Want to learn web development as a beginner, but not sure where to start?

It's hard to know the best way to learn coding, because there are a ton of resources out there. But right now, all you need is the basics of web development– a general explanation with some direction on where to go next.

First, here are the steps that you'll need to follow as a beginner web developer.

Steps to learn web development basics:

1. Learn the basics of how websites work, front-end vs back-end, and using a code editor
2. Learn basic HTML, CSS, and JavaScript
3. Learn tools: package managers, build tools, version control
4. Learn Sass, responsive design, JavaScript frameworks
5. Learn back-end basics: servers and databases, programming languages

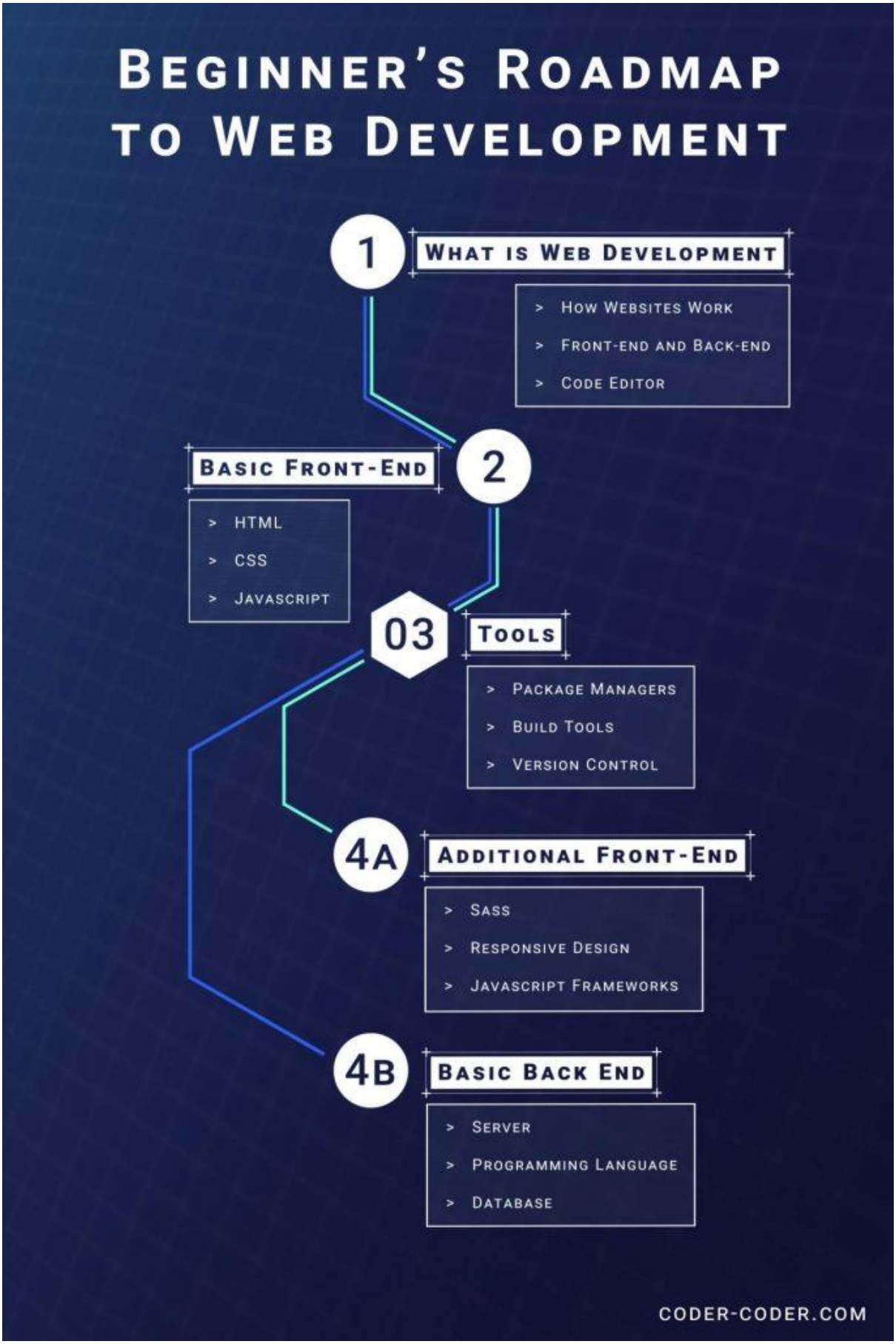
I recommend doing Steps 1, 2, and 3 in order. Then, depending on whether you want to focus on more front-end or back-end, you can do steps 4a or 4b in any order.

