EVERYTHING YOU NEED TO KNOW ABOUT PROGRAMMING LANGUAGES IN 2016

INTRO

If you want to become a Full Stack Developer, you need to decide which programming language you're going to learn. And if you've done some research and looked into the dozens of web development technologies out there, you've probably noticed two things:

1. There are a ton of differing opinions about the technologies that developers should be focusing on
2. A lot of developers feel really strongly about whatever opinion they have on this topic

The world of programming languages can be difficult to navigate, so we decided to create this guide that can help you figure out which languages you should be learning and how you can learn in a way that makes sense.

But where exactly do you start? And how do you separate what matters from what doesn't?

In order to figure out which programming languages are relevant for your goals, you need to understand all technologies that go into creating modern web applications.

This guide will help you understand everything that's out there for you to learn; what you should learn, when you should learn it, and what you should start learning today to put everything in motion. By the end, you'll understand how to navigate the ever-changing industry while focusing on the things that actually matter. **We'll cover 10 main sections.**

TABLE OF CONTENTS

1. THE SKY-HIGH LOOK AT MODERN WEB APPLICATIONS

There are a number of different aspects of modern web applications that all full-stack developers need to learn. But first, what is a web application?

WEB APPLICATIONS

A web application is a program that generates dynamic (i.e. constantly changing) web pages for each web request that comes in. They often will have forms that accept user input. If you want to do any of the below, you'll need to build a web application:

- Allows users to log in to an account
- Display user-generated data on a page (photo uploads, comments, tweets, etc.)
- Allow different users to interact with the same data

Facebook, Twitter, Instagram, Pinterest, and Yelp are examples of web applications.

DATABASES

Databases store the information that powers our application. Our web application will connect to our database in the access data. There are many different types of databases and ways to store user-generated data.

FRONT-END

The front-end of our web application is very close to the things that users see. Front-end code is usually very visually-oriented. Changing the code at this level will impact what users see.

BACK-END

Back-end code is the glue that allows us to take abstract things, like user requests for a specific page on the internet and database connections, and split things out in a format that is visually processable (front-end). In most cases, it's where the most complex logic lives.

Full-stack developers are expected to understand each of these layers. Each layer also has a variety of different technologies that can be used.

Now, let's do a deep-dive into each of the layers, talk about the problems they need to solve, and work through the different solutions (technologies) that are out there. Once we fully understand the problems, we'll be able to make a decision about the correct tool for the job.

2. FRONT-END LANGUAGES

It's 2016, and people expect the web applications that they visit to have slick user interfaces. By learning things on the front-end, you'll be able to build the type of fluid interfaces that are in high demand. This is a skill that all web developers need to have. Here's exactly what you need to know if you want to become a web developer.

Web browsers (like Google Chrome, Internet Explorer, Safari, etc...) understand 3 different coding languages. Front-end code encompasses all the languages that browsers understand.

HTML

The building block of the internet. It's a way to code up documents in ways that web browsers understand. Every single website on the internet uses HTML in some fashion.

CSS

Describes how the page should look. It's the lipstick of the internet.

JAVASCRIPT

Sometimes you'll want to change the contents of a page or visual representation of a page without doing a full page refresh. In order to do this, you'll need to write code that runs inside a web browser. If you want to be a full-stack developer, you need to know the technologies that web browsers know how to run: JavaScript, HTML, and CSS.

Let's walk through how JavaScript actually works inside web browsers.

VANILLA JAVASCRIPT

Modern web browsers all support the ability to modify the content on the page and use cool ways to get data from different locations without reloading the page using JavaScript.

The Javascript that web browsers understand required many lines of code to do very simple tasks. To get around these shortcomings, an awesome developer implemented a way to make JavaScript easier for web developers. He called it jQuery.

JAVASCRIPT JQUERY

Most modern web applications use jQuery because it makes JavaScript development a lot easier. jQuery essentially allows you to write just a few lines of code to achieve the same thing that would take dozens of lines of code without it. With a little bit of jQuery and JavaScript, you can implement a lot of custom functionality that runs in the browser.

3. DATABASE SYSTEMS

Users expect web applications to keep track of a certain state of the application. If you were using your computer, you would use files to store information. With web applications, we save information that users provide in a database. There are many different types of databases, and if you're going to be a web developer, you're going to need to learn how to use them. In this section, we'll go through a rundown of the different database options out there.

Web applications generally need to store user entered data in some way. Instead of storing data in files, web applications store data into a database that allow users to add and extract data from the database.

There are two main categories of databases. Relational database and NoSQL databases. Both have pros and cons.

