Front-end Handbook

Frontend Masters

Published with GitBook

Table of Contents

Front-end Developer Handbook

Written by Cody Lindley sponsored by — Frontend Masters

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

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, CTO's, instructors, and head hunters to gain insights into the practice of front-end development.

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

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

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

The handbook is divided into three parts.

Part I: The front-end practice

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

Part II: Learning front-end development

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

Part III: Front-end development tools

Part three briefly explains and identifies tools of the trade.

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

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

contribute content, suggestions, and fixes on github:

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

What is a front-end developer?

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

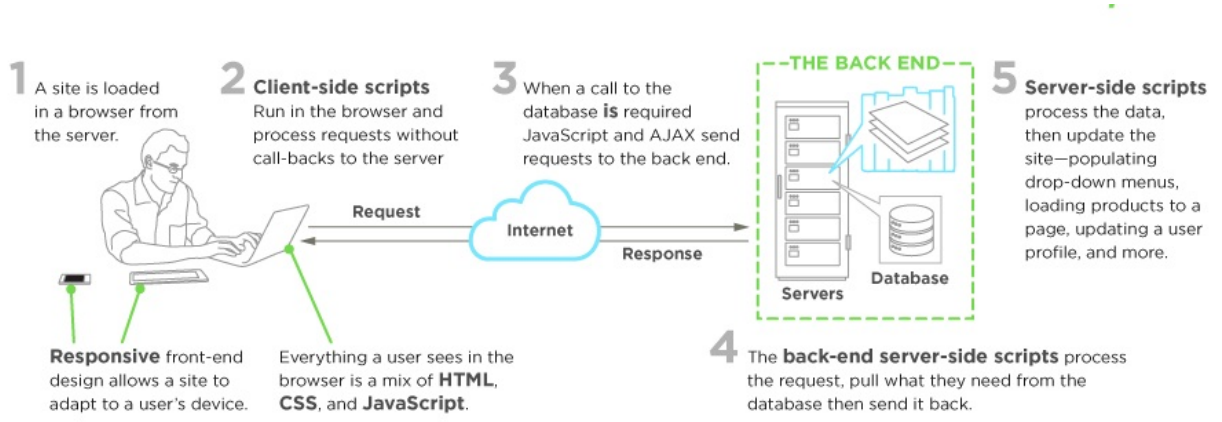

image source: https://www.upwork.com/hiring/development/front-end-developer/

Typically, a person enters into the field of front-end development by learning to develop HTML, CSS, and JS code, which runs in a web browser, headless browser, webview, or as compilation input for a native runtime environment. The four run times scenarios are explained below.

A web browser is software used to retrieve, present, and traverse information on the WWW. Typically, browsers run on a desktop, laptop, 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:

- Chrome
- Internet Explorer
- Firefox
- Safari

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

The most common headless browsers are:

- PhantomJS
- slimerjs

- trifleJS

Webviews are used by a native OS 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.

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)

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

Some examples of these environments are:

- NativeScript
- React Native

Part I. The front-end practice

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

Front-end jobs titles

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

Front-end Developer/Engineer (aka Front-end Web Developer/Engineer, Client-side Developer/Engineer, Front-end Software Developer/Engineer or UI Engineer)

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.

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 JavaScript (optionally...Application) Developer

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 will have solid experience building front-end applications).

Front-end Web Designer

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

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

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

Mobile/Tablet Front-end Developer

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

Front-end SEO Expert

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

Front-end Accessibility Expert

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

Front-end Dev. Ops

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

Front-end Testing/QA

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

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

end.

Web technologies employed by front-end developers

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

The following web technologies are employed by front-end developers:

- Hyper Text Markup Language (aka HTML)
- Cascading Style Sheets (aka CSS)
- Document Object Model (aka DOM)
- JavaScript Programming Language (aka: ECMAScript 6, ES6, JavaScript 2015)
- Web API's (aka HTML5 and friends or Browser API's)
- Hypertext Transfer Protocol (aka HTTP)
- Uniform Resource Locator's (aka URL)
- JavaScript Object Notation (aka JSON)
- Web Content Accessibility Guidelines (aka WCAG) & Accessible Rich Internet Applications (aka ARIA)

These technologies are defined below with the relevant documentation and specifications. For a comprehensive list of all web related specifications have a look at platform.html5.org.

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.org

Most relevant specifications / documentation:

- HTML5 from W3C : 5th major revision of the core language of the World Wide Web
- The elements of HTML from the Living Standard
- The HTML Syntax from the Living Standard
- All W3C HTML Spec
- HTML element reference
- HTML attribute reference
- Global attributes

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.org

Most relevant specifications / documentation:

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

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.org

Most relevant specifications / documentation:

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

JavaScript Programming Language (aka: ECMAScript 6, ES6, JavaScript 2015)

> 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.org

Most relevant specifications / documentation:

- ECMAScript® 2015 Language Specification

Web API's (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

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.org

Most relevant specifications:

- Hypertext Transfer Protocol -- HTTP/1.1
- Hypertext Transfer Protocol version 2 draft-ietf-httpbis-http2-16

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), [3] 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.[4][5] 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.org

Most relevant specifications:

- Uniform Resource Locators (URL)
- URL Living Standard

JavaScript Object Notation (aka JSON)

> JSON, 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.org

Most relevant specifications:

- Introducing JSON
- The JSON Data Interchange Format
- JSON API

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

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

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

Front-end dev skills

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

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

Beyond HTML, CSS, DOM, JavaScript, HTTP/URL, and browser development, a front-end developer could be skilled in one 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 API's (e.g Restful API)
- Web Font Embedding
- Scalable Vector Graphics (aka SVG)
- Regular Expressions
- Content Strategy
- Microdata / Microformats
- Task Runners, Build Tools, Process Automation Tools
- Responsive Web Design
- Object Oriented Programming
- Application Architecture
- Modules
- Dependency Managers
- Package Managers
- JavaScript Animation
- CSS Animation
- Charts / Graphs
- UI widgets
- Code Quality Testing
- Code Coverage Testing
- Code Complexity Analysis
- Integration Testing
- Command Line / CLI
- Templating Strategies
- Templating Engines
- Single Page Applications
- XHR Requests (aka AJAX)
- Web/Browser Security
- HTML Semantics
- Browser Developer Tools

Front-end developers develop for...

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

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

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

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

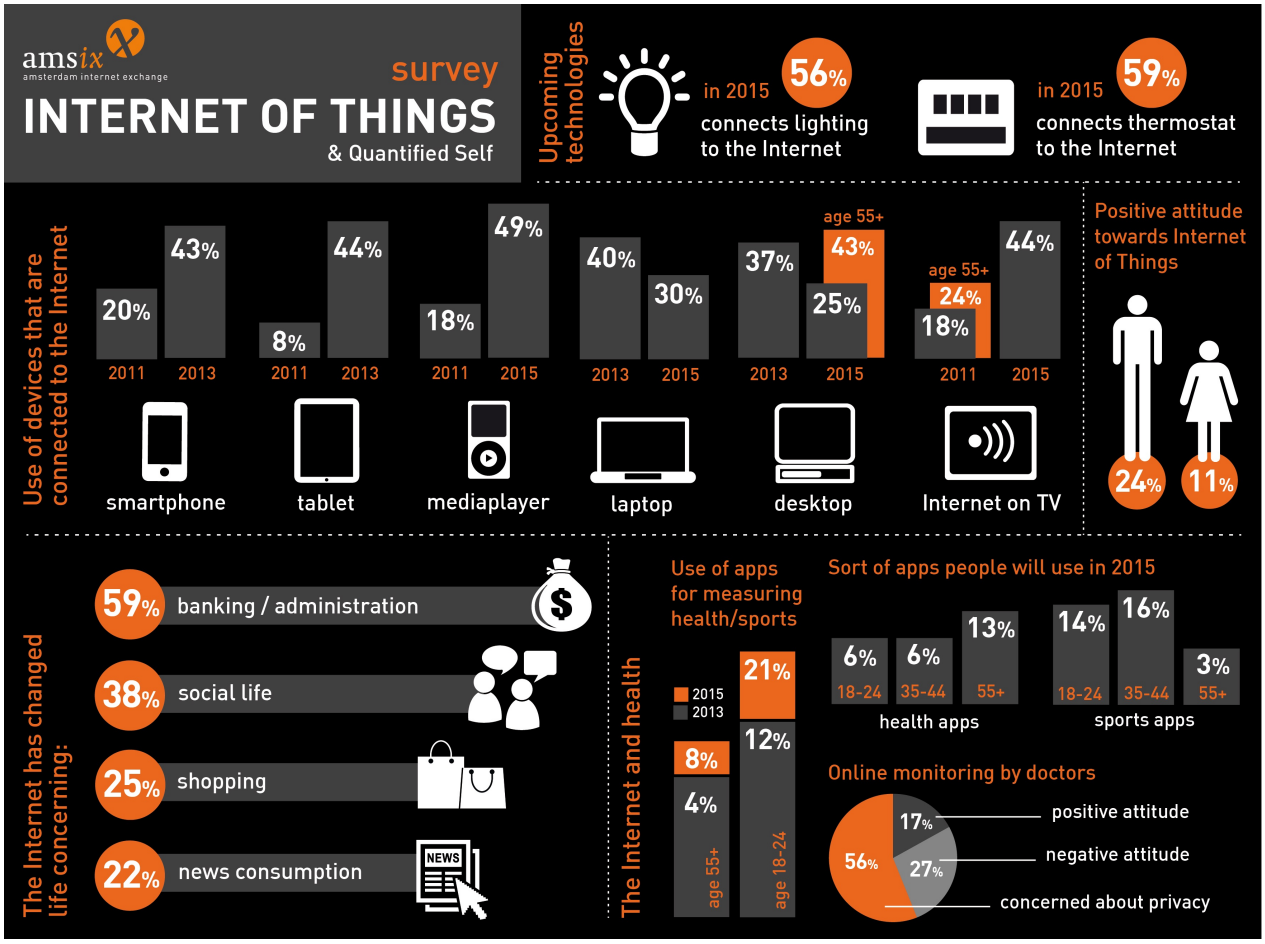

image source: https://ams-ix.net/newsitems/87

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

- A web browser (examples: Chrome, IE, Safari, Firefox) running on an OS.
- A headless browser (examples phantomJS) driven from a CLI running on an OS.
- A WebView/browser tab (think iframe) embedded within a native application as a runtime with bridge to native API's. 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 API's. The UI will make use of native UI parts (e.g. iOS native controls) not web technologies. (examples: NativeScript, React Native)

Front-end on a team

A front-end developer is typically only one player on a team that designs and develops web sites, web applications, or native applications running on web technologies. (Note: A developer who builds everything was once called a "web master" but as of late these rare and mythical developers are called "full-stack developers").