I personally think it's good idea for front-end web developers to know at least a bit of back-end, and vice versa. At the very least, learning the basics of both will help you figure out if you like front-end or back-end web development better ☺

Roadmap to learn web development (infographic)

Here's a helpful infographic showing you all the steps in the roadmap to learn web development as a beginner!

Beginner's Roadmap to Web Development



Now, let’s jump right into the first step!

1: What is web development?

Before we get into actual coding, let’s first take a look at some general information on what web development is: how websites work, the difference between front and back-end, and using a code editor.

How do websites work?

All websites, at their most basic, are just a bunch of files that are stored on a computer called a server. This server is connected to the internet. You can then load that website through a browser (like Chrome, Firefox, or Safari) on your computer or your phone. Your browser is also called the client in this situation.

So, every time that you’re on the internet, you (the client) are getting and loading data (like cat pics) from the server, as well as submitting data back to the server (*load moar cat pics!*) This back and forth between the client and the server is the basis of the internet.

Anything that you can access in your browser is something that a web developer built. Some examples are small business websites and blogs on the simpler side, all the way up to very complex web apps like AirBnb, Facebook and Twitter.

What’s the difference between front-end and back-end?

The terms “front end,” “back end,” and “full stack” web developer describe what part of the client/server relationship you’re working with.

“Front end” means that you’re dealing mainly with the client side. It’s called the “front end” because it’s what you can see in the browser. Conversely, the “back end” is the part of the website that you can’t really see, but it handles a lot of the logic and functionality that is necessary for everything to work.

One way you can think about this is that front-end web development is like the “front of house” part of a restaurant. It’s the section where customers come to see and experience the restaurant– the interior decor, seating, and of course, eating the food.

On the other hand, back-end web development is like the “back of house” part of the restaurant. It’s where deliveries and inventory are managed, and the process to create the food all happens. There’s a lot of things behind the scenes that the customers won’t see, but they will experience (and hopefully enjoy) the end product– a delicious meal!

Fun illustrations aside, both front and back end web development serve different but very important functions.

Using a code editor

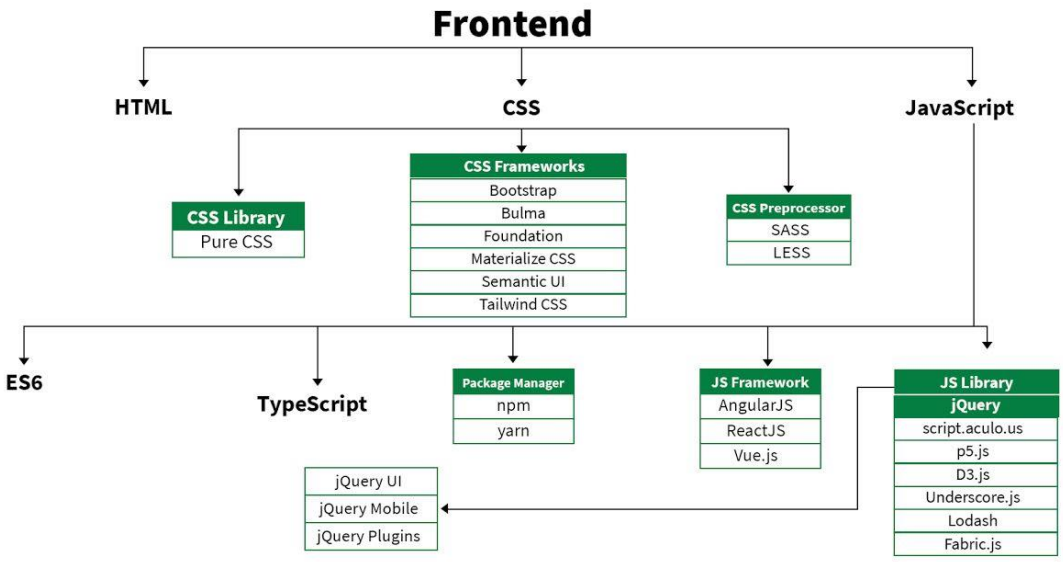
When you build a website, the most essential tool that you will use is your code editor or IDE (Integrated Development Environment). This tool allows you to write the markup and code that will make up the website.