RELATIONAL DATABASES

Relational databases use a language called SQL (pronounced sequel), which is the Structured Query Language as a protocol to deal with the database. There are several implementations of SQL that are generally used in production: **MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL.**

You can connect to these databases with almost any programming language pretty easily. Although each implementation has slight quirks that cause it to be slightly different than the other implementations, the implementations are basically identical with small changes that are pretty easy to pick up.

Once you learn one implementation of SQL, you should be able to be comfortable in other ones. It's much like learning English in the UK or the US, there are some differences, but the fundamental language is usually the same.

When you think about the data, it will be stored in a manner similar to an excel sheet: there are tables in the database, which has columns and rows. And similar to keeping your excel columns understandable, SQL requires you to specify the columns headers up-front.

SQL is a tried and true database and is used by many modern web applications. Yahoo! uses PostgreSQL extensively, and Google and Facebook both use MySQL.

One of the big developments in writing database driven applications is known as "ORM"s or "Object Relational Mappings". Systems are in place to convert regular code that is written in a programming language to the exact SQL commands that are needed to communicate with the database. ORMs for SQL databases exist in most programming languages.

ORMs make it very simple to get the data you need out of a database, or push additional data into the database. This means instead of writing SQL code, which is written in a different language than the other code in your application, the code that you write will be in the same programming language that you're already used to and will be a lot easier to read than the SQL commands the database actually needs in order to run.

NOSQL DATABASES

NoSQL databases are a relatively new phenomenon. Instead of having a defined structure, they allow you to have lists of databases and push anything into the list regardless of the structure.

Inconsistencies can exist in how the data is stored. It's generally considered a bad idea to do so. You'll generally want entries to have a consistent structure, but the NoSQL database is flexible and will accept anything.

There are a number of different NoSQL databases: **MongoDB, CouchDB, RavenDB, Redis.**

Currently, we don't really have a standard for how data should be stored inside of NoSQL databases, so each NoSQL database will be considerably different. That means if you learn one type of NoSQL database, it won't translate well to any other implementations.

MongoDB is one of the most popular implementations of NoSQL databases. It has a few features that allow for some interesting properties while scaling up an application to millions of users with hundreds of millions of database requests (or building applications at so-called WebScale).

Mongo's features make it a good solution for ad servers and other applications that need to handle hundreds of millions of requests a day. On the flip side, it makes it a poor choice for things like marketplaces, eCommerce sites, or other sites that handle less than 10 million requests per day (which is most web applications).

NOSQL USAGE IN THE REAL WORLD

For every 10 web developers who write applications with SQL, there will be around 1 web developer who is writing applications with NoSQL.

In short: if you're a web developer you're probably going to need to work with SQL databases eventually, so you better start with learning what everyone else is using already.

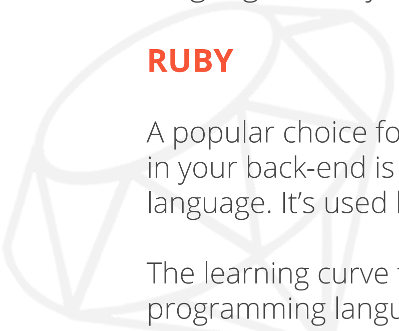
4. BACK-END LANGUAGES

When you're building any web application, you need to have a server to power and make it dynamic. The server that powers our web application will run on a specific technology and will likely perform most of the work that the web application needs to do. Because of this, one of the most critical components of a web developer's toolkit will be a programming language. The issue is that there are so many programming languages out there, and picking the right one can feel intimidating. In this section, we'll give you a rundown of key programming languages and help you understand what you need to know.

Programming languages and web frameworks will allow you to write code that will allow you to store information in a database, retrieve the data, and determine what a user should see (like which avatar should be in the top-nav of the website). First, you need to pick a programming language to use inside a server. Each programming language has popular web application frameworks as well.

The most popular back-end programming languages that you'll typically hear about are:

A POPULAR CHOICE FOR A BACK-END LANGUAGE IS RUBY



RUBY

A popular choice for a programming language to use in your back-end is ruby. It's a good option to choose for a first programming language. It's used by companies like AirBnB, BaseCamp, Groupon and Github.

The learning curve to master Ruby is less steep in comparison to other programming languages, and after just a short period, you're able to write and execute Ruby programs.Ruby really stays out of your way as a developer, and if you're going to be learning advanced algorithms it's a great language to work with them. Think of it as the springboard to learning other programming languages later on at an accelerated pace.

NODEJS (JAVASCRIPT)

You can also use JavaScript in the back-end as well. When JavaScript is used in the back-end it's called "NodeJS". NodeJS is exactly the same as JavaScript, except it indicates that it is JavaScript that is not running in the front-end.