A bare bones team for building professional sites or software on the web will minimally contain the following roles.

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

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

It is assumed that the team is being directed by a project lead of 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:

- Visual Designers
- Interface Design/Interaction Designer/Information Design
- **SEO strategist**
- Front-end Developer
- **DevOps Engineers**
- Back-end Developer
- **API Developer**
- **Database Administrator**
- **QA Engineer / Testers**

Generalist myth

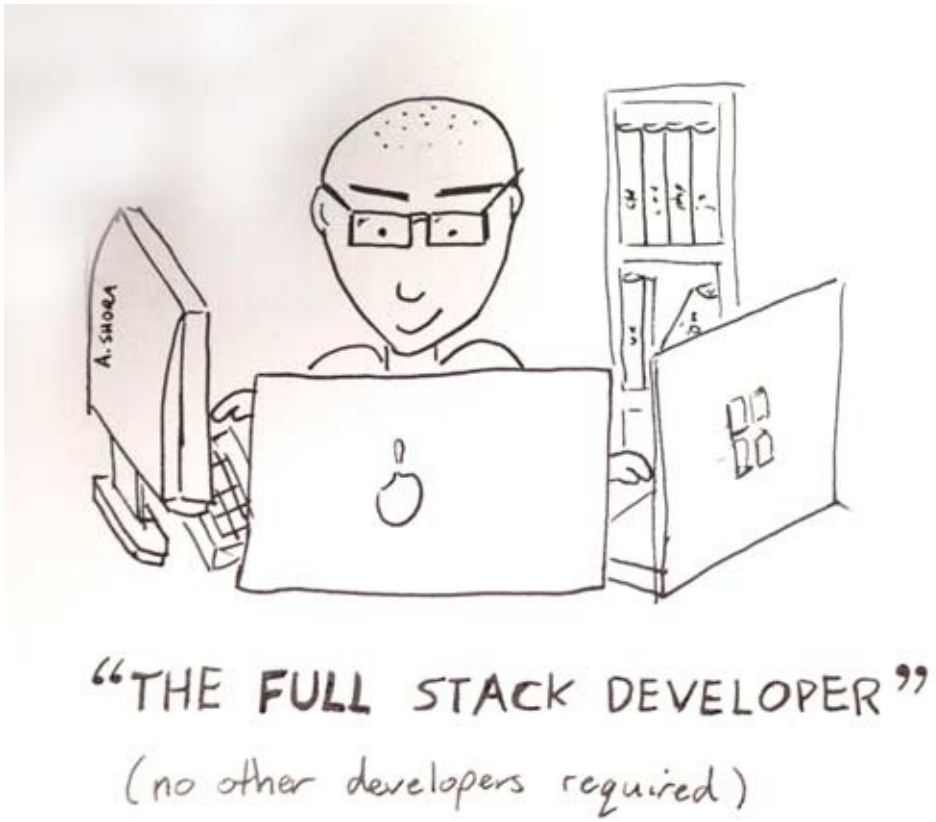

image source: http://andyshora.com/full-stack-developers.html

The roles required to design and develop a web solution require a deep skill set and vast experience in the area of visual design, UI/interaction design, front-end development, and back-end development. Any person (aka generalist or full-stack developer/designer) who can fill one or more of these 4 roles at a professional level is a rare exception to the rule.

Pragmatically, you should seek to be, or seek to hire, an expert in one of these roles. Those who claim to operate at an expert level at one or more of these roles are exceptionally rare and more than likely mythical, given modern stacks.

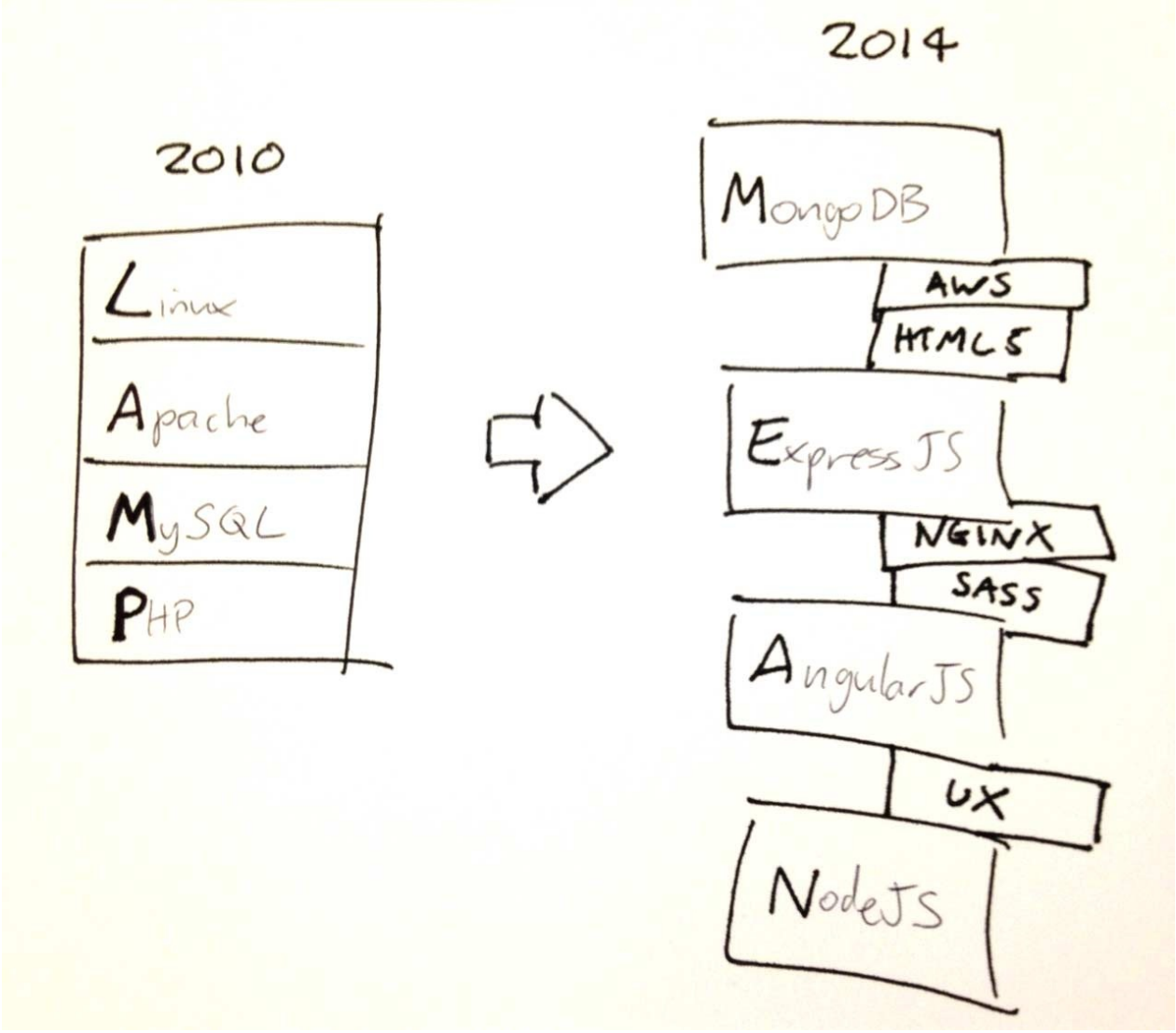

image source: http://andyshora.com/full-stack-developers.html

Front-end interviews

Questions you may get asked:

- Front-end Job Interview Questions
- Interview Questions for front-end-Developer
- 10 Interview Questions Every JavaScript Developer Should Know *Front End Quiz* Javascript Quiz

Questions you ask:

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

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.

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

Front-end salaries

The national average in the U.S for a front-end developer is $75k.

An experienced front-end developer potentially can 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).

