**Front End** **Development Guidebook**

# What Is a Front-End Development?

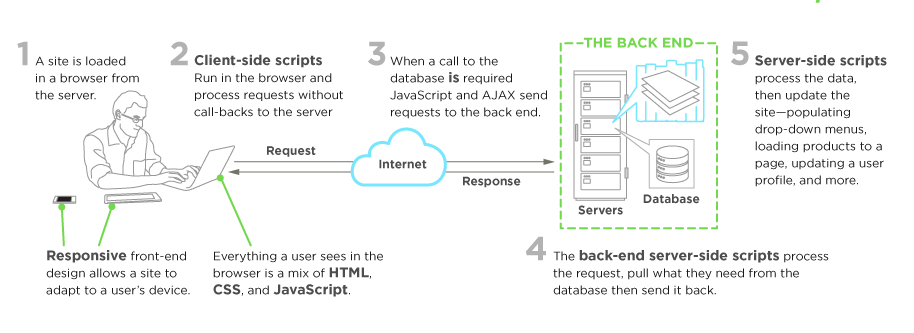
Front-end web development, also known as client-side development is the practice of producing HTML, CSS and JavaScript for a website or Web Application so that a user can see and interact with them directly. The challenge associated with front end development is that the tools and techniques used to create the front end of a website change constantly and so the developer needs to constantly be aware of how the field is developing.

The objective of designing a site is to ensure that when the users open up the site they see the information in a format that is easy to read and relevant. This is further complicated by the fact that users now use a large variety of devices with varying screen sizes and resolutions thus forcing the designer to take into consideration these aspects when designing the site. They need to ensure that their site comes up correctly in different browsers (cross-browser), different operating systems (cross-platform) and different devices (cross-device), which requires careful planning on the side of the developer.

— [*Wikipedia*](https://en.wikipedia.org/wiki/Front-end_web_development)

##### HTML, CSS, & JavaScript:

A front-end developer architects and develops websites and applications using web technologies (i.e., HTML, CSS, DOM, and JavaScript), which run on the [web platform](https://en.wikipedia.org/wiki/Open_Web_Platform) or act as compilation input for non-web platform environments (i.e., [NativeScript](https://www.nativescript.org/)).



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](https://en.wikipedia.org/wiki/Web_browser), [headless browser](https://en.wikipedia.org/wiki/Headless_browser), [WebView](http://developer.telerik.com/featured/what-is-a-webview/), or as compilation input for a native runtime environment. These four run times scenarios are explained below.

##### Web Browsers

A web browser is software used to retrieve, present, and traverse information on the [WWW](https://en.wikipedia.org/wiki/World_Wide_Web). 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](http://www.google.com/chrome/)
* [Internet Explorer](https://en.wikipedia.org/wiki/Internet_Explorer) (Note: not [Edge](http://dev.modern.ie/), referring to IE 9 to IE 11)
* [Firefox](https://www.mozilla.org/firefox/)
* [Safari](http://www.apple.com/safari/)

##### Headless Browsers

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

The most common headless browsers are:

* [PhantomJS](http://phantomjs.org/)
* [slimerjs](http://slimerjs.org/)
* [trifleJS](http://triflejs.org/)

##### Webviews

[Webviews](http://developer.telerik.com/featured/what-is-a-webview/) are used by a native OS, in a native application, to run web pages. Think of a [webview](http://developer.telerik.com/featured/what-is-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](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIWebView_Class/), [android](http://developer.android.com/reference/android/webkit/WebView.html), [windows](https://msdn.microsoft.com/library/windows/apps/windows.ui.xaml.controls.webview.aspx)).

The most common solutions for [webview](http://developer.telerik.com/featured/what-is-a-webview/) development are:

* [Cordova](https://cordova.apache.org/) (typically for native phone/tablet apps)
* [NW.js](https://github.com/nwjs/nw.js) (typically used for desktop apps)
* [Electron](http://electron.atom.io/) (typically used for desktop apps)

##### Native from Web Tech

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

Some examples of these environments are:

* [NativeScript](https://www.nativescript.org/)
* [React Native](https://facebook.github.io/react-native/)

###### NOTES:

Make sure you are clear what what exactly is meant by the "web platform". Read, ["The Web platform: what it is"](http://tess.oconnor.cx/2009/05/what-the-web-platform-is) and read the, ["Open Web Platform"](https://en.wikipedia.org/wiki/Open_Web_Platform) Wikipedia page.