There are quite a few good options out there, but currently the most popular code editor is VS Code. VS Code is a more lightweight version of Visual Studio, Microsoft's main IDE. It's fast, free, easy to use, and you can customize it with themes and extensions. Other code editors are Sublime Text,Atom and Vim

If you're just getting started, though, I'd recommend checking out VS Code, which you can download from their website.

Now that we've covered some of the broader concepts in what web development is, let's get into more of the details– starting with the front end.

2: Basic front-end



The front-end of a website is made up of three types of files: HTML, CSS, and JavaScript. These files are what is loaded in the browser, on the client side.

Let’s take a closer look at each one of them.

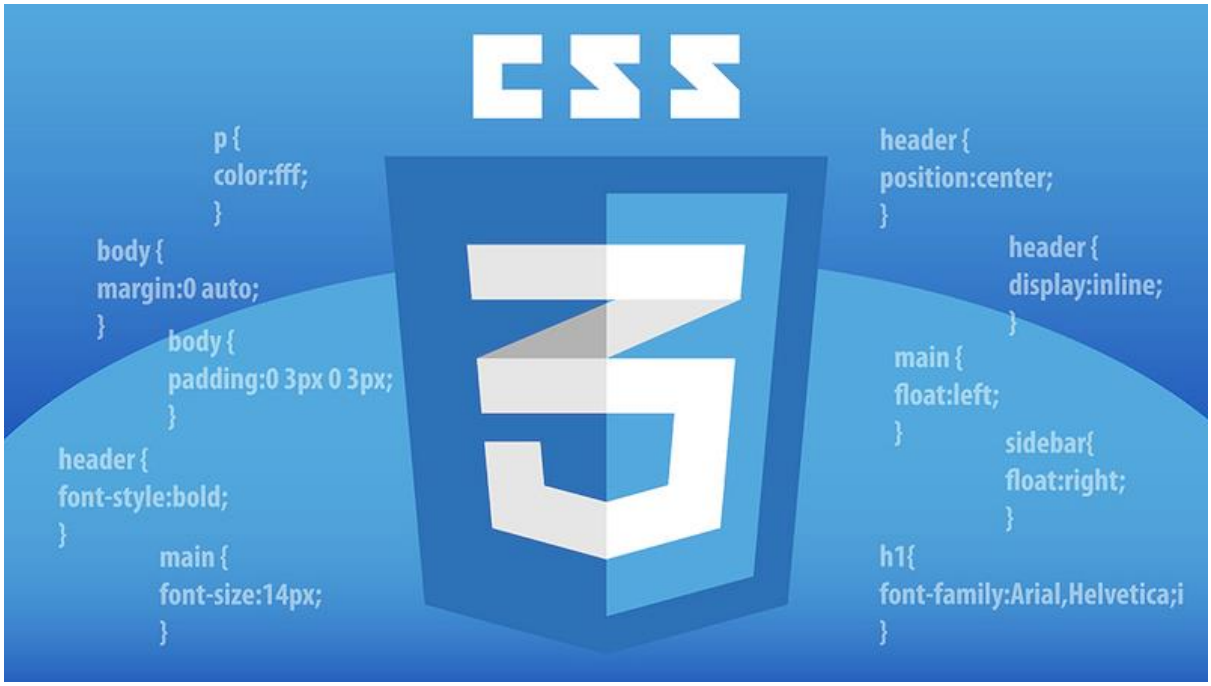
HTML

HTML, or HyperText Markup Language, is the foundation of all websites. It’s the main file type that is loaded in your browser when you look at a website. The HTML file contains all the content on the page, and it uses tags to denote different types of content.