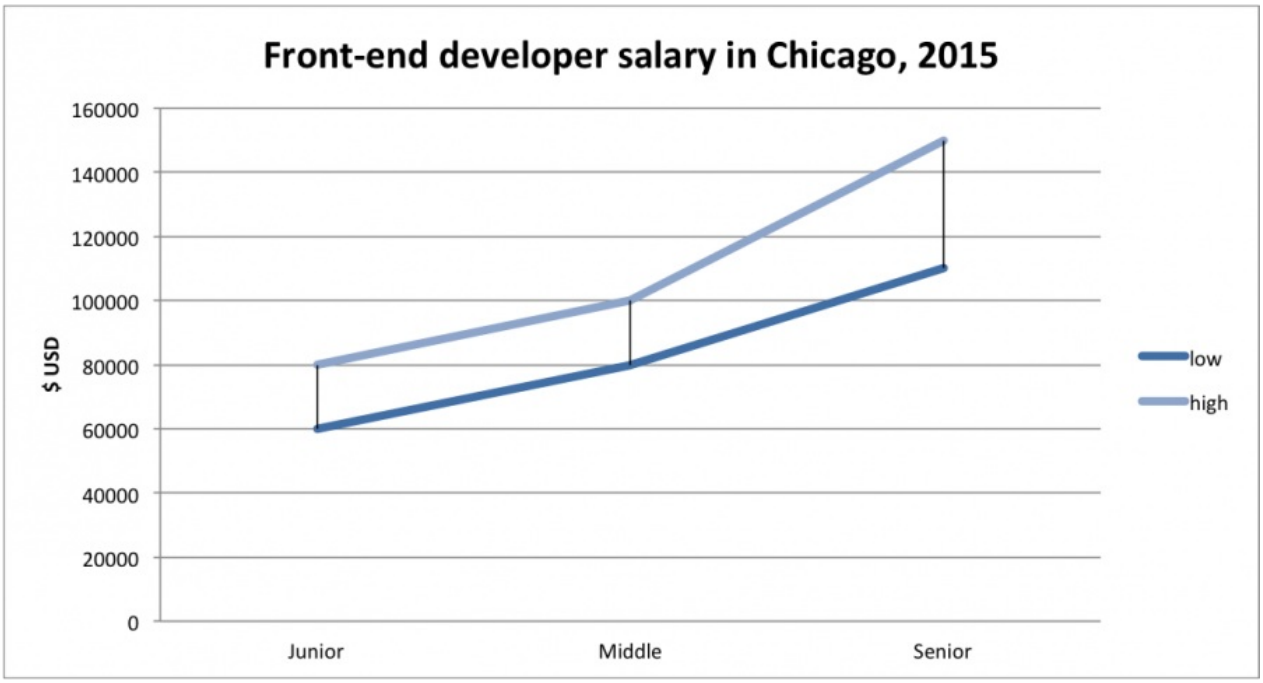

image source: http://intersog.com/blog/chicago-tech-salary-guide-2015/

How front-end developers are made

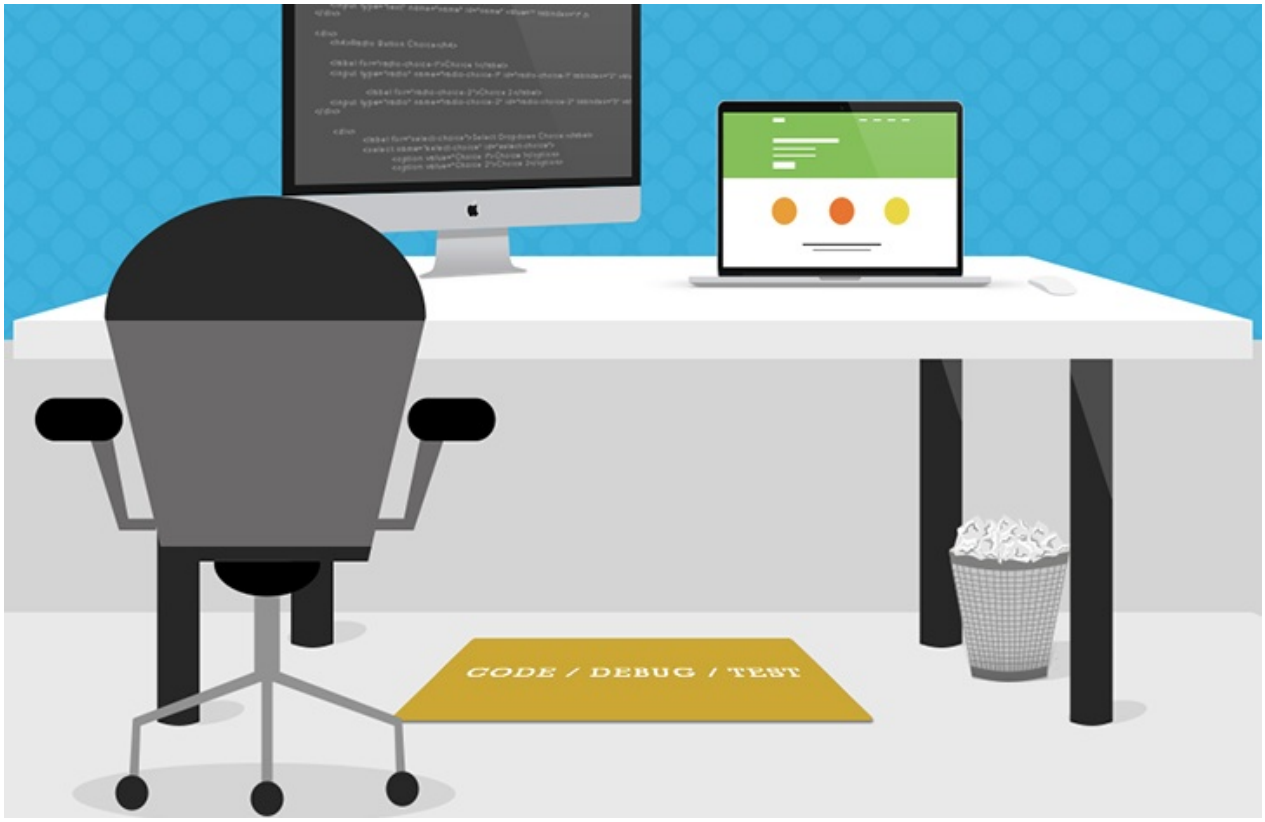

image source: http://cdn.skilledup.com/wp-content/uploads/2014/11/life-of-front-end-developer-infographic-Secondary.jpg

How exactly does one become a front-end developer? Well, it's complicated. Still today you can't go to college and expect to graduate with a degree in front-end engineering. And, I rarely hear of or meet front-end developers who suffered through what is likely a deprecated computer science degree or graphic design degree to end up writing HTML, CSS, and JavaScript professionally. In fact, most of the people working on the front-end, even today, generally seem to be self taught and not traditionally trained as a programmer. Why is this the case?

A front-end developer is not a focused visual designer or an interaction designer. Design school is not exactly the place to hone front-end skills. A front-end developer is not exactly a traditionally trained computer science graduate either. Focusing on either doesn't prepare a person for front-end development. And, in fact, following the traditional paths for either in the higher education systems of America (i.e. College) can derail a person from potentially finding a doorway into a practice where actual experience is king. Today, if you want to be a front-end developer, you teach yourself or you take what is likely a non accredited program, course, bootcamp, or class.

A front-end engineer crafts the skeleton that the user interface rests upon. They must, at times, care as much about the interaction design as they do about the underlying code that creates the UI interactions. Therefore, many in practice today do not come to front-end engineering with programming skills, but, instead, from the other direction. That is, front-end development seems to be filled with more designer types turned developer than programmer types turned front-end developer. Of course, as JavaScript has grown up, so has the desire by more traditionally trained programmers to bring their knowledge to the front-end practice. If you are not aware, front-end developers have not always been considered by "real" programmers as, well, programmers. But times are a changing.

With all of that said, I believe that the path to a career as a front-end developer is very much an unknown process. What I can say is that to become a front-end engineer one must know and use HTML, CSS, DOM, and JavaScript at a high level without ignoring interaction design or traditional programming know how. In fact, from my experience, the best front-end developers often have a mastery understanding of interaction design and programming, but from the context of the web platform (i.e. browsers, HTML, CSS, DOM, and JavaScript). And for whatever reason, this knowledge is often found not given. That is to say, front-end engineering still seems very much to be a practice made up of self taught people, as opposed to a field that corresponds directly to an educational focus from an organized and accredited higher learning situation.

If you were to set out today to become a front-end developer I would loosely strive to follow the process outlined below. The process assumes you are your own best teacher.

1. Learn, roughly, how the web works. Make sure you know the "what" and "where" of Domains, DNS, URL's, HTTP, networks, browsers, servers/hosting, databases, JSON, data API's, HTML, CSS, DOM, and JavaScript. The goal is to make sure you loosely know how it all works together and exactly what each part is doing. Focus on the high level outlines for front-end architectures. Start with simple web pages and briefly study native web applications (aka SPA's).
2. Learn HTML, CSS, Accessibility, and SEO.
3. Learn the fundamentals of UI design patterns, interaction design, user experience design, and usability.
4. Learn the fundamentals of programming
5. Learn JavaScript
6. Learn JSON and data API's
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.
10. Learn Node.js

Where you actually stop in the process is what will separate a front-end HTML/CSS developer from an expert level front-end application/JavaScript developer.

A short word of advice on learning. Learn the actual underlying technologies, before learning abstractions. Don't learn jQuery, learn the DOM. Don't learn SASS, learn CSS. Don't learn HAML, learn HTML. Don't learn coffeeScript, learn JavaScript. Don't learn Handlebars, learn JavaScript ES6 templates. Don't just use Bootstrap, learn UI patterns. When getting your start, you should fear most things that conceal complexity. Abstracts in the wrong hands can give the appearance of advanced skills, while all the time hiding the fact that a developer has an inferior understanding of the basics or underlying concepts.

The rest of this book contains point a reader to the resources and tools to follow the suggested process. It is assumed that you are not only learning, but also doing as you learn and investigate tools. Some suggest only doing. 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.

I should mention that lately a lot of non-accredited front-end code schools/bootcamps have emerged. These avenues of becoming a front-end developer are teacher directed in classroom (virtual and physical) courses, which follow a more traditional style of learning from an official instructor (i.e. syllabus, test, quizzes, projects, team projects, grades etc..). I have more to say about these institutions in the direct learning section of this handbook. In brief, 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 what is actually free, and hold you accountable for learning it, you might consider an organized course. Otherwise, I am not aware of any other profession that is practically free for the taking with an internet connection and a burning desire for knowledge.

If you want to get started immediately I'd suggest the following general overviews of the practice of front-end development:

- Frontend Guidelines [read]
- Being a web developer [read]
- Isobar Front-end Code Standards [read]
- Web Fundamentals [read]
- Front-end Curriculum [read]
- freeCodeCamp [interact]
- Planning a Front-end JS Application [watch]
- So, You Want to be a Front-End Engineer [watch]
- Front End Web Development Career Kickstart [watch][$]
- Front End Web Development: Get Started [watch][$]

- Front-End Web Development Quick Start With HTML5, CSS, and JavaScript [watch][$]
- Introduction to Web Development [watch][$]
- Foundations of Front-End Web Development [watch][$]
- Lean Front-End Engineering [watch][$]
- A Baseline for Front-End [JS] Developers: 2015 [read]
- Learn Front End Web Development [watch][$]
- Front-End Dev Mastery [watch][$]
- Front-End Web Developer Nanodegree [watch][$]

Part II: Learning

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

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

Self directed learning

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

The resources include free material and paid material. Paid material will be indicated with [$].