**Front-End Jobs Titles**

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

**Front-End Developer**

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

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

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

**CSS/HTML Developer**

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

**Front-End Web Designer**

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

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

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

**Mobile/Tablet Front-End Developer**

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

**Front-End SEO Expert**

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

**Front-End Accessibility Expert**

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

**Front-End Dev. Ops**

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

**Front-End Testing/QA**

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

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

# Web Technologies Employed by Front-End Developers



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

1. Uniform Resource Locators (aka URLs)
2. Hypertext Transfer Protocol (aka HTTP)
3. Hyper Text Markup Language (aka HTML)
4. Cascading Style Sheets (aka CSS)
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)

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](https://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*](https://en.wikipedia.org/wiki/HTML)

Most relevant specifications / documentation:

* [All W3C HTML Spec](http://www.w3.org/standards/techs/html#w3c_all)
* [The elements of HTML from the Living Standard](https://html.spec.whatwg.org/multipage)
* [Global attributes](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes)
* [HTML 5.2 from W3C](http://w3c.github.io/html/)
* [HTML attribute reference](https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes)
* [HTML element reference](https://developer.mozilla.org/en-US/docs/Web/HTML/Element)
* [The HTML Syntax](https://html.spec.whatwg.org/multipage/syntax.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*](https://en.wikipedia.org/wiki/Cascading_Style_Sheets)

Most relevant specifications / documentation:

* [All W3C CSS Specifications](http://www.w3.org/Style/CSS/current-work#roadmap)
* [Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification](https://drafts.csswg.org/css2/)
* [CSS reference](https://developer.mozilla.org/en-US/docs/Web/CSS/Reference)
* [Selectors Level 3](http://www.w3.org/TR/css3-selectors/)

##### 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*](https://en.wikipedia.org/wiki/Document_Object_Model)

Most relevant specifications / documentation:

* [Document Object Model (DOM) Level 3 Events Specification](https://www.w3.org/TR/DOM-Level-3-Events/)
* [DOM Living Standard](https://dom.spec.whatwg.org/)
* [W3C DOM4](https://www.w3.org/TR/2015/REC-dom-20151119/)

##### JavaScript Programming Language (aka ECMAScript 262)

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

— [*Wikipedia*](https://en.wikipedia.org/wiki/JavaScript)

Most relevant specifications / documentation:

* [ECMAScript® 2017 Language Specification](https://tc39.github.io/ecma262/)

##### 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*](https://developer.mozilla.org/en-US/docs/Web/API)

Most relevant documentation:

* [Web API Interfaces](https://developer.mozilla.org/en-US/docs/Web/API)

##### 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*](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)

Most relevant specifications:

* [Hypertext Transfer Protocol -- HTTP/1.1](https://tools.ietf.org/html/rfc2616)
* [HTTP/2](https://http2.github.io/)

##### 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*](https://en.wikipedia.org/wiki/Uniform_Resource_Locator)

Most relevant specifications:

* [Uniform Resource Locators (URL)](http://www.w3.org/Addressing/URL/url-spec.txt)
* [URL Living Standard](https://url.spec.whatwg.org/)

##### JavaScript Object Notation (aka JSON)

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

— [*Wikipedia*](https://en.wikipedia.org/wiki/JSON)

Most relevant specifications:

* [Introducing JSON](http://json.org/)
* [JSON API](http://jsonapi.org/)
* [The JSON Data Interchange Format](http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf)

##### 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*](https://en.wikipedia.org/wiki/Accessibility)

* [Accessible Rich Internet Applications (WAI-ARIA) Current Status](http://www.w3.org/standards/techs/aria#w3c_all)
* [Web Accessibility Initiative (WAI)](http://www.w3.org/WAI/)
* [Web Content Accessibility Guidelines (WCAG) Current Status](http://www.w3.org/standards/techs/wcag#w3c_all)

