The Ultimate Beginners Guide To Web Development — Lessons from the Web Lead of an Amazon Web Service

My friends love learning. This essay is an amalgamation of emails I have sent friends and things I have picked up in my journey to learn web development [1]. By the end of this guide you'll be able to answer the following questions:

- "How might I build a personal website in an afternoon?"
- "How might I start web development as a career?"
- "How might I become a future expert in web development?"

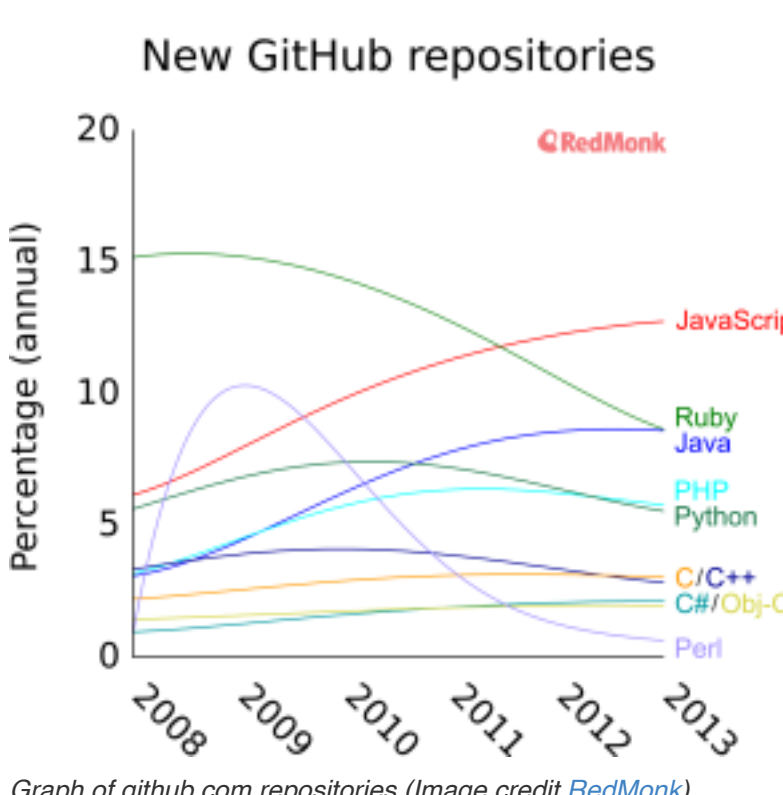
Here's a summary of what you'll learn by section:

1. **How to understand the basics** of a webpage and the internet
2. **How to structure learning** on web development fundamentals
3. **How to engage and grow** your expertise

A lot of these lessons are pulled from how I have learned. At the end of this guide, you'll have the resources necessary to grow in area of your web development expertise (even if it is just for an afternoon).

> Everything in life either grows or dies
> *Anthony Robbins*

Web development is centered around communities of developers sharing and collaborating together to build software. You'll see links to Github, the world's leading open community to share code with "friends, colleagues, classmates, and complete strangers."



Graph of github.com repositories (Image credit RedMonk)

In the remainder of this guide, you'll learn how you can grow from a beginniner to an expert in web development. This is by no means a comprehensive list of resources. Constraints are powerful in forcing maximum results with minimal investment. This guide lists resources useful for results. This guides teaches an approach on learning and exploring that **you**, **web developer**, will be able to use for what you're trying to do.

I want to emphasize action:

- If you have an interest in web development, you should block at least ONE hour for The Basics section and return to reading this guide at that time.
- Else if you're looking to learn web development as a hobby or as a career, you may want to print this document (pdf) and bookmark it for easy reference. The most important part of **learning a craft** is building a consistent habit, for more on structuring habits, see [2].

I have found the hardest part of learning is starting. The sections below are ordered by increasing difficulty.

The Basics

I have sent various emails to friends who want to learn web development. HOWEVER, my results over the past 10 years show about 75% who want to learn web development actually want to do something quite different. They want to build a web presense and register a domain name.

If this is your goal, you should use a site generator that lets you share your site without touching any code

- AboutMe
- Wordpress
- Weebly
- Wix
- Webs

But wait, how does the internet work? Introduction on how the browser and internet (by Google researchers).

Excerpt:

> The internet is a global network of computers. It is millions of computers around the world, all connected. People often think of the internet as a cloud in space. In reality, every computer in the "inter-network", or internet, is connected by actual wires – ethernet cables, phone lines, and fiber optic wiring on the ocean floor!