For example, you can use tags to create headline titles, paragraphs, bulleted lists, images, and so on. HTML tags by themselves do have some styles attached, but they are pretty basic, like what you would see in a Word document.

Just getting started with HTML? Check out this tutorial on building a very simple website using just HTML.

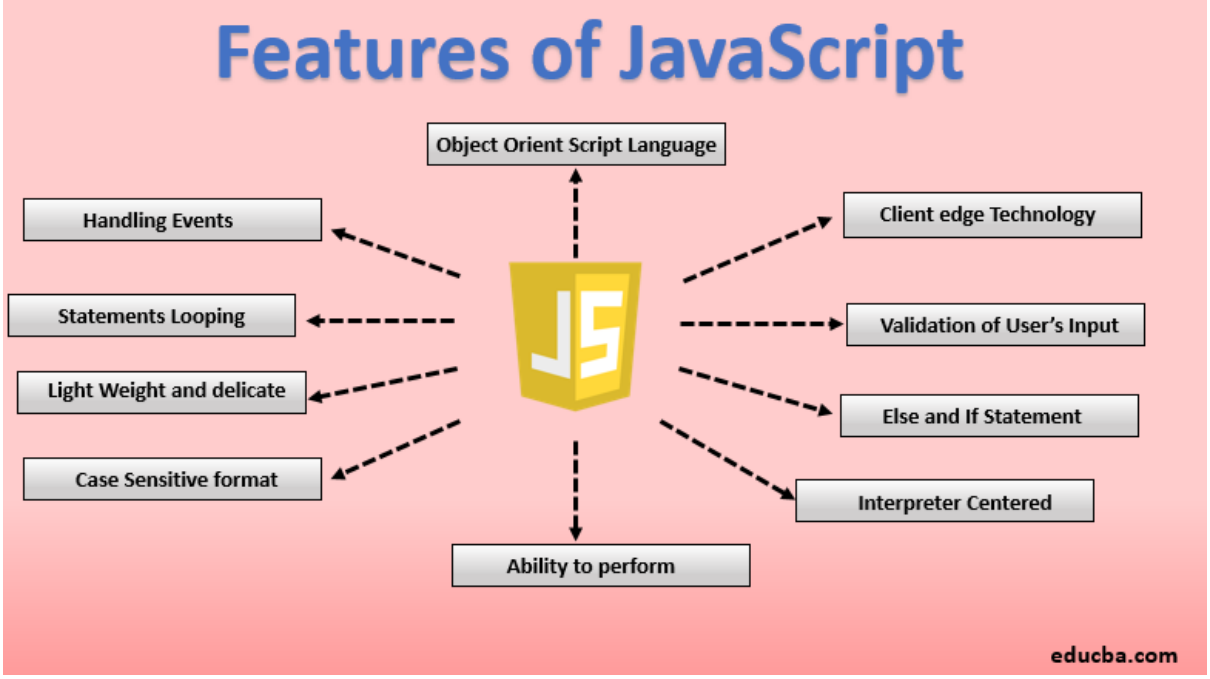
CSS



CSS, or Cascading Style Sheets, lets you style that HTML content so it looks nice and fancy. You can add colors, custom fonts, and layout the elements of your website however you want them to look. You can even create animations and shapes with CSS!

There is a lot of depth to CSS, and sometimes people tend to gloss over it so they can move on to things like JavaScript. However, I can't overestimate the importance of understanding how to convert a design into a website layout using CSS. If you want to specialize in front-end, it's essential to have really solid CSS skills.

