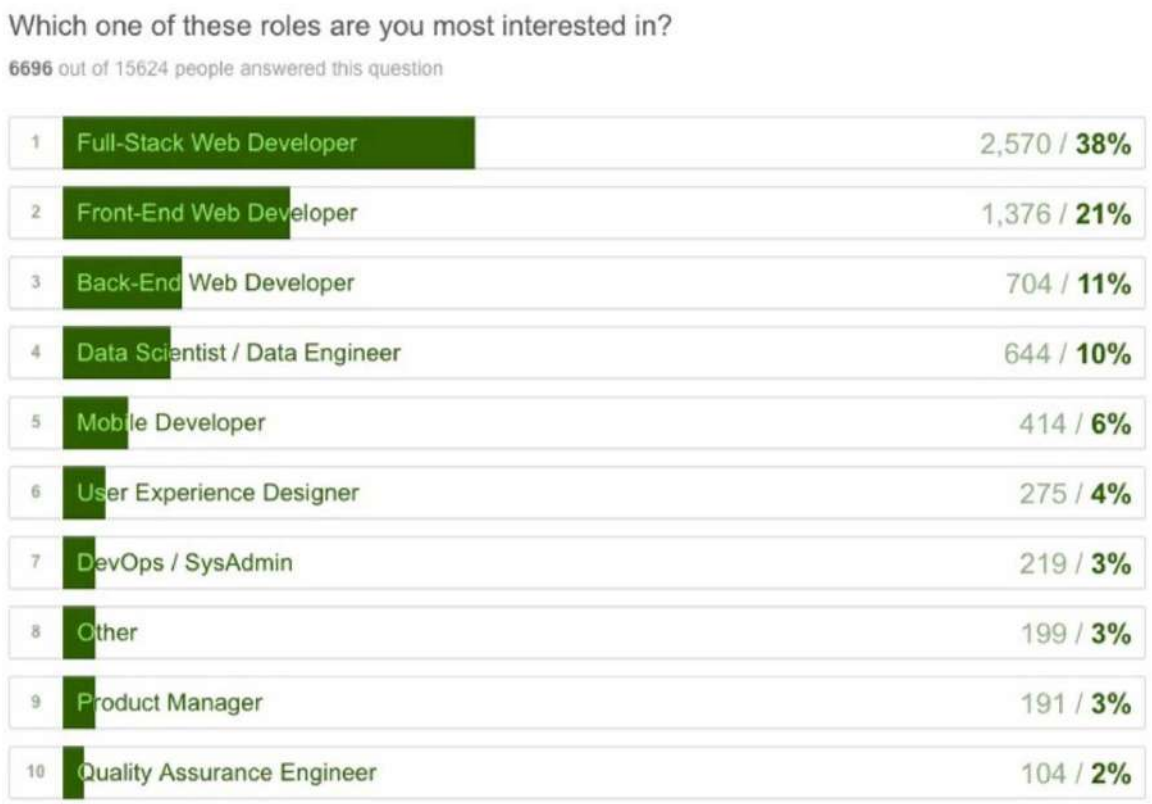


Image source: https://medium.freecodecamp.com/we-asked-15-000-people-who-they-are-and-how-theyre-learning-to-code-4104e29b2781#.ngcpn8nlz

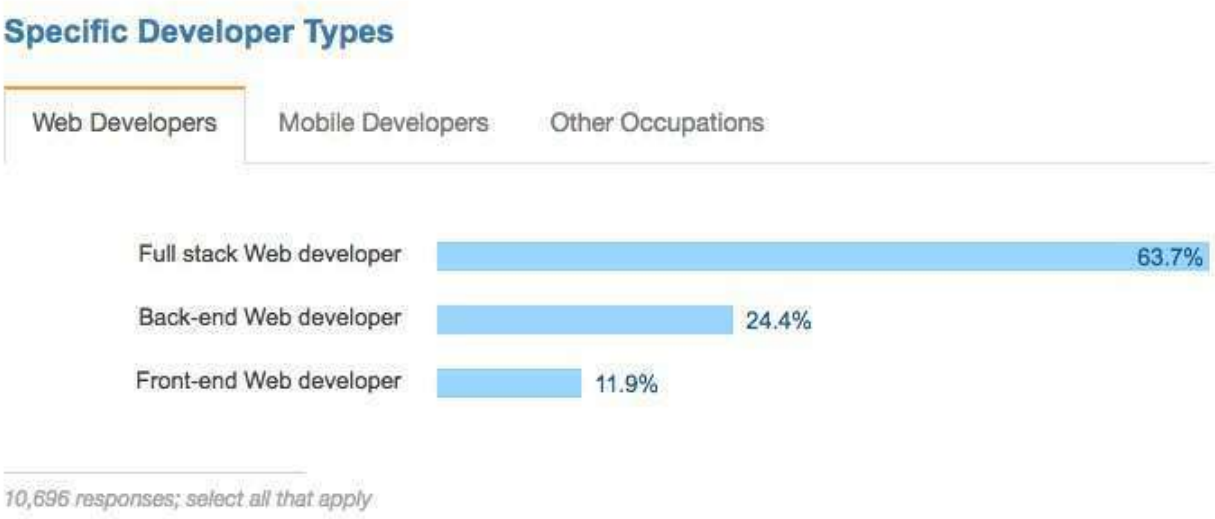


Image source: https://insights.stackoverflow.com/survey/2017#developer-profile-specific-developer-types

Front-End Interviews

Questions you may get asked:

- 10 Interview Questions Every JavaScript Developer Should Know
- Front-End Job Interview Questions
- Front End Web Development Quiz
- Interview Questions for Front-End-Developer
- JavaScript Web Quiz
- The Best Frontend JavaScript Interview Questions (written by a Frontend Engineer))
- Front End Interview Handbook

Questions you ask:

- An open source list of developer questions to ask prospective employers

Preparing:

- Preparing for a Front-End Web Development Interview in 2017
- Interview Cake [$]
- Cracking the front-end interview
- Front End Interview Handbook

Front-End Job Boards

A plethora of technical job listing outlets exist. The narrowed list below are currently the most relevant resources for finding a specific front-end position/career.

- angularjobs.com
- authenticjobs.com
- careers.stackoverflow.com
- css-tricks.com/jobs
- codepen.io/jobs/
- frontenddeveloperjob.com
- glassdoor.com
- jobs.emberjs.com
- jobs.github.com
- weworkremotely.com

NOTES:

Looking for a remote front-end Job, check out these Remote-friendly companies

Front-End Salaries

The national average in the U.S for a mid-level front-end developer is somewhere between $75k and 100k.

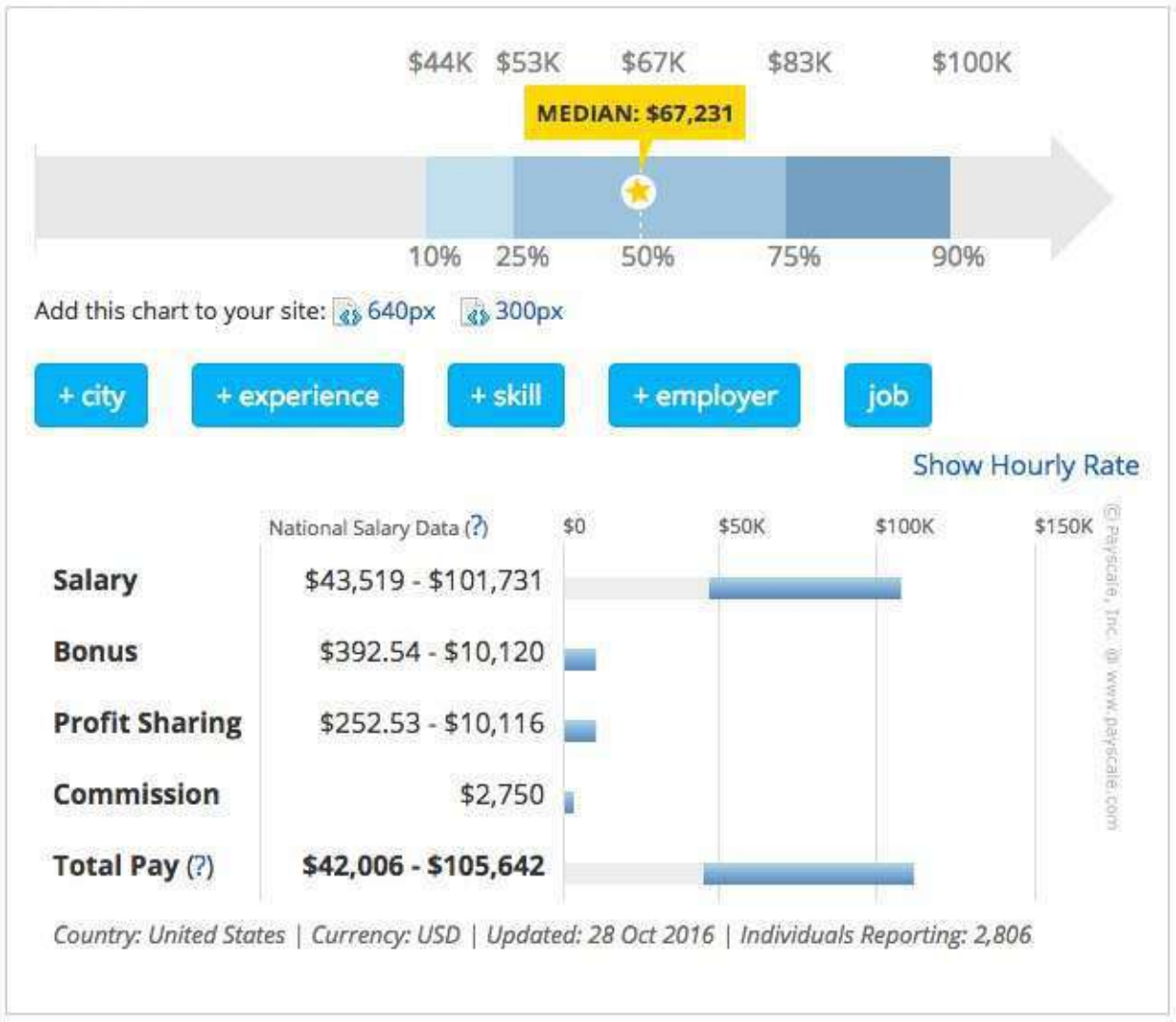


Image source: https://www.glassdoor.com/Salaries/front-end-developer-salary-SRCH_KO0,19.htm

Image source: https://www.indeed.com/salaries/Front-End-Developer-Salaries

Of course when you first start expect to enter the field at around 43k depending upon location and portfolio.

NOTES:

A lead/senior front-end developer/engineer can potentially live wherever they want (i.e., work remotely) and make over $150k a year (visit angel.co, sign-up, review front-end jobs over $150k or examine the salary ranges on Stack Overflow Jobs).

How Front-End Developers Are Made

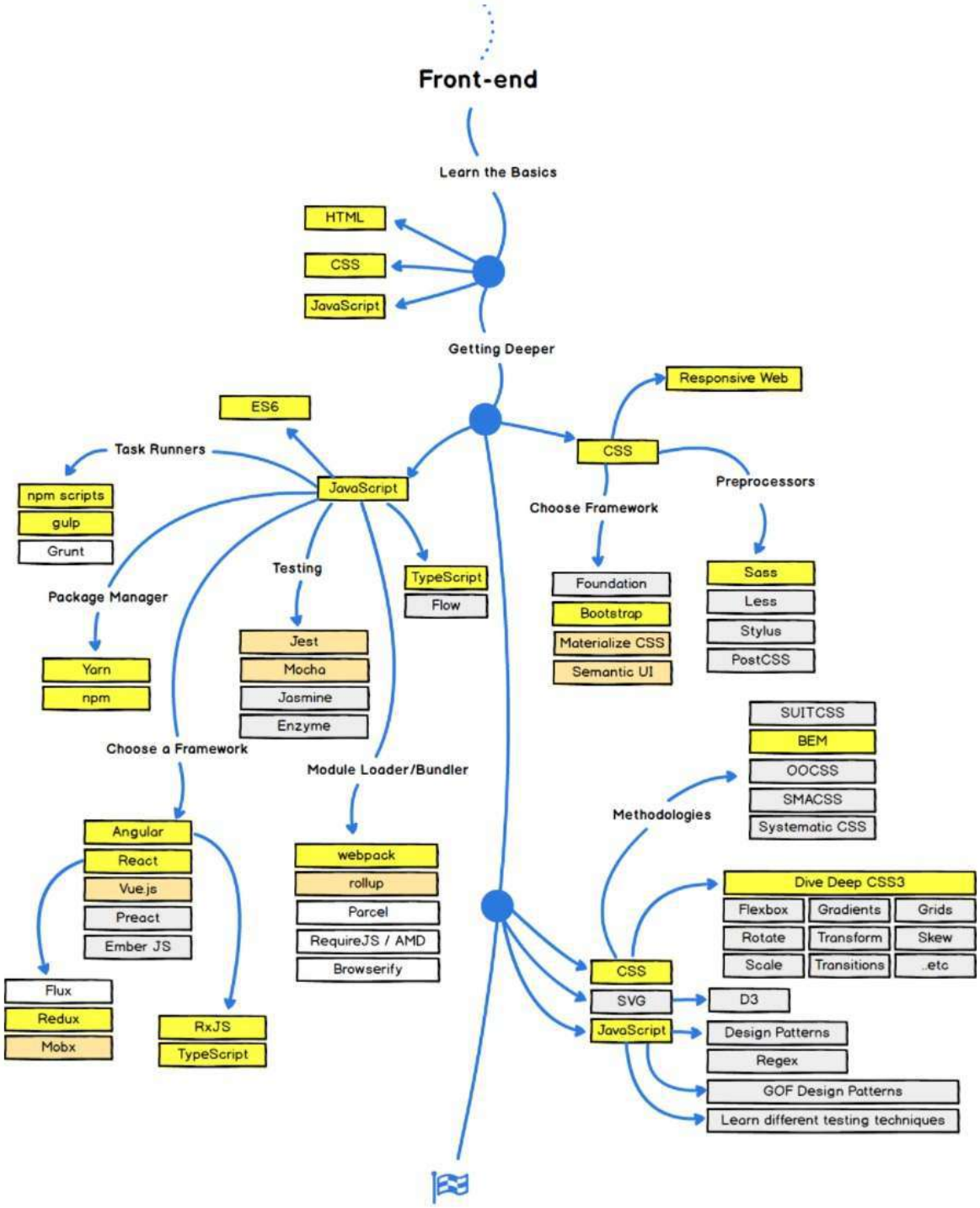


Image source: https://github.com/kamranahmedse/developer-roadmap

How exactly does one become a front-end developer? Well, it's complicated. Still today you can't go to college and expect to graduate with a degree in front-end engineering. And, I rarely hear of or meet front-end developers who suffered through what is likely a deprecated

computer science degree or graphic design degree to end up writing HTML, CSS, and JavaScript professionally. From my perspective, most of the people working on the front-end today generally seem to be self taught from the ground up or cross over into the front-end from design or traditional computer science fields.

If you were to set out today to become a front-end developer I would loosely strive to follow the process outlined below (Part two, "Learning Front-End Dev", dives into more details on learning resources).