# Front-End Dev Skills



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

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

* Content Management Systems (aka CMS)
* Node.js
* Cross-Browser Testing
* Cross-Platform Testing
* Unit Testing
* Cross-Device Testing
* Accessibility / WAI-ARIA
* Search Engine Optimization (aka SEO)
* Interaction or User Interface Design
* User Experience
* Usability
* E-commerce Systems
* Portal Systems
* Wireframing
* CSS Layout / Grids
* DOM Manipulation (e.g., jQuery)
* Mobile Web Performance
* Load Testing
* Performance Testing
* Progressive Enhancement / Graceful Degradation
* Version Control (e.g., GIT)
* MVC / MVVM / MV\*
* Functional Programming
* Data Formats (e.g., JSON, XML)
* Data APIs (e.g Restful API)
* Web Font Embedding
* Scalable Vector Graphics (aka SVG)
* Regular Expressions
* Content Strategy
* Microdata / Microformats
* Task Runners, Build Tools, Process Automation Tools
* Responsive Web Design
* Object-Oriented Programming
* Application Architecture
* Modules
* Dependency Managers
* Package Managers
* JavaScript Animation
* CSS Animation
* Charts / Graphs
* UI Widgets
* Code Quality Testing
* Code Coverage Testing
* Code Complexity Analysis
* Integration Testing
* Command Line / CLI
* Templating Strategies
* Templating Engines
* Single Page Applications
* XHR Requests (aka AJAX)
* Web/Browser Security
* HTML Semantics
* Browser Developer Tools

# Front-End Developers Develop For...

A front-end developer crafts HTML, CSS, and JS that typically runs on the [web platform](http://tess.oconnor.cx/2009/05/what-the-web-platform-is) (e.g. a web browser) delivered from one of the following operating systems (aka OSs):

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

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

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

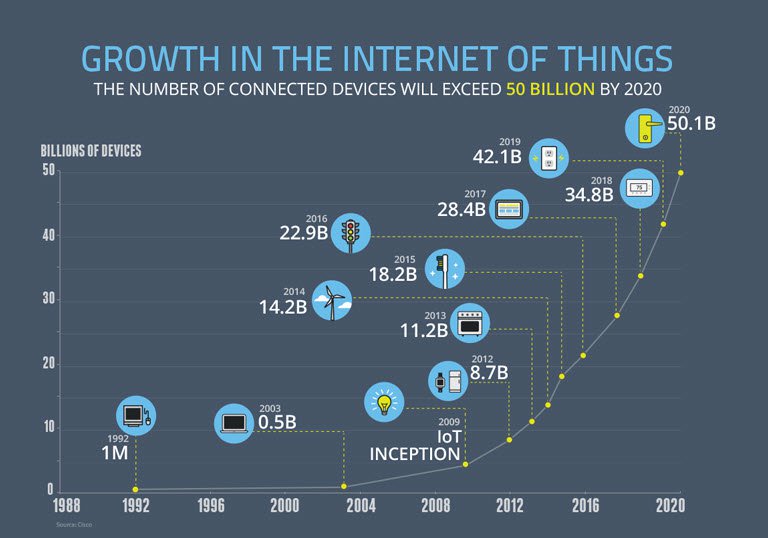


Image source: [*https://www.enterpriseirregulars.com/104084/roundup-internet-things-forecasts-market-estimates-2015/*](https://www.enterpriseirregulars.com/104084/roundup-internet-things-forecasts-market-estimates-2015/)

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](http://outdatedbrowser.com/en)).
* A [headless browser](https://en.wikipedia.org/wiki/Headless_browser) (examples: [phantomJS](http://phantomjs.org/)).
* A [WebView](http://developer.telerik.com/featured/what-is-a-webview/)/browser tab (think iframe) embedded within a native application as a runtime with bridge to native APIs. WebView applications typically contain a UI constructed from web technologies. (i.e., HTML, CSS, and JS). (examples: [Apache Cordova](https://cordova.apache.org/), [NW.js](http://nwjs.io/), [Electron](http://electron.atom.io/))
* 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](https://www.nativescript.org/), [React Native](https://facebook.github.io/react-native/))

# Front-End on a Team

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

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

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

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

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

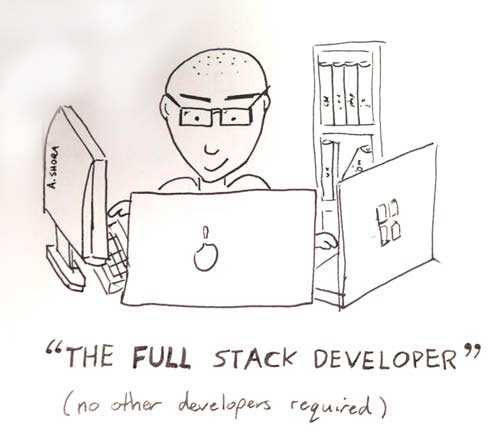
A larger web team might include the following roles not shown above:

* SEO Strategists
* DevOps Engineers
* API Developers
* Database Administrators
* QA Engineers / Testers

###### NOTES:

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

# Generalist/Full-Stack Myth



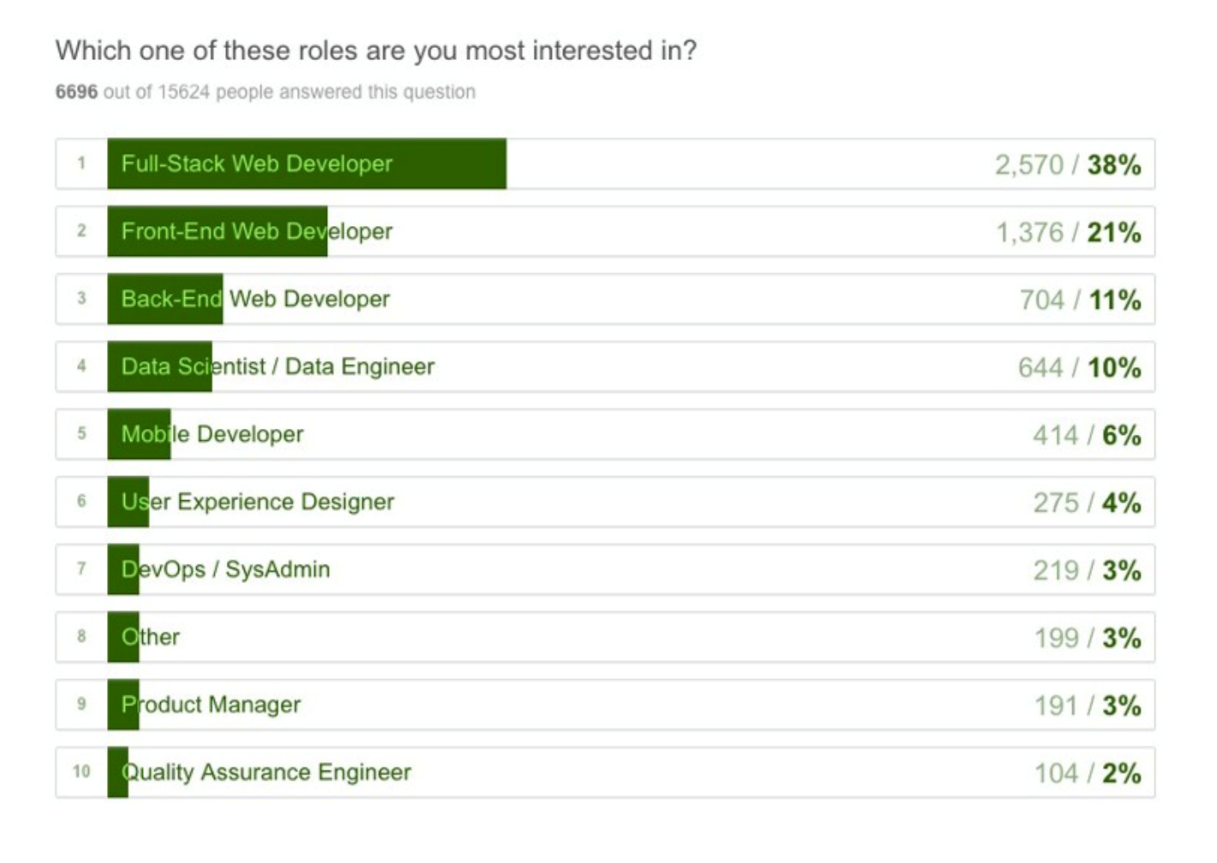
The roles required to design and develop a web solution require a deep skill set and vast experience in the area of visual design, UI/interaction design, front-end development, and back-end development. Any person who can fill one or more of these 4 roles at a professional level is an extremely rare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