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

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

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

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

- How the Internet Works [watch]
- WHAT IS THE INTERNET? or, "You Say Tomato, I Say TCP/IP" [read]
- How does the Internet work [read]
- How does the Internet Work? [read]
- How the Internet Works in 5 Minutes [watch]
- How The Web Works [watch][$]

Learn web browsers

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

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

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

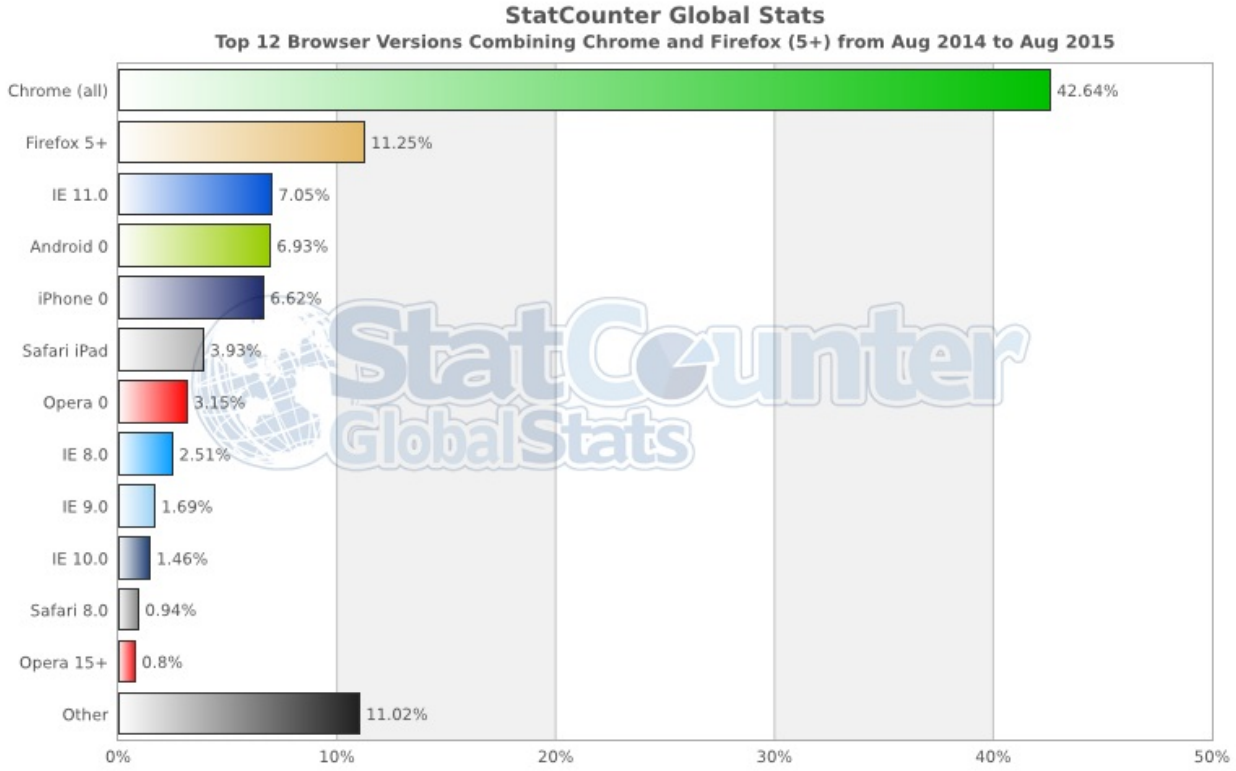

image source: http://gs.statcounter.com/#all-browser_version_partially_combined-ww-monthly-201408-201508-bar

Evolution of browsers & web technologies (i.e. API's)

- www.evolutionoftheweb.com [read]

- Timeline of web browsers [read]

The most commonly used headless browser are:

- PhantomJS (engine: Webkit + SquirrelFish)
- slimerjs (engine: Gecko + SpiderMonkey)
- TrifleJS (engine: Trident + Chakra)

How browsers work

- 20 Things I Learned About Browsers and the Web [read]
- How Browsers Work: Behind the scenes of modern web browsers [read]
- Fast CSS: How Browsers Lay Out Web Pages [read]

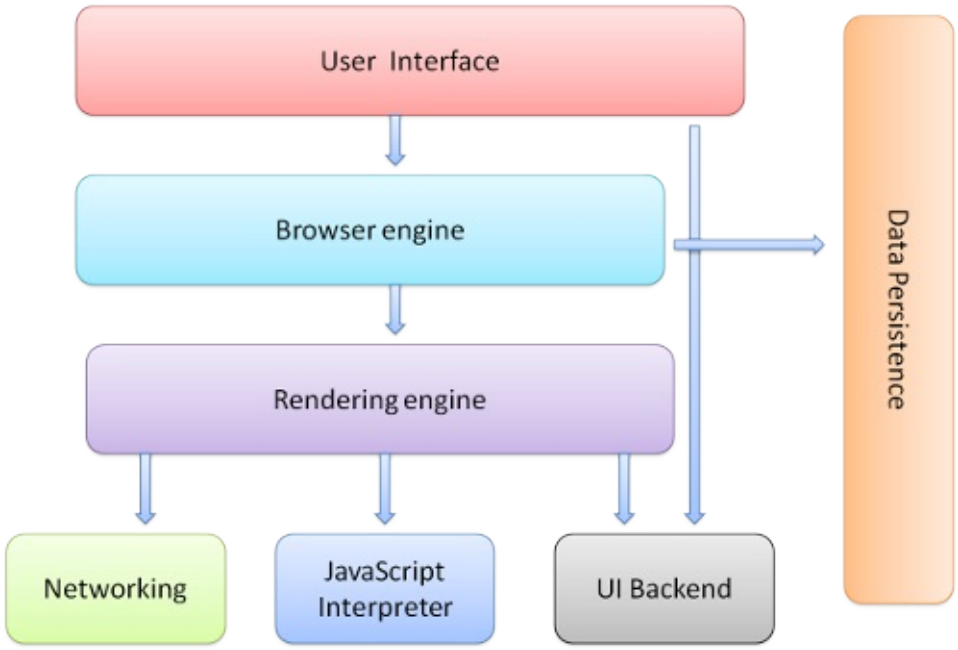

image source: http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

Optimizing for browsers:

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

Browser security

- Browser Security Handbook [read]
- HTML5 Security Cheatsheet [read]
- Frontend Security [watch]
- Security for Web Developers: Using JavaScript, HTML, and CSS [read][$]
- The Tangled Web: A Guide to Securing Modern Web Applications read

Comparing browsers

- Comparison of web browsers [read]

Developing for browsers

In the past, a front-end developer spent a lot of time making code work in several different browsers. This was once a bigger issue than it is today, unless you have to write code for older browsers (i.e. <IE8). This still remains an issue today, just not one that demands so much of the front-end developers time and brain cycles. The fact of the matter is modern abstractions (e.g. jQuery, pre-processors, transpilers) have done away with a lot of browser inconsistency issues.

Evergreen browsers

The latest versions of browsers are considered evergreen browsers. That is, in theory they are suppose to automatically update themselves silently without prompting the user. This move towards self updating browsers has been in reaction to the slow process of eliminating older browsers. Older browsers are complicated to develop for given their deviations from the commonalities between modern browsers (i.e. new specifications and this rate of change).

Picking a browser

As of today, most front-end developers use Chrome and the tools available to a developer, "Chrome Dev Tools". However, all of the browsers offer a flavor of developer tools. Picking one to use for development is a subjective matter. The more important issue is knowing which browsers you have to support and testing in each as you develop. Select whichever browser makes sense to your brain and gets the job done. I suggest using Chrome simply because the developer tools are consistently improving and at this time contain the most robust features.

Browser hacks

- browserhacks.com [read]

Learn Domain Name System (aka DNS)

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

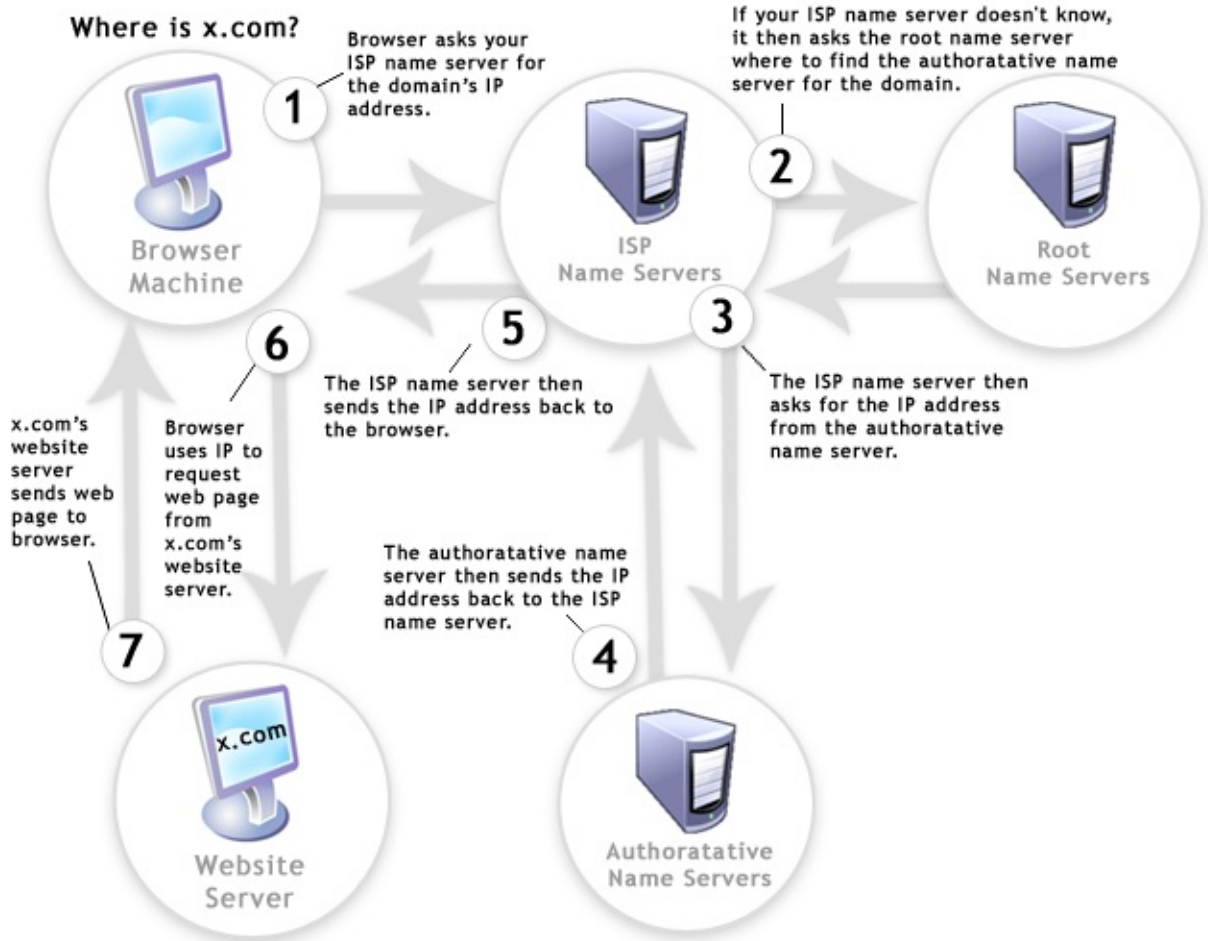

image source: http://www.digital-digest.com/blog/DVDGuy/wp-content/uploads/2011/11/how_dns_works.jpg