1. Learn, roughly, how the web works. Make sure you know the "what" and "where" of Domains, DNS, URLs, HTTP, networks, browsers, servers/hosting, JSON, data APIs, HTML, CSS, DOM, and JavaScript. Don't dive deep on anything, just understand the parts and loosely how they fit together. Focus on the high level outlines for front-end architectures. Start with simple web pages and briefly study front-end applications (aka SPAs)
2. Learn HTML
3. Learn CSS
4. Learn JavaScript
5. Learn DOM
6. Learn JSON and data APIs
7. Learn the fundamentals of user interface design (i.e. UI patterns, interaction design, user experience design, and usability).
8. Learn CLI/command line
9. Learn the practice of software engineering (i.e., Application design/architecture, templates, Git, testing, monitoring, automating, code quality, development methodologies).
10. Get opinionated and customize your tool box with whatever makes sense to your brain (e.g. Webpack, React, and Mobx).
11. Learn Node.js

A short word of advice on learning. Learn the actual underlying technologies, before learning abstractions. Don't learn jQuery, learn the DOM. Don't learn SASS, learn CSS. Don't learn HAML, learn HTML. Don't learn CoffeeScript, learn JavaScript. Don't learn Handlebars, learn JavaScript ES6 templates. Don't just use Bootstrap, learn UI patterns.

Lately a lot of non-accredited, expensive, front-end code schools/bootcamps have emerged. These avenues of becoming a front-end developer are typically teacher directed courses, that follow a more traditional style of learning, from an official instructor (i.e., syllabus, test, quizzes, projects, team projects, grades, etc.). Keep in mind, if you are considering an expensive training program, this is the web! Everything you need to learn is on the web for the taking, costing little to nothing. However, if you need someone to tell you how to take and learn what is actually free, and hold you accountable for learning it, you might consider an

organized course. Otherwise, I am not aware of any other profession that is practically free for the taking with an internet connection, a hundred dollars a month for screencasting memberships, and a burning desire for knowledge.

For example if you want to get going today, consuming one or more of the following self-directed resources below can work:

- 2016/2017 MUST-KNOW WEB DEVELOPMENT TECH [watch]
- A Beginner's Guide to Front-End Programming [read & watch][free to $]
- Become a Front-End Web Developer [watch][$]
- Front-End Curriculum [read]
- freeCodeCamp [interact]
- So, You Want to be a Front-End Engineer [watch]
- Front End Web Development Career Kickstart [watch][$]
- Front End Web Development: Get Started [watch][$]
- Front-End Web Development Quick Start With HTML5, CSS, and JavaScript [watch][$]
- Introduction to Web Development [watch][$]
- Foundations of Front-End Web Development [watch][$]
- Lean Front-End Engineering [watch][$]
- Learn Front End Web Development [watch][$]
- Front-End Dev Mastery [watch][$]
- Front-End Web Developer Nanodegree [watch][$]
- Full Stack for Front End Engineers [$]

If you are not a self motivated individual and need a more structured approach you should consider a directed learning path.

When getting your start, you should fear most things that conceal complexity. Abstractions in the wrong hands can give the appearance of advanced skills, while all the time hiding the fact that a developer has an inferior understanding of the basics or underlying concepts.

The remaining parts of this book will point the reader to potential resources that could be used to learn front-end development and the tools used when practicing front-end development. It is assumed that on this journey you are not only learning, but also doing as you learn and investigate tools. Some suggest only doing to learn. While others suggest only learning about doing. I suggest you find a mix of both that matches how your brain works and do that. But, for sure, it is a mix! So, don't just read about it, do it. Learn, do. Learn, do. Repeat indefinitely because things change fast. This is why learning the fundamentals, and not abstractions, are so important.

Part II: Learning

Part two identifies self-directed (i.e., at your own pace when you want) and directed (i.e., formal class room specific times and dates) resources for learning to become a front-end developer.

Note that just because a learning resource is listed, or a category of learning is documented, I am not suggesting that a front-end developer learn everything. That would be absurd. Choose your own slice of expertise within the profession. I'm providing the possibilities of what could be mastered in the field.

Self Directed Learning

This section focuses on free and paid resources (video training, books, etc.) that an individual can use to direct their own learning process and career as a front-end developer.

The learning resources mentioned (articles, books, videos, screencasts etc..) will include both free and paid material. Paid material will be indicated with [$].

The author believes that anyone with the right determination and dedication can teach themselves how to be a front-end developer. All that is required is a computer connected to the web and some cash for books and online video training.

Below are a few video learning outlets (tech focused) I generally recommend pulling content from:

- codecademy.com
- codeschool.com
- egghead.io
- eventedmind.com
- Frontend Masters
- Freecodecamp
- Khan Academy
- laracasts.com
- lynda.com [careful, quality varies]
- mijingo.com
- pluralsight.com [careful, quality varies]
- Treehouse
- tutsplus.com
- Udacity [careful, quality varies]

Learn Internet/Web

> The Internet is a global system of interconnected computer networks that use the Internet protocol suite (TCP/IP) to link several billion devices worldwide. It is a network of networks that consists of millions of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies. The Internet carries an extensive range of information resources and services, such as the inter-linked hypertext documents and applications of the World Wide Web (WWW), electronic mail, telephony, and peer-to-peer networks for file sharing.
>
> — *Wikipedia*

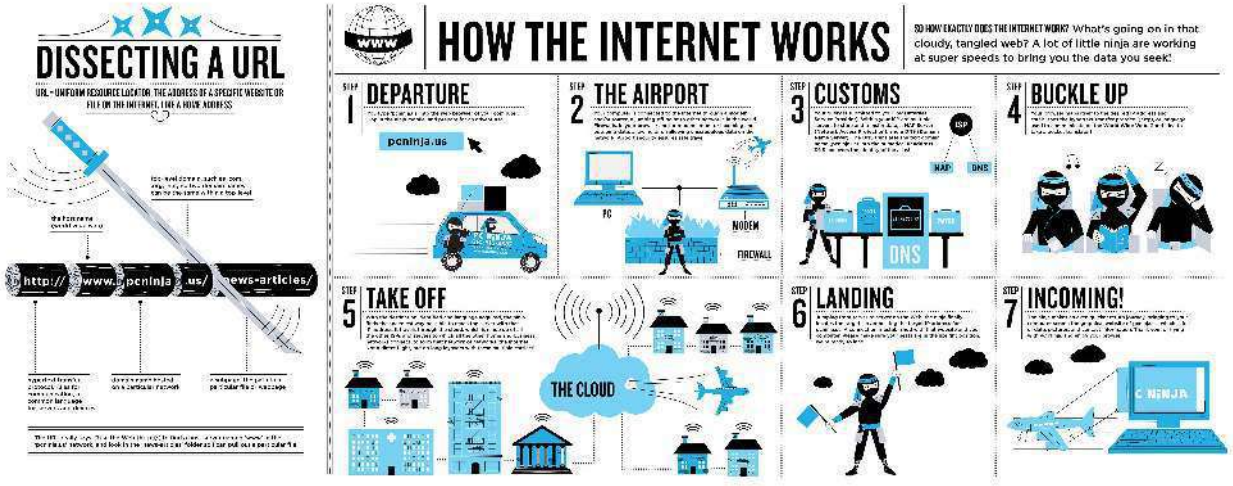


Image source: https://www.helloitsliam.com/2014/12/20/how-the-internet-works-infographic/

- What is the Internet? [watch]
- How the Web works [read]
- How does the Internet work? https://developer.mozilla.org/en-US/docs/Learn/Common_questions/How_does_the_Internet_work and http://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm [read]
- How the Internet Works [watch]
- How the Internet Works in 5 Minutes [watch]
- How the Web Works [watch]
- What Is the Internet? Or, "You Say Tomato, I Say TCP/IP" [read]
- Don’t Fear the Internet

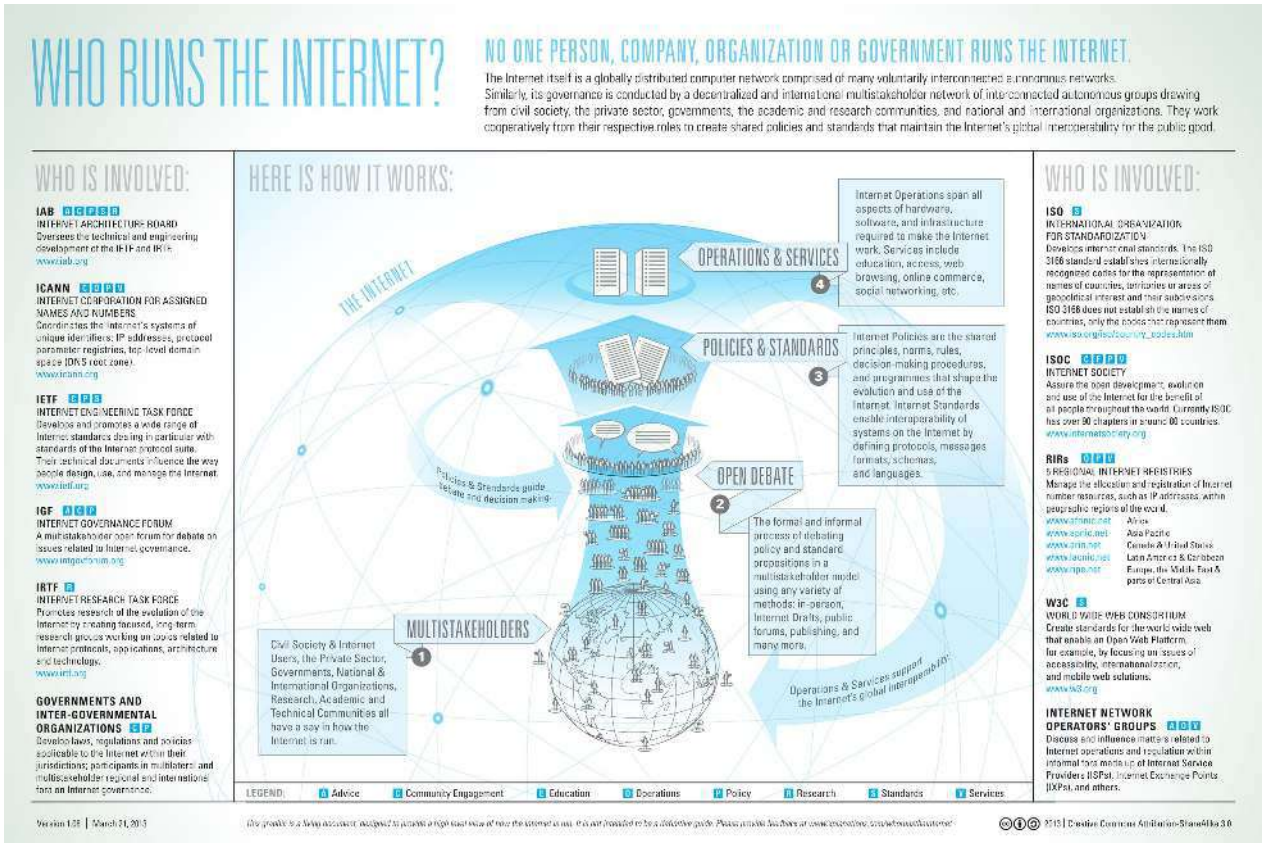


Image source: http://www.bitrebels.com/technology/find-out-who-runs-the-internet-chart/

Learn Web Browsers

> A web browser (commonly referred to as a browser) is a software application for retrieving, presenting, and traversing information resources on the World Wide Web. An information resource is identified by a Uniform Resource Identifier (URI/URL) and may be a web page, image, video or other piece of content. Hyperlinks present in resources enable users easily to navigate their browsers to related resources. Although browsers are primarily intended to use the World Wide Web, they can also be used to access information provided by web servers in private networks or files in file systems.
>
> — *Wikipedia*

The most commonly used browsers (on any device) are:

1. Chrome (engine: Blink + V8)
2. Firefox (engine: Gecko + SpiderMonkey)
3. Internet Explorer (engine: Trident + Chakra)
4. Safari (engine: Webkit + SquirrelFish)

Image source: http://gs.statcounter.com/browser-market-share

Evolution of Browsers & Web Technologies (i.e., APIs)

- evolutionoftheweb.com [read]
- Timeline of web browsers [read]

The Most Commonly Used Headless Browser Are:

- Headless Chromium (engine: Blink + V8)
- PhantomJS (engine: Webkit + SquirrelFish)
- SlimerJS (engine: Gecko + SpiderMonkey)
- TrifleJS (engine: Trident + Chakra)

How Browsers Work

- 20 Things I Learned About Browsers and the Web [read]
- Fast CSS: How Browsers Lay Out Web Pages [read]
- How Browsers Work: Behind the scenes of modern web browsers [read]
- Quantum Up Close: What is a browser engine?
- So How Does the Browser Actually Render a Website [watch]
- What forces layout / reflow [read]
- What Every Frontend Developer Should Know About Webpage Rendering [read]

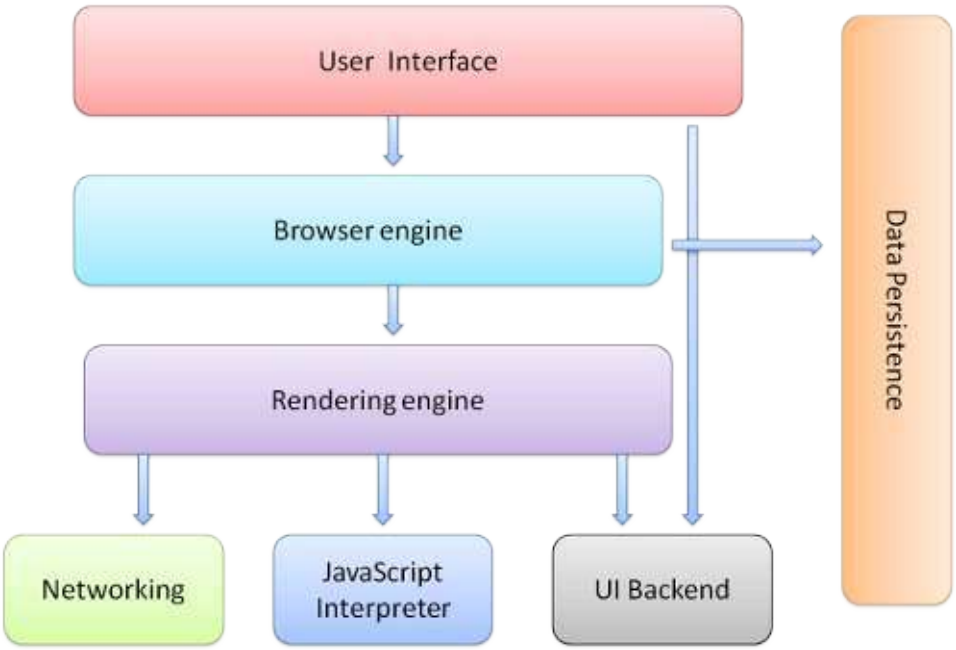


Image source: http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

Optimizing for Browsers:

- Browser Rendering Optimization [watch]
- Website Performance Optimization [watch]

Comparing Browsers

- Comparison of Web Browsers [read]

Browser Hacks

- browserhacks.com [read]

Developing for Browsers

In the past, front-end developers spent a lot of time making code work in several different browsers. This was once a bigger issue than it is today. Today, abstractions (e.g., jQuery, React, Post-CSS, Babel etc...) combined with modern browsers make browser development fairly easy. The new challenge is not which browser the user will use, but on which device they will run the browser.