JavaScript



JavaScript is a programming language that was designed to run in the browser. Using JavaScript, you can make your website dynamic, meaning it will respond to different inputs from the user, or other sources.

For example, you can build a “Back to Top” button that when the user clicks it, they’ll scroll back up to the top of the page. Or you can build a weather widget that will display today’s weather based on the user’s location in the world.

Especially if you want to develop your skills later on with a JavaScript framework like React, you’ll understand more if you take the time to learn regular vanilla JavaScript first. It’s a really fun language to learn, and there’s so much you can do with it!

Where to learn HTML, CSS and JavaScript?

When people ask me where to learn web development, I will usually recommend they check out one of the following resources: So if you really like learning from videos, here are a few other options:

Zero to master academy is created by Andrei Neagoie, one of the highest rated coding instructors on Udemy. Andrei now has his own course platform with courses covering full-stack web development, JavaScript, Python, React, and even freelancing and coding interviewing. The benefit is that you pay a monthly or yearly fee to get access to every single one of the courses on the ZTM platform.

Andrei is excellent at explaining complex topics, and I highly recommend checking out his complete web Development bootcamp course that takes you from beginner to advanced topics in web development.

If you're more of a fan of one-off video courses, there are some free and paid options:

Wes Bos has free courses on learning Flexbox,css grid and java script that are excellent. I just went through his CSS Grid course, and it was really thorough and also fun. Wes is a great teacher!

Udemy is an online learning platform with a lot of great courses as well. One in particular that you might like is The advance CSS and Sass cource by Jonas Schmedtmann– this paid course covers CSS grid, flexbox, responsive design, and other CSS topics!

Once you've learned the basics, one of the best ways to improve your skills is to practice building projects! One place you can do that is Devprojects by codemetorThey have a collection of free projects that you can submit solutions to and also get feedback from other developers on the platform!

There are also a ton of free video resources on YouTube:

Traversy Media, probably the biggest web development channel out there, has an HTML Crash Course and CSS Crash Cource for beginners.

DesignCourse, a channel focused on web design and front-end, has HTML & CSS TURORIAL for beginners as well.

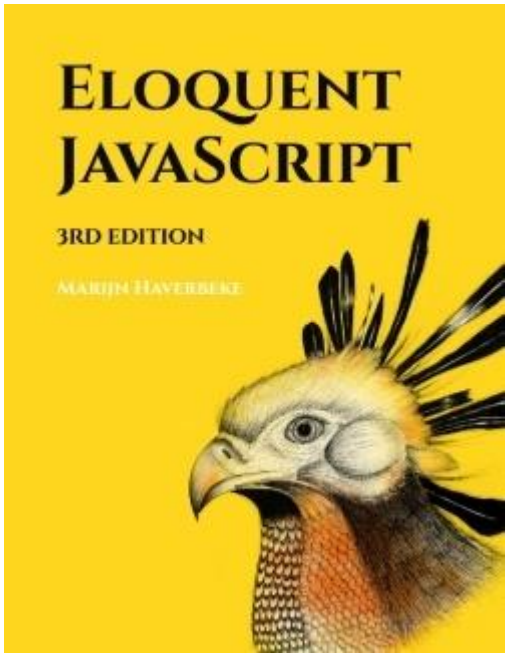
FreeCodeCamp has their own YouTube channel, with videos like a Learn javscript for beginners cource and other in-depth courses.

And of course, I have my own YouTube channel Coder Coder where I create videos on front-end web development tutorials! Check out my 7-part playlist on building a responsive website from scratch with HTML, SCSS, and JavaScript:

Books and articles on web development

If you're more of a reading person, I would highly recommend the following:

The incredibly popular Jon Duckett books on HTML & CSS, and JavaScript & jQuery. These books are not your dense, run-of-the-mill textbooks at all. They are beautifully designed, really well-written, and have lots of photos and images to help teach the material.