Detailed links

- What makes Chrome or Firefox or Internet Explorer work? What is HTML and CSS?
  - Follow tutorials on HTML Dog. Alternatively W3Schools
  - Search Quora. Quora is a resource to find guides to start. E.g. On CSS; On Learning HTML / CSS / AJAX
  - Search StackOverflow. StackOverflow is the go to resource for many developers. As you start to search for detailed questions, you'll find a StackOverflow page with the answer. These are often more technical in nature than Quora
- Want a more in-depth look on the internet? A white paper from Stanford

Diving Deep

The resources in this section are meant for those who want to start learning the fundamentals of web development. This is perhaps the **most important part** of this article. These resources are part of a long term learning process that starts with learning how to structure your learning. At the completion of this section, you should have a basic knowledge of how to deconstruct and play with most web sites you use. Furthermore, you will have a set of resources to continually learn web development.

Set up your development environment

Learning a new environment is hard. This is a walk through of my basic environment set up. Whenever I have a new machine. I download SublimeText, Google Chrome, iTerm2

- **Chrome Extensions**: Each of these extensions enhance the Chrome user experience
  - PageSpeed. This is an extension to allow you to use Google's pagespeed insights to make your website faster Google Developer's Guide
  - AngularJS Batarang. This is an extension to debug AngularJS applications
  - EditThisCookie. Cookies are a way to keep information about a website you're using. This is the technology that lets you open gmail without logging in each time
- **Plugins for Sublime**: Each of these plugins enhance the Sublime user experience
  - Package Control
  - jshint
  - sublimelinter
  - sidebarenhancements
  - brackethighlighter
  - theme-soda
  - Angularjs
  - GitGutter
  - DocBlockr
  - Markdown-editor
  - Gist
- **More?**
  - Sublime Text Customization Guides: From Scotch.io; From Hongkiat
  - Web Development Environment Set-up Guides: From @nicolashery; From damln

Note: I don't dive deep on all the tools mentioned as they are beyond the scope of this guide (and others do a much better job of it).

Learn your web console

The console allows you to interact with HTML and Javascript interactively. This will let you change what you see and see what is happening behind the scenes as you go through a web page.

- On ng-book
- By Team Tree House
- By Google Devtools

Read these guides for a basic understanding of the console. You'll return to the console often as you continue to develop. If you master the basics of inspecting HTML and debugging Javascript, you'll reap progressively larger benefits as you continue to learn. Especially as you develop more complex applications, you will have huge dividends in time savings if you master the console.

Knowing how to use the debugger is like having Quicksilver's superpower from X-Men. You can stop Javascript execution, change variables, manipulate HTML, and much more. (Image credit EW)

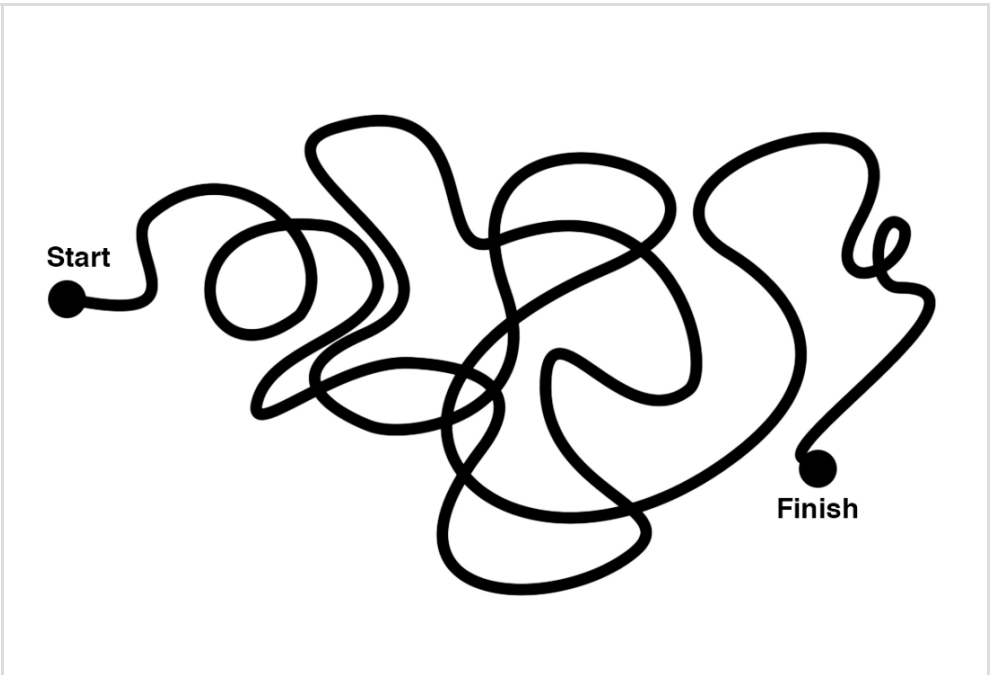
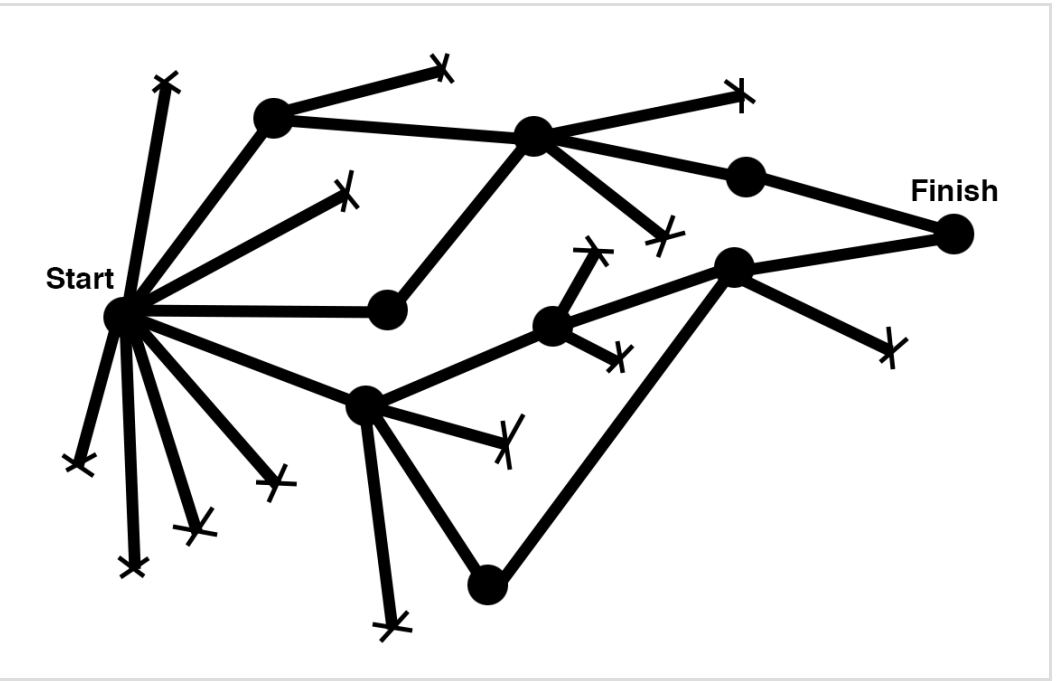
Structure your learning