Evergreen Browsers

The latest versions of most modern browsers are considered evergreen browsers. That is, in theory they are suppose to automatically update themselves silently without prompting the user. This move towards self updating browsers has been in reaction to the slow process of eliminating older browsers that do not auto-update.

Picking a Browser [1]

As of today, most front-end developers use Chrome and "Chrome Dev Tools" to develop front-end code. However, the most used modern browsers all offer a flavor of developer tools. Picking one to use for development is a subjective choice. The more important issue is knowing which browsers, on which devices, you have to support and then testing appropriately.

ADVICE:

[1] I suggest using Chrome because the developer tools are consistently improving and at this time contain the most robust features.

Learn Domain Name System (aka DNS)

> The Domain Name System (DNS) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates domain names, which can be easily memorized by humans, to the numerical IP addresses needed for the purpose of computer services and devices worldwide. The Domain Name System is an essential component of the functionality of most Internet services because it is the Internet's primary directory service.
>
> — *Wikipedia*

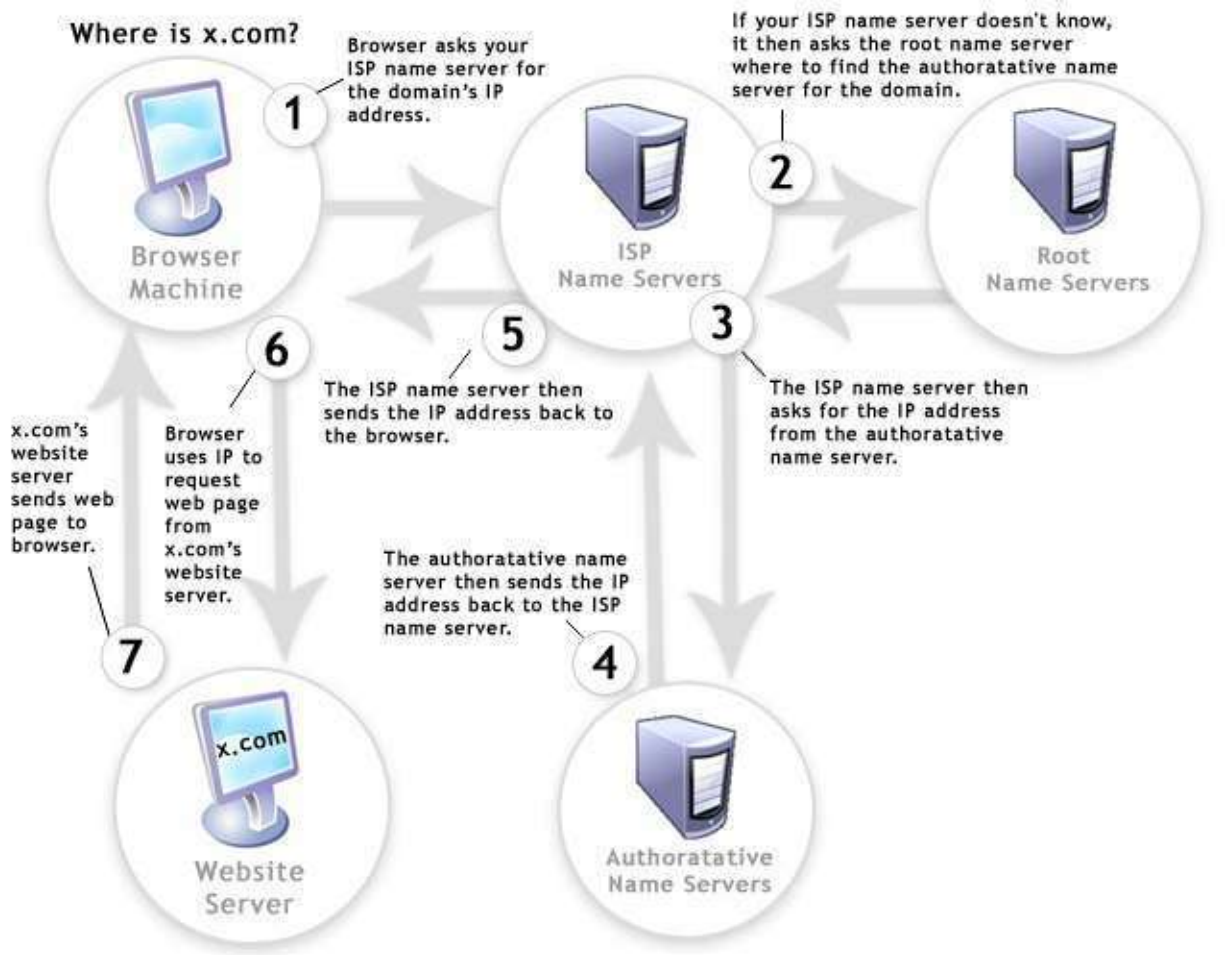


Image source: http://www.digital-digest.com/blog/DVDGuy/wp-content/uploads/2011/11/how_dns_works.jpg

- An Introduction to DNS Terminology, Components, and Concepts [read]
- DNS Explained [watch]
- How DNS Works [read]

- The Internet: IP Addresses and DNS [watch]
- What is a domain name? [read]

Learn HTTP/Networks (Including CORS & WebSockets)

> **HTTP** - The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.
>
> — *Wikipedia*
>
> **CORS** - Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources (e.g., fonts) on a web page to be requested from another domain outside the domain from which the resource originated.
>
> — *Wikipedia*
>
> **WebSockets** - WebSocket is a protocol providing full-duplex communication channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011, and the WebSocket API in Web IDL is being standardized by the W3C.
>
> — *Wikipedia*

HTTP Specifications

- HTTP/2
- Hypertext Transfer Protocol -- HTTP/1.1

HTTP Docs

- MDN HTTP [read]

HTTP Videos/Articles/Tutorials

- High Performance Browser Networking: What Every Web Developer Should Know About Networking and Web Performance [read]
- MDN: An overview of HTTP [read]
- HTTP: The Definitive Guide (Definitive Guides) [read][$]
- HTTP/2 Frequently Asked Questions [read]
- HTTP Fundamentals [watch][$]
- HTTP/2 Fundamentals [watch][$]
- HTTP: The Protocol Every Web Developer Must Know - Part 1 [read]
- HTTP: The Protocol Every Web Developer Must Know - Part 2 [read]

- HTTP Succinctly [read]

HTTP Status Codes

- HTTP Status Codes
- HTTP Status Codes in 60 Seconds [watch]

CORS Specifications

- Cross-Origin Resource Sharing

CORS

- CORS in Action [read][$]
- HTTP Access Control (CORS) [read]

WebSockets

- Connect the Web With WebSockets [watch]
- WebSocket: Lightweight Client-Server Communications [read][$]
- The WebSocket Protocol [read]

Learn Web Hosting

> A web hosting service is a type of Internet hosting service that allows individuals and organizations to make their website accessible via the World Wide Web. Web hosts are companies that provide space on a server owned or leased for use by clients, as well as providing Internet connectivity, typically in a data center. Web hosts can also provide data center space and connectivity to the Internet for other servers located in their data center, called colocation, also known as Housing in Latin America or France.
>
> — *Wikipedia*

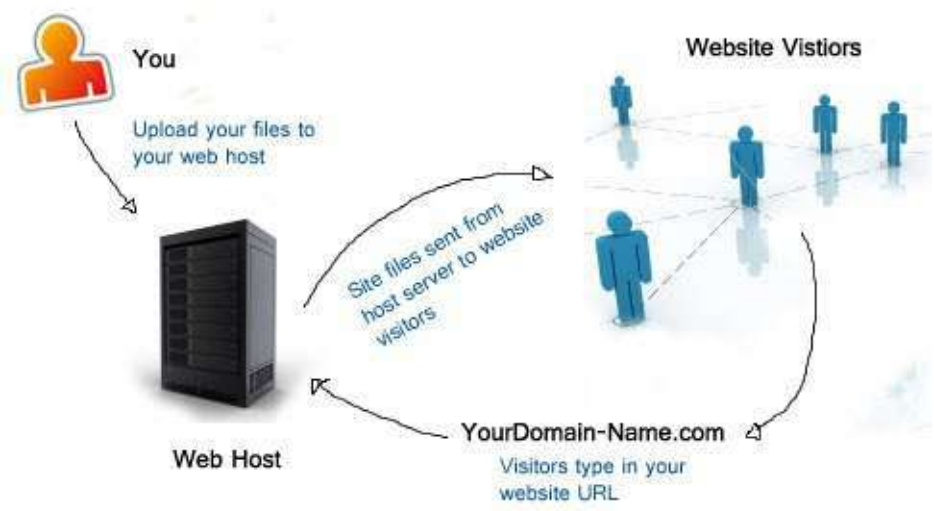
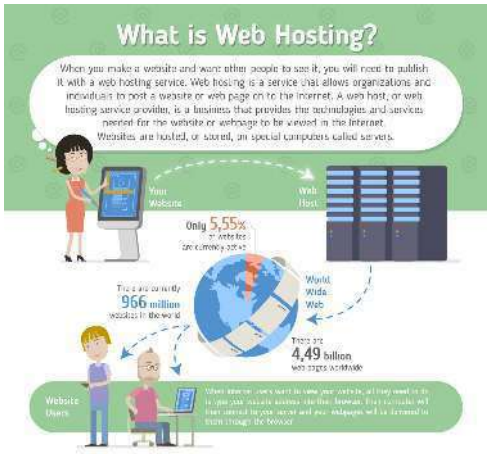
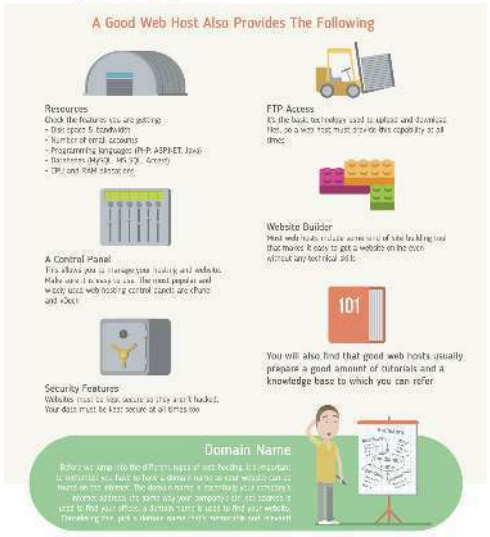


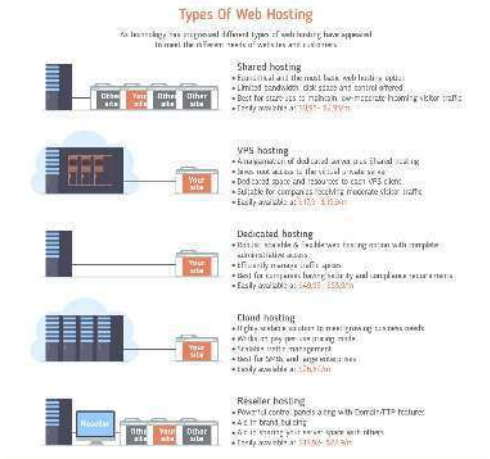
Image source: http://www.alphaelite.com.sg/sitev2/images/stories/webhostdemo.jpg

General Learning:

- Web Hosting Services Explained [read]
- Web Hosting 101: Get Your Website Live on the Web in No Time [video]


What is Web Hosting?
When you make a website and want other people to see it, you will need to publish it with a web hosting service. Web hosting is a service that allows organizations and individuals to post a website or web page on to the Internet. A web host, or web hosting service provider, is a business that provides the technologies and services needed for the website or webpage to be viewed in the Internet.
Websites are hosted, or stored, on special computers called servers.
Your Website
Web Host
Only 5,55%
There are currently nearly 966 million websites in the world
World Wide Web
There are 4,49 billion
Website Users


A Good Web Host Also Provides The Following
Resources
FTP Access
Website Builder
A Control Panel
You will also find that good web hosts usually prepare a good amount of tutorials and a knowledge base to which you can refer
Security Features
Domain Name


Types Of Web Hosting
Shared hosting
VPS hosting
Dedicated hosting
Cloud hosting
Reseller hosting


BEFORE SIGNING UP WITH A WEB HOST, YOU SHOULD SHOP AROUND AND GET A FEELING FOR WHAT IS AVAILABLE.
You should use a variety of criteria before making your choice
Experience
Money Back Guarantee
Reputation
Uptime
Location
Extras
Support
Backups
Cost
Terms & Conditions
Firstsiteguide

Image source: https://firstsiteguide.com/wp-content/uploads/2016/06/what-is-web-hosting-infographic.jpg

Learn General Front-End Development

General Learning:

- Become a Front-End Web Developer [watch][$]
- Being a web developer [read]
- Foundations of Front-End Web Development [watch]
- freeCodeCamp [interact]
- Front-End Curriculum [read]
- Front-End Dev Mastery [watch][$]
- Front-End Web Developer Nanodegree [watch][$]
- Front End Web Development Career Kickstart [watch][$]
- Front End Web Development: Get Started [watch][$]
- Front-End Web Development Quick Start With HTML5, CSS, and JavaScript [watch][$]
- Front-End Web Development: The Big Nerd Ranch Guide [read][$]
- Frontend Guidelines [read]
- Introduction to Web Development [watch][$]
- Isobar Front-End Code Standards [read]
- Lean Front-End Engineering [watch][$]
- Learn Front End Web Development [watch][$]
- Planning a Front-End JS Application [watch]
- So, You Want to Be a Front-End Engineer [watch]

General Front-End Newsletters, News Outlets, & Podcasts:

- The Big Web Show
- Front-End Dev Weekly
- Front End Happy Hour
- frontendfront.com
- FrontEnd Focus
- Front End Newsletter
- Mobile Web Weekly
- Open Web Platform Daily Digest
- Pony Foo Weekly
- shoptalkshow.com
- The Web Ahead
- The Web Platform Podcast
- webtoolsweekly.com

Learn User Interface/Interaction Design

> **User Interface Design** - User interface design (UI) or user interface engineering is the design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centered design).
>
> — *Wikipedia*
>
> **Interaction Design Pattern** - A design pattern is a formal way of documenting a solution to a common design problem. The idea was introduced by the architect Christopher Alexander for use in urban planning and building architecture, and has been adapted for various other disciplines, including teaching and pedagogy, development organization and process, and software architecture and design.
>
> — *Wikipedia*
>
> **User Experience Design** - User Experience Design (UXD or UED or XD) is the process of enhancing user satisfaction by improving the usability, accessibility, and pleasure provided in the interaction between the user and the product. User experience design encompasses traditional human–computer interaction (HCI) design, and extends it by addressing all aspects of a product or service as perceived by users.
>
> — *Wikipedia*
>
> **Human–Computer Interaction** - Human–computer interaction (HCI) researches the design and use of computer technology, focusing particularly on the interfaces between people (users) and computers. Researchers in the field of HCI both observe the ways in which humans interact with computers and design technologies that lets humans interact with computers in novel ways.
>
> — *Wikipedia*

Minimally I'd suggest reading the following canonical texts on the matter so one can support and potential build usable user interfaces.

- About Face: The Essentials of Interaction Design [read][$]
- Design for Hackers: Reverse Engineering Beauty [read][$]
- Design for Non-Designers [watch]
- Designing Interfaces [read][$]

- Designing Web Interfaces: Principles and Patterns for Rich Interactions [read][$]
- Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability [read][$]

Learn HTML & CSS

> **HTML** - HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.
>
> — *Wikipedia*
>
> **CSS** - Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. Although most often used to change the style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.
>
> — *Wikipedia*

Liken to constructing a house, one might consider HTML the framing and CSS to be the painting & decorating.