JavaScript has a number of complicated language features that make it a good choice to learn once you've mastered a different programming language.

Things that make JavaScript hard to learn as a first language are anonymous functions, callbacks, prototypical inheritance. Complicated topics like these are deeply ingrained into JavaScript, so even to understand basic code you need to understand a few very complex computer science topics.

It's much easier to learn JavaScript after becoming a competent programmer in other languages first.

PHP

PHP came out around the same time as ruby, and saw a lot of early success. Now, however, companies that run PHP code are generally using older technologies instead of cutting edge ones. PHP is very polarizing. People either love it, or people think that it's an old, dying language.

Wordpress is built off PHP, which has played a significant role in its success. Because of this, many PHP developers are hired to modify PHP blogs. These jobs can be interesting at first, but eventually, they become really mundane. Working in Wordpress can be boring because it doesn't give you the opportunity to use creative problem solving and do real programming.

PYTHON

Python has become popular in data science and scientific computing. It's used by companies like Google and Yahoo!

Python's syntax looks very familiar to Ruby, so if you learn one language it's pretty easy to pick up the other. There's one aspect of Python that developers like to discuss, and it's that it is "white space sensitive." This means that the space character needs to be perfect for your code to work. People tend to either love this or hate it.

C++

C++ is a low-level system language that requires developers to really think about the hardware level of the code. Programmers working with C++ need to think about where data gets stored. This is called memory management (sometimes pointer arithmetic), and it's very difficult to learn. But once you understand it, you understand how computers work at a very deep level.

C++ is based off of the C language, which is very similar to C++. Most other programming languages are build in the C language. However, there really aren't that many jobs for developers working with C or C++. Still, it's an important part of the programming language discussion.

JAVA

Java is the language that backs much of the non-consumer facing aspects of large corporations.

Java runs on top of the JVM, which is the Java Virtual Machine. The JVM is incredibly robust and scales very well. Java's syntax is heavily based on C++, but it provides garbage collection with C++ style syntax. This generally makes Java easier to write and work with than C++.

.NET (C#)

.NET (C#) is used a lot by big companies that build desktop applications. C#'s syntax is very similar to Java's syntax. They're basically identical. .NET has an awesome IDE (integrated developer environment) called Visual Studio. In general, it makes developers' lives easier.

There are some other specialized languages that are used by developers who are very specialized, but we'll get into those later on in this guide.

That was a lot. But try not to be intimidated by the number of languages out there. Once you learn a first language, you'll find that learning your additional languages will come so much easier. So don't worry about picking an incorrect language, it won't set you back too far. Your primary goal should be to just get started and pick something that will get you coding.

We'll get more into how exactly to do that in the last section of this guide. For now, keep reading to get a rundown of web application frameworks.

Ready to start coding right now? Click the button below to sign up for our free 2-week intro course.

GET 2 WEEKS FREE

5. WEB APPLICATION FRAMEWORKS

Web application frameworks help developers do things that happen many times in most web applications with only a few lines of code (instead of dozens or even hundreds of lines of code.) These frameworks make life easier as a developer because they allow you to focus on features that you care about, rather than mundane, repetitive tasks.

If you're building a web application, you should definitely use a web application framework to help you do the work. Just like with programming languages, you'll need to choose a framework to learn. In general, if you've picked a programming language, there's a common web application framework that you should learn, too.

The main web application framework that developers are using in 2016 include:

RUBY ON RAILS

Ruby on Rails is a web framework that focuses on developer happiness. Its core principle is Convention over Configuration. What does this mean?

Even though there are many ways you can do a particular thing, there is a certain way that the ruby community will deem "correct." If you do things the correct way, your life will be easier. You'll find that once you understand how one application works, you'll notice that other applications that are written by other developers will be organized in a remarkably similar manner.

EXPRESSJS

ExpressJS is a web framework that runs on NodeJS. It's an un-opinionated web framework, meaning that it doesn't make your life easier if you do things the "right way."

Overall, remember that once you learn one web framework, you'll be able to learn other frameworks very quickly. The hard stuff to learn about web application development applies to all frameworks equally: you need to understand databases, REST, HTTP requests and more.

If you're good at Rails, you'll be able to pick up Node/ExpressJS very quickly. If you're good at ExpressJS, you'll be able to pick up Rails quickly.

DJANGO

Django is an MVC framework, very similar to Ruby on Rails. A ruby on rails developer could pick-up the basics of Django fairly quickly and the reverse is true too.