A critical juncture in learning is a habit to jump start your learning process. For example, I like to learn in the mornings. I describe my morning routine here. When I am learning a new craft, I will block at least 30 minutes every morning to learn and practice this craft prior to leaving for work. For more on building a habit see: [2]

Courses:

- Udacity: HTML & CSS; Javascript; Web Development
- Codeacademy: Basics HTML & CSS; JQuery; Interactive Website
- NetTutorials: Web Development
- CodeCombat: Learn to Code By Playing a Game
- Coursera: Human-Computer Interaction; Web Applications Note: these two must be taken in a course setting

For structured courses or books: **limit what doesn't work**. Use this list as a list to sample from, and sample the course or book to learn what learning style works for you. For a book, you only need to read 30-40 pages to decide if a programming book's author's style works for you. Even if someone else recommended a book, it may not be ideal for you, everyone's learning style and background is different. The same thing applies for courses. Programming books and courses vary in quality of learning depending on your learning style and skillset. For example, Mastering AngularJS is great if you have experience in Javascript and AngularJS. This book jumps from high level concepts for beginners to specific solutions to detailed pain points geared for experts. This type of jump may be disorienting if you're new to AngularJS.

Don't do this	Do this
	

When It comes to your learning, use this technique of sampling then eliminating what doesn't work. Image Credit Julie Zhou, originally used in a fantastic article on Junior versus Senior Designers

Future Experts

This section is related to a base set of resources are critical to specializing in Javascript and AngularJS.

Step 1: On Javascript and Web Technologies

Read these resources as you're developing your project (or are going through the courses listed in the Structured Learning Section).

- Crockford: JavaScript The Good Parts
- Javascript Garden
- Essential Javascript Design Patterns
- Google Developer's Guide for Angular
- Learning JQuery & Bootstrap
- Reintroduction to Javascript from Mozilla
- Secrets of the Javascript Ninja

Step 2: On Angular JS

There are many flavors of Javascript frameworks [3]. The one I am most familiar with is AngularJS. As of this writing, AngularJS has one of most active communities on Github and many developers swear by it. For a showcase of applications built on AngularJS: https://builtwith.angularjs.org/. This framework introduces language constructs that engineers familiar with iOS and Android development use that are traditionally missing in Javascript. AngularJS is the right framework for many teams that want to rapidly iterate and maintain a single page application.