General Learning:

- Absolute Centering in CSS [read]
- codecademy.com HTML & CSS [interact]
- CSS Positioning [watch][$]
- Front End Web Development: Get Started [watch][$]
- Front-End Web Development Quick Start With HTML5, CSS, and JavaScript [watch][$]
- HTML and CSS: Design and Build Websites [read][$]
- HTML Document Flow [watch][$]
- HTML Mastery: Semantics, Standards, and Styling [read][$]
- Interneting is Hard [read]
- Intro to HTML/CSS: Making webpages [watch]
- Learn to Code HTML & CSS [read]
- Learn CSS Layout [read]
- MarkSheet [read]
- MDN: HTML [read]
- MDN: CSS [read]
- Semantic HTML: How to Structure Web Pages [watch]

- Solid HTML Form Structure [watch]
- Understanding the CSS Box Model [watch]
- Resilient Web Design [read]

Mastering CSS:

- A Complete Guide to Flexbox [read]
- CSS Diner [interact]
- CSS Selectors from CSS4 till CSS1 [read]
- CSS Secrets: Better Solutions to Everyday Web Design Problems [read][$]
- CSS3 [read]
- CSS3 In-Depth [watch][$]
- What the Flexbox?! A Simple, Free 20 Video Course That Will Help You Master CSS Flexbox [watch]

References/Docs:

- CSS Triggers...a Game of Layout, Paint, and Composite
- cssreference.io
- cssvalues.com
- Default CSS for Chrome Browser
- Head - A list of everything that could go in the of your document
- HTML Attribute Reference
- MDN CSS Reference
- MDN HTML Element Reference

Glossary/Vocabulary:

- CSS Glossary - Programming Reference for CSS Covering Comments, Properties, and Selectors
- CSS Vocabulary
- HTML Glossary Programming Reference for HTML elements

Standards/Specifications:

- All W3C CSS Specifications
- All W3C HTML Spec
- Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification
- CSS Indexes - A listing of every term defined by CSS specs
- The Elements of HTML from the Living Standard
- Global Attributes
- The HTML Syntax from the Living Standard
- HTML 5.2 from W3C

- Selectors Level 3

Architecting CSS:

- Atomic Design [read]
- BEM
- ITCSS
- OOCSS [read]
- SMACSS [read][$]
  - Scalable Modular Architecture for CSS (SMACSS) [watch][$]
- SUIT CSS
- rscss

Authoring/Architecting Conventions:

- CSS code guide [read]
- css-architecture
- cssguidelin.es [read]
- Idiomatic CSS [read]
- MaintainableCSS [read]
- Standards for Developing Flexible, Durable, and Sustainable HTML and CSS [read]

HTML/CSS Newsletters:

- CSS Weekly
- Frontend Focus

NOTES:

Writing CSS in JS was hot this year. Make sure you are aware of the use case. And why some think it is overused.

Learn Search Engine Optimization

> Search engine optimization (SEO) is the process of affecting the visibility of a website or a web page in a search engine's unpaid results — often referred to as "natural," "organic," or "earned" results. In general, the earlier (or higher ranked on the search results page), and more frequently a site appears in the search results list, the more visitors it will receive from the search engine's users. SEO may target different kinds of search, including image search, local search, video search, academic search, news search and industry-specific vertical search engines.
>
> — *Wikipedia*

how-does-seo-work.png

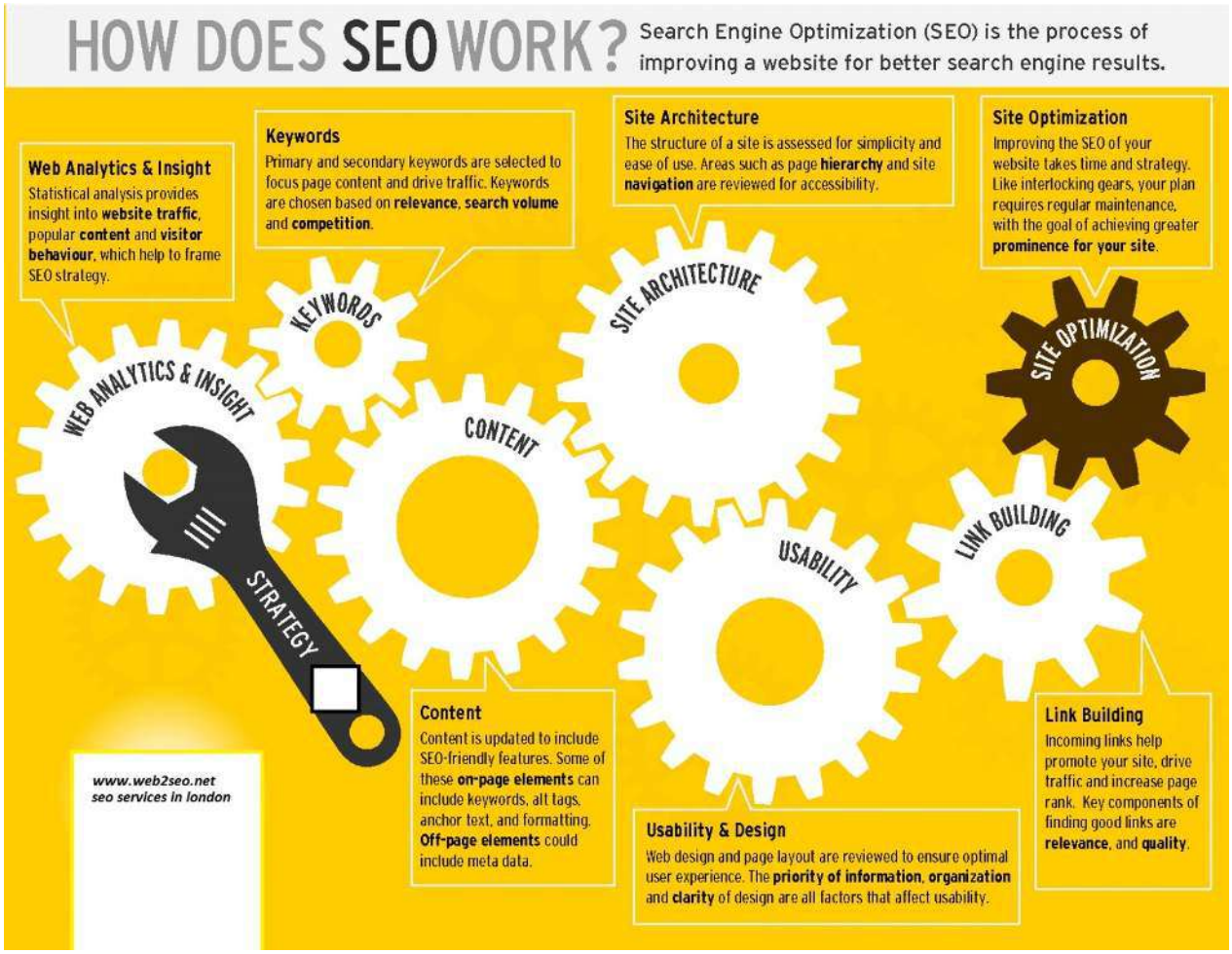


Image source: https://visual.ly/community/infographic/computers/how-does-seo-work

General Learning:

- Google Search Engine Optimization Starter Guide [read]

- Modern SEO [watch][$]
- SEO Fundamentals From David Booth [watch][$]
- SEO Fundamentals From Paul Wilson [watch][$]
- SEO Tutorial For Beginners in 2016 [read]
- SEO for Web Designers [watch][$]

Learn JavaScript

> JavaScript is a high level, dynamic, untyped, and interpreted programming language. It has been standardized in the ECMAScript language specification. Alongside HTML and CSS, it is one of the three essential technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern web browsers without plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage or graphics facilities, relying for these upon the host environment in which it is embedded.
>
> — *Wikipedia*

Getting Started:

- Ten Things A Serious JavaScript Developer Should Learn
- codecademy.com JavaScript [interact]
- MDN: JavaScript [read]
- javascript.info
- JavaScript Enlightenment [read]
- Eloquent JavaScript [read]

General Learning:

- Speaking JavaScript [read]
- You Don't Know JS: Up & Going [read]
- You Don't Know JS: Types & Grammar [read]
- You Don't Know JS: Scope & Closures [read]
- Gentle explanation of 'this' keyword in JavaScript [read]
- You Don't Know JS: this & Object Prototypes [read]
- Modern JavaScript Cheatsheet - Cheatsheet for the JavaScript knowledge you will frequently encounter in modern projects. [read]

Mastering:

- Setting up ES6 [read]
- ES6 FOR EVERYONE! [watch][$]
- Exploring ES6 [read]
- You Don't Know JS: ES6 & Beyond [read]
- Understanding ECMAScript 6: The Definitive Guide for JavaScript Developers [read][$]

- ES6: The Right Parts [watch][$]
- Exploring ES2016 and ES2017 [read]
- JavaScript Regular Expression Enlightenment [read]
- Using Regular Expressions [watch][$]
- You Don't Know JS: Async & Performance [read]
- JavaScript with Promises [read][$]
- Test-Driven JavaScript Development [read][$]
- JS MythBusters [read]
- Robust JavaScript

Functional JavaScript:

- Functional Programming Jargon
- funfunfunction: Functional programming in JavaScript [watch]
- Functional-Light-JS [read]
- Functional Programming in JavaScript: How to improve your JavaScript programs using functional techniques [read]
- Mostly adequate guide to FP (in javascript) [read]
- Professor Frisby Introduces Composable Functional JavaScript [watch]
- JavaScript Allongé [read][$]
- Hardcore Functional Programming in JavaScript [watch][$]
- Functional-Lite JavaScript [watch][$]

References/Docs:

- MDN JavaScript Reference
- MSDN JavaScrip Reference

Glossary/Encyclopedia/Jargon:

- The JavaScript Encyclopedia
- JavaScript Glossary
- Simplified JavaScript Jargon

Standards/Specifications:

- How to Read the ECMAScript Specification
- ECMAScript® 2015 Language Specification
- ECMAScript® 2016 Language Specification
- ECMAScript® 2017 Language Specification
- ECMAScript® 2018 Language Specification
- Status, Process, and Documents for ECMA262

Style:

- Airbnb JavaScript Style Guide
- JavaScript Standard Style
- JavaScript Semi-Standard Style

JavaScript Newsletters, News, & Podcasts:

- Echo JS
- ECMAScript Daily
- ES.next News
- JavaScript Air
- JavaScript Jabber
- JavaScript Kicks
- JavaScript Live
- JavaScript Weekly
- JavaScript.com

Deprecated JS Learning Resources:

- Crockford on JavaScript - Volume 1: The Early Years [watch]
- Crockford on JavaScript - Chapter 2: And Then There Was JavaScript [watch]
- Crockford on JavaScript - Act III: Function the Ultimate [watch]
- Crockford on JavaScript - Episode IV: The Metamorphosis of Ajax [watch]
- Crockford on JavaScript - Part 5: The End of All Things [watch]
- Crockford on JavaScript - Scene 6: Loopage [watch]
- JavaScript Patterns [read][$]
- The Principles of Object-Oriented JavaScript [read][$]
- JavaScript Modules [read]
- Functional JavaScript: Introducing Functional Programming with Underscore.js [read][$]
- The Good Parts of JavaScript and the Web [watch][$]
- High Performance JavaScript (Build Faster Web Application Interfaces) [read][$]
- Advanced JavaScript [watch][$]

JS Explorers:

- JavaScript Array Explorer
- JavaScript Object Explorer

Learn Web Animation

General Learning:

- Advanced SVG Animation [$][watch]
- Adventures in Web Animations [$][watch]
- Animating With Snap.svg [$][watch]
- Animation in CSS3 and HTML5 [$][watch]
- Create Animations in CSS [read & watch]
- CSS Animation in the Real World [$][watch]
- Foundation HTML5 Animation with JavaScript [$][read]
- Learn to Create Animations in JavaScript [read & watch]
- Motion Design with CSS [$][watch]
- State of the Animation 2015 [watch]
- Web Animation using JavaScript: Develop & Design (Develop and Design) [$][read]

Standards/Specifications:

- Web Animations

Learn DOM, BOM, & jQuery

> **DOM** - The Document Object Model (DOM) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML, and XML documents. The nodes of every document are organized in a tree structure, called the DOM tree. Objects in the DOM tree may be addressed and manipulated by using methods on the objects. The public interface of a DOM is specified in its application programming interface (API).
>
> — *Wikipedia*
>
> **BOM** - The Browser Object Model (BOM) is a browser-specific convention referring to all the objects exposed by the web browser. Unlike the Document Object Model, there is no standard for implementation and no strict definition, so browser vendors are free to implement the BOM in any way they wish.
>
> — *Wikipedia*
>
> **jQuery** - jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. jQuery is the most popular JavaScript library in use today, with installation on 65% of the top 10 million highest-trafficked sites on the Web. jQuery is free, open-source software licensed under the MIT License.
>
> — *Wikipedia*

The ideal path, but certainly the most difficult, would be to first learn JavaScript, then the DOM, then jQuery. However, do what makes sense to your brain. Most front-end developers learn about JavaScript and then DOM by way of first learning jQuery. Whatever path you take, just make sure JavaScript, the DOM, and jQuery don't become a black box.

