# FRONT End DEVELOPER Handbook 2019



Frontend Masters

www.dbooks.org

# Front-end Developer Handbook 2019

**Written by Cody Lindley**

*Sponsored by Frontend Masters, advancing your skills with in-depth, modern front-end engineering courses*

## Overview:

This is a guide that everyone can 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 2019.

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. This is currently the fourth year an edition has been released.

**What is in this Handbook**:

Chapter `0` provides a lite recap of the year in front-end development and what may be to come. Chapter `1` & `2` aim to give a brief overview of the discipline and practice of front-end development. Chapters `3` & `4` organize and recommend learning paths and resources. Chapter `5` organizes and list the tools used by front-end developers and Chapter `6` highlights front-end information outlets.

**Contribute content, suggestions, and fixes on github**:

www.dbooks.org

https://github.com/FrontendMasters/front-end-handbook-2019

# Chapter 0. Recap of 2018 and Looking Forward

## 0.1 — Recap of Front-end Development in 2018

- React had several notable releases this past year that included, lifecycle methods, context API, suspense, and React hooks.
- Microsoft buys Github. Yeah, that happened.
- Fonts created by CSS became a thing.
- What I used to call front-end driven apps, gets labeled "serverless". Unfortunately, this term is overloaded. However, the term JAMstack does seem to be resonating with developers.
- Google offered some neat tools this year to help make webpages load faster, i.e. squoosh and quicklink.
- Vue gets more Github stars than React this year. But React remains dominant in terms of use.
- A solution similar to React, without a virtual DOM or JSX, is introduced RE:DOM.
- Alternatives to NW.js and Electron show up, DeskGap and Neutralino.js.

- In 2017 the great divide between a front-end HTML & CSS developer v.s. front-end application developer is realized/verbalized. In 2018 that divide has grown wider and deeper and more people start to feel the divide.
- This year, like most recent years, was stock full 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. Radi.js, DisplayJS, Stimulus, Omi, Quasar.
- JavaScript frameworks start offering their own languages that compile to JavaScript (e.g. Mint).
- CodeSandbox evolves to become the dominant solution for online code sharing.
- CSS Grid and CSS Flexbox are fully supported in modern browsers and get taken for some serious rides. But many are left wondering when to use which one and how.
- Many realize the long terms costs of bolted on type systems (e.g. TypeScript and Flow). Some concluded bolted on systems are not unlike bolted on module systems (i.e. AMD/Require.js) and come with more issues than solutions. Minimally, many developers realize that if types are needed in large code bases, that bolted on systems are not ideal in comparison to languages that have them baked in (e.g. Reason, Purescript, Elm).
- CSS Variables gain browser support among modern web browsers
- The flavors of CSS in JS exploded and some question the practice.
- ES modules are now usable in modern browsers and dynamic imports are close behind. We are even seeing a shift in tooling around this fact.
- Many realize that end to end testing is the starting point of doing tests correctly in large part due to Cypress (i.e. Cypress first, then Jest).
- While Webpack was heavily used again this year, many developers found Parcel to be easier to get up and running.
- One of the most important questions asked this year was, what is the cost of JavaScript.
- Babel 7 was released this year. That's a big deal because the last major release was almost three years ago.

www.dbooks.org

- The reality of too much JavaScript change too fast is realized and people start talking about what you need to know before you can even learn something like React. The fight is real.
- Most developers found GraphQL, via Apollo, and see it as the next evolution for data API's.
- Gulp and friends definitely took a back seat to NPM/Yarn run. But this did not stop Microsoft from getting in the game with Just.
- This year, one can not only lint/hint HTML, CSS, and JavaScript they can lint/hint the web itself.
- The 2018 Front-End Tooling survey is worth reading if only to realize just how much jQuery is still used.
- It can't be denied TypeScript gained a lot of users this year.
- VScode, dominates as the code editor of choice.

## 0.2 — In 2019, Expect...

- Hopefully, more of this to come. "Stepping away from Sass".
- Still a good idea to keep an eye on and learn about the up coming additions (and potential additions) to CSS via https://cssdb.org
- The WebP image format from Google could reach support from all modern browsers this year.
- Prepack will continue to cook.
- GraphQL will continue to gain massive adoption.
- The, "State of JavaScript" survey authors will add a "State of CSS" survey in 2019.
- Keep an eye on Web Animations API.
- Someone you know will try and convince you to use TypeScript.

- Babel will get some competition from swc-project.
- The case for, JAMStack's will continue.
- Chasing the one code base to many platforms will continue.
- More developers will turn to languages like ReasonML over JavaScript/TypeScript for large code bases.
- More, largely used projects will start to shed jQuery in favor of native DOM solutions.
- Web Components! At this point, I have no idea how Web Components will play out. Reality is they are not going away, and they have not gained a lot of momentum/usage once the hype ended.

# Chapter 1. What Is a Front-end Developer?

This chapter provides a baseline explanation for front-end development and the front-end developer discipline.

> *Front-end web development, also known as client-side development is the practice of producing HTML, CSS and JavaScript for a website or*

www.dbooks.org

*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*

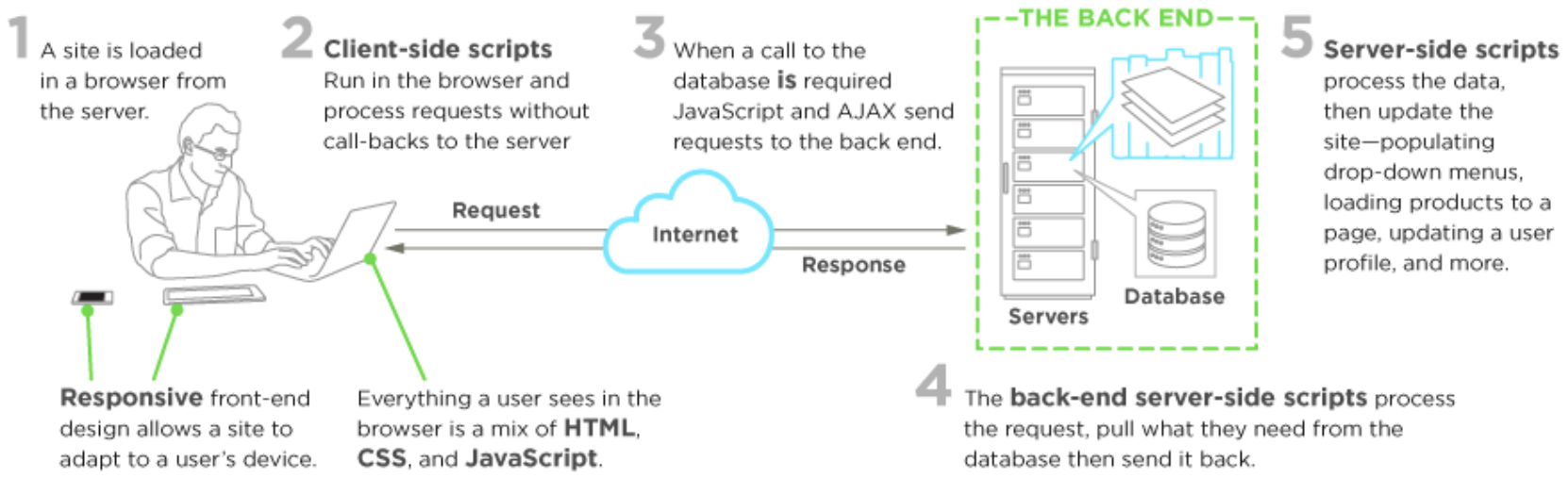



*Image source: https://www.upwork.com/hiring/development/front-end-developer/*

## A Front-end Developer...

A front-end developer architects and develops websites and web applications using

web technologies (i.e., HTML, CSS, and JavaScript), which typically runs on the Open Web Platform or acts as compilation input for non-web platform environments (i.e., React Native).

A person enters into the field of front-end development by learning to build a website or web application which relies on HTML, CSS, and JavaScript and commonly runs in a 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 programmatically from the

www.dbooks.org

command line that can retrieve and traverse web page code.

The most common headless browsers are:

- Headless Chromium
- Zombie
- slimerjs
- puppeteer

**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 (i.e. web platform). 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
- NativeScript

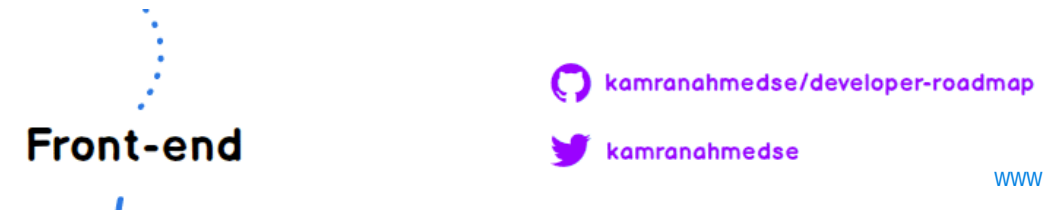
> **Notes:**
>
> 1. Make sure you are clear what exactly is meant by the "web platform". Read the, "Open Web Platform" Wikipedia page. Explore the many technologies that make up the web platform.

# Chapter 2. The Practice of Front-end Development: Overview

This chapter will break down and broadly describes the practice of front-end engineering starting with, "How Front-End Developers Are Made".

## 2.1 - How Front-End Developers Are Made

How exactly does one become a front-end developer? Well, it's complicated. Just consider this road map:




www.dbooks.org

## Learn the Basics

- Learn the basics
- Writing Semantic HTML
- Basic SEO
- Accessibility

**HTML**

**CSS**

- Learn the basics
- Making Layouts
- Media Queries
- Learn CSS 3

Floats
Positioning
Display
Box Model
CSS Grid
Flex Box

**Basics of JavaScript**

- Syntax and Basic Constructs
- Learn DOM Manipulation
- Learn Fetch API / Ajax (XHR)
- ES6+ and modular JavaScript

Understand the concepts Hosting, Event Bubbling, Scope Prototype, Shadow DOM, strict, how browsers work, DNS, HTTP

**Package Managers**

- npm
- yarn

npm improved a lot, post v5+, but is still behind yarn in some features; nothing serious though. Pick any!

**CSS Pre-processors**

- SASS
- PostCSS
- Less

PostCSS is not a pre-processor but can be used as one. Go for SASS and revisit PostCSS later. There is still some Less in the market but I won't go for it if I was starting in 2019.