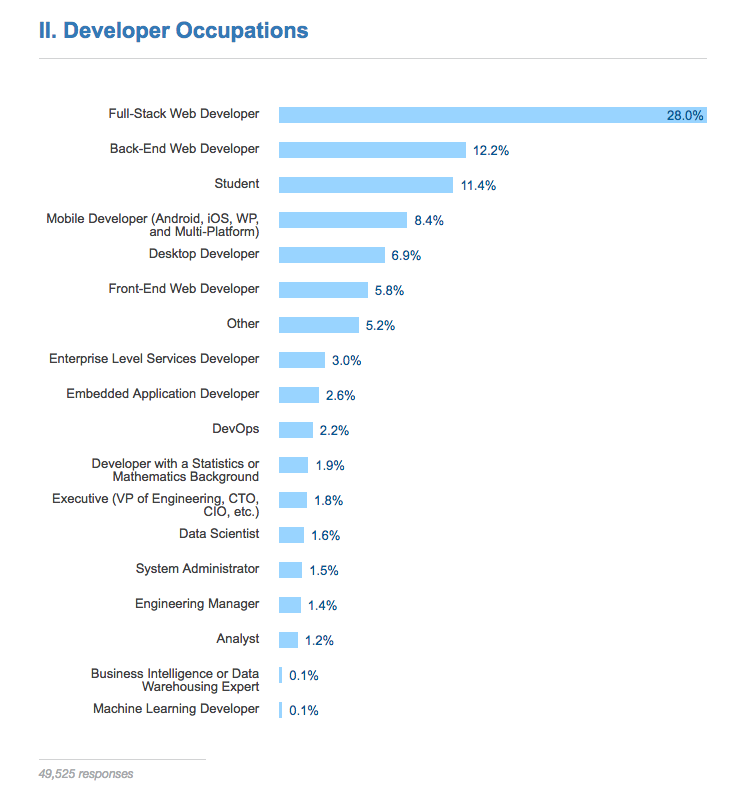
Pragmatically, you should seek to be, or seek to hire, an expert in one of these roles (i.e. Visual Design, Interaction Design/IA, Front-end Dev, Back-end Dev). Those who claim to operate at an expert level at one or more of these roles are exceptionally rare and more than likely mythical.

However, given that JavaScript has infiltrated all layers of a technology stack (e.g. React, node.js, express, couchDB, gulp.js etc...) finding a full-stack JS developer who can code the front-end and back-end is becoming less mythical. Typically, these full stack developers only deal with JavaScript. A developer who can code the front-end, back-end, API, and database isn't as absurd as it once was (excluding visual design, interaction design, and CSS). Still mythical in my opinion, but not as uncommon as it once was. Thus, I wouldn't recommend a developer set out to become a "full stack" developer. In rare situations it can work. But, as a general concept for building a career as a Front-end Developer, I'd focus on front-end technologies.

###### NOTES:

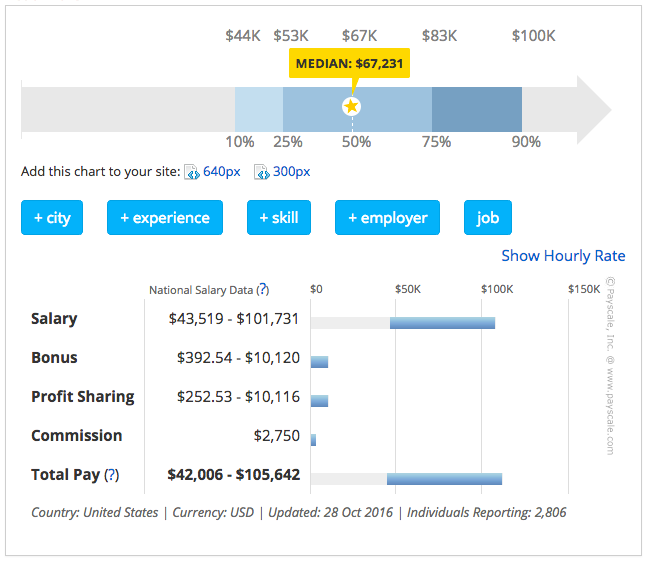
The term "Full-Stack" developer has come to take on several meanings. So many, that not one meaning is clear when the term is used. Just consider the results from the two surveys shown below. These results would lead one to believe that the majority of developers are full-stack developers. But, in my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