General Learning:

- Codecademy.com jQuery [watch]
- The Document Object Model [read]
- HTML/JS: Making Webpages Interactive [watch]
- HTML/JS: Making Webpages Interactive with jQuery [watch]
- jQuery Enlightenment [read]
- What is the DOM? [read]

Mastering:

- AdvancED DOM Scripting: Dynamic Web Design Techniques [read][$]

- Advanced JS Fundamentals to jQuery & Pure DOM Scripting [watch][$]
- Douglas Crockford: An Inconvenient API - The Theory of the DOM [watch]
- DOM Enlightenment [read][$] or read online for free
- Fixing Common jQuery Bugs [watch][$]
- jQuery-Free JavaScript [watch][$]
- jQuery Tips and Tricks [watch][$]

References/Docs:

- jQuery Docs
- Events
- DOM Browser Support
- DOM Events Browser Support
- HTML Interfaces Browser Support
- MDN Document Object Model (DOM)
- MDN Browser Object Model
- MDN Document Object Model
- MDN Event reference
- MSDN Document Object Model (DOM)

Standards/Specifications:

- Document Object Model (DOM) Level 3 Events Specification
- Document Object Model (DOM) Technical Reports
- DOM Living Standard
- W3C DOM4

Learn Web Fonts, Icons, & Images

> Web typography refers to the use of fonts on the World Wide Web. When HTML was first created, font faces and styles were controlled exclusively by the settings of each Web browser. There was no mechanism for individual Web pages to control font display until Netscape introduced the `<font>` tag in 1995, which was then standardized in the HTML 3.2 specification. However, the font specified by the tag had to be installed on the user's computer or a fallback font, such as a browser's default sans-serif or monospace font, would be used. The first Cascading Style Sheets specification was published in 1996 and provided the same capabilities.
>
> The CSS2 specification was released in 1998 and attempted to improve the font selection process by adding font matching, synthesis and download. These techniques did not gain much use, and were removed in the CSS2.1 specification. However, Internet Explorer added support for the font downloading feature in version 4.0, released in 1997. Font downloading was later included in the CSS3 fonts module, and has since been implemented in Safari 3.1, Opera 10 and Mozilla Firefox 3.5. This has subsequently increased interest in Web typography, as well as the usage of font downloading.
>
> — *Wikipedia*

Fonts:

- A Comprehensive Guide to Font Loading Strategies [read]
- Beautiful Web Type a Showcase of the Best Typefaces from the Google Web Fonts Directory [read]
- Quick Guide to Webfonts via @font-face [read]
- MDN: Web fonts [read]
- Responsive Typography [watch][$]
- Typography for the Web [watch][$]

Icons:

- [read] [watch]

Images:

- MDN: Images in HTMLb [read]
- MDN: Responsive images [read]
- SVG ON THE WEB - A Practical Guide [read]

Learn Accessibility

> Accessibility refers to the design of products, devices, services, or environments for people with disabilities. The concept of accessible design ensures both “direct access” (i.e., unassisted) and "indirect access" meaning compatibility with a person's assistive technology (for example, computer screen readers).
>
> Accessibility can be viewed as the "ability to access" and benefit from some system or entity. The concept focuses on enabling access for people with disabilities, or special needs, or enabling access through the use of assistive technology; however, research and development in accessibility brings benefits to everyone.
>
> Accessibility is not to be confused with usability, which is the extent to which a product (such as a device, service, or environment) can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.
>
> Accessibility is strongly related to universal design which is the process of creating products that are usable by people with the widest possible range of abilities, operating within the widest possible range of situations. This is about making things accessible to all people (whether they have a disability or not).
>
> — *Wikipedia*

General Learning:

- 9 tips to get bare minimum of web accessibility
- Foundations of UX: Accessibility [watch][$]
- How HTML elements are supported by screen readers [read]
- Introduction to Web Accessibility - Google Open Online Education [watch]
- Introduction to Web Accessibility - WAI [read]
- Universal Design for Web Applications: Web Applications That Reach Everyone [read][$]
- Web Accessibility: Getting Started [watch][$]
- A Web for Everyone [read][$]
- Web Accessibility [watch][$]
- A11ycasts [watch]
- Accessibility by Google - Udacity course [watch]

Standards/Specifications:

- Accessible Rich Internet Applications (WAI-ARIA) Current Status
- Web Accessibility Initiative (WAI)
- Web Content Accessibility Guidelines (WCAG) Current Status

Learn Web/Browser APIs

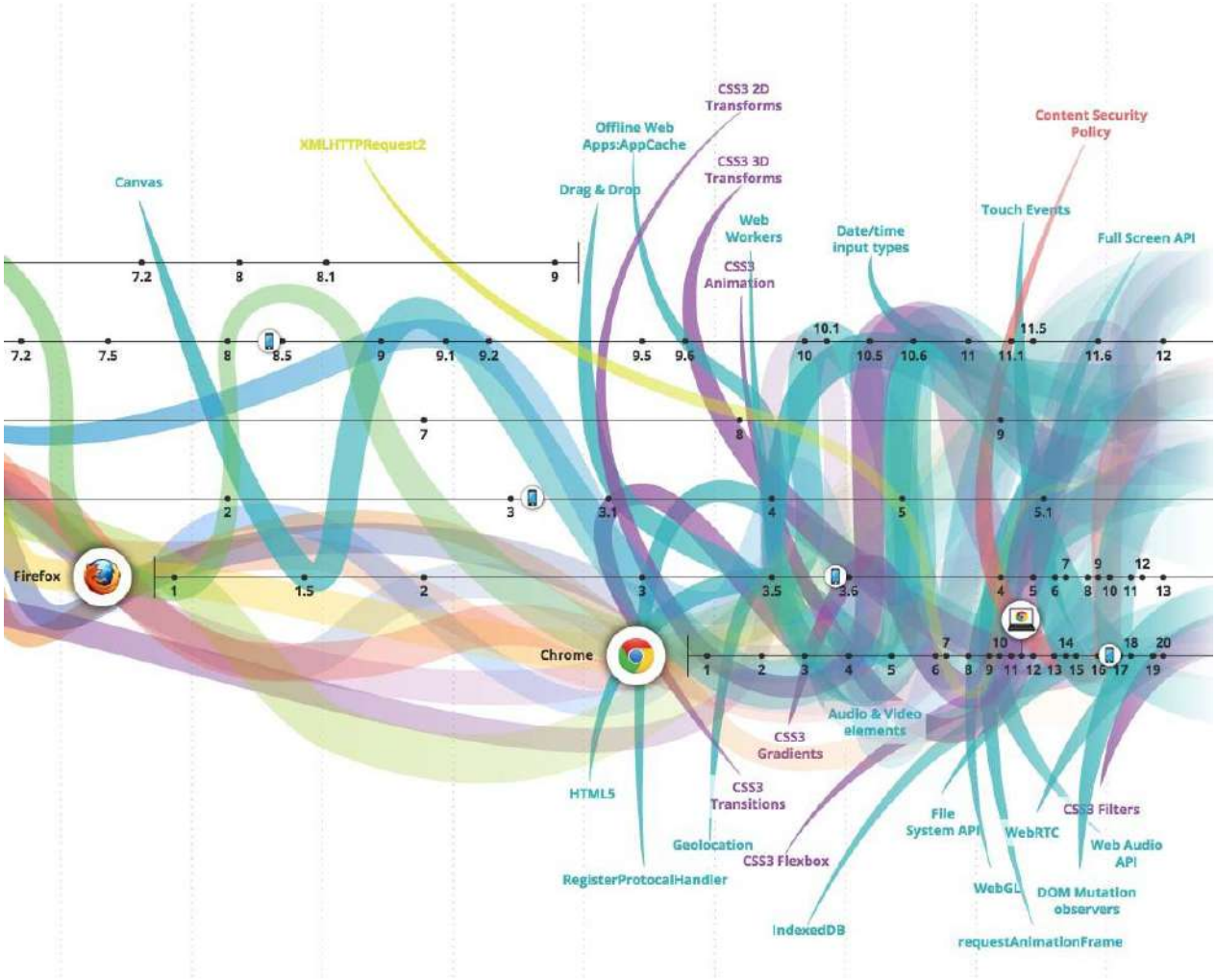


Image source: http://www.evolutionoftheweb.com/

The BOM (Browser Object Model) and the DOM (Document Object Model) are not the only browser APIs that are made available on the web platform inside of browsers. Everything that is not specifically the DOM or BOM, but an interface for programming the browser could be considered a web or browser API (tragically in the past some of these APIs have been called HTML5 APIs which confuses their own specifics/standardize with the actual HTML5 specification specifying the HTML5 markup language). Note that web or browser APIs do include device APIs (e.g., `Navigator.getBattery()`) that are available through the browser on tablet and phones devices.

You should be aware of and learn, where appropriate, web/browser APIs. A good tool to use to familiarize oneself with all of these APIs would be to investigate the HTML5test.com results for the 5 most current browsers.

MDN has a great deal of information about web/browser APIs.

- MDN Web API Reference
- MDN Web APIs Interface Reference - All Interfaces, Arranged Alphabetically
- MDN WebAPI - Lists Device Access APIs and Other APIs Useful for Applications

Keep in mind that not every API is specified by the W3C or WHATWG.

In addition to MDN, you might find the following resources helpful for learning about all the web/browser API's:

- The HTML 5 JavaScript API Index
- HTML5 Overview
- platform.html5.org

Learn JSON (JavaScript Object Notation)

> JSON, (canonically pronounced sometimes JavaScript Object Notation), is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is the primary data format used for asynchronous browser/server communication (AJAJ), largely replacing XML (used by AJAX).
>
> Although originally derived from the JavaScript scripting language, JSON is a language-independent data format. Code for parsing and generating JSON data is readily available in many programming languages.
>
> The JSON format was originally specified by Douglas Crockford. It is currently described by two competing standards, RFC 7159 and ECMA-404. The ECMA standard is minimal, describing only the allowed grammar syntax, whereas the RFC also provides some semantic and security considerations. The official Internet media type for JSON is application/json. The JSON filename extension is .json.
>
> — *Wikipedia*

General Learning:

- Introduction to JavaScript Object Notation: A To-the-Point Guide to JSON [read][$]
- json.com [read]
- What is JSON [watch]

References/Docs:

- json.org [read]

Standards/Specifications:

- ECMA-404 The JSON Data Interchange Format
- RFC 7159 The JavaScript Object Notation (JSON) Data Interchange Format
- STD 90 - RFC 8259 - The JavaScript Object Notation (JSON) Data Interchange Format, DECEMBER 2017

Architecting:

- JSON API

NOTES:

JSON apis are being replaced by things like GraphQL and Falcor.

Learn JS Templates

A JavaScript template is typically used, but not always with a MV* solution to separate parts of the view (i.e., the UI) from the logic and model (i.e., the data or JSON).

- ES6 Template Literals, the Handlebars killer? [read]
- Getting Started with nunjucks [read]
- Instant Handlebars.js [read][$]
- JavaScript Templating with Handlebars [watch][$]
- Learn Handlebars in 10 Minutes or Less [read]
- Lodash Templates [docs]

NOTES:

Note that JavaScript 2015 (aka ES6) has a native templating mechanism called "Templates strings". Additionally, templating as of late has been replaced by things like JSX, a template element, or HTML strings.

ADVICE:

If I was not using React & JSX I'd first reach for JavaScript "Templates strings" and when that was lacking move to nunjucks.

Learn Static Site Generators

Static site generators, typically written using server side code (i.e., ruby, php, python, nodeJS, etc.), produce static HTML files from static text/data + templates that are intended to be sent from a server to the client statically without a dynamic nature.

General Learning:

- JAMstack [read]
- Static Site Generators [read]
- Working with Static Sites - Bringing the Power of Simplicity to Modern Sites [read][$]

Learn Computer Science via JS

- Four Semesters of Computer Science in Six Hours [video][$]
- Computer Science in JavaScript [read]
- Collection of classic computer science paradigms, algorithms, and approaches written in JavaScript [read]
- Algorithms and Data Structures in JavaScript [watch][$]

Learn Front-End Application Architecture

General Learning: [1]

- JavaScript Application Design [read][$]
- Programming JavaScript Applications [read]
- Grab Front End Guide [read]
- A set of best practices for JavaScript projects
- Spellbook of Modern Web Dev
- JavaScript Stack from Scratch

Deprecated Learning Materials:

- Build an App with React and Ampersand [watch]
- Building Modern Single-Page Web Applications [watch][$]
- Eloquent JavaScript: Modules [read]
- Field Guide to Web Applications [read]
- Frontend Guidelines Questionnaire [read]
- Human JavaScript [read]
- Nicholas Zakas: Scalable JavaScript Application Architecture [watch]
- Organizing JavaScript Functionality [watch][$]
- Patterns for Large-Scale JavaScript Application Architecture [read]
- Terrific [read]
- UI Architecture [watch][$]
- Web UI Architecture [watch][$]

NOTES:

Not a lot of general content is being created on this topic as of late. Most of the content offered for learning how to build front-end/SPA/JavaScript applications presupposes you've decided up a tool like Angular, Ember, React, or Aurelia.

ADVICE:

[1] In 2018 learn React and Mobx.

Learn Data (i.e. JSON) API Design

- API Design in Node.js (using Express & Mongo) [watch][$]
- Build APIs You Won't Hate [$][read]
- JSON API [read]
- RESTful Web API Design with Node.JS - Second Edition [$][read]

Learn React

Learning React:

- The Beginner's Guide to ReactJS [read]
- React.js Introduction For People Who Know Just Enough jQuery To Get By [read]
- 13 things you need to know about React [read]
- Tutorial: Intro To React [read]
- ReactJS For Stupid People [read]
- Complete Intro to React, v3 (feat. Redux, Router & Flow) [watch][$]
- React [read]
- React Enlightenment [read]
- REACT JS TUTORIAL #1 - Reactjs Javascript Introduction & Workspace Setup [watch]

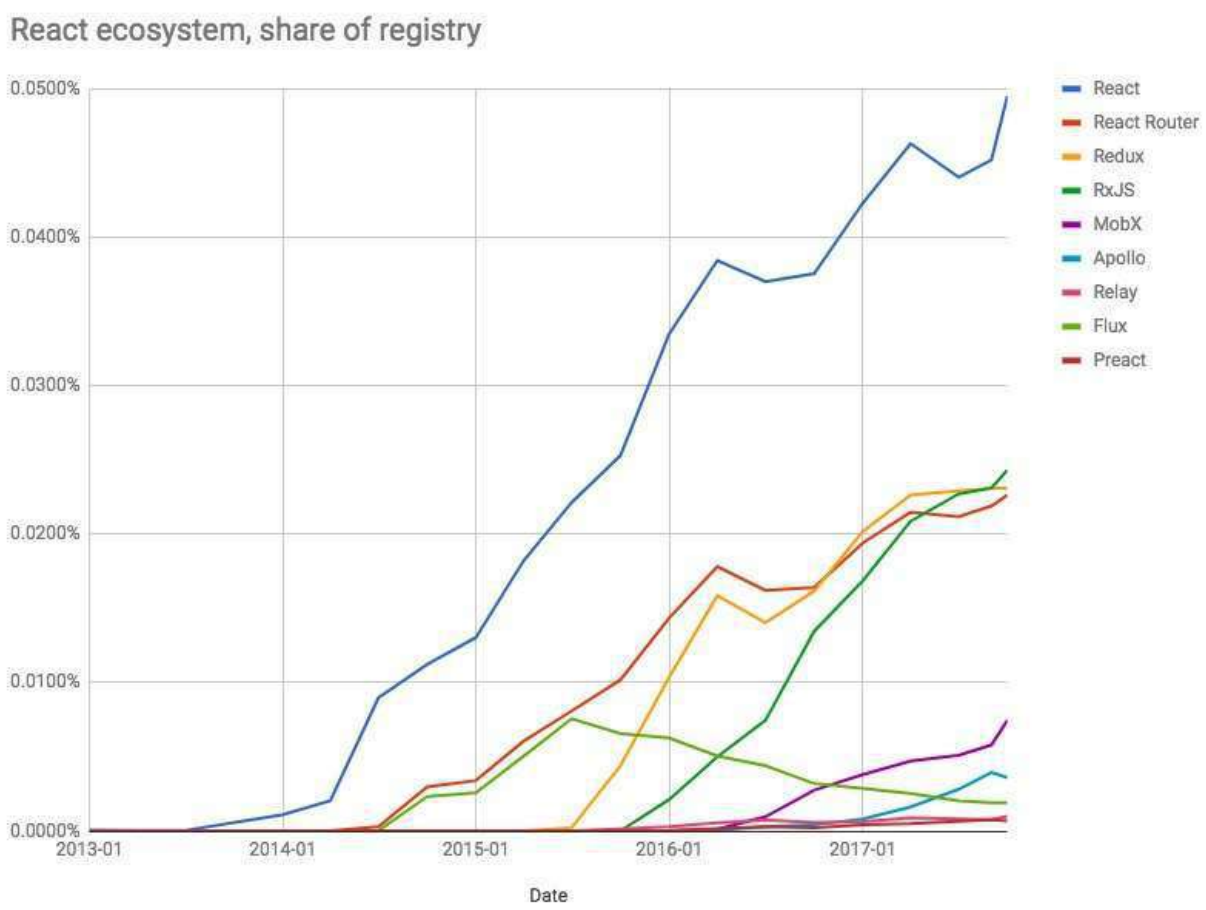
Mastering React:

- Build Your First Production Quality React App [watch][$]
- Advanced React Component Patterns [watch][$]
- React Patterns [read]
- 8 Key React Component Decisions [read]
- React + Mobx codebase containing real world examples (CRUD, auth, advanced patterns, etc) that adheres to the RealWorld spec and API. [code]
- An Introduction to React Router v4 and its Philosophy Toward Routing [read]

NOTES:

Once you have a good handle on React move on to learning a more robust state management solution like MobX. If you are an experienced developer with Functional Programming knowledge look at Redux. If you need help understanding the role of state management beyond React's `setState` watch, "Advanced State Management in React (feat. Redux and MobX)".