**CSS Frameworks**

- Bootstrap
- Materialize CSS
- Bulma
- Semantic UI

**CSS Architecture**

- BEM
- OOCSS
- SMACSS

With modern front-end frameworks, there is more push push towards CSS in JS methodologies with which you are not going to need these. However, you should still learn BEM at-least, which would prove helpful in the long run

**Build Tools**

**Task Runners**

- npm scripts
- gulp

**Linters and Formatters**

- Prettier
- ESLint
- JSHint
- JSLint
- JSCS

**Module Bundlers**

- Webpack
- Parcel
- Rollup

- Redux
- MobX

These are not specific to React though, you can use them in any framework or app

**Pick a Framework**

- React.js
- Angular
- Vue.js

- RxJS
- ngrx

- Vuex

Before you start this, you should have a good understanding of what single page applications are how they work and what are some of the positive and

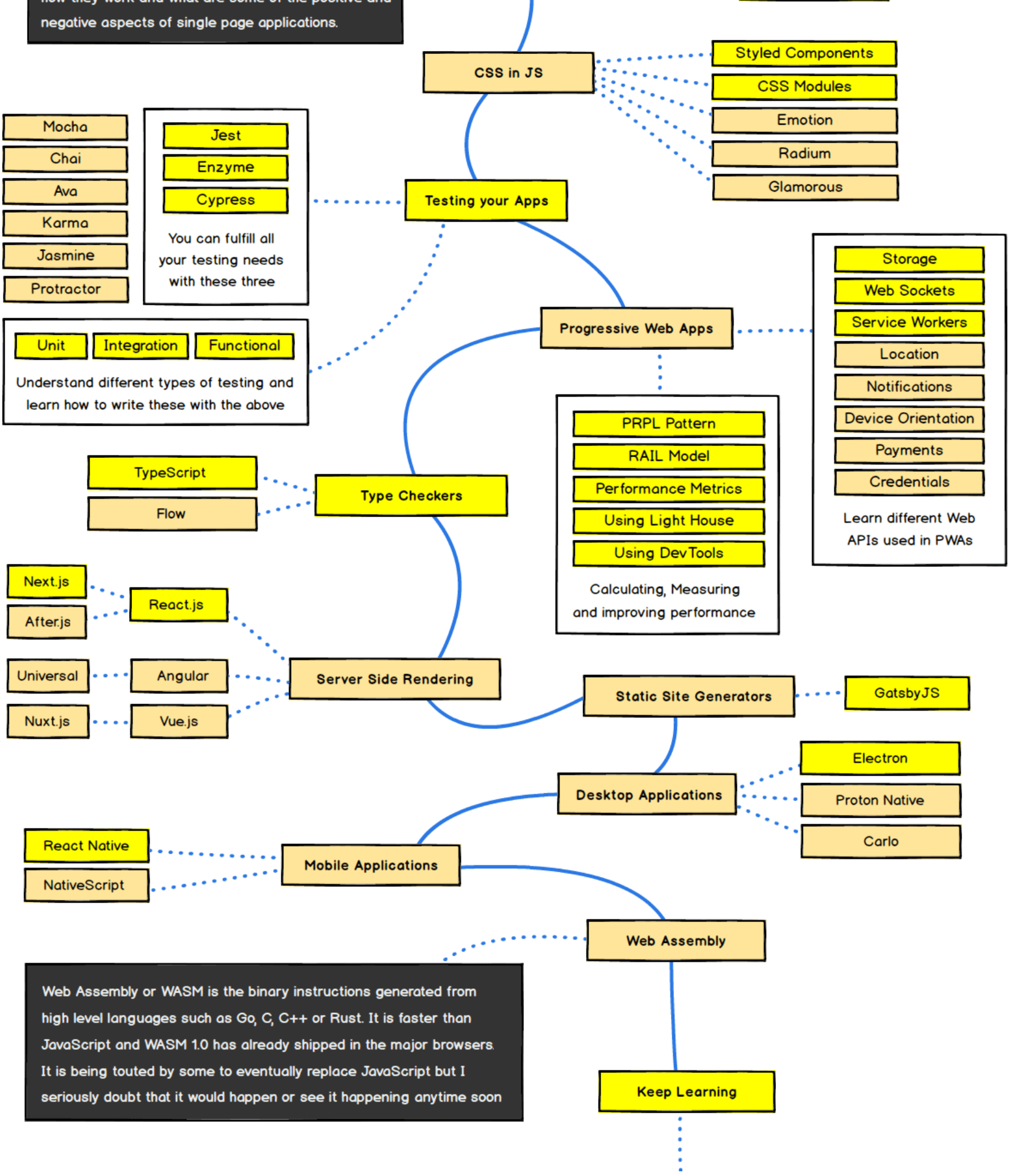



*Image source: https://github.com/kamranahmedse/developer-roadmap*

Today, in general, one 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

www.dbooks.org

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 space from design or 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 (Chapter 3 and Chapter 4 will dive into more details on learning resources).

1. Learn, roughly, how the web platform works. Make sure you know the "what" and "where" of HTML, CSS, DOM, JavaScript, Domains, DNS, URLs, HTTP, browsers, and servers/hosting. Don't dive deep on anything just yet, just aim to understand the parts at play and how they loosely fit together. Start by building simple web pages.
2. Learn HTML
3. Learn CSS
4. Learn JavaScript
5. Learn DOM
6. Learn the fundamentals of user interface design (i.e. UI patterns, interaction design, user experience design, and usability).
7. Learn CLI/command line
8. Learn the practice of software engineering (i.e., Application design/architecture, templates, Git, testing, monitoring, automating, code quality, development methodologies).
9. Get opinionated and customize your tool box with whatever makes sense to your brain (e.g. Webpack, React, and Mobx).
10. 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 JSX, learn HTML. Don't learn TypeScript, 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 low cost to free, and hold you accountable for learning it, you should consider a traditional instructor lead class room setting. Otherwise, I am not aware of any other profession that is practically free for the taking with an internet connection, a couple of 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:

- Getting started with the Web [read]
- So, You Want to be a Front-End Engineer [watch]
- Frontend Masters Learning Paths [watch][$]
- Introduction to Web Development [watch][$]
- Treehouse Techdegree [watch][$]
- Front-End Web Developer Nanodegree [watch][$]
- Become a Front-End Web Developer [watch][$]
- freeCodeCamp [interactive][watch]

When getting your start, you should fear most things that conceal complexity.

www.dbooks.org

Abstractions (e.g. jQuery) 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.

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.

## 2.2 - Front-End Job Titles

A great divide has been brewing in the front-end developer space for several years between two very different types of so-called front-end developers. On the one side, you have JavaScript-focused programmers who write JavaScript for front-end runtimes that likely have computer science skills with a software development history. They more than likely view HTML and CSS as an abstraction (i.e. JSX and CSS in JS). On the other side, you have, most likely, non-computer science educated developers who focus on HTML, CSS, and JavaScript as it specifically pertains to the UI. In 2019, when entering or trying to understand the front-end developer space you will absolutely feel this divide. The term front-end developer is on the verge of meaninglessness without clarifying words to address what type of front-end developer is being discussed.

Below is a list and description of various front-end job titles (Keep in mind titles are hard). 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 computer science knowledge 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 software 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 possess front-end skills (i.e., HTML & CSS) but also professional design (Visual Design and Interaction Design) skills.

**UI (User Interface) 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).

www.dbooks.org

**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 quality.

**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:**

1. 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.

## 2.3 - Baseline Web Technologies Employed by Front-End Developers




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 or MDN Web APIs.

The nine technologies just mentioned are defined below along with a link to the

www.dbooks.org

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 5.3 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*

www.dbooks.org

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:

- DOM Living Standard
- W3C DOM4
- UI Events

**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*

www.dbooks.org

> *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® 2018 Language Specification
- All ECMAScript Language Specifications

**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)**

> *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).*

www.dbooks.org

— *Wikipedia*

- Web Accessibility Initiative (WAI)
- Web Content Accessibility Guidelines (WCAG) Current Status

## 2.4 - Potential Front-end Developer Skills




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

A basic to advanced understanding of HTML, CSS, DOM, JavaScript, HTTP/URL, and web browsers is assumed for any type of professional front-end developer role.

Beyond the skills just mentioned, a front-end developer might also be specifically 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

www.dbooks.org

- Scalable Vector Graphics (aka SVG)
- Regular Expressions
- 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
- Web/Browser Security
- Browser Developer Tools

## 2.5 - 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 (i.e. MacOS)
- Ubuntu (or some flavor of Linux)
- 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.)

www.dbooks.org

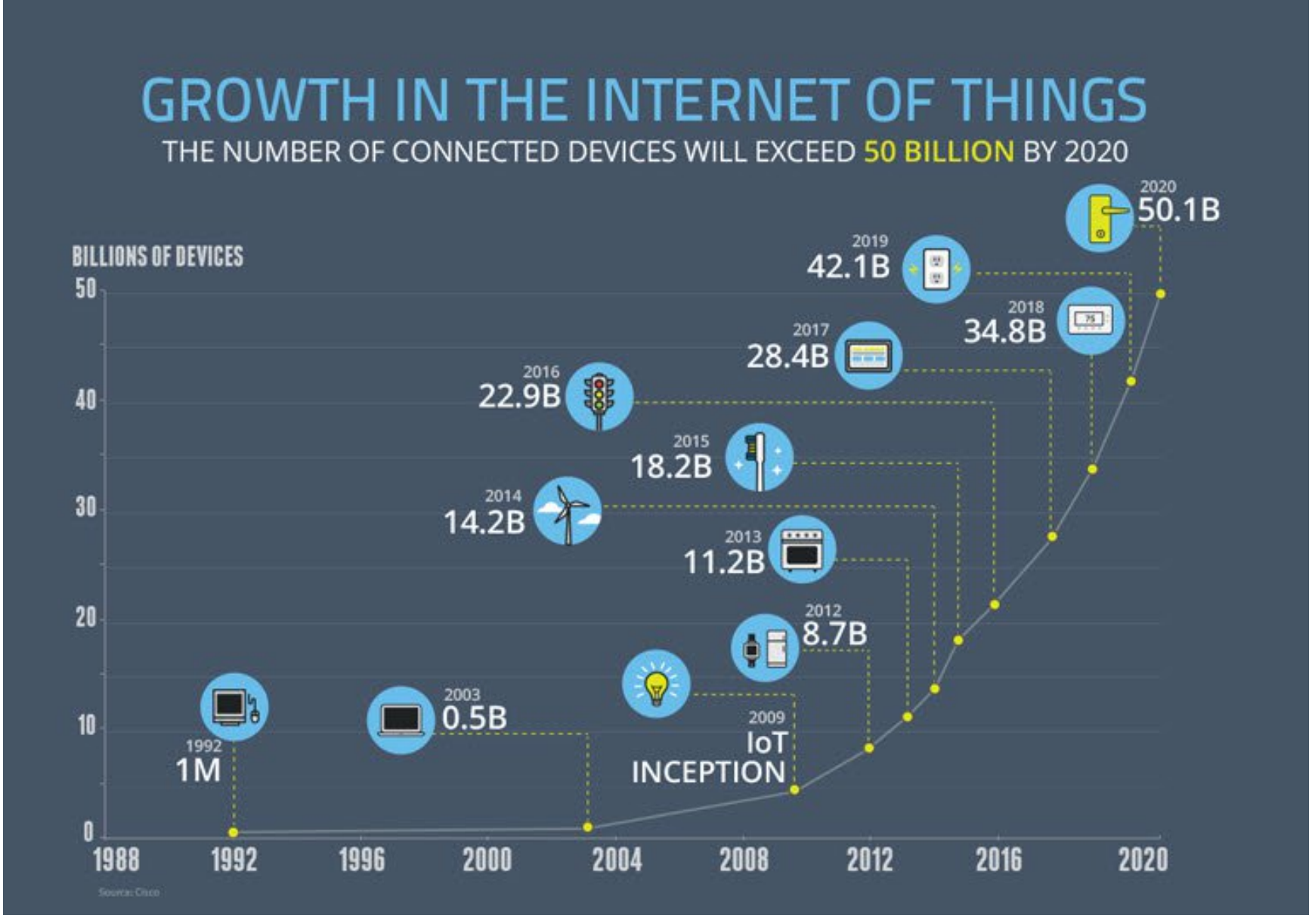



*Image source: https://www.enterpriseirregulars.com/104084/roundup-internet-things-forecasts-market-estimates-2015/*

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: Headless Chromium).
- A WebView/browser tab (think iframe) embedded within a native application as a runtime with a 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)

# 2.6 - 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 the device)
- Back-End Developer (i.e., writes code that runs on the 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
- Performance Engineers
- API Developers
- Database Administrators

www.dbooks.org

- QA Engineers / Testers

## 2.7 - Generalist/Full-Stack Myth




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 might lead one to believe that being a full-stack developer is commonplace. But, in my almost 20 years of experience, this is anything but the case in a professional context.