- DNS Explained [watch]
- How DNS works [read]

Learn HTTP/networks (including CORS & WebSockets)

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

HTTP

- HTTP: The Protocol Every Web Developer Must Know - Part 1 [read]
- HTTP: The Protocol Every Web Developer Must Know - Part 2 [read]
- HTTP Fundamentals [watch][$]
- HTTP Succinctly [read]
- High Performance Browser Networking: What every web developer should know about networking and web performance [read]

CORS

- HTTP Status Codes in 60 Seconds [watch]
- HTTP access control (CORS) [read]
- CORS in Action [read][$]

WebSockets

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

Learn web hosting

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

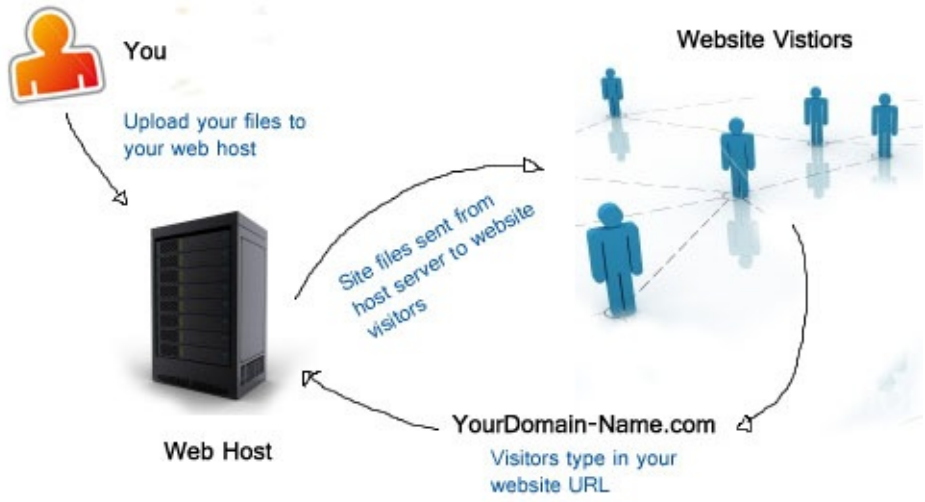

image source: http://www.alphaelite.com.sg/sitev2/images/stories/webhostdemo.jpg

General learning:

- Web Hosting Beginner Guide [read]
- Web Hosting For Dummies [read][$]
- Ultimate Guide to Web Hosting [read]

Learn general front-end development

General learning:

- Frontend Guidelines [read]
- Being a web developer [read]
- Isobar Front-end Code Standards [read]
- Web Fundamentals [read]
- Front-end Curriculum [read]
- freeCodeCamp [interact]
- Planning a Front-end JS Application [watch]
- So, You Want to be a Front-End Engineer [watch]
- Front End Web Development Career Kickstart [watch][$]
- Front End Web Development: Get Started [watch][$]
- Front-End Web Development Quick Start With HTML5, CSS, and JavaScript [watch][$]
- Introduction to Web Development [watch][$]
- Foundations of Front-End Web Development [watch][$]
- Lean Front-End Engineering [watch][$]
- A Baseline for Front-End [JS] Developers: 2015 [read]
- Learn Front End Web Development [watch][$]
- Front-End Dev Mastery [watch][$]
- Front-End Web Developer Nanodegree [watch][$]

General front-end newsletters, news outlets, & podcasts:

- shoptalkshow.com
- frontendfront.com
- webtoolsweekly.com
- O'Reilly Web Platform Radar
- The Wdb Platform Podcast
- The Web Ahead
- The Big Web Show
- Fresh Brewed Frontend
- Mobile Web Weekly
- Open Web Platform Daily Digest
- FRONT-END DEV weekly

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 build proper user interfaces.

- Designing Interfaces [read][$]
- About Face: The Essentials of Interaction Design [read][$]
- Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability [read][$]

Learn HTML & CSS

> **HTML** - HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages.[1] 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.org
>
> **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.org

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

General learning:

- codecademy.com HTML & CSS [interact]
- Intro to HTML/CSS: Making webpages [watch]
- Learn to Code HTML & CSS [read]
- Learn CSS Layout [read]
- Front End Web Development: Get Started [watch][$]
- HTML and CSS: Design and Build Websites [read][$]
- Front-End Web Development Quick Start With HTML5, CSS, and JavaScript [watch][$]
- HTML Mastery: Semantics, Standards, and Styling [read][$]
- CSS Positioning [watch][$]
- HTML Document Flow [watch][$]
- Introduction to HTML5 and CSS3 [watch][$]
- Absolute Centering in CSS [read]
- Understanding the CSS Box Model [watch]
- Solid HTML Form Structure [watch]
- Semantic HTML: How to Structure Web Pages [watch]

Mastering:

- CSS Selectors from CSS4 till CSS1 [read]
- CSS Diner [interact]
- CSS3 In-Depth [watch][$]
- atozcss.com/ [watch]
- A Complete Guide to Flexbox [read]

References/docs:

- htmlelement.info
- MDN CSS reference
- MDN HTML element reference
- cssvalues.com/
- CSS TRIGGERS...A GAME OF LAYOUT, PAINT, AND COMPOSITE
- HTML attribute reference

Glossary:

- HTML Glossary Programming reference for HTML elements
- CSS Glossary - Programming reference for CSS covering Comments, Properties, and Selectors

Standards/specifications:

- HTML5 from W3C : 5th major revision of the core language of the World Wide Web
- The elements of HTML from the Living Standard
- The HTML Syntax from the Living Standard
- All W3C HTML Spec
- Global attributes
- Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification
- Selectors Level 3
- All W3C CSS Specifications

Architecting CSS:

- oocss [read]
- SMACSS [read][$]
- SMACSS [watch][$]
- Atomic Design [read]

Authoring conventions:

- idiomatic-css [read]
- Standards for developing flexible, durable, and sustainable HTML and CSS [read]
- cssguidelin.es [read]

- Google HTML/CSS Style Guide

HTML/CSS newsletters:

- HTML 5 Weekly
- CSS Weekly

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

General learning:

- SEO Fundamentals from David Booth [watch][$]
- SEO Fundamentals from Paul Wilson [watch][$]
- SEO for Web Designers [watch][$]
- Google Search Engine Optimization Starter Guide [read]
- SEO Tutorial For Beginners 2015 [read]

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.org

General learning:

- codecademy.com JavaScript [interact]
- Professional JavaScript for Web Developers [read][$]
- JavaScript Enlightenment [read]
- The Principles of Object-Oriented JavaScript [read][$]
- Speaking JavaScript [read]
- You Don't Know JS: Up & Going [read]
- Understanding ECMAScript 6 [read]
- JavaScript Patterns [read][$]
- JS.Next: ES6 [watch][$]
- 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 Modules [read]

Mastering:

- Functional JavaScript: Introducing Functional Programming with Underscore.js [read][$]
- ECMA-262 by Dmitry Soshnikov [read]
- Advanced JavaScript [watch][$]
- JavaScript the Good Parts [watch][$]
- You Don't Know JS: Scope & Closures [read]

- You Don't Know JS: this & Object Prototypes [read]
- You Don't Know JS: Types & Grammar [read]
- You Don't Know JS: Async & Performance [read]
- You Don't Know JS: ES6 & Beyond [read]
- Eloquent JavaScript [read]
- Test-Driven JavaScript Development [read][$]
- JavaScript Allongé [read][$]
- JavaScript with Promises [read][$]
- High Performance JavaScript (Build Faster Web Application Interfaces) [read][$]
- JavaScript Regular Expression Enlightenment [read]
- Using Regular Expressions [watch][$]

References/docs:

- MDN JavaScript reference

Glossary/encyclopedia:

- JavaScript Glossary
- The JavaScript Encyclopedia

Standards/specifications:

- ECMAScript® 2015 Language Specification
- Status, process, and documents for ECMA262

Style:

- Node.js Style Guide
- Principles of Writing Consistent, Idiomatic JavaScript
- Airbnb JavaScript Style Guide

JavaScript newsletters, news, & podcasts:

- Javascript Jabber
- JavaScript Weekly
- Echo JS
- JavaScript Kicks
- javascript.com
- FiveJS
- JavaScript Live

Learn web animation

General learning:

- Adventures in Web Animations [$][watch]
- Animating With Snap.svg [$][watch]
- Animation in CSS3 and HTML5 [$][watch]
- CSS Animation in the Real World [$][watch]
- Foundation HTML5 Animation with JavaScript [$][read]
- Web Animation using JavaScript: Develop & Design (Develop and Design) [$][read]
- The State Of Animation 2014

Standards/specifications:

- Web Animations

Learn DOM, BOM, & jQuery

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

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, then DOM, or jQuery don't become a black box.