Eloquent JavaScript is another book that I really like. You can read it for free on their website or buy a paper copy from Amazon if you like physical books. I have this one myself, and I really like it!

And last but not least, some websites that have great articles and other resources are:

- Mozilla Developer Network
- CSS Tricks
- Smashing Magazine

3: Tools

Let’s get into some other front-end technologies now. As we mentioned, HTML, CSS, and JavaScript are the basic building blocks of front-end web development. In addition to them, there are a few other tools that you’ll want to learn.

Package managers

Package managers are online collections of software, much of it open source. Each piece of software, called a package, is available for you to install and use in your own projects.

You can think about them like plugins– instead of writing everything from scratch, you can use helpful utilities that other people have written already.

The most popular package manager is called npm or Node Package Manager, but you can also use another manager called .Yarn Both are good options to know and use, although it’s probably best to start out with npm.

If you’re curious to learn more, you can read this article on the basic of using npm.

Build tools

Module bundlers and build tools like Webpack, Gulp, or Parcel, are another essential part of the front-end workflow.

On a basic level, these tools run tasks and process files. You can use them to compile your Sass files to CSS, transpile your ES6 JavaScript files down to ES5 for better browser support, run a local web server, and many other helpful tasks.Gulp a task runner, has a suite of npm packages that you can use to compile and process your files Webpack is a super powerful bundler that can do everything Gulp can do plus more. It’s used a ton in JavaScript environments, particularly with JavaScript Frameworks (which we’ll get to in a bit). One down side of Webpack is that it requires a lot of configuration to get up and running, which can be frustrating for beginners.Parcel is a newer bundler like Webpack, but it comes pre-configured out of the box, so you can literally get it going in just a few minute And you won’t have to worry as much about configuring everything.

If you want to learn more about Webpack check out the following YouTube videos:

- Crash course in Webpack by DesignCourse
- 10-part series on Webpack by Colt Steele

Version control

Version control (also called source control) is a system that keeps track of every code change that you make in your project files. You can even revert to a previous change if you make a mistake. It's almost like having infinite save points for your project, and let me tell you, it can be a huge lifesaver.

The most popular version control system is an open source system called Git Using Git, you can store all your files and their change history in collections called repositories.

You may have also heard of Git Hub which is an online hosting company owned by Microsoft where you can store all your Git repositories.

To learn Git and GitHub, GitHub has some online guidest hat explain how to get up and running. Traversy Media also has a YouTube video explaining how Git works.

4a: Additional front-end

Once you have the basics of front-end down, there are some more intermediate skills that you will want to learn. I recommend that you look at the following: Sass, responsive design, and a JavaScript framework.

Sass

Sass is an extension of CSS that makes writing styles more intuitive and modular. It's a really powerful tool. With Sass, you can split up your styles into multiple files for better organization, create variables to store colors and fonts, and use mixins and placeholders to easily reuse styles.

Even if you just utilize some of the basic features, like nesting, you will be able to write your styles more quickly and with less headache.

You can learn more about Sass in this Scotch.io tutorial as well as a You Tube video by Dev Ed.

Responsive design

Responsive design-- ensures that your styles will look good on all devices– desktops, tablets, and mobile phones. The core practices of responsive design include using flexible sizing for elements, as well as utilizing media queries to target styles for specific devices and widths.

For example, instead of setting your content to be a static 400px wide, you can use a media query and set the content to be 50% width on desktop and 100% on mobile.

Building your websites with responsive CSS is a must these days, as mobile traffic is outpacing desktop traffic in many cases.

JavaScript frameworks

Once you have the basics of vanilla JavaScript down, you may want to learn one of the JavaScript frameworks (especially if you want to be a full-stack JavaScript developer).

These frameworks come with pre-built structures and components that allow you to build apps more quickly than if you started from scratch.

Currently, you have three main choices: React, Angular, and Vue.

React (technically a library), was created by Facebook and is the most popular framework right now. You can get started learning by going to the React.js website. If you're interested in a premium React course, both Tyler McGinnins and Wes Bos have great courses for beginners.