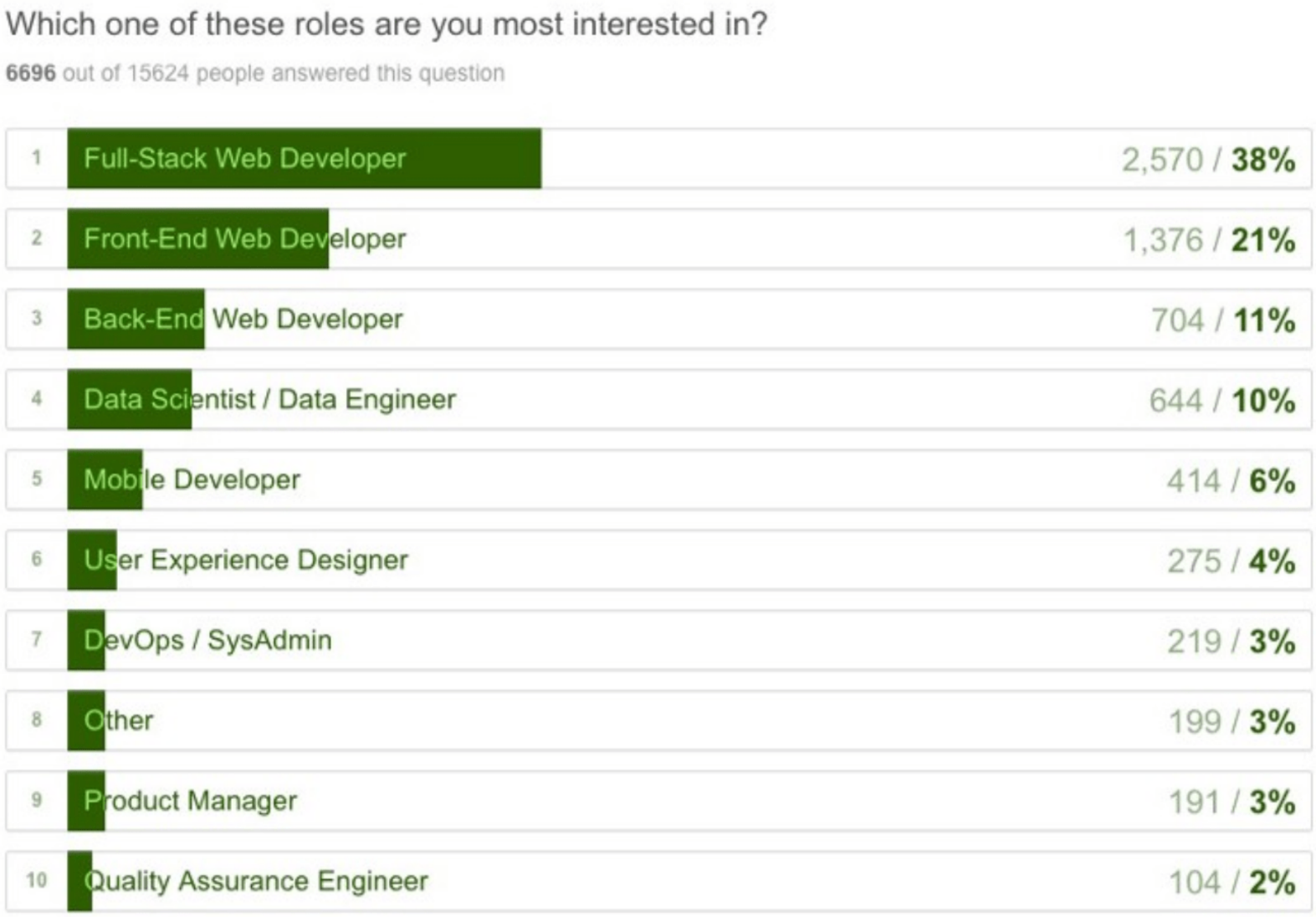
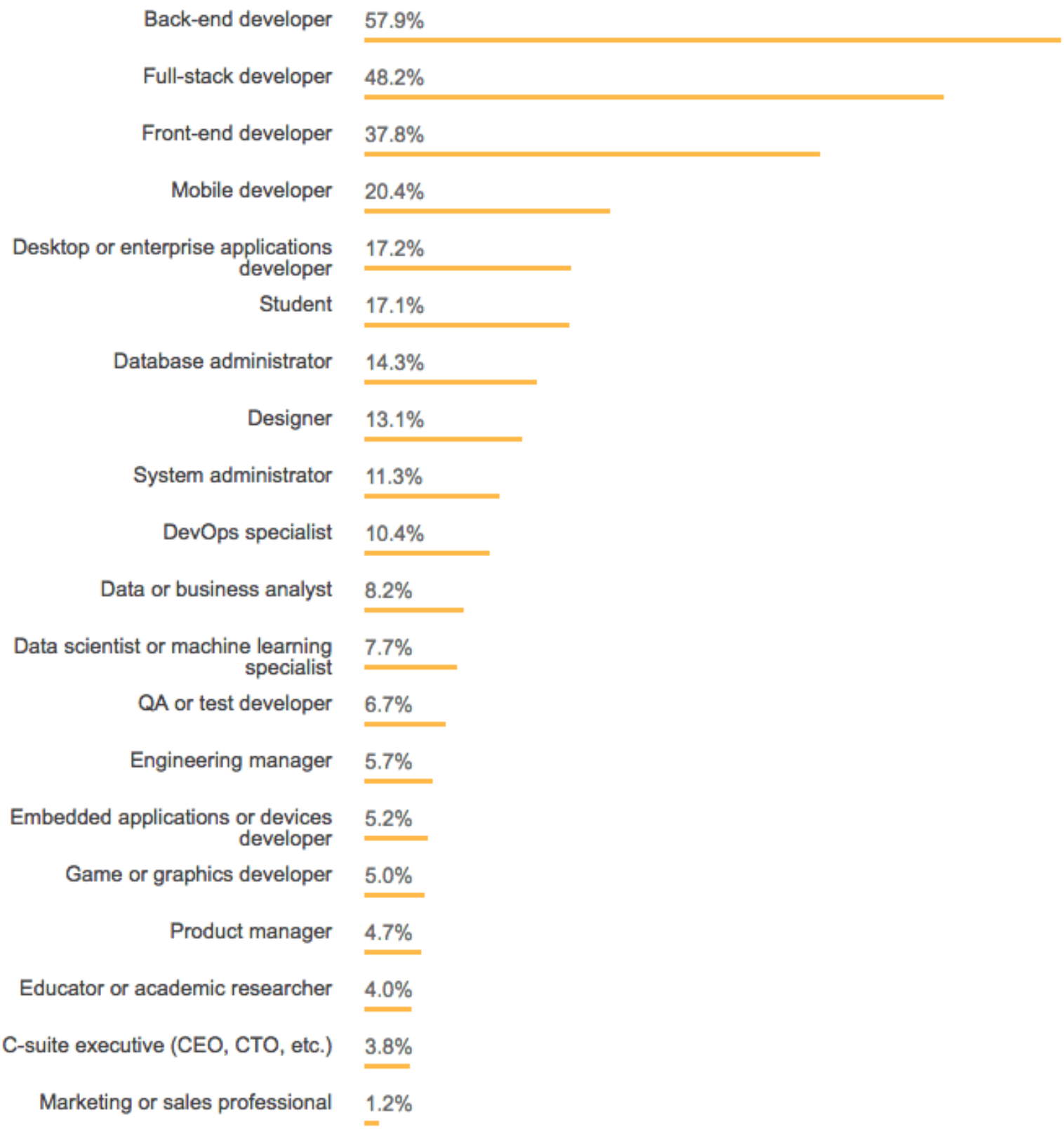



*Image source: https://medium.freecodecamp.com/we-asked-15-000-people-who-they-are-and-how-theyre-learning-to-code-4104e29b2781#.ngcpn8nlz*

www.dbooks.org

# Developer Roles

## Developer Type




*92,098 responses; select all that apply*

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

The roles to design and develop a website or web application require a deep set of skills 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.

However, given that JavaScript has infiltrated all layers of a technology stack (i.e. Node.js) 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.

## 2.8 - Front-End Interviews

**Preparing:**

- Preparing for a Front-End Web Development Interview in 2017
- Cracking the front-end interview
- Front End Interview Handbook
- Decoding the Front-end Interview Process

www.dbooks.org

**Quiz's:**

- Front End Web Development Quiz
- JavaScript Web Quiz

**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
- The Best Frontend JavaScript Interview Questions (written by a Frontend Engineer))

**Questions you ask:**

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

## 2.9 - 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.

- authenticjobs.com
- careers.stackoverflow.com
- css-tricks.com/jobs
- frontenddeveloperjob.com
- glassdoor.com
- jobs.github.com

- linkedin.com
- remote.co
- weworkremotely.com
- www.smashingmagazine.com/jobs/

> **Notes:**
>
> 1. Want to work remotely as a front-end developer checkout these remote-friendly companies.

## 2.10 - Front-End Salaries

The national average in the U.S for a mid-level front-end developer is somewhere between $65k and 100k.

Of course when you first start expect to enter the field at around 40k depending upon location and experience.

> **Notes:**
>
> 1. 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).

# Chapter 3. Learning Front-end Dev: Self

www.dbooks.org

# Directed Resources/Recommendations

This chapter highlights the many 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 identified (articles, books, videos, screencasts etc..) will include both free and paid material. Paid material will be indicated with [$].