General learning:

- The Document Object Model [read]
- codecademy.com jQuery [watch]
- jQuery Enlightenment [read]
- HTML/JS: Making webpages interactive [watch]
- HTML/JS: Making webpages interactive with jQuery [watch]

Mastering:

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

References/docs:

- MDN Document Object Model
- MDN Event reference
- jQuery Docs
- MDN Browser Object Model
- msdn Document Object Model (DOM)

Standards/specifications:

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

Learn web fonts & icons

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

General learning:

- Typography for the Web [watch][$]
- Learn-web-fonts [read]
- Quick guide to webfonts via @font-face [read]
- Responsive Typography [watch][$]
- Beautiful Web Type A showcase of the best typefaces from the Google web fonts directory. [read]

Learn accessibility

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

General learning:

- Universal Design for Web Applications: Web Applications That Reach Everyone [read][$]
- Web Accessibility: Getting Started [watch][$]
- Introduction to Web Accessibility [read]
- A Web for Everyone [read][$]
- Foundations of UX: Accessibility[watch][$]
- Introduction to Web Accessibility [watch]

Standards/specifications:

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

Learn web/browser API's

The BOM and the DOM are not the only browser API's 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 API's have been called HTML5 api's which confuses their own specifics/standardize with the actual HTML5 specification specify the HTML5 markup language). Note that web or browser API's do include device API's (e.g. `Navigator.getBattery()`) that are available through the browser on tablet and phones devices.

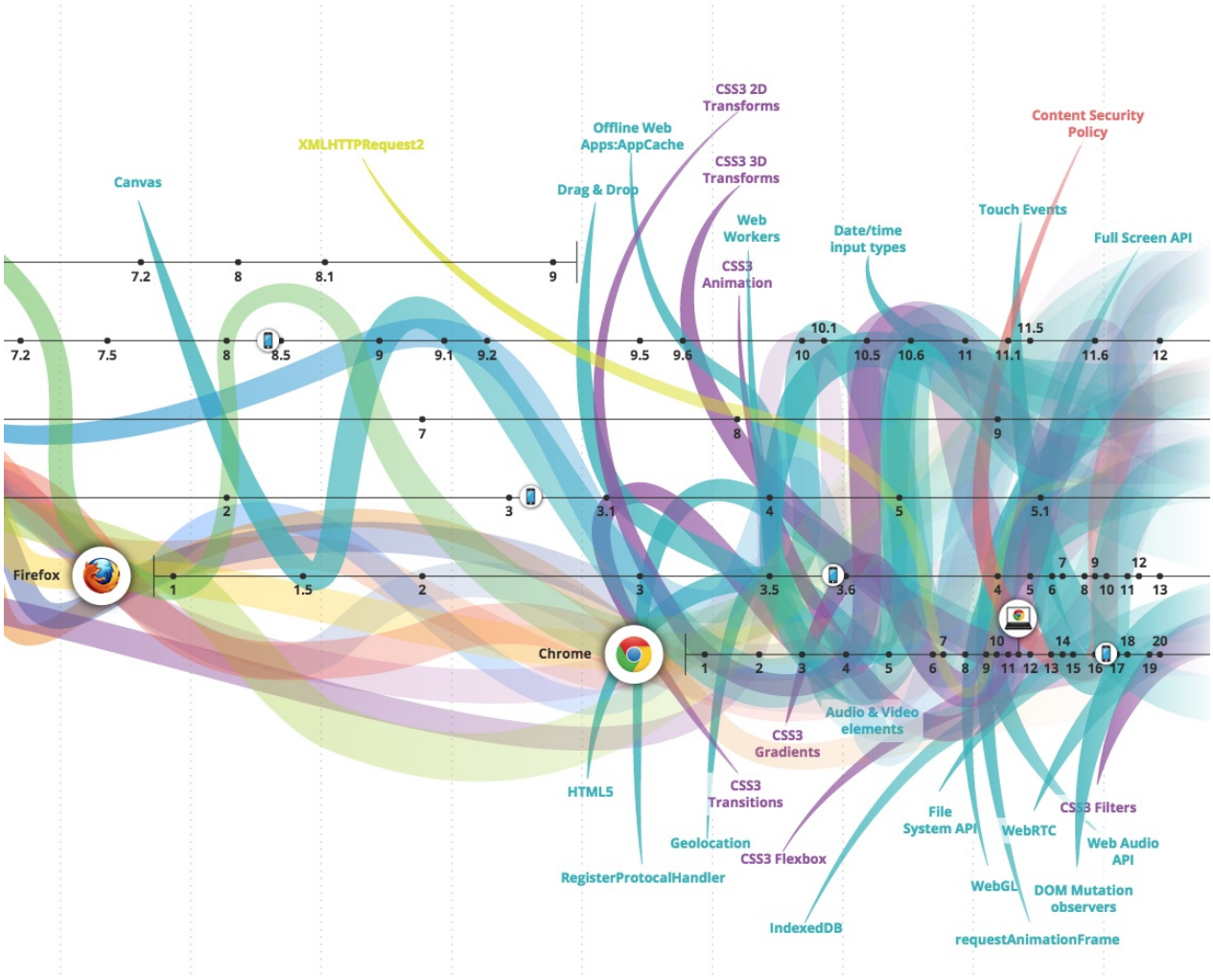

image source: http://www.evolutionoftheweb.com/

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

Learn:

- DIVE INTO HTML5 [read]
- Pro HTML5 Programming [read]

Learn audio:

- HTML5 Canvas [read]
- Fun With Web Audio [watch]
- Add Sound to Your Site With Web Audio [watch]

Learn canvas:

- Web Audio API [read]

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

- MDN Web API reference
- MDN WebAPI - lists device access APIs and other APIs useful for applications.
- MDN Web APIs interface reference - all interfaces, arranged alphabetically.

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:

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

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:

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

References/docs:

- json.org/ [read]

Standards/specifications:

- ECMA-404 The JSON Data Interchange Format
- RFC 7159 The JavaScript Object Notation (JSON) Data Interchange Format

Architecting:

- JSON API

Learn static site generators

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

General learning:

- Static Site Generators [read]

Learn front-end application architecture

- JavaScript Application Design [read][$]
- Build an App with React and Ampersand [watch][$]
- Human JavaScript [read]
- Programming JavaScript Applications [read]
- Building Modern Single-Page Web Applications [watch][$]
- Organizing JavaScript Functionality [watch][$]
- Eloquent JavaScript: modules [read]
- Web UI Architecture [watch][$]
- Field Guide to Web Applications [read]
- UI Architecture [watch][$]
- Terrific [read]
- Nicholas Zakas: Scalable JavaScript Application Architecture [watch]
- Patterns For Large-Scale JavaScript Application Architecture [read]
- A field guide to static apps [read]

Learn Interface/API design

Data (most likly JSON) API's

- Build APIs You Won't Hate [$][read]
- JSON API [read]

JavaScript API's

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

Learn web developer tools

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

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

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

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

Chrome web developer tools docs:

- Per-Panel Documentation
- Command Line API Reference
- Keyboard & UI Shortcuts Reference
- Settings

News/newsletters/podcasts/tips:

- Dev Tips

Learn command line

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

General learning:

- codecademy: Learn the Command Line [watch]
- The Command Line Crash Course [read]
- Meet the Command Line [watch][$]
- Learn Enough Command Line to Be Dangerous [read] [free to $]

Mastering:

- Advanced Command Line Techniques [watch][$]

Learn node.js

> Node.js is an open-source, cross-platform runtime environment for developing server-side web applications. Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, Linux, FreeBSD, NonStop,[3] IBM AIX, IBM System z and IBM i. Its work is hosted and supported by the Node.js Foundation,[4] a collaborative project at Linux Foundation.[5]
>
> 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:

- Introduction to Node.js from Evented Mind [watch][$]
- The Art of Node [read]
- Node.js Basics [watch][$]
- io.js and Node.js Next: Getting Started [watch][$]
- Learn all the nodes [watch]
- The Node Beginner Book [read][$]
- Introduction to Node.js [watch][$]
- Node.js in Practice [read][$]
- Nodeschool.io [code]

Learn module loading and dependency management

General learning:

- Creating JavaScript Modules with Browserify [watch][$]
- Webpack Fundamentals [watch][$]
- browserify-handbook [read]
- Choose ES6 modules Today! [read]

References/docs:

- browserify
- system.js
- webpack

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:

- Bower Fundamentals [watch][$]
- Package Managers: An Introductory Guide For The Uninitiated Front-End Developer [read]
- Up and Running with NPM, the Node Package Manager[watch][$]
- npm Basics [watch][$]
- The npm Book [read]
- The Mystical & Magical SemVer Ranges Used By npm & Bower [read]

References/docs:

- bower
- jspm.io
- npm

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 current modern solution for version control is Git. Learn it!

General learning:

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

Mastering:

- Advanced Git Tutorials [read]
- Pro Git [read]

References/docs:

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

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:

- JavaScript Build Automation With Gulp.js [watch][$]
- Getting Started with Gulp [read][$]
- Rapid Gulp [watch][$]
- Learning Gulp - Getting started with the font end factory [read]
- Gulp Basics [watch][$]
- Using npm as a Task Runner [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 both.