There are additional frameworks that run inside web browsers. *Keep reading to learn about what developers call "front-end frameworks."*

6. FRONT-END FRAMEWORKS AND TECHNOLOGIES

Earlier, we explained how JavaScript is essential to making responsive, attractive, and fluid web applications. Front-end frameworks provide a way to organize JavaScript code. Front-end frameworks make writing JavaScript code a bit easier and give you patterns that you can follow. Learning the core JavaScript without a framework first, though, is really important.

JavaScript without a framework uses a method known as DOM manipulation, which is basically changing the content of the page dynamically.

Here are the key front-end frameworks that developers are working with in 2016:

ANGULARJS (NOW ANGULARJS 2)

Angular is a framework that is backed by Google and it's used by a variety of companies. Large companies and the Fortune 500 use it as well as startups.

Angular 2 is a complete rewrite of the Angular framework and old Angular code won't run in this version.

REACTJS

React is a cutting edge framework that was released by Facebook. It uses an advanced concept of flux to manage the information that can change during the lifecycle of a web page. React is really cutting edge and is constantly changing. New React technologies keep coming out each day.

FRONT-END FRAMEWORKS ALLOW DEVELOPERS TO BETTER ORGANIZE THEIR CODE.

EMBERJS

Ember is a technology that is built using ruby on rails philosophies (convention over configuration, etc.) and is intuitive for rails developers to learn. Ember is built by the same people who are core contributors to the rails framework and this framework has a bright future as well.

Front-end frameworks overall allow developers to better organize their code, however, it's unclear which frameworks will be most popular in the future and many developers debate which framework is best.

Now that we've covered a ton of the different technologies that developers are using, we want to talk about them in the context of real-world developer jobs. *Keep reading to learn about the technology profiles of developers at different types of companies all over the world.*

7. PROFESSIONAL DEVELOPER PROFILES

With so many different types of technologies out there, it can be overwhelming to figure out where to start. There are some core technology profiles of developers who work at typical companies that you should probably know about. The key is to understand that programming is about learning new technologies, languages, and frameworks every single day. Because everything is changing all of the time, it's far more important to learn how to learn programming, rather than learn specific programming technologies.

That said, it can be helpful to understand which key technology are typically used at different types of companies. This doesn't mean that you should definitely pick a specific language just because you think that you want to work at a certain type of company. But it's helpful to understand the landscape so that you can keep yourself educated and aware of the trends and changes that are going on in the industry.

Here are what developers at different types of companies are generally working with:

SOFTWARE DEVELOPER AT A BIG CORPORATION

Big corporations typically work with these programming languages:

Java

- Java is used by a lot of large corporations. Java Spring, as well as Hibernate, are particularly in demand.
- Groovy and Grails are built off of Java, but they don't have high demand in most organizations.

C#

- Big companies that ask their dev teams to build Desktop applications that run on Windows generally use C#. It's one of the go-to languages of companies that are nervous about open sources technology and only want to use technology that is backed by a big player like Microsoft.

VB.net

- It's common in the financial services industry for developers to use Visual Basic.NET.

WEB DEVELOPER AT STARTUPS AND MID-SIZED COMPANIES

Startups and mid-sized companies typically work with the following programming languages:

Ruby

- Written to optimize for developer happiness and effectiveness of building applications quickly. Because ruby was written to make coder's lives easier it's a wonderful language to code in. It's used by products like Airbnb, Basecamp, Groupon, and GitHub.

- Whether you're looking to build web applications to launch your startup idea or work for a start-up, small, or mid-sized company, Ruby is a safe bet.
- There are also a ton of boutique consulting agencies that build out web application rapidly for Fortune 500 companies, using the same technologies as startups. Most of these use the Ruby framework.

Python

- Python is a coding language that has become popular in data science and scientific computing. It's used by companies like Google and Yahoo!

JavaScript

- JavaScript is the language of web browsers. It was famously designed in 10 days and quickly became ubiquitous because it was distributed through web browsers - all you need is a web browser and you can run JavaScript code. It's the language that runs the front-end code of websites that powers advanced user interaction. The technology NodeJS also allows you to run the code in servers as well.

APP DEVELOPER

Developers working on iPhone apps typically work with these languages:

Objective-C & Swift (and now Swift 2)

- New iPhone applications are built primarily in the Swift programming language, but you may need to use a little bit of Objective-C. Swift is a fairly new technology, and only a few years ago, it didn't exist. If you wanted to write an iPhone application without using Swift, you would have to use Objective-C, which is a bit of a clunky, unusual programming language.

Developers working on Android apps typically work with these languages:

Java

- Developers building Android apps use Java and the Android SDK.