## 3.1. - 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*

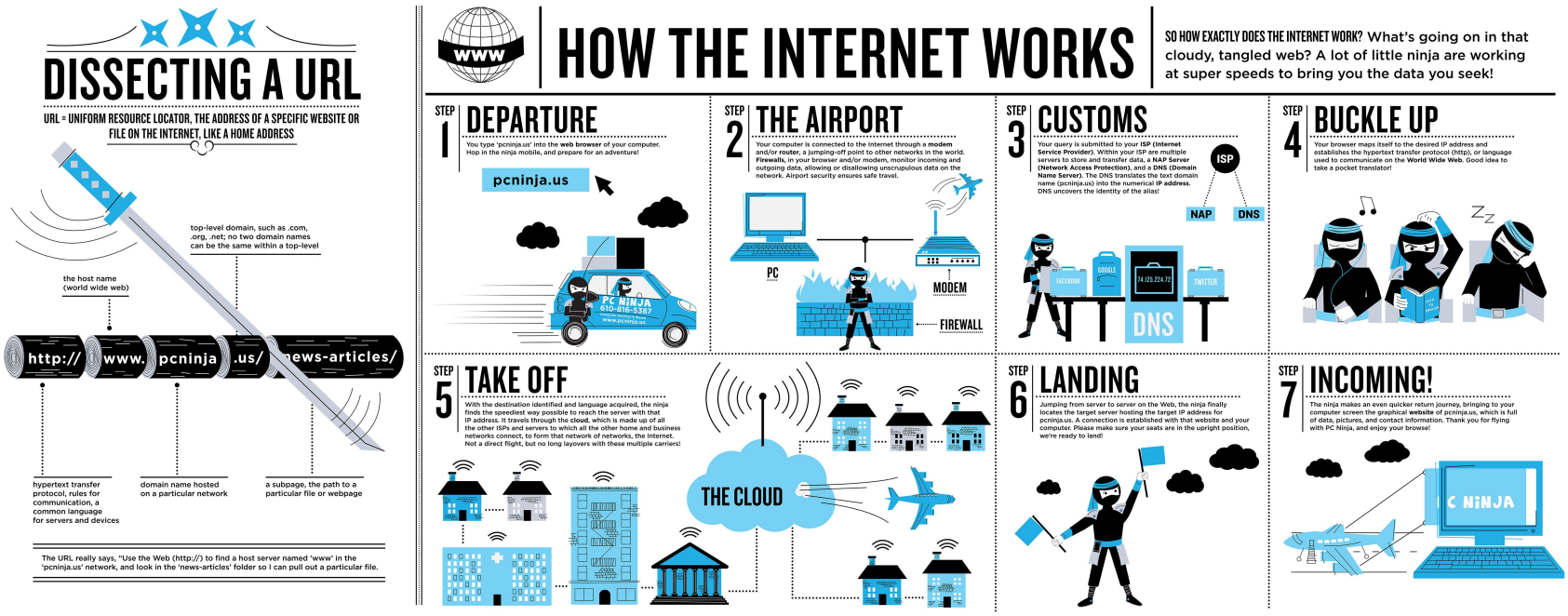



*Image source: https://www.helloitsliam.com/2014/12/20/how-the-internet-works-infographic/*

- What is the Internet? [watch]
- Internet Fundamentals [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

www.dbooks.org

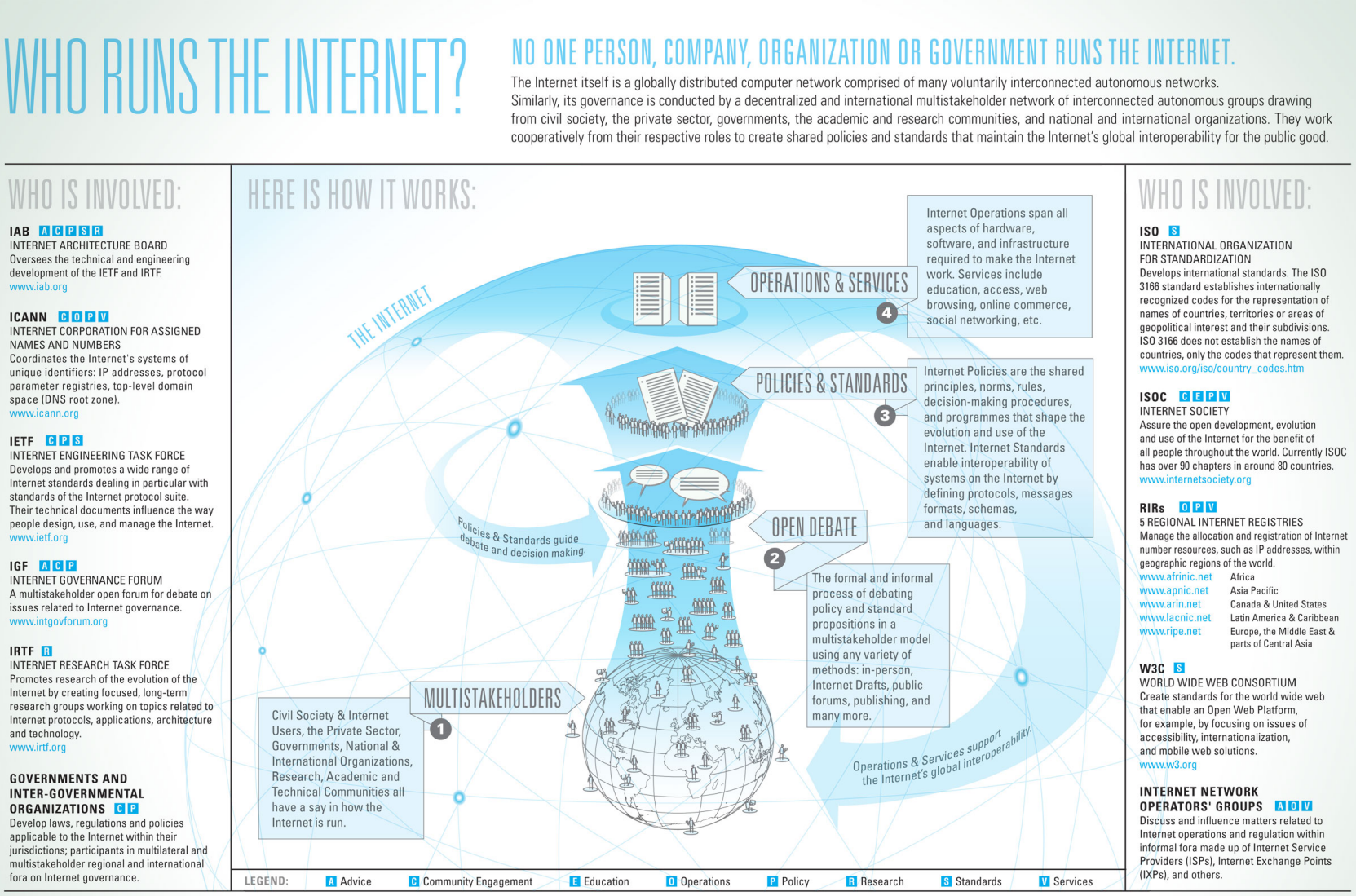



*Image source: http://www.bitrebels.com/technology/find-out-who-runs-the-internet-chart/*

## 3.2. - 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 desktop and mobile) are:**

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

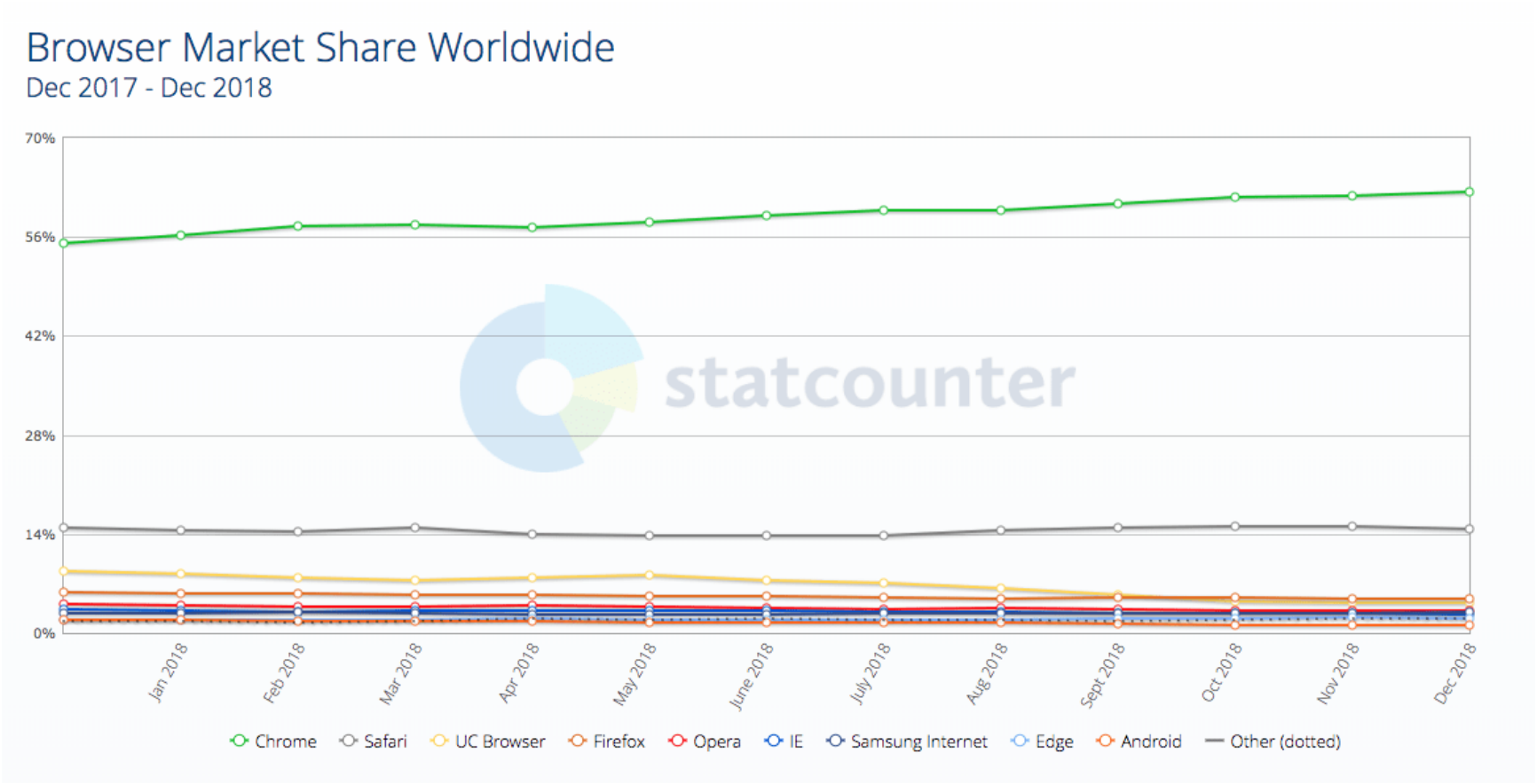



*Image source: http://gs.statcounter.com/browser-market-share*

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

- evolutionoftheweb.com [read]

www.dbooks.org

- Timeline of web browsers [read]

**The Most Commonly Used Headless Browser Are:**

- Headless Chromium (engine: Blink + V8)
- SlimerJS (engine: Gecko + SpiderMonkey)

**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]

**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., React, Webpack, 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 supposed 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**

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.