Read:

- Give Grunt the Boot! A Guide to Using npm as a Build Tool
- Task automation with npm run
- Build Tools vs npm Scripts: Why Not Both?
- Using npm as a build system for your next project
- How to Use npm as a Build Tool

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:

- Website Performance [watch][$]
- High Performance Web Sites: Essential Knowledge for Front-End Engineers [read][$]
- Even Faster Web Sites: Performance Best Practices for Web Developers [read][$]
- Web Performance Daybook Volume 2 [read][$]
- JavaScript Performance Rocks [read]
- Performance Calendar [read]
- PageSpeed Insights Rules [read]
- Website Performance Optimization [watch]
- Browser Rendering Optimization [watch]
- Using WebPageTest [read][$]
- Web Performance: The Definitive Guide [read]
- perf.rocks

Learn JS 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:

- Test-Driven JavaScript Development [read][$]
- Testable JavaScript [read][$]
- JavaScript Testing Recipes [read][$]
- Front-End First: Testing and Prototyping JavaScript Apps [watch][$]
- Let's Code: Test-Driven JavaScript [watch][$]

Learn headless browsers

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

- PhantomJS Cookbook [read][$]
- Getting Started with PhantomJS [read][$]
- Rapid PhantomJS [watch][$]
- PhantomJS for Web Automation [watch]

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:

- offlinefirst.org [read]
- Creating HTML5 Offline Web Applications [read]
- Offline First [read]
- Everything you need to know to create offline-first web apps. [read]
- Your first offline web app [read]

Learn security

- Browser Security Handbook [read]
- HTML5 Security Cheatsheet [read]
- Frontend Security [watch]
- Security for Web Developers: Using JavaScript, HTML, and CSS [read][$]
- The Tangled Web: A Guide to Securing Modern Web Applications [read][$]
- Web Security Basics [read]

Learn multi-thing development

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

A site or application can run on a wide range of computers, laptops, tablets and phones, as well as a handful of other things (watches, thermostats, fridges etc...). How you determine what things you'll support and how you will develop to support those things is called a multi-thing development strategy. Below, I list the most common multi-thing development strategies.

- Build a responsive (RWD) web site/app for all things.
- Build a RESS (responsive web design with server-side components) web site/app for all things.
- Build an adaptive/progressively enhanced web site/app for all things.
- Build a website, web app, native app, or hybrid-native app for each individual thing or a grouping of things.
- Attempt to retrofit something you have already built using bits and parts from strategies 1, 2 or 3. This could be as simple as sprinkling in some screen-size agnostic UI parts or attempting to fully support other things with the entire UI.

General learning:

- Adaptive Web Design [read][$]
- Designing with progressive enhancement [read][$]
- Responsive Typography [watch][$]
- Responsive Web Design [watch][$]
- Responsive HTML Email Design [watch][$]
- Designing Multi-Device Experiences: An Ecosystem Approach to User Experiences across Devices [read][$]
- Responsive Web Design Fundamentals [watch]
- Responsive Images [watch]
- Mobile Web Development [watch]

Directed learning

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

Directed learning

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

If you can't afford a directed education, a self directed education using screencasts, books, and articles is a viable alternative to learn front-end development for the self-driven individual.

company	course	price	on site	remote
Iron Yard	Front End Engineering	12,000	multiple locations	
Udacity	Front-End Web Developer Nanodegree	200 monthly	multiple locations	yes
The New York Code + Design Academy	Front End 101	2,000	New York, NY	
Portland Code School	Advanced Front-end Developer Tools	2,000	Portland, OR	
BLOC	Become a Frontend Developer	4,999		yes
Thinkful	Frontend Web Development	300 per month		yes
General Assembly	Frontend Web Development	3,500	multiple locations	
Hackbright Academy	Front-End Web Development	3,000	San Francisco, CA	
HackerYou	Front-end Web Development Immersive	7,000 - 7,910	Toronto, Canada	
The Flatiron School	Introduction to Front-End Web Development	3,500	New York, NY	
Austin Coding Academy	The Front End Track	1,490 per class	Austin, TX	
Codeup	Night Front-End Bootcamp	8,500	San Antonio, Texas	
Betamore	Front-end Web Development	8,500	Baltimore, MD	
Codify Academy	Front-end Web Development	4,000	multiple locations	
DecodeMTL	Learn Front-end Web Development	2,500	Montreal, QC	
gr8code	Front-End Bootcamp	10,000	Tampa, FL	
LearningFuze	Part-Time Front-End Development Course	2,500	Irvine, CA	

Front-end developers to learn from

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

Front-end newsletters, news sites, & podcasts

General front-end newsletters, news, & podcasts:

- shoptalkshow.com
- frontendfront.com
- webtoolsweekly.com
- O'Reilly Web Platform Radar
- The Wdb Platform Podcast
- The Web Ahead
- The Big Web Show
- Fresh Brewed Frontend
- Mobile Web Weekly
- Open Web Platform Daily Digest
- FRONT-END DEV weekly
- Web Design Weekly
- Front-end News in 5 Minutes
- TTL
- Viewsources
- Web Components Weekly

HTML/CSS newsletters:

- HTML 5 Weekly
- CSS Weekly

JavaScript newsletters, news, & podcasts:

- Javascript Jabber
- JavaScript Weekly
- Echo JS
- JavaScript Kicks
- javascript.com
- FiveJS
- JavaScript Live

Part III: Tools

Part three briefly explains and identifies tools of the trade.

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

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

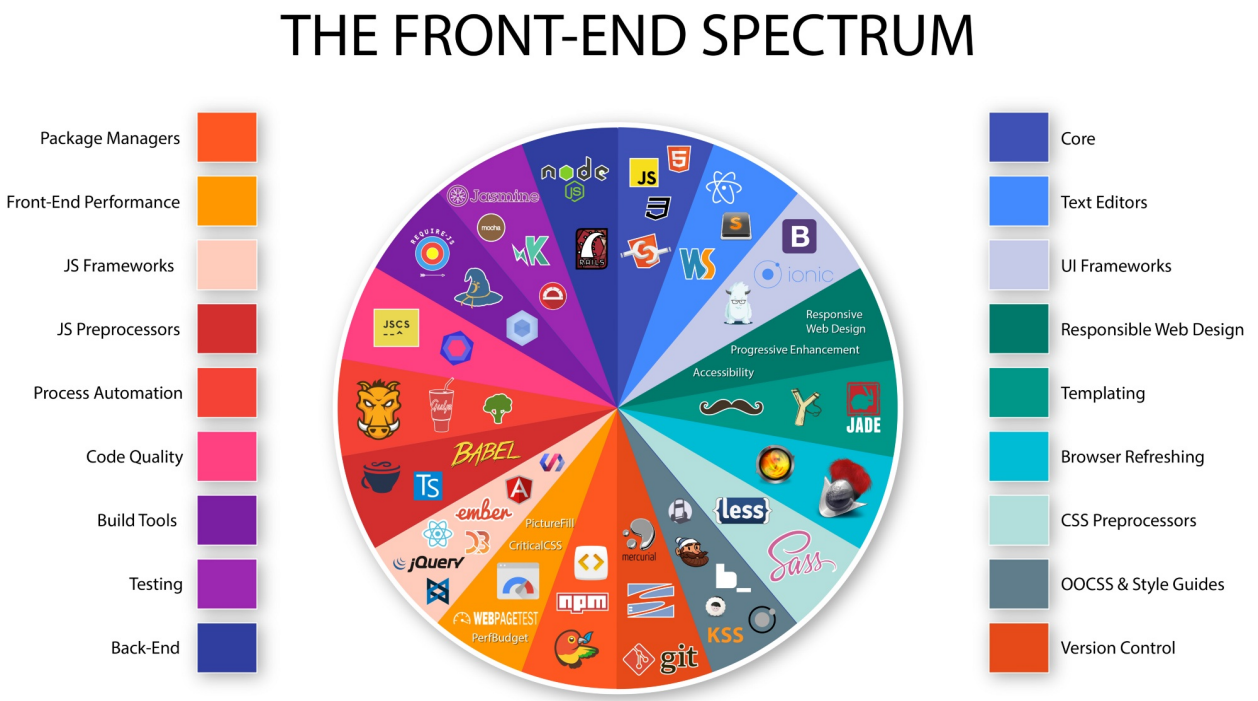

image source: https://medium.com/@withinsight1/the-front-end-spectrum-c0f30998c9f0

General front-end development tools

Development tools:

- screensiz.es
- placehold.it
- codeKit
- prepros
- Browsersync
- ish. 2.0.
- Wraith

Online code editors:

- jsbin.com
- jsfiddle.net
- liveweave.com
- es6fiddle.net
- codepen.io
- Plunker

Tools for finding tools:

- stackshare.io
- javascripting.com
- built with
- microjs.com
- The Tool Box
- unheap.com
- stylesheets.co

Doc/API browsing tools

Tools to browser common developer documents and developer API references.

- devdocs.io
- Dash [OS X, iOS][$]
- Velocity [Windows][$]
- Zeal [Windows, Linux]

SEO Tools

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

Prototyping & wireframing tools

Creating:

- Balsamiq Mockups [$]
- Justinmind [$]
- UXPin [free to $]

Collaboration / presenting:

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

Diagramming tools

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

HTTP/network tools

- Charles [$]
- Fiddler
- Postman
- Chrome DevTools Network Panel

Learn Code Editors

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

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

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

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

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

I personally use and recommend Sublime Text with the following plug-ins.

- Package Control
- AutoFileName
- SublimeLinter
  - SublimeLinter-json
  - SublimeLinter-jshint
  - SublimeLinter-html-tidy
- SideBarEnhancements
- Terminal
- BracketHighlighter
- Color Highlighter
- CSS3

- HTMLAttributes
- StringEncode
- HTML-CSS-JS Prettify

Here are some resources for learning Sublime:

- Sublime Productivity [read][$]
- Sublime Text Power User Book [read][$]
- Sublime Text 3 From Scratch [watch][$]
- Perfect Workflow in Sublime Text 2 [watch][requires login, but free]

If you need a free alternative to Sublime (cost $70) you might consider atom or brackets.