Note below the trends in the React Ecosystem (from npm registry) as you move past React and learn tools that work alongside React.


React ecosystem, share of registry
0.0500%
0.0400%
0.0300%
0.0200%
0.0100%
0.0000%
2013-01
2014-01
2015-01
2016-01
2017-01
Date
React
React Router
Redux
RxJS
MobX
Apollo
Relay
Flux
Preact

Learn State Management

- State management in JavaScript [read]
- Advanced State Management in React (feat. Redux and MobX) [watch][$]
- React js tutorial - How Redux Works [watch]
- MobX + React is AWESOME [watch]

Learn Progressive Web App

> Unlike traditional applications, progressive web apps are a hybrid of regular web pages (or websites) and a mobile application. This new application model attempts to combine features offered by most modern browsers with the benefits of mobile experience.
>
> In 2015, designer Frances Berriman and Google Chrome engineer Alex Russell coined the term "Progressive Web Apps" to describe apps taking advantage of new features supported by modern browsers, including Service Workers and Web App Manifests, that let users upgrade web apps to be first-class applications in their native OS.
>
> According to Google Developers, these characteristics are:
>
> - Progressive - Work for every user, regardless of browser choice because they're built with progressive enhancement as a core tenet.
> - Responsive - Fit any form factor: desktop, mobile, tablet, or forms yet to emerge.
> - Connectivity independent - Service workers allow work offline, or on low quality networks.
> - App-like - Feel like an app to the user with app-style interactions and navigation.
> - Fresh - Always up-to-date thanks to the service worker update process.
> - Safe - Served via HTTPS to prevent snooping and ensure content hasn't been tampered with.
> - Discoverable - Are identifiable as "applications" thanks to W3C manifests[6] and service worker registration scope allowing search engines to find them.
> - Re-engageable - Make re-engagement easy through features like push notifications.
> - Installable - Allow users to "keep" apps they find most useful on their home screen without the hassle of an app store.
> - Linkable - Easily shared via a URL and do not require complex installation.
>
> — *Wikipedia*

- A Beginner's Guide To Progressive Web Apps [read]
- Progressive Web Apps [read]
- Getting Started with Progressive Web Apps [watch][$]
- Building a Progressive Web App [watch][$]
- Intro to Progressive Web Apps by Google [watch]
- Native Apps are Doomed [read]
- Why Native Apps Really are Doomed: Native Apps are Doomed pt 2 [read]
- Your First Progressive Web App [read]

- Progressive Web Applications and Offline [watch][$]

Learn JS API Design

- Designing Better JavaScript APIs [read]
- Writing JavaScript APIs [read]

Learn Web Developer Tools

> Web development tools allow web developers to test and debug their code. They are different from website builders and IDEs in that they do not assist in the direct creation of a webpage, rather they are tools used for testing the user facing interface of a website or web application.
>
> Web development tools come as browser add-ons or built in features in web browsers. The most popular web browsers today like, Google Chrome, Firefox, Opera, Internet Explorer, and Safari have built in tools to help web developers, and many additional add-ons can be found in their respective plugin download centers.
>
> Web development tools allow developers to work with a variety of web technologies, including HTML, CSS, the DOM, JavaScript, and other components that are handled by the web browser. Due to the increasing demand from web browsers to do more popular web browsers have included more features geared for developers.
>
> — *Wikipedia*

While most browsers come equipped with web developer tools, the Chrome developer tools are currently the most talked about and widely used.

I'd suggest learning and using the Chrome web developer tools, simply because the best resources for learning web developer tools revolves around Chrome DevTools.

Learn Chrome Web Developer Tools:

- Chrome Developer Tools [watch][$]
- Explore and Master Chrome DevTools [watch]
- Mastering Chrome Developer Tools [watch][$]
- Using The Chrome Developer Tools [watch][$]
- Learning Chrome Web Developer Tools [watch][$]

Chrome Web Developer Tools Docs:

- Command Line API Reference
- Keyboard & UI Shortcuts Reference
- Per-Panel Documentation
- Configure and Customize DevTools

News/Newsletters/Podcasts/Tips:

- Dev Tips

Learn Command Line

> A command-line interface or command language interpreter (CLI), also known as command-line user interface, console user interface, and character user interface (CUI), is a means of interacting with a computer program where the user (or client) issues commands to the program in the form of successive lines of text (command lines).
>
> — *Wikipedia*

General Learning:

- The Bash Guide [read]
- Codecademy: Learn the Command Line [watch]
- Command Line Power User [watch]
- Learn Enough Command Line to Be Dangerous [read] [free to $]
- Meet the Command Line [watch][$]

Mastering:

- Advanced Command Line Techniques [watch][$]
- Introduction to Bash, VIM & Regex [watch][$]

Learn Node.js

> Node.js is an open-source, cross-platform runtime environment for developing server-side web applications. Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, Linux, FreeBSD, NonStop, IBM AIX, IBM System z and IBM i. Its work is hosted and supported by the Node.js Foundation, a collaborative project at Linux Foundation.
>
> Node.js provides an event-driven architecture and a non-blocking I/O API designed to optimize an application's throughput and scalability for real-time web applications. It uses Google V8 JavaScript engine to execute code, and a large percentage of the basic modules are written in JavaScript. Node.js contains a built-in library to allow applications to act as a web server without software such as Apache HTTP Server, Nginx or IIS.
>
> — *Wikipedia*

General Learning:

- The Art of Node [read]
- Introduction to Node.js [watch][$]
- Introduction to Node.js from Evented Mind [watch]
- io.js and Node.js Next: Getting Started [watch][$]
- Learning Node: Moving to the Server-Side [read][$]
- Learn You The Node.js [self-guided workshops]
- Node.js Basics [watch][$]
- Node.js in Practice [read][$]
- Real-time Web with Node.js [watch]
- REST & GraphQL API Design in Node.js, v2 (using Express & MongoDB) [watch][$]
- Learn Node [watch][$]

Learn Modules

General Learning:

- jsmodules.io
- ES6 Modules in Depth [read]
- Exploring JS - Modules [read]

References/Docs:

- MDN - export
- MDN - import

NOTES:

We are still waiting on wide support in browsers for loading modules. Until then you can have a look at, "ES Module Loader Polyfill", "JavaScript Loader Standard", and ECMAScript modules in browsers.

Learn Module loaders/bundlers

Webpack:

- Webpack [read]
- Webpack Deep Dive [watch][$]
- Webpack Fundamentals [watch][$]
- Survivejs.com Webpack Book [read]

Rollup:

- Rollup [read]
  - Microbundle

Parcel

- Parcel [read]

NOTES:

It is not uncommon for developers to use a tool like Gulp for bundling JS modules. However, many of the Gulp plugins simply use Webpack or Rollup under the hood.

Learn Package Manager

> A package manager or package management system is a collection of software tools that automates the process of installing, upgrading, configuring, and removing software packages for a computer's operating system in a consistent manner. It typically maintains a database of software dependencies and version information to prevent software mismatches and missing prerequisites.
>
> — *Wikipedia*

General Learning:

- An introduction to how JavaScript package managers work
- The Mystical & Magical SemVer Ranges Used By npm & Bower [read]
- Package Managers: An Introductory Guide For The Uninitiated Front-End Developer [read]
- npm docs
- yarn docs

Learn Version Control

> A component of software configuration management, version control, also known as revision control or source control, is the management of changes to documents, computer programs, large web sites, and other collections of information. Changes are usually identified by a number or letter code, termed the "revision number," "revision level," or simply "revision." For example, an initial set of files is "revision 1." When the first change is made, the resulting set is "revision 2," and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged.
>
> — *Wikipedia*

The most common solution used for version control today is Git. Learn it!

General Learning:

- codeschool.com [interact]
- Getting Git Right [read]
- Git Fundamentals [watch][$]
- learn Enough Git [read]
- Ry's Git Tutorial [read]

Mastering:

- Advanced Git Tutorials [read]
- Pro Git [read]
- Learn Git Branching [interact]

References/Docs:

- https://git-scm.com/doc
- git-cheatsheet

Learn Build and Task Automation

> Build automation is the process of automating the creation of a software build and the associated processes including: compiling computer source code into binary code, packaging binary code, and running automated tests.
>
> — *Wikipedia*

General Learning:

- Getting Started with Gulp [read][$]
- Gulp Basics [watch][$]
- JavaScript Build Automation With Gulp.js [watch][$]

References/Docs:

- Gulp

ADVICE:

Gulp is great. However, you might only need `npm run`. Before turning to additional complexity in your application stack ask yourself if `npm run` can do the job. If you need more, use Gulp.

Read:

- Give Grunt the Boot! A Guide to Using npm as a Build Tool
- How to Use npm as a Build Tool
- Task Automation with npm Run
- Using npm as a Build System for Your next Project
- Using npm as a Task Runner [watch][$]
- Why I Left Gulp and Grunt for npm Scripts
- Why npm Scripts?

Learn Site Performance Optimization

> Web performance optimization, WPO, or website optimization is the field of knowledge about increasing the speed in which web pages are downloaded and displayed on the user's web browser. With the average internet speed increasing globally, it is fitting for website administrators and webmasters to consider the time it takes for websites to render for the visitor.
>
> — *Wikipedia*

General Learning:

- Browser Rendering Optimization [watch]
- Even Faster Web Sites: Performance Best Practices for Web Developers [read][$]
- High Performance Web Sites: Essential Knowledge for Front-End Engineers [read][$]
- JavaScript Performance Rocks [read][$]
- PageSpeed Insights Rules [read]
- perf-tooling.today [read]
- Performance Calendar [read]
- perf.rocks [read]
- Using WebPageTest [read][$]
- Web Performance Daybook Volume 2 [read][$]
- Website Performance [watch][$]
- Website Performance Optimization [watch]
- Front-End Performance Checklist 2018 [PDF, Apple Pages] [read]

Learn Testing

> **Unit Testing** - In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application.
>
> — *Wikipedia*
>
> **Functional Testing** - Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional testing usually describes what the system does.
>
> — *Wikipedia*
>
> **Integration Testing** - Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.
>
> — *Wikipedia*

General Learning:

- Front-End First: Testing and Prototyping JavaScript Apps [watch][$]
- Let's Code: Test-Driven JavaScript [watch][$]
- JavaScript Testing [watch]
- JavaScript Testing Recipes [read][$]
- Testable JavaScript [read][$]
- Test-Driving JavaScript Applications: Rapid, Confident, Maintainable Code[read][$]
- Test-Driven JavaScript Development [read][$]
- The Way of the Web Tester: A Beginner's Guide to Automating Tests [read][$]
- Testing and Modular Front-End [watch][$]
- Testing JavaScript Applications (feat. React and Redux) [watch][$]
- Learn Javascript Unit Testing With Mocha, Chai and Sinon [watch][$]

Learn Headless Browsers

> A headless browser is a web browser without a graphical user interface.
>
> Headless browsers provide automated control of a web page in an environment similar to popular web browsers, but are executed via a command line interface or using network communication. They are particularly useful for testing web pages as they are able to render and understand HTML the same way a browser would, including styling elements such as page layout, color, font selection and execution of JavaScript and AJAX which are usually not available when using other testing methods. Google stated in 2009 that using a headless browser could help their search engine index content from websites that use AJAX.
>
> — *Wikipedia*

- Getting Started with Headless Chrome [readme]

NOTES:

PhantomJS is no longer maintained, Headless Chrome steps in.

Learn Offline Development

Offline development (aka offline first) is an area of knowledge and discussion around development practices for devices that are not always connected to the Internet or a power source.

General Learning:

- Creating HTML5 Offline Web Applications [read]
- Everything You Need to Know to Create Offline-First Web Apps [read]
- Offline First [read]
- offlinefirst.org [read]
- The Offline Cookbook [read]
- Offline Quickstart[read]

Learn Web/Browser/App Security

- Browser Security Handbook [read]
- Frontend Security [watch]
- Hacksplaining [read]
- HTML5 Security Cheatsheet [read]
- HTTP Security Best Practice [read]
- Identity and Data Security for Web Development: Best Practices read
- Security for Web Developers: Using JavaScript, HTML, and CSS [read][$]
- The Basics of Web Application Security [read]
- The Internet: Encryption & Public Keys [watch]
- The Internet: Cybersecurity & Crime [watch]
- The Tangled Web: A Guide to Securing Modern Web Applications [read][$]
- Web Security Basics [read]
- Web security [read]

Learn Multi-Device Development

Image source: http://bradfrost.com/blog/post/this-is-the-web/

A website or web application can run on a wide range of computers, laptops, tablets and phones, as well as a handful of new devices (watches, thermostats, fridges, etc.). How you determine what devices you'll support and how you will develop to support those devices is

called, "multi-device development strategy". Below, I list the most common multi-device development strategies.

- Build a responsive (RWD) web site/app for all devices.
- Build an adaptive/progressively enhanced web site/app for all devices.
- Build a website, web app, native app, or hybrid-native app for each individual device or a grouping of devices.
- Attempt to retrofit something you have already built using bits and parts from strategies 1, 2 or 3.

General Learning:

- A book Apart Pack - Responsive Web Design [read][$]
- A Book Apart Pack - Design For Any Device [read][$]
- Adaptive Web Design [read][$]
- Designing with Progressive Enhancement [read][$]
- Mobile Web Development [watch]
- Responsive HTML Email Design [watch][$]
- Responsive Images [watch]
- Responsive Typography [watch][$]
- Responsive Web Design [watch][$]
- Responsive Web Design Fundamentals [watch]

Responsive Newsletters, News, & Podcasts:

- Responsive Web Design Podcast
- Responsive Web Design Newsletter

Directed Learning

This section focuses on directed learning via schools, courses, programs and bootcamps.

Directed Learning

The table below contains instructor led, paid, front-end courses, programs, schools, and bootcamps.

However, if you can't afford a directed education (can be very expensive), a self directed education using screencasts, books, and articles is a viable alternative to learn front-end development for the self-driven individual.

company	course	price	on site	remote	duration
Betamore	Front-end Web Development	3,000	Baltimore, MD		10 weeks
BLOC	Become a Frontend Developer	4,999		yes	16 weeks @ 25hr/wk or 32 weeks @ 10hr/wk
General Assembly	Frontend Web Development	3,500	multiple locations		3 hrs/day 2 days/wk for 8 weeks
HackerYou	Front-end Web Development Immersive	7,000 - 7,910	Toronto, Canada		9 weeks
The New York Code + Design Academy	Front End 101	2,000	New York, NY		8 weeks part-time
Thinkful	Frontend Web Development	300 per month		yes	15 hrs/wk for 3 months
Turing School of Software & Design	Front-End Engineering	20,000	Denver, CO		7 months full time
Udacity	Front-End Web Developer Nanodegree	200 monthly	multiple locations	yes	6 months 12hrs/wk

NOTES:

For a complete list of courses/bootcamps look at switchup.org or coursereport.com.