Must reads

- Official Documentation
- O'Reilly AngularJS book
- Mastering AngularJS - for intermediate developers

Diving deeper

- A **very** comprehensive list of resources on AngularJS. This collection is constantly updated
- Tutorials + videos: Organized tutorial via Thinkster.io (uses Egghead.io tutorials) + part 2 with full stack components. Individual video tutorials: https://egghead.io/
- Angular Recipes

Step 3: On becoming a Browser Whisperer

Reads

- On Testing: Beautiful Testing; Application testing via yearofmoo
- Mozilla Developer's Guide to Javascript
- ACM: Best Practices on the Move: Building Web Apps for Mobile Devices while this essay is a collection of recommendations for making websites better for mobile devices, it serves as a nice template for modern performant web applications
- ACM: How Fast is Your Website this essay gives a nice overview for how to understand performance within web applications
- Cross-Origin Resource Sharing: This wikipedia article has a great set of links related to how to understand why this is important and methods to perform Cross Origin Resource Sharing for your website.
- Fundamentals of Algorithms by GeeksForGeeks has my favorite collection of algorithms on the web. It has details on run time for you to remember computer science fundamentals (and a great primer if you are interviewing)

Specific to AngularJS

- Angular UI: set of useful tools built for Angular. For example, this project has a subproject for integration of Twitter's Bootstrap components
- Understand data binding via StackOverflow by one of the creators of Angular
- Improving performance with data bound components with BindOnce
- Approaches on Combining Facebook React with AngularJS [1] ; [2]
- Another approach to improving performance for ngRepeats

What is the hardest part of developing for the web? What are pitfalls that every web developer should know?

The hardest part of developing on the web is constant change. To succeed in delivering a great customer experience, you sometimes feel like you're hitting a flying target with a bow and arrow while on horseback. Browsers / browser versions / frameworks change all the time. For example, something that worked last week may not work this week due to a dependency change in your framework that seemingly has nothing to do with a tool you're using within your project, and this problem only pops up in IE 10.

The easiest pitfall of development is not testing. At the highest levels, testing can be a critical piece to ensure customer success. Entire books are dedicated to the topic of testing and beyond the scope of this beginners guide. Testing may be often overlooked in young development teams. Browser tests are particularly hard because browsers are interpretations of how the web protocols should work. There is no guarantee of consistency between browsers and browser versions.

How can I practice development everyday or reinforce my learning?

Learn widely. Web development is a moving target with many different areas to specialize in. Here are some ways to stay engaged with different communities

- Newsletters: Official Mailing List, AngularJS Daily, ng-newsletter
- Engineering Blogs: Google, AWS, LinkedIn, Twitter, Facebook, Github, High Scalability Blog, AirBNB

Find a mentor. Chances are you know someone who does this professionally or unprofessionally. Ask them for code reviews or problems you're having. Talking aloud is an incredibly powerful way to solve problems. Thinking as an engineer means moving between different levels of abstraction. Someone who has 20 years of experience sees an entirely different set of problems when I ask this person a question than I do when someone asks me the same question.

Build something and stretch your skills. You can choose the tools in what you build every day in your practice. Github has a variety of codebaes for you to play with that are a "checkout" away. Even if you never end up releasing or integrating someone else's project into your development projects, you have the opportunity to play with something cool.

Was this useful to you? Send me an email at frxchen@(google's email provider).com with feedback!

Special thanks to Abhishek Mantha for critical insights on how he learned development, and Amaan Penang, Nate Ngerebara for feedback on early drafts of this guide.

Notes

[1] **Why am I writing this?** I have learned a lot in the past few years and wanted to share a set of resources with my friends who wanted to learn web development and internet friends.

I started web development at 12—at that time, people "surfed the world wide web." For most of my academic and professional career, I was a researcher where I created software as part of my research. I built systems to study social influence on opinion at RAND, mobile environmental impact at UCLA, and behavioral theories on mobile phones at Stanford).

Development at a company or startup requires a different set of skills. I studied Computer Science and researched in Human-Computer Interaction, both of which required development. Yet, the craft of development is very different from academia. The focus of development is much more team oriented. From early 2013, I led the web team on Amazon Zocalo. Zocalo is a secure enterprise sharing and collaboration service that uses AngularJS as its Javascript frontend framework.

[2] **On developing habits for learning** I have found these to be good systems to model for learning new skills:

- Tim Ferriss's 4 Hour Chef - this is a book on learning, that also teaches cooking
- BJ Fogg's Tiny Habits - this is a habit program on learning how to develop simple habits
- Lift - this is a social network for habits

[3] **On picking the right Javascript framework** I focus on AngularJS in this guide. To prevent frontend framework choice paralysis, AngularJS is the only Javascript framework presented. AngularJS has been hugely popular and useful for many developers who use it. If you're interested in what else is out there, I would suggest **TodoMVC** TodoMVC implements the same To-do application using many popular Javascript frontend frameworks. This allows for a comparison and bootstraps the ability for people to prototype in different flavors of frontend frameworks.

More? Subscribe to my "Habits, Design, and Learning" mailing list by clicking here!