Online code editors:

- codeanywhere [free to $]
- Koding [free to $]
- Cloud9 [free to $]

Sharable/runnable code editors:

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

- liveweave.com
- codepen.io [free to $]
- Plunker
- es6fiddle.net
- jsbin.com [free to $]
- jsfiddle.net

Browser Tools

JS browser utilities:

- URI.js
- platform.js
- history.js
- html2canvas

General reference tools to determine if X browser supports X:

- caniuse.com
- HTML5 Please
- HTML5 test
- Browserscope
- webbrowsercompatibility.com
- iwanttouse.com/
- Platform status
- Browser support for broken/missing images
- Big JS-Compatibility-Table
- jscc.info

Browser development/debug tools:

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

Synchronized browser tools:

- Browsersync

Browser coding tools to determine if X browser supports X:

- Modernizr
- ES Feature Tests

Broad browser polyfills/shims:

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

Browser hosted testing/automation:

- browserstack [$]
- browserling [$]
- Sauce labs [$]
- Selenium
- CrossBrowserTesting.com [$]

Headless browsers:

- PhantomJS
- slimerjs
- TrifleJS

Headless browser automation:

- nightwatchjs
- casperJS
- Nightmare

Browser hacks:

- browserhacks.com

HTML Tools

HTML templates/boilerplates/starter kits:

- HTML5 Boilerplate
- Mobile boilerplate
- Web Starter Kit Boilerplate & Tooling for Multi-Device Development
- dCodes
- HTML5 Bones
- Email-Boilerplate
- Pears

HTML polyfill:

- html5shiv

Transpiling:

- jade
- HAML
- Markdown

References:

- HTML Interfaces Browser Support
- HTML Entity Lookup
- HTMLelement.info
- Elements
- Element attributes
- HTML Arrows

Linting/hinting:

- html5-lint
- HTMLHint
- html-inspector

Optimizer:

- HTML Minifier

Online creation/generation/experimentation tools:

- tablesgenerator.com

Authoring conventions:

- Principles of writing consistent, idiomatic HTML
- HTML code guide

Workflow:

- Emmet

Trending HTML repositories on Github this month:

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

CSS Tools

Desktop & mobile CSS frameworks:

- Semantic UI
- Foundation
- Bootstrap
- Metro UI
- Pure.css
- Concise
- Materialize
- Material Design Lite (MDL)
- Base

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. - http://cssreset.com/

- Eric Meyer’s “Reset CSS” 2.0
- Normalize

Transpiling:

- SASS/SCSS
- stylus
- PostCSS & cssnext
- rework & myth
- pleeease.io

References:

- css3test.com
- css4-selectors.com
- css3clickchart.com
- cssvalues.com
- CSS TRIGGERS...A GAME OF LAYOUT, PAINT, AND COMPOSITE

- MDN CSS reference
- overapi.com CSS cheatsheet

Linting/hinting:

- CSS Lint
- stylelint

Code formatter/beautifier:

- CSScomb
- cssfmt

Optimizer:

- csso
- clear-css
- cssnano

Online creation/generation/experimentation tools:

- Ultimate CSS Gradient Generator
- Enjoy CSS
- CSS matic
- patternify.com
- patternizer.com
- CSS arrow please
- flexplorer
- Flexbox Playground

Architecting CSS:

- oocss [read]
- SMACSS [read][$]
- Atomic Design [read]

Authoring conventions:

- idiomatic-css [read]
- CSS code guide [read]
- cssguidelin.es [read]
- Google HTML/CSS Style Guide

Trending CSS repositories on Github this month:

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

DOM Tools

DOM libraries/frameworks:

- jQuery
- Zepto.js
- keypress
- clipboard.js
- tether.io

DOM performance tools:

- DOMMonster

References:

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

DOM polyfills/shims:

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

Virtual DOM:

- jsdom
- virtual-dom

JavaScript Tools

JS utilities:

- lodash
- underscore.js
- Moment.js
- string.js
- Numeral.js
- accounting.js
- xregexp.com
- Math.js
- wait
- async
- format.js

Transpiling (ESX to ESX):

- Babel

JavaScript compatibility checker:

- http://jscc.info/

Linting/hinting:

- jshint
- eshint
- JSLint
    - jslinterrors.com

Unit testing:

- Mocha
- QUnit
- Jasmine
    - Jest

Testing assertions for unit testing:

- should.js
- Chai

- expect.js

Test spies, stubs, and mocks for unit testing:

- sinon.js

Code style linter:

- JSCS

Code formater/beautifier:

- jsfmt
- esformatter
- js-beautify

Performance testing:

- jsperf
- benchmark.js

Visualization, static analysis, complexity, coverage tools:

- jscomplexity.org
- istanbul
- Blanket.js
- Coveralls [$]
- Plato
- escomplex
- Esprima

Optimizer:

- UglifyJS 2

Obfuscate:

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

Sharable/runnable code editors:

- jsbin.com [free to $]
- jsfiddle.net
- es6fiddle.net

Online Regular expression editors/visual tools:

- regex101
- regexper
- debuggex
- RegExr

Authoring conventions:

- Khan JavaScript Style Guide
- Principles of Writing Consistent, Idiomatic JavaScript
- Airbnb JavaScript Style Guide

Trending JS repositories on Github this month:

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

Most depended upon packages on NPM:

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

Static site generators tools

JS site generators:

- Metalsmith
- harp

JS blog site generators:

- hubpress.io
- Hexo.io

Site generator listings:

- staticsitegenerators.net
- www.staticgen.com

App (desktop, mobile, tablet etc..) tools

Front-end app frameworks:

- AngularJS & Batarang
- Backbone & Marionette
- React & Flux & React Developer Tools
- Vue.js & vue-loader & vue-router
- Ember & Ember Inspector
- Ampersand.js
- Knockout
- Aurelia
- Polymer & Iron Elements & Paper Elements

Full-stack JS app platforms:

- Meteor
- Hood.ie
- MEAN

Mobile web ui/site/app frameworks

These solutions can be used anywhere including in a webview (i.e. web platform and browser engine) app:

- Ratchet
- Kendo UI Mobile
- Mobile Angular UI
- Framework7

Native Hybrid mobile webview (i.e. browser engine driven) frameworks:

These solution typically use Cordova, crosswalk, or a custom webview as bridge to native api's.

- ionic
- onsen.io

Native Hybrid mobile development webview (i.e. browser engine driven) environments/platforms/tools:

These solution typically use Cordova, crosswalk, or a custom webview as bridge to native api's.

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

Native desktop webview (i.e. browser engine driven) app frameworks:

- NW.js
- Electron

Native mobile app frameworks (aka JavaScript Native apps)

These solutions use a JS engine at runtime to inerput JS and bridget that to native api's. No browser engine or webview is used. The ui is constructed from native UI components.

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

References:

- todomvc.com

App seeds/starters/boilerplates:

- React Starter Kit
- Ember starter-kit
- NG6-starter
- Angular 2 Webpack Starter
- hjs-webpack

Scaffolding tools

Client side Scaffolding is concerned with generating a starter template for the application as a whole, rather than generating code to access a database.

- Yeoman
- Slush

Templating tools

Just Templating:

- Mustache.js
- Handlebars
  - htmlbars
- Nunjuncks
- Transparency
- doT.js

Templating and reactive data binding:

- Rivets.js
- paperclip.js
- riot
- vue.js
- ractive.js
- react.js
- RxJS
- knockout

UI widgets & components tools

Desktop & Mobile:

- Kendo UI [free to $]
- Webix [$]
- Semantic UI
- Metro UI
- Bootstrap
- Materialize
- Material ui
- Polymer Paper Elements

Desktop (NW.js and Electron):

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

Mobile focused:

- Ratchet
- Kendo UI Mobile
- Mobile Angular UI
- Framework7

Data visualization (e.g. charts) tools

JS libraries:

- d3
- sigmajs

Widgets & Components:

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

Services:

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

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

General:

- Two.js
- Fabric.js

Canvas:

- Paper.js
- EaselJS

SVG:

- svg.js
- Snap.svg
- Raphaël
- d3

Webgl:

- three.js
- pixi.js

Animation tools

- Velocity.js
- snabbt.js
- TweenJS
- Dynamics.js
- GreenSock-JS

Polyfills/shims:

- web-animations-js

JSON tools

Online editors:

- JSONmate

Query tools:

- DefiantJS
- ObjectPath
- JSON Mask

Generating mock JSON tools:

- JSON Generator
- Mockaroo

Online JSON mocking API tools:

- Mocky
- FillText.com
- JSONPlaceholder
- mockable.io

Local JSON mocking API tools:

- json-server

JSON specifications/schemas:

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

Testing framework tools

- Karma
- Intern

Front-end data storage tools

- YDN-DB
- forerunner
- AlaSQL
- LokiJS
- lovefield
- Dexie.js
- localForage
- pouchdb

Module/package loading tools

- SystemJS
- webpack
- Browserify
- rollup.js

Module/package repository tools

- NPM
- Bower
- jspm.io
- spmjs

Web/cloud/static hosting tools

- AWS [$]
- Heroku [free to $]
- DigitalOcean [$]
- Modulus [$]
- DIVSHOT [free to $]
- netlify [free to $]
- surge [free to $]

Project management & code hosting tools

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

Collaboration & communication tools

- Slack & screenhero [free to $]
- Skype [free to $]
- Google Hangouts

Code/Github collaboration & communication:

- Gitter [free to $]

Content Management Hosted/API tools

API CMS (i.e. content delivery CMS) tools:

- prismic.io [free to $]
- contentful [$]
- Cosmic JS [free to $]

Hosted CMS tools:

- LightCMS [$]
- Surreal CMS [$]
- Page Lime [$]
- Cushy CMS [free to $]

Static CMS tools:

- webhook.com

Back-end as a service (aka BAAS) tools for front-end developers

Data/back-end management as a service:

- Firebase [free to $]
- Parse [free to $]
- kinvey [free'ish to $]
- Back& [free to $]
- Pusher [free to $]

User management as a service:

- UserApp [free to $]
- hull [$]
- auth0 [$]

Offline tools

- upup
- offline.js
- pouchdb
- hood.ie

Security Tools

Coding tool:

- DOMPurify
- XSS

References:

- HTML5 Security Cheatsheet

Tasking (aka build) tools

Tasking/build tools:

- Gulp
- Grunt

Tasking/build and more tools:

- Brunch
- Mimosa

Deployment tools

- FTPLOY [free to $]
- Travis CI [free to $]
- codeship [free to $]
- Bamboo [$]
- Springloops [free to $]
- surge
- sync ninja

Site/app monitoring tools

Uptime:

- pingdom [free to $]
- Uptime Robot
- Uptrends [$]

General:

- New Relic

JavaScript error reporting/monitoring

- Raygun [$]
- errorception [$]
- sentry [free to $]
- {track:js} [$]

Performance tools

Reporting:

- WEIGHTOF.IT
- Web Page Test
- GTmetrix
- Speed Curve [$]
- Chrome Devtools Timeline
- sitespeed.io

JS tools:

- ImageOptim-CLI
- imagemin

Budgeting:

- performancebudget.io