Front-End Developers to Learn From

The notion that you should follow an individual to learn about front-end development is slowly becoming pointless. The advanced practitioners of front-end development generate enough content that you can simply follow the community/leaders by paying attention to the front-end "news" outlets (via Newsletters, News, & Podcasts).

Front-End Newsletters, News Sites, & Podcasts

General Front-End Newsletters, News, & Podcasts:

- The Big Web Show
- Dev Tips
- Front End Happy Hour
- Front-End Front
- FrontEnd Focus
- The Frontside Podcast
- Mobile Web Weekly
- Non Breaking Space Show
- Web Platform News Weekly
- ShopTalk Show
- UX Design Newsletter
- The Versioning Show by SitePoint
- The Web Ahead
- Web Development Reading List
- The Web Platform Podcast
- Web Tools Weekly
- Fresh Brewed Frontend
- Pony Foo Weekly

HTML/CSS Newsletters:

- Accessibility Weekly
- CSS Weekly
- CSS-Tricks
- csslayout.news

JavaScript Newsletters, News, & Podcasts:

- Awesome JavaScript Newsletter
- Echo JS
- ECMAScript Daily
- ES.next News
- JavaScript Jabber
- JavaScript Kicks

- JavaScript Live
- JavaScript Weekly
- JavaScript.com
- JSter
- React Status

Graphic and Animation Newsletters and Podcasts

- Motion and Meaning
- Responsive Images Community Group Newsletter
- SVG Immersion Podcast
- Web Animation Weekly

NOTES:

Need more Newsletters, News Sites, & Podcasts look at Awesome Newsletter.

Part III: Front-end Developer Tools

Part three briefly explains and identifies tools of the trade.

Make sure you understanding the category that a set of tools falls within, before studying the tools themselves.

Note that just because a tool is listed, or a category of tools is documented, this does not equate to an assertion on my part that a front-end developer should learn it and use it. Choose your own toolbox. I'm just providing the common toolbox options.

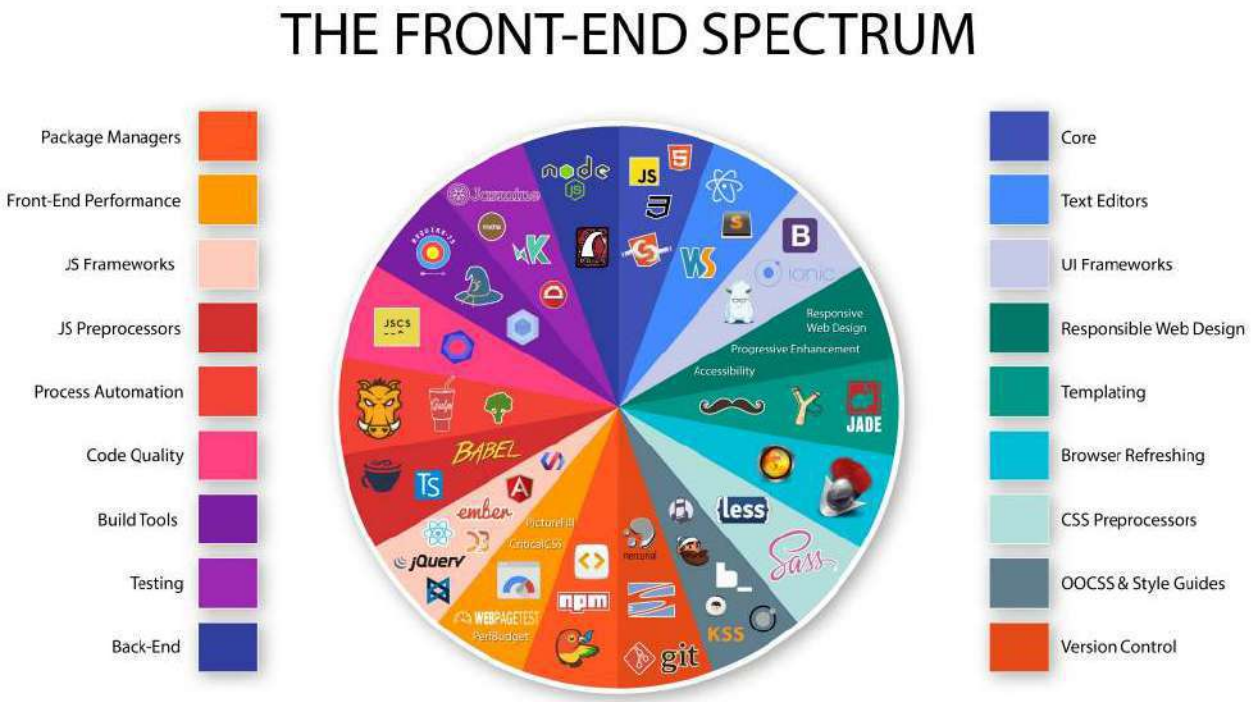


Image source: https://medium.com/@withinsight1/the-front-end-spectrum-c0f30998c9f0

Doc/API Browsing Tools

Tools to browser common developer documents and developer API references.

- Dash [OS X, iOS][$]
- DevDocs
- Velocity [Windows][$]
- Zeal [Windows, Linux]

Cheatsheets:

- devhints.io

SEO Tools

- Keyword Tool
- Google Webmasters Search Console
- Varvy SEO tool

Tools for Finding SEO Tools:

- SEO Tools - The Complete List

Prototyping & Wireframing Tools

Creating:

- Axure [$]
- Balsamiq Mockups [$]
- Justinmind [$]
- Moqups [$]
- proto.io [$]
- UXPin [free to $]

Collaboration / Presenting:

- InVision [free to $]
- Conceptboard [free to $]
- myBalsamiq [$]

Diagramming Tools

- draw.io [free to $]
- Cacoo [free to $]
- gliffy [free to $]
- sketchboard.io [$]

HTTP/Network Tools

- Charles [$]
- Chrome DevTools Network Panel
- Insomnia [free - $]
- Mitmproxy [free]
- Paw [$]
- Postman [free - $]

Code Editing Tools

> A source code editor is a text editor program designed specifically for editing source code of computer programs by programmers. It may be a standalone application or it may be built into an integrated development environment (IDE) or web browser. Source code editors are the most fundamental programming tool, as the fundamental job of programmers is to write and edit source code.
>
> — *Wikipedia*

Front-end code can minimally be edited with a simple text editing application like Notepad or TextEdit. But, most front-end practitioners use a code editor specifically design for editing a programming language.

Code editors come in all sorts of types and size, so to speak. Selecting one is a rather subjective engagement. Choose one, learn it inside and out, then get on to learning HTML, CSS, DOM, and JavaScript.

However, I do strongly believe, minimally, a code editor should have the following qualities (by default or by way of plugins):

1. Good documentation on how to use the editor
2. Report (i.e., hinting/linting/errors) on the code quality of HTML, CSS, and JavaScript.
3. Offer syntax highlighting for HTML, CSS, and JavaScript.
4. Offer code completion for HTML, CSS, and JavaScript.
5. Be customizable by way of a plug-in architecture
6. Have available a large repository of third-party/community plug-ins that can be used to customize the editor to your liking
7. Be small, simple, and not coupled to the code (i.e., not required to edit the code)

Code Editors: [1]

- Atom
- Sublime Text [$]
- WebStorm [$]
- Visual Studio Code

Online Code Editors:

- PaizaCloud [free to $]
- AWS Cloud9 [$]
- Codeanywhere [free to $]

Shareable & Runnable Code Editors:

Used to share limited amounts of immediately runnable code. Not a true code editor but a tool that can be used to share small amounts of immediately runnable code in a web browser.

- CodePen [free to $]
- jsbin.com [free to $]
- jsfiddle.net
- StackBliz
- codeSandbox

ADVICE:

[1] I recommending using Visual Studio Code because of the quality of the tool and the continuous improvements made to the editor that likely won't stop or slow due to the fact that Microsoft is behind the tool. It is widely used:

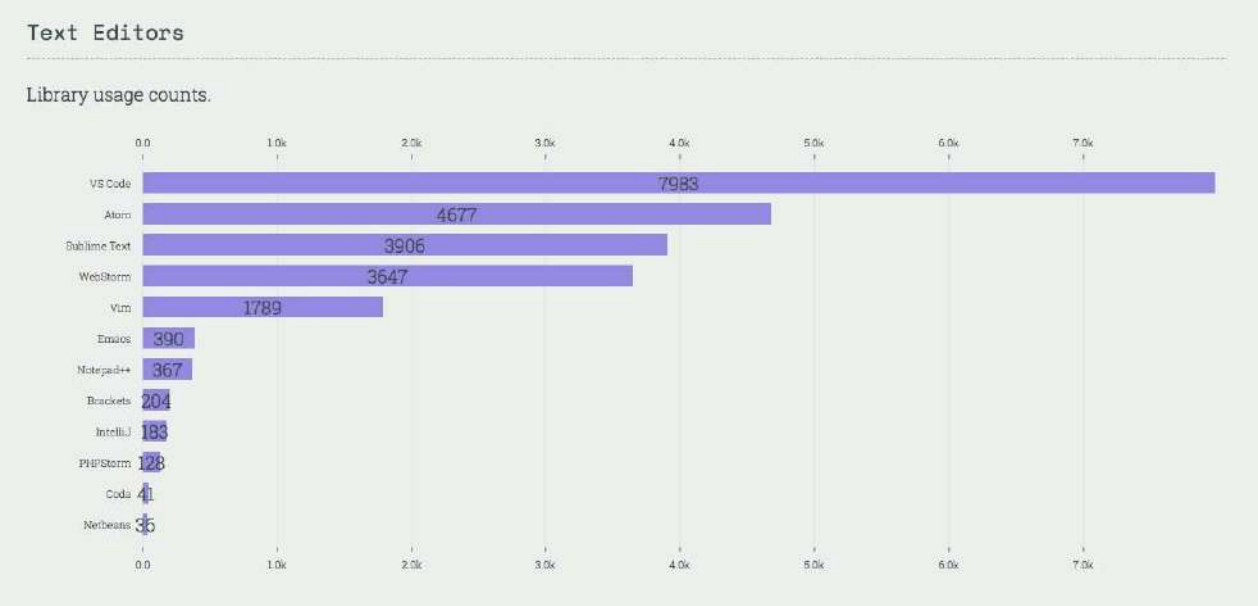


Image source: https://stateofjs.com/2017/other-tools/

Browser Tools

JS Utilities to fix Browsers:

- History.js
- html2canvas
- Platform.js
- URI.js

General Reference Tools to Determine If X Browser Supports X:

- Browser support for broken/missing images
- Browserscope
- caniuse.com
- Firefox Platform Status - Implementation & standardization roadmap for web platform features
- HTML5 Please
- HTML5 Test
- iwanttouse.com
- Platform Status
- web-platform-tests dashboard
- whatwebcando.today

Browser Development/Debug Tools:

- Chrome Developer Tools (aka DevTools)
  - Per-Panel Documentation
  - Command Line API Reference
  - Keyboard & UI Shortcuts Reference
  - Settings
- Firefox Developer Tools
- IE Developer tools (aka F12 tools)
- Safari Web Inspector
- Vorlon.js

JavaScript Utilities to Determine If X Browser Supports X:

- Feature.js
- Modernizr

Broad Browser Polyfills/Shims:

- console-polyfill
- HTML5 Cross Browser Polyfills
- fetch
- socket.io
- SockJS
- webcomponents.js
- webshim

Hosted Testing/Automation for Browsers:

- Browserling [free to $]
- BrowserStack [$]
- CrossBrowserTesting.com [$]
- Ghost Inspector [free to $]
- Nightcloud.io
- Sauce Labs [$]

Headless Browsers:

- PhantomJS
  - PhantomCSS
- slimerjs
- Zombie.js
- Headless Chromium

Browser Automation:

Used for functional testing and monkey testing.

- CasperJS
- Nightmare
- TestCafe

Browser Hacks:

- browserhacks.com

Browser Syncing Tools:

- Browsersync

Browser List:

Share target browsers between different front-end tools, like Autoprefixer, Stylelint and babel-preset-env.

- Browserslist
  - http://browserl.ist/

HTML Tools

HTML Templates/Boilerplates/Starter Kits:

- dCodes
- Email-Boilerplate
- HTML5 Boilerplate
- HTML5 Bones
- Mobile boilerplate

HTML Polyfill:

- html5shiv

Transpiling:

- Pug
- Markdown

References:

- Element attributes
- Elements
- HTML Arrows
- HTML Entity Lookup
- htmlreference.io
- HEAD - A free guide to elements

Linting/Hinting:

- HTMLHint
- html-inspector

Optimizer:

- HTML Minifier

Online Creation/Generation/Experimentation Tools:

- tablesgenerator.com

Authoring Conventions:

- HTML Code Guide

- Principles of Writing Consistent, Idiomatic HTML

Workflow:

- Emmet

HTML Outliner:

- HTML 5 Outliner

Trending HTML Repositories on GitHub This Month:

https://github.com/trending?l=html&since=monthly

CSS Tools

CSS Utilities:

- Basscss
- Skeleton
- Shed
- Tailwind CSS
- Tachyons

CSS Frameworks (utilities + UI):

- Base
- Bulma
- Bootstrap 4
- Concise
- Foundation
- Material Design Lite (MDL)
- Metro UI
- Mini.css
- Mobi.css
- Picnic
- Pure.css
- Semantic UI
- Shoelace
- Spectre.css

Mobile Only CSS Frameworks:

- Ratchet

CSS Reset:

> A CSS Reset (or “Reset CSS”) is a short, often compressed (minified) set of CSS rules that resets the styling of all HTML elements to a consistent baseline.
>
> — *cssreset.com*

- Eric Meyer's “Reset CSS” 2.0
- Normalize
- sanitize.css

Transpiling:

- pleeease.io
- PostCSS & cssnext
- rework & myth

References:

- CSS Cursors
- css3test.com
- css3clickchart.com
- cssreference.io
- CSS Indexes - A listing of every term defined by CSS specs
- css4-selectors.com
- css4 Rocks
- CSS TRIGGERS...A GAME OF LAYOUT, PAINT, AND COMPOSITE
- CSS Tricks Almanac
- cssvalues.com
- MDN CSS Reference
- CSS Cheat Sheet

Linting/Hinting:

- CSS Lint
- stylelint

Code Formatter/Beautifier:

- CSScomb
- CSSfmt

Optimizer:

- clear-css
- cssnano
- CSSO
- purgecss

Online Creation/Generation/Experimentation Tools:

- CSS Arrow Please
- CSS Matic
- Enjoy CSS
- Flexbox Playground

- flexplorer
- patternify.com
- patternizer.com
- Ultimate CSS Gradient Generator

Architecting CSS:

- Atomic Design [read]
- BEM
- ITCSS
- OOCSS [read]
- SMACSS [read][$]
  - Scalable Modular Architecture for CSS (SMACSS) [watch][$]
- SUIT CSS
- rscss

Authoring/Architecting Conventions:

- CSS code guide [read]
- css-architecture [read]
- cssguidelin.es [read]
- Idiomatic CSS [read]
- MaintainableCSS [read]
- Standards for Developing Flexible, Durable, and Sustainable HTML and CSS [read]
- Airbnb CSS / Sass Styleguide [read]