## 3.3 - 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*

www.dbooks.org

> *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*

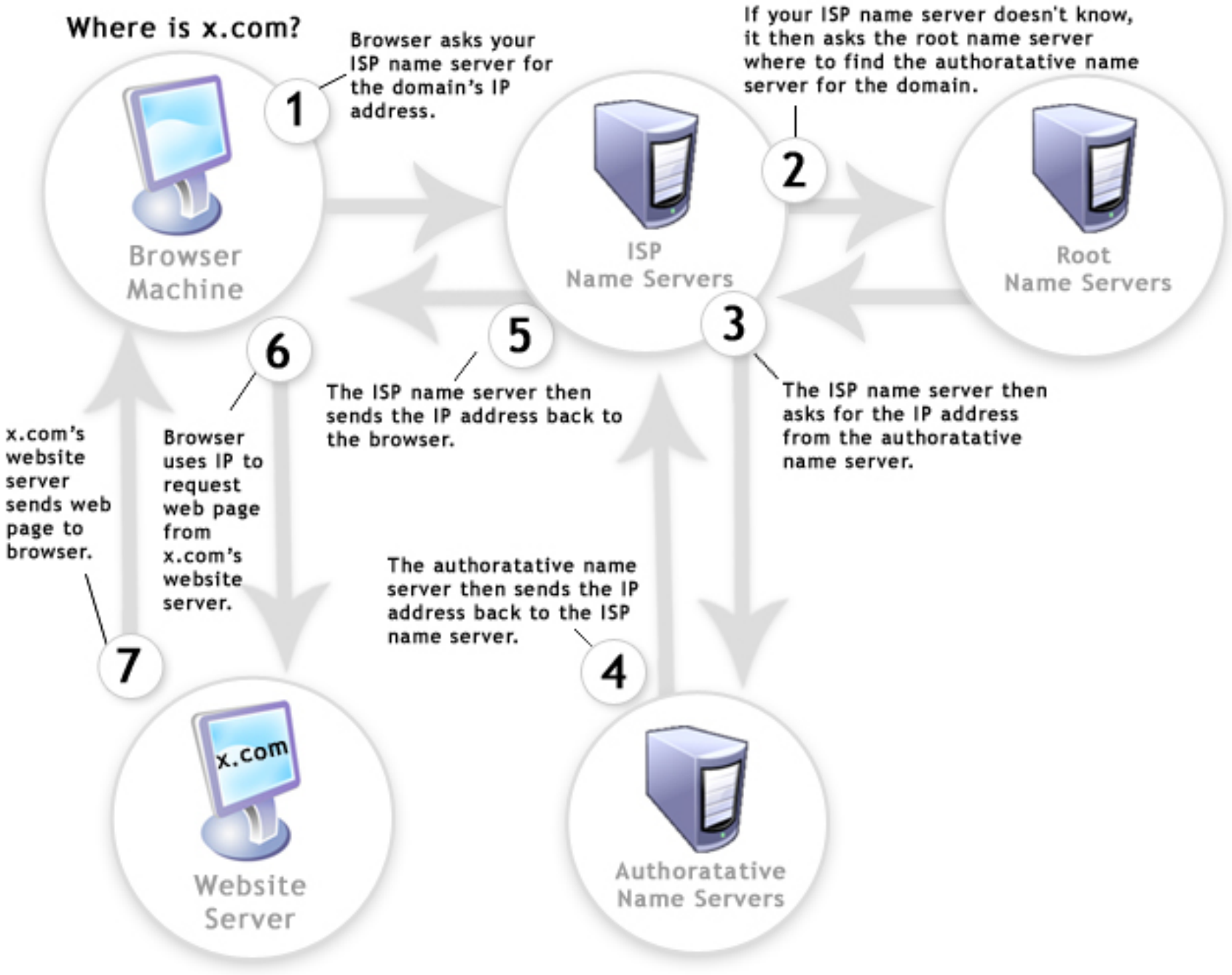



*Image source: http://www.digital-digest.com/blog/DVDGuy/wp-content/uploads/2011/11/how_dns_works.jpg*

- 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]

## 3.4 - 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*

### 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]

www.dbooks.org

- 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*** *- 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*

### CORS Specifications

- Cross-Origin Resource Sharing

### CORS

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

> ***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*

### WebSockets

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

## 3.5 - Learn Web Hosting

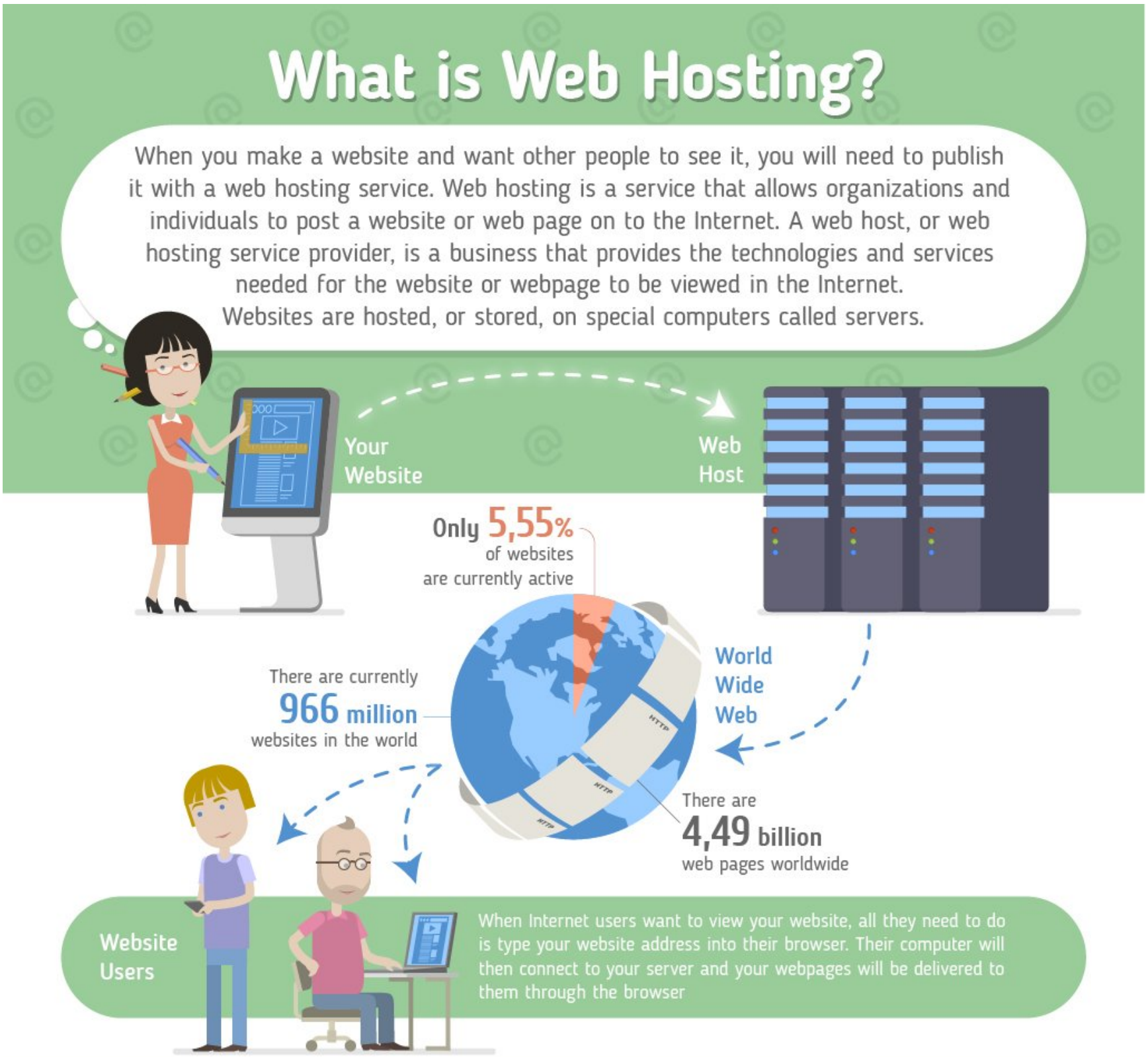
> *A web hosting service is a type of Internet hosting service that allows individuals and organizations to make their website accessible via the World Wide Web. Web hosts are companies that provide space on a server owned or leased for use by clients, as well as providing Internet*

www.dbooks.org

> *connectivity, typically in a data center.*
>
> — *Wikipedia*

**General Learning:**

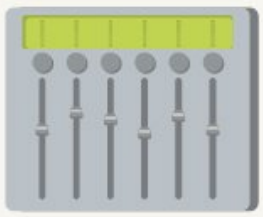
- Web Hosting 101: Get Your Website Live on the Web in No Time [video]





### Resources

Check the features you are getting:
- Disk space & bandwidth
- Number of email accounts
- Programming languages (PHP, ASP.NET, Java)
- Databases (MySQL, MS SQL, Access)
- CPU and RAM allocations



### A Control Panel

This allows you to manage your hosting and website. Make sure it is easy to use. The most popular and widely used web hosting control panels are cPanel and vDeck



### Security Features

Websites must be kept secure so they aren't hacked. Your data must be kept secure at all times too



### FTP Access

It's the basic technology used to upload and download files, so a web host must provide this capability at all times



### Website Builder

Most web hosts include some kind of site building tool that makes it easy to get a website online even without any technical skills




**You will also find that good web hosts usually prepare a good amount of tutorials and a knowledge base to which you can refer**

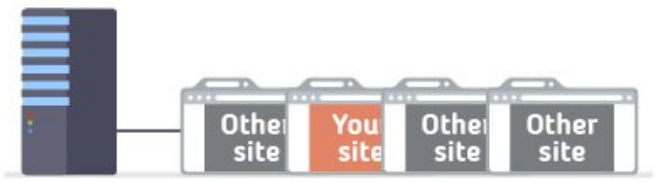



## Domain Name

Before we jump into the different types of web hosting, it's important to remember you have to have a domain name so your website can be found on the Internet. The domain name is essentially your company's Internet address; the same way your company's str eet address is used to find your offices, a domain name is used to find your website. Considering this, pick a domain name that's memorable and relevant!

## Types Of Web Hosting

As technology has progressed different types of web hosting have appeared to meet the different needs of websites and customers




### Shared hosting

- Economical and the most basic web hosting option
- Limited bandwidth, disk space and control offered
- Best for start-ups to maintain low-moderate incoming visitor traffic
- Easily available at $0,95- $2,95/m




### VPS hosting

- Amalgamation of dedicated server plus Shared hosting
- Gives root access to the virtual private server
- Dedicated space and resources to each VPS client
- Suitable for companies receiving moderate visitor traffic

www.dbooks.org

- Suitable for companies receiving moderate visitor traffic
- Easily available at $17,9- $19,9/m



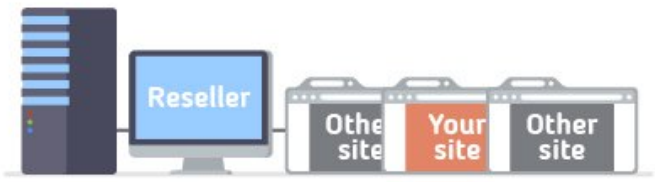

## Dedicated hosting

- Robust, scalable & flexible web hosting option with complete administrative access
- Efficiently manage traffic spikes
- Best for companies having security and compliance requirements
- Easily available at $49,95- $53,9/m




## Cloud hosting

- Highly scalable solution to meet growing business needs
- Works on pay-per-use pricing model
- Scalable traffic management
- Best for SMBs and large enterprises
- Easily available at $26,54/m




## Reseller hosting

- Powerful control panels along with Domain/FTP features
- Aid in brand building
- Aid in sharing your server space with others
- Easily available at $19,92- $22,9/m

# 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
How long has the company been in business?

### Reputation
Check online reviews to find out how satisfied customers are

### Location
Choose a host that's located in the same country as your web audience. This will give you Search Engine Optimization benefits and loads faster

### Support
All web hosts offer support, but it's best to choose one that has 24/7 support as you never know when you might need help. It's also helpful if they offer support in a number of channels, such as phone, email and live chat

### Cost
Shouldn't be your only

### Money Back Guarantee
Hopefully you won't want your money back, but this offer shows the company has confidence in its services

### Uptime
Does the web host guarantee a certain amount of uptime? This refers to how often your site is actually online, which should be close to 100%!