Angular was the first big framework, and it was created by Google. It's still very popular, even though it has been surpassed by React recently. You can start learning Angular on their Website.Gary from DesignCourse also has an Angular crash cource on youtube.

Vue is a newer framework created by Evan You, a former Angular developer. While it is smaller in use than React and Angular, it is growing quickly and is also considered easy and fun to use. You can get up and running with it on the Vue website

Which framework should you learn?

You might be wondering now, "Ok, well, which framework is the best?"

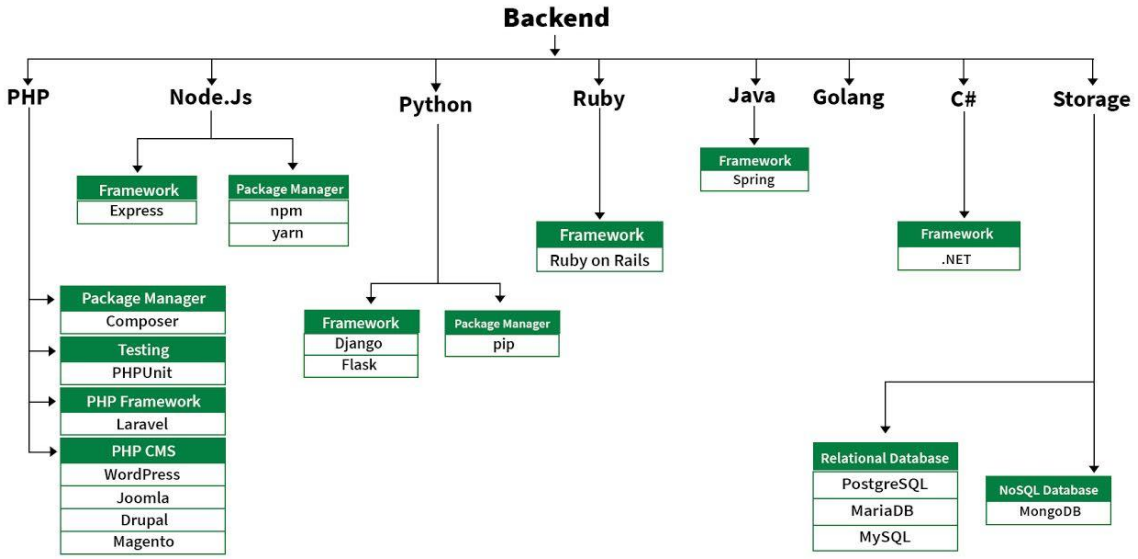
The truth is, they are all good. In web development, there's almost never a single choice that is 100% the best choice for every person and every situation.

Your choice will most likely be determined by your job, or simply by which one you enjoy using the most. If your end goal is to land a job, try researching which framework seems to be the most common in potential job listings.

Don't worry too much about which framework to choose. It's more important that you learn and understand the concepts behind them. Also, once you learn one framework it will be easier to learn other ones (similar to programming languages).

Let's move on now to our last section: back-end web development!

4b: Basic back-end



The back-end, or server-side of web development, is made up of three main components: the server, a server-side programming language, and the database.

Server

As we mentioned at the very beginning, the server is the computer where all the website files, the database, and other components are stored.

Traditional servers run on operating systems such as Linux or Windows. They're considered "centralized" because everything– the website files, back-end code, and data are stored together on the server.

Nowadays there are also serverless architectures, which is a more decentralized type of setup. This type of application splits up those components and leverages third party vendors to handle each of them.

Despite the name, though, you still do need some kind of server, to at least store your website files. Some examples of serverless providers are Amazon Web Services or Netlify.

Serverless setups are popular because they are fast, cheap, and you don't need to worry about server maintenance. They're great for simple static websites that don't require a traditional server-side language. However, for very complex applications the traditional server setup might be a better option.

To learn more about serverless setups, Netlify has an informative blog post that takes you through all the steps to setup a static website with deployment.