Style Guide Resources:

- Frontify [$]
- SC5 STYLE GUIDE GENERATOR
- styleguide-generators
- Catalog

Trending CSS Repositories on GitHub This Month:

https://github.com/trending?l=css&since=monthly

DOM Tools

DOM Libraries/Frameworks:

- Bliss
- jQuery
  - You Don't Need jQuery
- Zepto
- cash
- Umbrella JS

DOM Utilities:

- Keypress
- Tether
- clipboard.js

DOM Event Tools:

- Keyboard Event Viewer

DOM Performance Tools:

- Chrome DevTools Timeline
- DOM Monster

References:

- Events
- DOM Browser Support
- DOM Events Browser Support
- HTML Interfaces Browser Support
- MDN Document Object Model (DOM)
- MDN Browser Object Model
- MDN Document Object Model
- MDN Event reference
- MSDN Document Object Model (DOM)

DOM Polyfills/Shims:

- dom-shims
- Pointer Events Polyfill: a unified event system for the web platform

Virtual DOM:

- jsdom
- virtual-dom

JavaScript Tools

JS Utilities:

- accounting.js
- async
- axios
- chance
- date-fns
- format.js
- immutable
- is.js
- lodash
    - You-Dont-Need-Lodash-Underscore
- Luxon
- Math.js
- Moment.js
- Numeral.js
- Ramda
- RxJS
- string.js
- voca
- wait
- xregexp.com

Transpiling / Type Checking (ES *to ES*):

- Babel
- TypeScript
- Flow

Code-analysis Engine:

- Tern

Linting/Hinting & Style Linter:

- eslint

Unit Testing:

- AVA
- Jasmine
- Mocha
- Tape

Testing Assertions for Unit Testing:

- Chai
- expect.js
- should.js

Test Spies, Stubs, and Mocks for Unit Testing:

- sinon.js
- Kakapo.js

Code Formater/Beautifier:

- esformatter
- js-beautify
- jsfmt
- prettier

Performance Testing:

- benchmark.js
- jsperf.com

Visualization, Static Analysis, Complexity, Coverage Tools:

- Coveralls [$]
- Esprima
- istanbul

Optimizer:

- Closure Compiler
- UglifyJS 2
- optimize-js
- Prepack

Obfuscate:

- Javascript Obfuscator [free to $]
- JScrambler [$]

Sharable/Runnable Code Editors:

- jsbin.com [free to $]
- jsfiddle.net

Online Regular Expression Editors/Visual Tools:

- debuggex
- regex101
- regexper
- RegExr

Authoring Convention Tools:

- Airbnb's ESLint config, following our styleguide
- Standard - ESLint Shareable Config

Trending JS Repositories on GitHub This Month:

https://github.com/trending?l=javascript&since=monthly

Most Depended upon Packages on NPM:

https://www.npmjs.com/browse/depended

Static Site Generators Tools

Site Generator Listings: [1]

- staticgen.com
- staticsitegenerators.net
- Metalsmith

ADVICE:

[1] Before using a static site generator framework consider using Gulp or npm scripts to orchestrate a custom solution or use a tool that makes use of Gulp for static site generation. e.g. Blendid

Accessibility Tools

Guides

- A11Y Style Guide
- Accessibility Guidelines Checklist
- Interactive WCAG 2.0
- 18F Accessibility Guide

Site Scanners

- aXe Browser Extension
- Chrome Accessibility Developer Tools
- Tenon Accessibility Tool
- WAVE Accessibility Tool

Color Contrast Testers

- Colorable
- Colorable Matrix
- Color Safe
- Color Ratio

Low-Vision Simulators

- SEE (Chrome)
- Spectrum (Chrome)
- NoCoffee (Chrome)

Screen Readers

- VoiceOver (Mac)
- JAWS (Win)
- NVDA (Win)
- Window-Eyes (Win)

- ChromeVox (Chrome extension)
- Basic screen reader commands

Readability Testers

- Expresso App
- Hemingway App
- Grammarly
- Readability Score
- MS Office

Articles

- Getting Started with ARIA
- Reframing Accessibility for the Web
- An Alphabet of Accessibility Issues
- Practical ARIA Examples
- MDN Accessibility Guide
- Enable accessibility panel in Chrome dev tools

App Frameworks (Desktop, Mobile, Tablet, etc.) Tools

Front-End App Frameworks: [1]

- AngularJS (i.e Angular 1.x.x) + Batarang
- Angular (i.e. Angular 2.0.0 +) + angular-cli
- Aurelia + Aurelia CLI
- Ember + embercli + Ember Inspector
- Polymer
- React + create-react-app + React Developer Tools
- Vue.js + vue-cli & Vue.js devtools
- Riot

Native Hybrid Mobile WebView (i.e., Browser Engine Driven) Frameworks:

These solutions typically use Cordova, crosswalk, or a custom WebView as a bridge to native APIs.

- ionic
- onsen.io

Native Hybrid Mobile Development Webview (i.e., Browser Engine Driven) Environments/Platforms/Tools:

These solutions typically use Cordova, crosswalk, or a custom WebView as a bridge to native APIs.

- Adobe PhoneGap [$]
- cocoon.io [free to $]
- ionic hub [free to $]
- kony [$]
- Monaca [$]

Native Desktop WebView (i.e., Browser Engine Driven) App Frameworks:

- Electron
- NW.js

Native Mobile App Frameworks (Aka JavaScript Native Apps)

These solutions use a JS engine at runtime to interpret JS and bridge that to native APIs. No browser engine or WebView is used. The UI is constructed from native UI components.

- Flutter
- NativeScript
- React Native
- tabris.js [free to $]
- trigger.io [$]
- weex

References & demo apps:

- todomvc.com
- RealWorld example apps [code]
- Frontend Guidelines Questionnaire
- Frontend Guidelines

Performance:

- js-framework-benchmark
- Front-End Performance Checklist 2018

ADVICE:

[1] If you are new to front-end/JavaScript application development I'd start with Vue.js. Then I'd work my way to React. Then I'd look at Angular 2+, Ember, or Aurelia.

If you are building a simple website that has minimal interactions with data (i.e. mostly a static content web site), you should avoid a front-end framework. A lot of work can be done with a task runner like Gulp and jQuery, while avoiding the unnecessary complexity of learning and using an app framework tool.

Want something smaller than React, consider Preact. Preact is an attempt to recreate the core value proposition of React (or similar libraries like Mithril) using as little code as possible, with first-class support for ES2015. Currently the library is around 3kb (minified & gzipped).

State Tools

- Redux
- Mobx
- mobx-state-tree
- Cerebral
- freactal
- unistore

Progressive Web App Tools:

- lighthouse
- Progressive Web App Checklist

GUI Development/Build Tools

- CodeKit
- Prepros

Templating/Data Binding Tools

Just Templating:

- doT.js
- art-template
- Nunjuncks

Templating and Reactive Data Binding:

- ractive.js
- react.js
  - preact
  - inferno
  - nerv
  - rax
- riot
- Rivets.js
- vue.js

Templating to Virtual DOM:

- JSX

UI Widget & Component Toolkits

On Web Platform: [1]

- Kendo UI for jQuery [free to $]
- Materialize
- Office UI Fabric
- Semantic UI
- UiKit
- Webix [$]

React Specific, On Web Platform:

- Ant Design
- Material ui
- Semantic-UI-React
- ExtReact [$]
- Fabric

Native Desktop/Laptop/Netbook Apps via Web Platform (i.e. used with NW.js and Electron):

- Photon
- React UI Components for OS X El Capitan and Windows 10

ADVICE:

[1] If you need a basic set of UI Widgets/Components start with Semantic UI. If you are building something that needs a grid, spreadsheet, or pivot grid you'll have to look at Kendo UI or Webix. Keep in mind that most of these solutions still require jQuery.

Data Visualization (e.g., Charts) Tools

JS Libraries:

- d3
- sigmajs

Widgets & Components:

- amCharts [free to $]
- AnyChart [Non-commercial free to $]
- C3.js
- Chartist-jsj
- Chart.js
- Epoch
- FusionCharts [$]
- Google Charts
- Highcharts [Non-commercial free to $]
- ZingChart [free to $]

Services (i.e. hosted data visualization services for embedding and sharing):

- ChartBlocks [free to $]
- Datawrapper
- infogr.am [free to $]
- plotly [free to $]

Graphics (e.g., SVG, canvas, webgl) Tools

General:

- Fabric.js
- Two.js

Canvas:

- EaselJS
- Paper.js

SVG:

- d3
- GraphicsJS
- Raphaël
- Snap.svg
- svg.js

WebGL:

- pixi.js
- three.js

Animation Tools

CSS and JavaScript Utilities:

- Animate Plus
- Animate
- Anime.js
- Animista.net
- Dynamics.js
- GreenSock-JS
- Kute.js
- Magic
- Micron.js
- Motion
- TweenJS
- Popmotion
- Velocity.js

Polyfills/Shims:

- web-animations-js

Animation References:

- canianimate.com

JSON Tools

Online Editors:

- JSONmate
- JSON Editor Online

Formatter & Validator:

- jsonformatter.org
- JSON Formatter & Validator

Query Tools:

- DefiantJS
- JSON Mask
- ObjectPath

Generating Mock JSON Tools:

- JSON Generator
- Mockaroo [free to $]

Online JSON Mocking API Tools:

- FillText.com
- FakeJSON [free to $]
- Jam API
- JSONPlaceholder
- jsonbin.io
- jsonbin.org
- mockable.io
- mockapi.io
- Mocky
- RANDOM USER GENERATOR

List of public JSON API's:

- A collective list of JSON APIs for use in web development

Local JSON Mocking API Tools:

- json-server

JSON Specifications/Schemas:

- json-schema.org & jsonschema.net
- {json:api}

Placeholder Content Tools

Images:

- placehold.it
- Satyr
- Placeimg
- Lorem Pixel
- CSS-Tricks Image Resources
- LibreStock
- Unsplash
- Place Beyoncé

Device Mockups:

- placeit.net
- mockuphone.com

Text:

- Meet the Ipsums
- catipsum.com
- baconipsum.com (API)

User Data:

- uinames.com
- randomuser.me

Testing Tools

Software Testing Frameworks:

- Intern
- Jest
  - majestic
  - Enzyme
  - Cheerio

Unit Testing:

- AVA
- Jasmine
- Mocha
- Tape

Testing Assertions for Unit Testing:

- Chai
- expect.js
- should.js

Test Spies, Stubs, and Mocks for Unit Testing:

- sinon.js
- Kakapo.js

Hosted Testing/Automation for Browsers:

- Browserling [$]
- BrowserStack [$]
- CrossBrowserTesting.com [$]
- Nightcloud.io
- Sauce Labs [$]

Integration/Functional Testing:

- Cypress cypress-react-unit-test
- Nightwatch
- WebDriver.io

Browser Automation:

- CasperJS
- Nightmare
- TestCafe

UI Testing Tools:

- gremlins.js
- Percy
- BackstopJS
- PhantomCSS
- Ghost Inspector
- diff.io

Automated dead link and error detectors:

- Monkey Test It

NOTES:

Testing frameworks typically offer more tools than just unit testing. If you are looking for JavaScript unit testing solutions look at JavaScript Tools.

Nice recap of testing JavaScript in 2017.

Front-End Data Storage Tools (i.e. Data storage solution in the client)

- AlaSQL
- Dexie.js
- LocalForage
- LokiJS
- Lovefield
- lowdb
- Pouchdb
- NeDB
- RxDB

Module Loading/Bundling Tools

- Parcel
- Rollup
  - Microbundle
- webpack
  - http://www.webpackbin.com/
- Fusebox
- Browserify

Notes:

Generally, when should I use Parcel, Webpack or Rollup?

Parcel—Small to medium sized projects (<15k lines of code)

Webpack—Large to enterprise sized projects.

Rollup—For NPM packages.

Module/Package Repository Tools

- NPM
- yarn
- PNPM

NOTES:

Keep an eye on turbo.

Hosting Tools

General

- AWS [$]
- DigitalOcean [$]
- WebFaction [$]

Static

- Firebase Hosting
- netlify [free to $]
    - Bitballoon
- Surge [free to $]
- Forge [$]

Local Hosting Tools:

Localname

Project Management & Code Hosting Tools

- Assembla [free to $]
- Bitbucket [free to $]
- Codebase [$]
- Github [free to $]
- GitLab [free to $]
- Unfuddle [$]

Collaboration & Communication Tools

- Slack & screenhero [free to $]
- appear.in
- Mattermost [free to $]

Code/GitHub Collaboration & Communication:

- Gitter [free to $]

Content Management Hosted/API Tools

Headless CMS Tools:

- Contentful [$]
- prismic.io [free to $]
- Headless

Self-hosted Headless CMS Tools:

- Cockpit

Hosted CMS:

- LightCMS [$]
- Surreal CMS [$]

Static CMS Tools:

- webhook.com
- Dato CMS
- siteleaf
- forestry.io

Back-end/API tools

Data/back-end as a service aka BAAS:

- Back& [free to $]
- Backendless
- Firebase [free to $]
- Pusher [free to $]
- restdb.io [free to $]
- MongoDB Stitch

Data/back-end

- GraphQL
  - Apollo
  - Relay
- Falcor
- RxDB

User Management as a Service:

- Auth0 [$]
- AuthRocket
- Okta

Search

- Algolia

Offline Tools

- Hoodie
- Offline.js
- PouchDB
- upup

NOTES:

For more tools look here.

Security Tools

Coding Tool:

- DOMPurify
- XSS

Security Scanners/Evaluators/Testers:

- Netsparker
- Websecurify
- OWASP ZAP

References:

- HTML5 Security Cheatsheet

Tasking (aka Build) Tools

Tasking/Build Tools: [1]

- Gulp

Opinionated Tasking/Build pipeline tools:

- Brunch

ADVICE:

[1] Before reaching for Gulp make sure npm scripts or yarn script won't fit the bill. Read, "Why I Left Gulp and Grunt for npm Scripts".

Deployment Tools

- Bamboo [$]
- Buddy [free to $]
- CircleCI [free to $]
- Codeship [free to $]
- Deploybot [free to $]
- Deployhq [free to $]
- FTPLOY [free to $]
- Now [free to $]
- Travis CI [free to $]
- Semaphore [free to $]
- Springloops [free to $]

Site/App Monitoring Tools

Uptime Monitoring:

- Monitority [free]
- Uptime Robot [free to $]

General Monitoring Tools:

- Pingdom [free to $]
- New Relic
- Uptrends [$]

JavaScript Error Reporting/Monitoring

- bugsnag [$]
- errorception [$]
- Honeybadger [$]
- Raygun [$]
- Rollbar [free to $]
- Sentry [free to $]
- TrackJS [$]

Performance Tools

Reporting:

- bundlephobia.com
- GTmetrix
- sitespeed.io
- Speed Curve [$]
- Web Page Test
- Sonarwhal
- Datadog [$]

JS Tools:

- imagemin
- ImageOptim-CLI

Budgeting:

- performancebudget.io

References/Docs:

- Jank Free
- Performance of ES6 features relative to the ES5

Checklist:

- The Front-End Checklist
- Front-End Performance Checklist 2018

Tools for Finding Tools

- built with
- frontendtools.com
- javascripting.com
- js.coach
- JSter
- microjs.com
- npms
- stackshare.io
- Unheap
- bestof.js.org