# Front-End Salaries

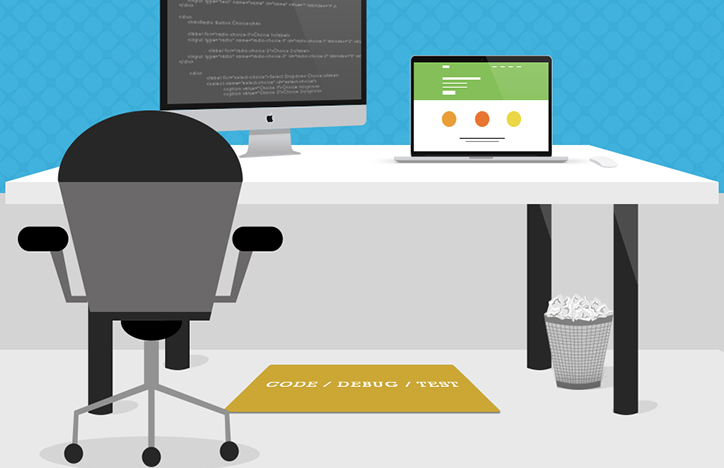
The national average in the U.S for a mid-level front-end developer is around [$75k](http://www.glassdoor.com/Salaries/front-end-web-developer-salary-SRCH_KO0,23.htm). Of course when you first start expect to enter the field at around 35k depending upon location and portfolio.



###### NOTES:

A lead/senior front-end developer/engineer can potentially live wherever they want (i.e., work remotely) and make over $150k a year (visit [angel.co](https://angel.co/jobs), sign-up, review front-end jobs over $150k or examine the salary ranges on [Stack Overflow Jobs](https://stackoverflow.com/jobs?q=front-end&sort=y)).

# How Front-End Developers Are Made



How exactly does one become a front-end developer? Well, it's complicated. Still today you can't go to college and expect to graduate with a degree in front-end engineering. And, I rarely hear of or meet front-end developers who suffered through what is likely a deprecated computer science degree or graphic design degree to end up writing HTML, CSS, and JavaScript professionally. From my perspective, most of the people working on the front-end today, generally seem to be self taught or come from a non accredited program, course, or bootcamp.

If you were to set out today to become a front-end developer I would loosely strive to follow the process outlined below (Part two, ["Learning Front-End Dev"](https://frontendmasters.gitbooks.io/front-end-handbook-2017/content/learning/self-direct-learning.html), dives into more details on learning resources).

1. Learn, roughly, how the web works. Make sure you know the "what" and "where" of Domains, DNS, URLs, HTTP, networks, browsers, servers/hosting, JSON, data APIs, HTML, CSS, DOM, and JavaScript. Don't dive deep on anything, just understand the parts and loosely how they fit together. Focus on the high level outlines for front-end architectures. Start with simple [web pages](https://github.com/h5bp/html5-boilerplate/blob/master/dist/index.html) and briefly study [front-end applications (aka SPAs)](http://developer.telerik.com/featured/front-end-driven-applications-new-approach-applications/)
2. Learn HTML
3. Learn CSS
4. [Learn JavaScript](https://youtu.be/QjKH1J77gjI?list=PL055Epbe6d5bQubu5EWf_kUNA3ef_qbmL)
5. Learn DOM
6. Learn JSON and data APIs
7. Learn the fundamentals of user interface design (i.e. UI patterns, interaction design, user experience design, and usability).
8. Learn CLI/command line
9. Learn the practice of software engineering (i.e., Application design/architecture, templates, Git, testing, monitoring, automating, code quality, development methodologies).
10. Get opinionated and customize your tool box with whatever makes sense to your brain (e.g. Webpack, React, and Redux).
11. Learn Node.js

A short word of advice on learning. [Learn the actual underlying technologies, before learning abstractions.](https://youtu.be/QjKH1J77gjI?list=PL055Epbe6d5bQubu5EWf_kUNA3ef_qbmL)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. Abstractions in the wrong hands can give the appearance of advanced skills, while all the time hiding the fact that a developer has an inferior understanding of the basics or underlying concepts.

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

Lately a lot of non-accredited, expensive, front-end code schools/bootcamps have emerged. These avenues of becoming a front-end developer are typically teacher directed courses, that follow a more traditional style of learning, from an official instructor (i.e., syllabus, test, quizzes, projects, team projects, grades, etc.). Keep in mind, if you are considering an expensive training program, this is the web! Everything you need to learn is on the web for the taking, costing little to nothing. However, if you need someone to tell you how to take and learn what is actually free, and hold you accountable for learning it, you might consider an organized course. Otherwise, I am not aware of any other profession that is practically free for the taking with an internet connection, a hundred dollars a month for screencasting memberships, and a burning desire for knowledge.