### Extras
Look for extras such as SimpleScripts Installer, free doname name, free advertising credits

### Backups
Look for a web host which will help you to backup your data regularly

### Terms & Conditions
Always read this! Make sure you

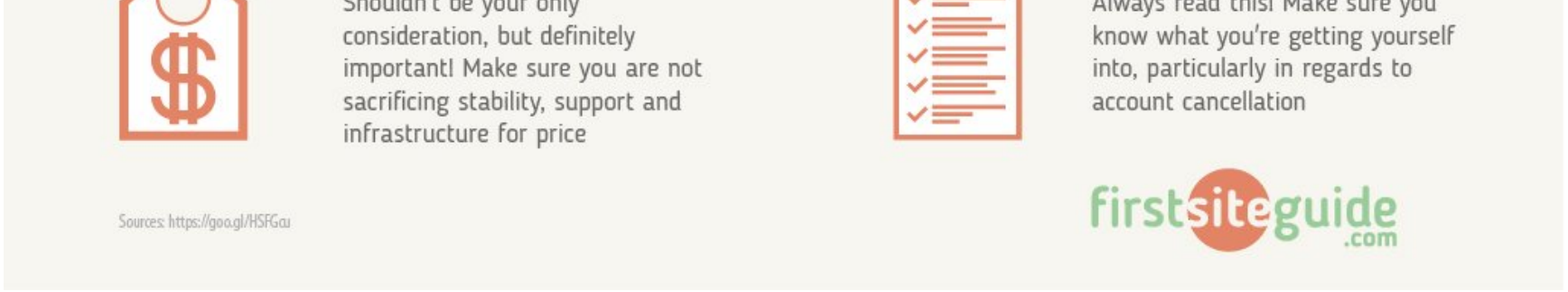



*Image source: https://firstsiteguide.com/wp-content/uploads/2016/06/what-is-web-hosting-infographic.jpg*

# 3.6 - Learn General Front-End Development

- Frontend Bootcamp / Days in the Web [read]
- Becoming a Career-Ready Web Developer
- Become a Front-End Web Developer [watch][$]
- Being a web developer [read]
- freeCodeCamp [interact]
    - learning front-end development during #100DaysOfCode [read]
- Front-End Web Developer Nanodegree [watch][$]
- 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][$]
- Front-End Web Development: The Big Nerd Ranch Guide [read][$]
- Complete Intro to Web Development [watch][$]
- Learn Front End Web Development [watch][$]
- So, You Want to Be a Front-End Engineer [watch]

www.dbooks.org

- codecademy.com: Web Development Path [interact][free to $]
- web.dev [read]

## 3.7 - 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][$]

## 3.8 - Learn HTML & CSS

www.dbooks.org

> ***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]
- CSS Positioning [watch][$]
- Introduction to Web Development (v2) [watch][$]

- 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 Grids and Flexbox for Responsive Web Design [watch][$]
- CSS Diner [interact]
- CSS Selectors from CSS4 till CSS1 [read]
- CSS Secrets: Better Solutions to Everyday Web Design Problems [read][$]
- CSS3 [read]

www.dbooks.org

- CSS In-Depth, v2 [watch][$]
- What the Flexbox?! A Simple, Free 20 Video Course That Will Help You Master CSS Flexbox [watch]
- 30 Seconds of CSS - A curated collection of useful CSS snippets you can understand in 30 seconds or less. [read]

**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]

www.dbooks.org

# 3.9 - 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*

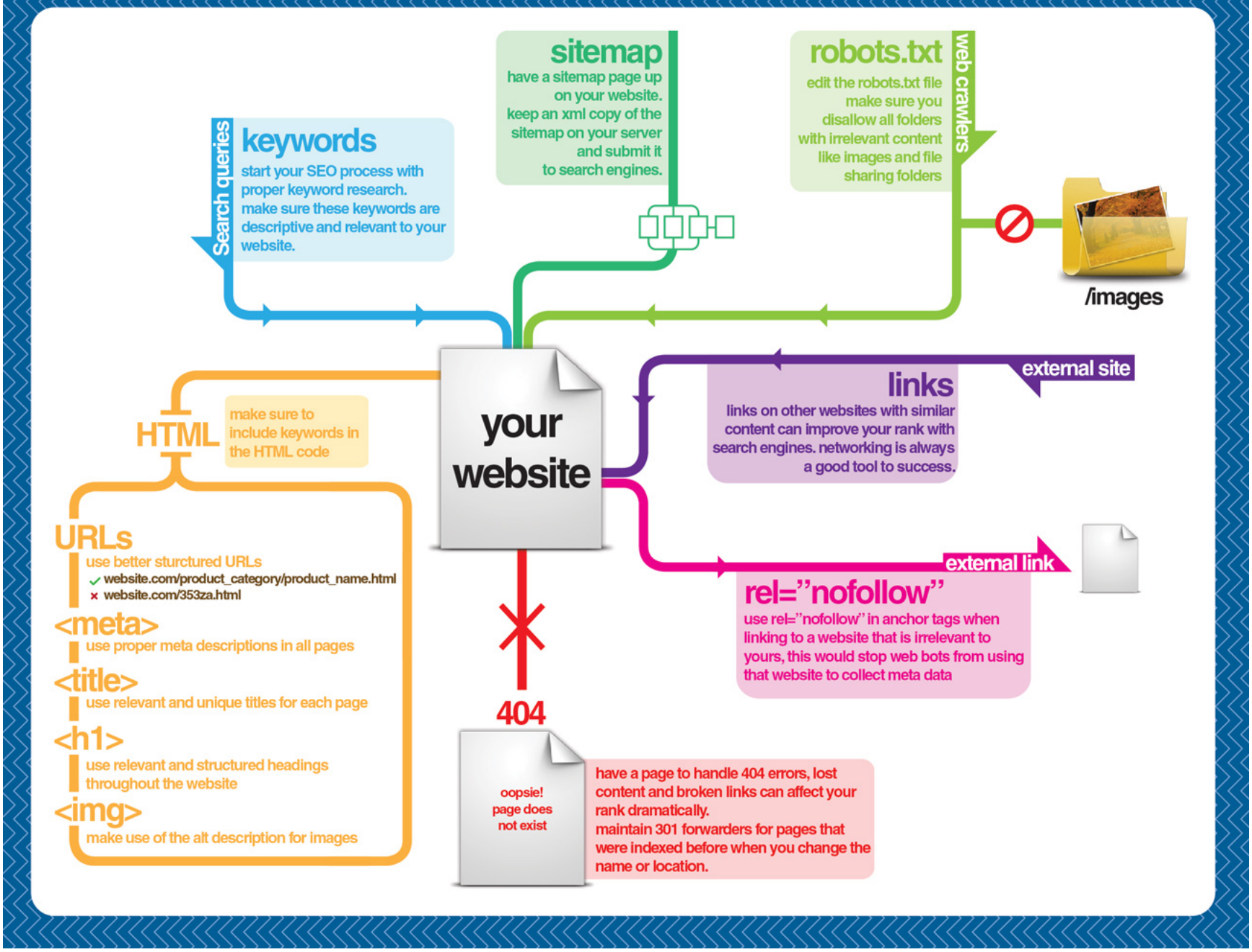



*Image source: https://visual.ly/community/infographic/computers/how-does-seo-work*

**General Learning:**

- Google Search Engine Optimization Starter Guide [read]
- Modern SEO [watch][$]
- SEO Fundamentals From David Booth [watch][$]
- SEO Fundamentals From Paul Wilson [watch][$]
- SEO Tutorial For Beginners in 2016 [read]
- SEO for Web Designers [watch][$]

www.dbooks.org

# 3.10 - 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:**

- MDN: JavaScript [read]
- javascript.info
- JavaScript Enlightenment [read]
- Eloquent JavaScript [read]

**General Learning:**

- Speaking JavaScript [read]
- JavaScript for impatient programmers [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]
- 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]
- JavaScript: The Hard Parts [watch][$]
- Deep Foundations of JavaScript (v3) [watch][$]

**Mastering:**

- Setting up ES6 [read]
- ES6 FOR EVERYONE! [watch][$]
- Exploring ES6 [read]
- You Don't Know JS: ES6 & Beyond [read]
- Understanding ECMAScript 6: The Definitive Guide for JavaScript Developers [read] [$]
- JavaScript: The Recent Parts [watch][$]
- Exploring ES2016 and ES2017 [read]
- Exploring ES2018 and ES2019 [read]
- JavaScript Regular Expression Enlightenment [read]
- Using Regular Expressions [watch][$]
- You Don't Know JS: Async & Performance [read]
- JavaScript with Promises [read][$]
- Test-Driven JavaScript Development [read][$]
- JS MythBusters [read]
- Robust JavaScript [read]

www.dbooks.org

- JavaScript Algorithms and Data Structures [read]
- 33 Concepts Every JavaScript Developer Should Know [read]
- doesitmutate.xyz [read]

**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][$]
- Functional-Lite JavaScript (v2) [watch][$]
- Hardcore Functional Programming in JavaScript [watch][$]

**References/Docs:**

- MDN JavaScript Reference
- MSDN JavaScript 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
- ECMAScript® 2019 Language Specification
- Status, Process, and Documents for ECMA262

**Style:**

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

**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][$]
- The Good Parts of JavaScript and the Web [watch][$]

www.dbooks.org

- High Performance JavaScript (Build Faster Web Application Interfaces) [read][$]

**JS Explorers/Visualizers:**

- JavaScript Array Explorer
- JavaScript Object Explorer
- JavaScript Visualizer

## 3.11 - Learn DOM, BOM, CSSOM & 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:**

- 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][$] or read online for free
- Fixing Common jQuery Bugs [watch][$]

www.dbooks.org

- 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)
- CSS Object Model (CSSOM)

**Standards/Specifications:**

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

# 3.12 - Learn Web Animation

**General Learning:**

- SVG Essentials and Animation, v2 [$][watch]
- Adventures in Web Animations [$][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

## 3.13 - 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*

www.dbooks.org

> *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 Web Typography, v2 [watch][$]
- Typography for the Web [watch][$]

**Icons:**

- [read] [watch]

**Images:**

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

## 3.14 - 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.*

www.dbooks.org

> *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][$]
- How HTML elements are supported by screen readers [read]
- Introduction to Web Accessibility - WAI [read]
- Universal Design for Web Applications: Web Applications That Reach Everyone [read][$]
- 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