Programming language

On the server, you need to use a programming language to write the functions and logic for your application. The server then compiles your code and conveys the result back to the client.

Popular programming languages for the web include PHP, Python, Ruby, C# and Java. There is also a form of server-side JavaScript– Node.js, which is a run-time environment that can run JavaScript code on the server.

There are also frameworks that you can use with each of these server-side languages. Just like the front-end JavaScript frameworks, these back-end frameworks are helpful tools that make building web apps much quicker.

Let's check out a list of the most commonly used programming languages for web development:

C#

C# was developed as Microsoft's competitor to Java. It's used to make web apps with the .NET framework, game development, and can even be used to create mobile apps.

Places to learn C#:
C# Programming Yellow Book by Rob Miles
C# Basics for Beginners on Udemy

Java

Java is one of the most popular programming languages, and is used in web apps as well as to build Android apps.

Places to learn Java:
University of Helsinki's MOOC
The Complete Java Developer Course on Udemy

Node.js

Node.js is a very popular technology (according to Stack Overflow's 2019 developer survey). One thing to note: it isn't technically a server-side language– it's a form of JavaScript that runs on the server using the Express.js framework.

Places to learn Node.js:
Node.js tutorial by Programming with Mosh
Learn Node by Wes Bos

PHP

PHP is the language that powers WordPress, so this might be a good choice if you think you will be working with small business websites, as many of them use WordPress. You can also build web apps with the Laravel framework.

Places to learn PHP:
Introduction to PHP by mmtuts
PHP for Beginners by Edwin Diaz on Udemy

Python

Python is growing in popularity, especially as it is used in data science and machine learning. It's also considered to be good for beginners, as its syntax is simpler than some other languages. If you want to build web apps, you can use the Diango or Flask frameworks.

Places to learn Python:
The Modern Python 3 Bootcamp by Colt Steele on Udemy
LearnPython.org

Ruby

Ruby is another language that has a syntax considered to be beginner-friendly as well as fun to learn. You can build web apps with the framework Ruby on Rails

Places to learn Ruby:
The Odin Project
Ruby on Rails Tutorial by Michael Hartl

Just like with the JavaScript frameworks, there's no #1 best programming language. Your choice should be based on either your personal interest and preference, as well as potential jobs– so do a little research on which might be a good choice *for you*.

Databases

Databases, as the name implies, are where you store information for your website. Most databases use a language called SQL (pronounced "sequel") which stands for "Structured Query Language."

In the database, data is stored in tables, with rows sort of like complex Excel documents. Then you can write queries in SQL in order to create, read, update, and delete data.

The database is run on the server, using servers like Microsoft SQL Server on Windows servers, and MySQL for Linux.

There are also NoSQL databases, which store the data in JSON files as opposed to the traditional tables. One type of NoSQL database is MongoDB, which is often used with React, Angular, and Vue applications.

Some examples of how data is utilized on websites are:

If you have a contact form on your website, you could build the form so that every time someone submits the form, their data is saved onto your database.

You can also user logins on the database, and write logic in the server-side language to handle checking and authenticating the logins.

Some resources to learn the basics of SQL are:

- The Complete SQL Bootcamp by Jose Portilla on Udemy
- SQLBolt

Some tips to leave you with…

Thanks for reading! I sincerely hope that this guide helps you get started learning web development.

A few tips that I have if you are going the self-taught route:

1. Don't try to learn everything at once. Pick one skill to learn at a time.
2. Don't jump around from tutorial to tutorial. As you're learning, it's ok to check out different resources to see which one you like best. But again, pick one and try to go all the way through it.
3. Know that learning web development is a long-term journey. Despite the stories you may have read of people going from zero to landing a web dev job in 3 months, I would aim more at 1 to 2 years to become job ready, if you're starting from the beginning.
4. Just watching a video course or reading a book won't automatically make you an expert. Learning the material is just the first step. Building actual websites and projects (even just demo ones for yourself) will help you to really cement your learning.

Best of luck as you start learning web development!