# 3.15 - Learn Web/Browser APIs

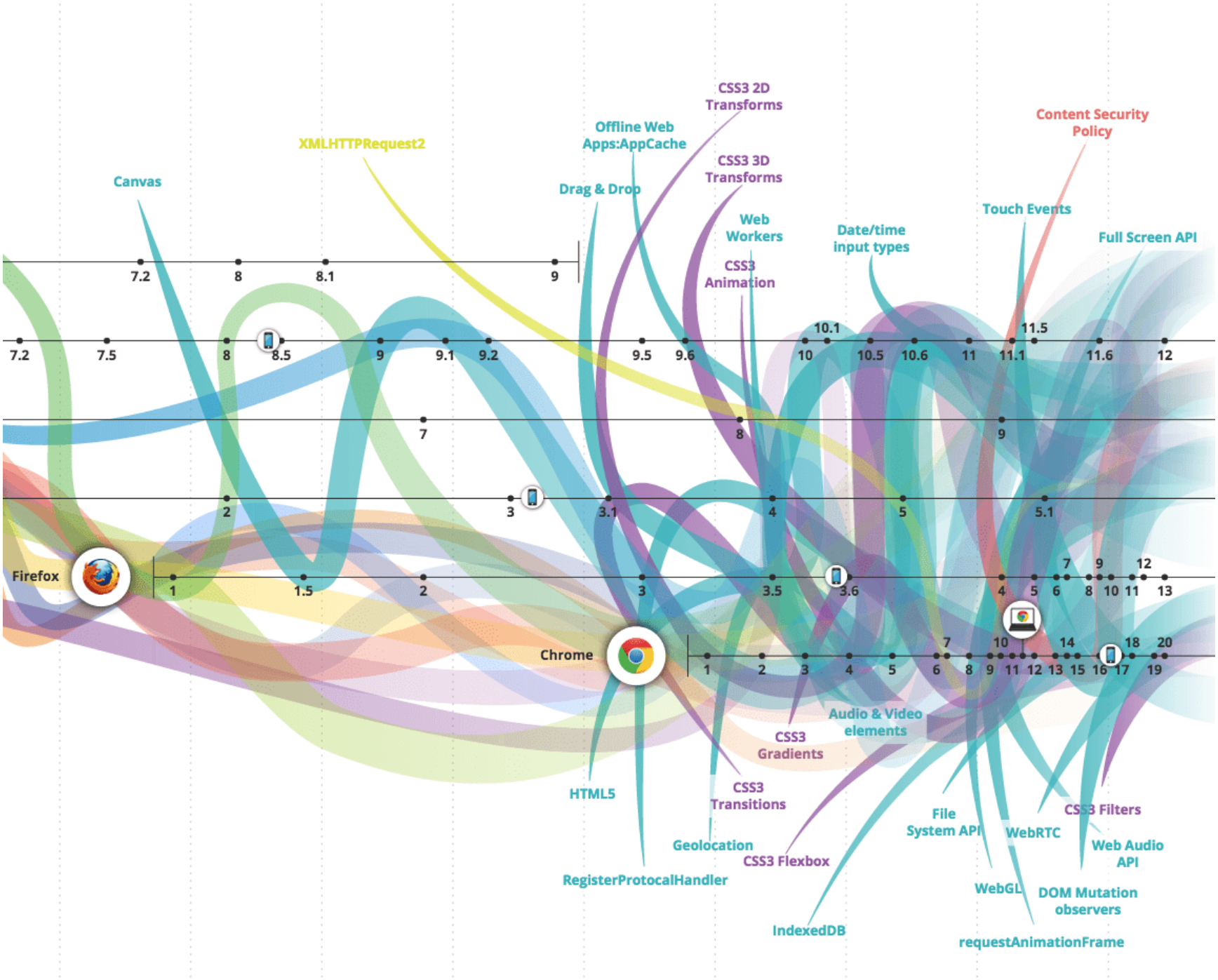



*Image source: http://www.evolutionoftheweb.com/*

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

www.dbooks.org

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

## 3.16 - 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

www.dbooks.org

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

**Architecting:**

- JSON API

## 3.17 - 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]

  [read][$]
- Lodash Templates [docs]

Note that JavaScript 2015 (aka ES6) added 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.

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

## 3.18 - 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] [$]

## 3.19 - Learn Computer Science via JS

- Four Semesters of Computer Science in Six Hours [video][$]
- Four Semesters of Computer Science in Six Hours: Part 2 [video][$]
- Computer Science in JavaScript [read]
- Collection of classic computer science paradigms, algorithms, and approaches written in JavaScript [read]
- A Practical Guide to Algorithms with JavaScript [watch][$]
- Introduction to Data Structures for Interviews [watch][$]
- JavaScript Algorithms and Data Structures Masterclass [watch][$]

## 3.20 - Learn Front-End Application Architecture

www.dbooks.org

**General Learning:**

- 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:**

- JavaScript Application Design [read][$]
- Build an App with React and Ampersand [watch]
- Field Guide to Web Applications [read]
- Frontend Guidelines Questionnaire [read]
- Human JavaScript [read]
- Nicholas Zakas: Scalable JavaScript Application Architecture [watch]
- Organizing JavaScript Functionality [watch][$]
- Patterns for Large-Scale JavaScript Application Architecture [read]
- Terrific [read]
- frontend case studies [read]

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.

My advice, in 2019 learn React and Mobx and Apollo/graphql.

## 3.21 - Learn Data (i.e. JSON) API Design

- API Design, v3 [watch][$]
- Build APIs You Won't Hate [$][read]
- JSON API [read]

## 3.22 - Learn React

**Learning React:**

- Tutorial: Intro To React [read]
- ReactJS For Stupid People [read]
- The Beginner's Guide to ReactJS [watch]
- Complete Intro to React v4 [watch][$]
- React 🎄 [read]
- React Patterns Video Subscription [watch][$]
- React Enlightenment [read]
- REACT JS TUTORIAL #1 - Reactjs Javascript Introduction & Workspace Setup [watch]

**Mastering React:**

- Build Your First Production Quality React App [watch][$]
- Advanced React Component Patterns [watch][$]
- Intermediate React [watch][$]
- React Patterns [read]
- 8 Key React Component Decisions [read]

www.dbooks.org

- React - Basic Theoretical Concepts [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]

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)".

## 3.23 - Learn Application 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]

## 3.24 - 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"*

www.dbooks.org

*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][$]

## 3.25 - Learn JS API Design

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

## 3.26 - Learn Browser 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*

www.dbooks.org

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][$]
- Explore and Master Chrome DevTools [watch]
- Mastering Chrome Developer Tools v2 [watch][$]
- 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

## 3.27 - Learn the Command Line (aka CLI)

> *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]
- Command Line Power User [watch]
- Learn Enough Command Line to Be Dangerous [read] [free to $]

**Mastering:**

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

## 3.28 - 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.*

www.dbooks.org

> *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][$]
- Real-time Web with Node.js [watch]
- API Design in Node.js, v3 [watch][$]
- Learn Node [watch][$]

## 3.29 - Learn Modules

**General Learning:**

- JavaScript for impatient programmers - Modules [read]
- ES6 Modules in Depth [read]
- Exploring JS - Modules [read]
- ES modules: A cartoon deep-dive [read]

**References/Docs:**

- MDN - export
- MDN - import

# 3.30 - Learn Module loaders/bundlers

**Webpack:**

- Webpack [read]
- Webpack 4 Fundamentals [watch][$]
- Survivejs.com Webpack Book [read]

**Rollup:**

- Rollup [read]
    - Microbundle

**Parcel**

- Parcel [read]

www.dbooks.org

## 3.31 - 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 [read]
- 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

## 3.32 - 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:**

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

**Mastering:**

- Git In-depth [watch][$]
- Advanced Git Tutorials [read]
- Pro Git [read]
- Learn Git Branching [interact]

**References/Docs:**

www.dbooks.org

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

## 3.33 - 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

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
- 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?

## 3.34 - 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]

www.dbooks.org

[$]

- 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][$]
- Website Performance [watch][$]
- Web Performance with Webpack 4 [watch][$]
- Website Performance Optimization [watch]
- Front-End Performance Checklist 2019 [PDF, Apple Pages, MS Word] [read]

## 3.35 - 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:**

- JavaScript Testing Practices and Principles [watch][$]
- Front-End First: Testing and Prototyping JavaScript Apps [watch][$]
- Let's Code: Test-Driven JavaScript [watch][$]
- JavaScript Testing [watch]
- JavaScript Testing Recipes [read][$]
- Testable JavaScript [read][$]
- Test-Driving JavaScript Applications: Rapid, Confident, Maintainable Code[read][$]

www.dbooks.org

- Test-Driven JavaScript Development [read][$]
- The Way of the Web Tester: A Beginner's Guide to Automating Tests [read][$]
- Testing React Applications, v2 [watch][$]
- Learn Javascript Unit Testing With Mocha, Chai and Sinon [watch][$]

## 3.36 - 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]

PhantomJS is no longer maintained, Headless Chrome steps in.

## 3.37 - 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]

## 3.38 - Learn Web/Browser/App Security

- Browser Security Handbook [read]
- Frontend Security [watch]
- Hacksplaining [read]
- HTML5 Security Cheatsheet [read]
- HTTP Security Best Practice [read]

www.dbooks.org

- 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]
- Web Security [watch][$]
- Full Stack for Front End Engineers [watch][$]

## 3.39 - 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

www.dbooks.org

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]
- CSS Grids and Flexbox for Responsive Web Design [watch][$]
- Responsive HTML Email Design [watch][$]
- Responsive Images [watch]
- Responsive Web Typography, v2 [watch][$]
- Responsive Web Design Fundamentals [watch]

# Chapter 4. Learning Front-end Dev: Instructor Directed

# Resources/Recommendations

This chapter highlights a few options for instructor directed learning via front-end development schools, courses, programs, and bootcamps.

The table below contains a small selection of instructor-led courses (i.e. programs, schools, and bootcamps). Use the table to get a general idea of what is available, the cost, duration, and locations of courses. (Be aware the information can change quickly)

| company | program | price estimate | on site | remote | duration |
|---|---|---|---|---|---|
| Betamore | Front-end Web Development | 3,000 | Baltimore, MD | | 10 weeks |
| BLOC | Become a Front-end Developer | 4,999 | | yes | 16 weeks @ 25hr/wk or 32 weeks @ 10hr/wk |
| General Assembly | Front-end Web Development | 3,500 | multiple locations | | 3 hrs/day 2 days/wk for 8 weeks |
| Thinkful | Front-end 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 |

**Notes:**

1. For a complete list of schools, courses, programs, and bootcamps to evaluate have a

www.dbooks.org

look at switchup.org or coursereport.com.

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.

# Chapter 5. Front-end Dev Tools

This chapter identifies the 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.

## 5.1 - Doc/API Browsing Tools

**Tools to browse common developer documents and developer API references.**

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

**Cheatsheets:**

- devhints.io

## 5.2 - SEO Tools

**General SEO Tools:**

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

**Tools for Finding SEO Tools:**

- SEO Tools - The Complete List
- CuratedSEOTools - Curated directory of the best SEO tools

## 5.3 - Prototyping & Wireframing Tools

**Creating:**

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

www.dbooks.org

- UXPin [free to $]

**Collaboration / Presenting:**

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

## 5.4 - Diagramming Tools

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

## 5.5 - HTTP/Network Tools

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

# 5.6 - 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.

www.dbooks.org

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:**

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

**Online Code Editors:**

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

**Shareable & Runnable Simple 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
- glitch

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:

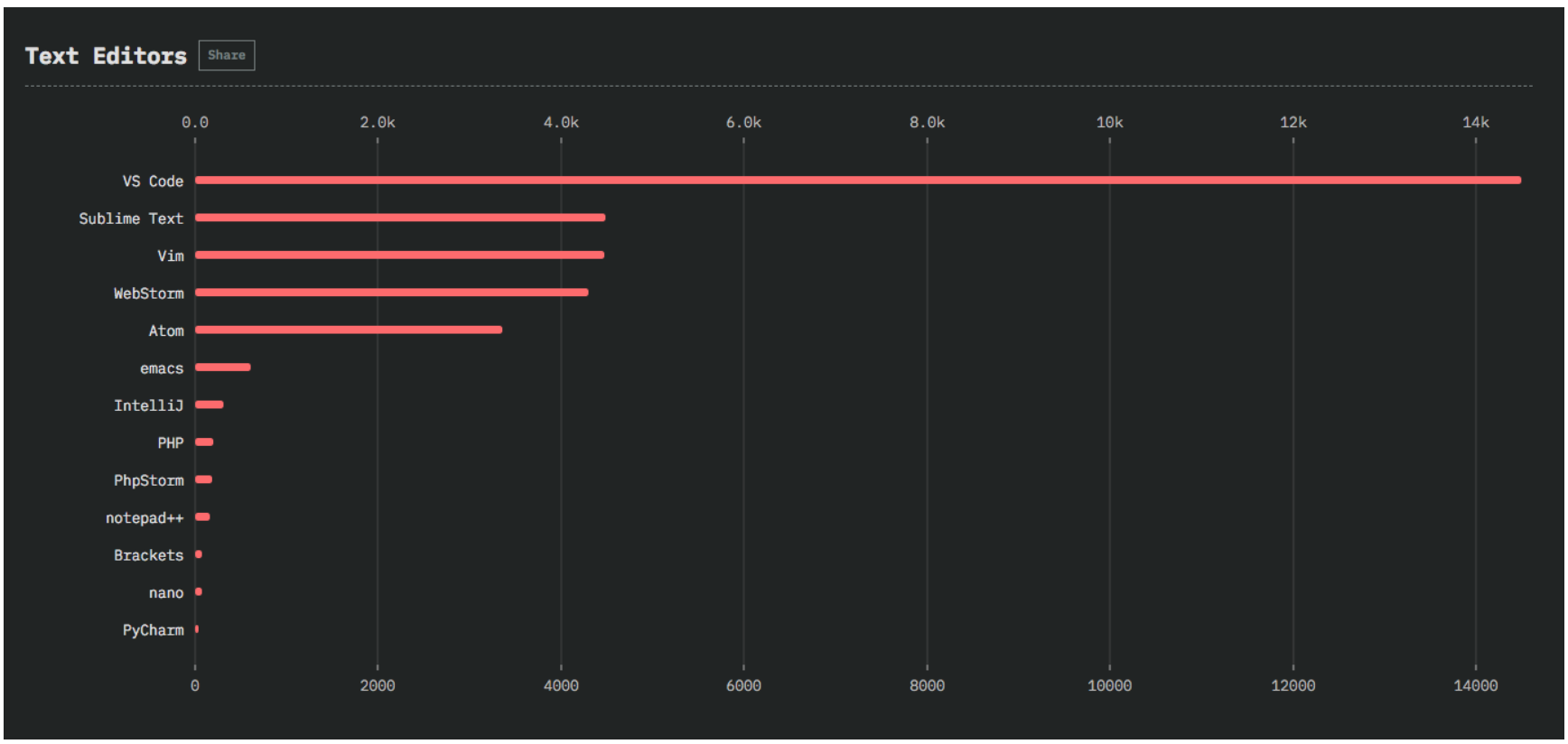



*Image source: https://2018.stateofjs.com/other-tools/text_editors*

# 5.7 - 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

www.dbooks.org

- Browserscope
- caniuse.com
- Firefox Platform Status - Implementation & standardization roadmap for web platform features
- HTML5 Please
- HTML5 Test
- iwanttouse.com
- 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
- 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:**

- slimerjs
- Zombie.js
- Headless Chromium

**Browser Automation:**

Used for functional testing and monkey testing.

- CasperJS
- Nightmare
- TestCafe

www.dbooks.org

**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/

# 5.8 - 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

www.dbooks.org

**Workflow:**

- Emmet

**HTML Outliner:**

- HTML 5 Outliner

**Trending HTML Repositories on GitHub This Month:**

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

# 5.9 - 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

www.dbooks.org

**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
- What's next for CSS?

**Linting/Hinting:**

- CSS Lint
- stylelint

**Code Formatter/Beautifier:**

- CSScomb

- CSSfmt

**Optimizer:**

- clean-css
- cssnano
- CSSO
- purgecss

**Online Creation/Generation/Experimentation Tools:**

- CSS Arrow Please
- CSS Matic
- Enjoy CSS
- 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

www.dbooks.org

- 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

**CSS in JS:**

- styled components
- Emotion
- Radium
- Aphrodite

**Trending CSS Repositories on GitHub This Month:**

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

# 5.10 - DOM Tools

**DOM Libraries/Frameworks:**

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

**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

www.dbooks.org

- 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

## 5.11 - JavaScript Tools

**JS Utilities:**

- accounting.js
- async
- axios
- chance
- date-fns

- dinero
- Finance.js
- format.js
- Howler.js
- immutable
- is.js
- lodash
    - You-Dont-Need-Lodash-Underscore
- Luxon
    - You don't (may not) need Moment.js
- Math.js
- Moment.js
- Numeral.js
- Ramda
- RxJS
- TheoremJS
- voca
- wait
- xregexp.com

**Transforming JavaScript Objects Tool:**

- transform-www

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

www.dbooks.org

- TypeScript

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

- Flow

**Transpiling (ES *to ES*):**

- Babel
- sucrase
- scw

**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
- Terser
- optimize-js
- Prepack

**Obfuscate:**

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

www.dbooks.org

**Sharable/Runnable Code Editors:**

- CodeSandbox [free to $]

**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

# 5.12 - Headless CMS Tools

**Site Generator Listings:**

- headless CMS

# 5.13 - Static Site Generators Tools

**Site Generator Listings:**

- staticgen.com
- staticsitegenerators.net

# 5.14 - 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

www.dbooks.org

- Color Ratio

### Low-Vision Simulators

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

### Screen Readers

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

## 5.15 - App Frameworks (Desktop, Mobile, Tablet, etc.) Tools

**Front-End App Frameworks:**

- 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:**

www.dbooks.org

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 App Frameworks:**

- Electron
- NW.js
- proton
- Neutralino.js
- DeskGap

**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]
- Front-end Guidelines Questionnaire
- Front-end Guidelines

**Performance:**

- js-framework-benchmark
- Front-End Performance Checklist 2019 [PDF, Apple Pages, MS Word] [read]

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).

## 5.16 - JavaScript App Manager

- JSUI

www.dbooks.org

## 5.17 - State Tools

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

## 5.18 - Progressive Web App Tools:

- lighthouse
- Progressive Web App Checklist

## 5.19 - GUI Development/Build Tools

- CodeKit
- Prepros
- KoalaApp [free]

# 5.20 - 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

# 5.21 - UI Widget & Component Toolkits

**On Web Platform:**

www.dbooks.org

- 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

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.

## 5.22 - 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 $]

# 5.23 - Graphics (e.g., SVG, canvas, webgl) Tools

**General:**

www.dbooks.org

- Fabric.js
- Two.js

**Canvas:**

- EaselJS
- Paper.js

**SVG:**

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

**WebGL:**

- pixi.js
- three.js

# 5.24 - 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

# 5.25 - JSON Tools

**Online Editors:**

- JSONmate
- JSON Editor Online

**Formatter & Validator:**

www.dbooks.org

- 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}

# 5.26 - Placeholder Content Tools

**Images:**

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

**Device Mockups:**

- placeit.net
- mockuphone.com

**Text:**

- Meet the Ipsums

www.dbooks.org

- catipsum.com
- baconipsum.com (API)

**User Data:**

- uinames.com
- randomuser.me

# 5.27 - 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
- Nightwatch
- WebDriver.io

**Browser Automation:**

- CasperJS
- Nightmare
- TestCafe

**UI Testing Tools:**

- gremlins.js
- Percy

www.dbooks.org

- BackstopJS
- PhantomCSS
- Ghost Inspector
- diff.io

**Automated dead link and error detectors:**

- Monkey Test It

**HTTP Stubbing**

- Polly.JS

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

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

# 5.29 - Module Loading/Bundling Tools

- Parcel
- Rollup
    - Microbundle
- webpack
    - Poi
    - jetpack
- Fusebox
- Browserify

# 5.30 - Module/Package Repository Tools

- NPM
- yarn
- PNPM

# 5.31 - Hosting Tools

### General

- AWS [$]

www.dbooks.org

- DigitalOcean [$]
- WebFaction [$]

**Static**

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

## 5.32 - Project Management & Code Hosting Tools

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

## 5.33 - Collaboration & Communication Tools

- Slack & screenhero [free to $]

- appear.in
- Mattermost [free to $]

**Code/GitHub Collaboration & Communication:**

- Gitter [free to $]

# 5.34 - Content Management Hosted/API Tools

**Headless CMS Tools:**

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

**Self-hosted Headless CMS Tools:**

- Cockpit
- Directus 7 App

**Hosted CMS:**

- LightCMS [$]
- Surreal CMS [$]

**Static CMS Tools:**

- webhook.com
- Dato CMS
- siteleaf

www.dbooks.org

- forestry.io

# 5.35 - Back-end/API tools

**Data/back-end as a service aka BAAS:**

- 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

## 5.36 - Offline Tools

- Hoodie
- Offline.js
- PouchDB
- upup
- Workbox

For more tools look here.

## 5.37 - Security Tools

**Coding Tool:**

- DOMPurify
- XSS

**Security Scanners/Evaluators/Testers:**

- Netsparker
- Websecurify
- OWASP ZAP

**References:**

www.dbooks.org

- HTML5 Security Cheatsheet

## 5.38 - Tasking (aka Build) Tools

**Tasking/Build Tools:**

- Gulp

**Opinionated Tasking/Build pipeline tools:**

- Brunch

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".

## 5.39 - 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 $]

## 5.40 - Site/App Monitoring Tools

**Uptime Monitoring:**

- Uptime Robot [free to $]

**General Monitoring Tools:**

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

## 5.41 - JavaScript Error Reporting/Monitoring

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

www.dbooks.org

# 5.42 - Performance Tools

**Reporting:**

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

**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 2019 [PDF, Apple Pages, MS Word] [read]

## 5.43 - Tools for Finding Tools

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

## 5.44 - Documentation Generation Tools

- docz
- ESDoc
- JSDoc
- documentjs

www.dbooks.org

# Chapter 6. Front-end Communities, Newsletters, News Sites, & Podcasts

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

- The Big Web Show
- Dev Tips
- Front End Happy Hour
- Front-End Front
- Front-end Focus
- Web Platform News Weekly
- ShopTalk Show
- UX Design Newsletter
- Web Development Reading List
- The Web Platform Podcast
- Web Tools Weekly
- Fresh Brewed Front-end
- Pony Foo Weekly
- CSS-Tricks
- syntax.

**HTML/CSS Newsletters:**

- Accessibility Weekly
- CSS Weekly
- csslayout.news

**JavaScript Newsletters, News, & Podcasts:**

- Awesome JavaScript Newsletter
- Echo JS
- ECMAScript Daily
- ES.next News
- JavaScript Jabber
- JavaScript Kicks
- JavaScript Weekly
- React Status
- JS Party
- JAMStack Radio
- My JavaScript Story

**Front-End Communities**

- http://fedsonslack.com/
- front-end on spectrum.

> **Notes:**
>
> 1. Need more Newsletters, News Sites, & Podcasts look at Awesome Newsletter.
> 2. Find local front-end development communities by searching https://www.meetup.com/

www.dbooks.org