All of that said, the individual technologies do not matter too much. Programming is about learning new technologies, languages, and frameworks every single day. The key is to learn one coding language really well, because once you do that, learning additional coding languages becomes a lot easier.

Keep reading on to the next section to figure out what you should be learning and why.

8. HOW TO FIGURE OUT WHAT TO LEARN

All the time, people ask us: “which programming language should I learn first?”

But if you’re thinking about making the correct long-term decision that will produce the least amount of friction and ultimately give you the highest chance of success, you should be asking an entirely different question. The right question to ask is:

What is the right tool for the job at hand?

The key to becoming an in-demand programmer is understanding the answer to this. And when you evaluate technologies, you should evaluate different technologies based on the merits of the technology- not the hype.

EVALUATE TECHNOLOGIES BASED ON MERITS, NOT HYPE.

Let’s think about a practical problem that you could face outside of the coding world.

PROBLEM: YOU NEED TO BUILD A BOOKSHELF

Potential solutions:

1. First, we could use a wide variety of different tools. We could use a saw to cut the wood, nails to connect the pieces of wood, a hammer to drive the nails through the wood, and wood glue in places that aren’t appropriate for nails. A drill could come in handy too, depending on the blueprints that we’re using.

2. We could just use a hammer. Instead of cutting the wood with a saw, we could use the opposite side of the hammer to split the pieces of wood into smaller pieces. We could connect the pieces of wood with nails we drive into the wood with the hammer. We could avoid using blueprints because those would just be another thing we’d have to learn to use.

Now, let’s return to our argument in this case.

“If all I need to learn is how to use a single tool, like a hammer, won’t building bookshelves be 4 times as easy for me as someone who uses a hammer, saw/glue, and blueprints? Doesn’t that mean, all things considered, if I work as hard as someone who uses all the other tools combined, I can produce 4 times the number of bookshelves?”

Obviously, this argument is completely ludicrous. Figuring out how to cut wood with a hammer is probably possible, but why would you want to do it, given there are better tools out there?

As someone starting out, it can be difficult to determine the right tool for the job. To do so, you need to understand the problems at hand and the problems that each tool solves. You can't do this by chasing the hot technology of the week. Instead, you need to figure out how to apply the different technologies in your developer toolkit at the right time to solve the different problems that you'll face as a developer.

In other words, you need to learn how to become an adaptable programmer. Keep reading to learn about the steps you need to take to make that a reality.

Want to start becoming an adaptable programmer? Start coding for free today by signing up for the 2-week Firehose Ruby intro course.

GET 2 WEEKS FREE

9. HOW TO BECOME AN ADAPTABLE PROGRAMMER

Adaptable programmers might not know every single one of the latest technologies, but they do know how to break down the solution to a problem into a series of steps that can be completed by a computer program. They know how to build a utilize their developer toolkit.

Employers invest in entry-level programmers for the long haul, and they hope that new hires will stay with the company for a year or longer. Only having one tool in your toolbox might be acceptable today, but what about a year from now when things are different.

Rather than learn the coolest new buzzword, smart programmers play the game a little differently, by maximizing for adaptability.

WHAT ALL IN-DEMAND PROGRAMMERS KNOW

All in-demand programmers are familiar with the fundamentals of programming. These are things that will likely stay true regardless of the changes in the programming ecosystem.

Since technology evolves so quickly, hiring someone who is excels at one specific fad is a lot less valuable than hiring someone who is adaptable and capable of working in a variety of environments.

IF YOU HAVE A STRONG FOUNDATION IN PLACE, YOU'LL BE ABLE TO LEARN WHATEVER IS THROWN AT YOU.

Just like a skyscraper, it's the foundation that allows you to eventually reach incredible heights as a programmer. This is what interviewers really care about. They know that if you have a strong foundation in place, you'll be able to learn whatever is thrown at you.

Let's break down the skills that you need to acquire to become a coveted adaptable programmer.

LEARN HOW TO BE GOOD AT SKILL PROGRAMMING IN A RAW PROGRAMMING LANGUAGE

You can learn many technologies (Rails, jQuery, Angular, ReactJS, etc.) that will provide a certain way to solve various web development problems. For 90% of the problems you'll face, you'll be able to find an elegant solution that is easy to solve using a framework. These are problems that can be solved with about 1 line of code.

These tools are awesome, and make our lives as developers so much easier.

The remaining problems (the other 10%) are problems that aren't easily solved with these tools. Although these make up only a small percentage of the problems you'll face, you'll often have to write dozens, even hundreds of lines of code, to do the tasks.

Being able to use a certain framework is valuable, but being an adaptable programmer, who is able to take any problem and solve it without the use of a cookie-cutter solution, is what is actually in demand. The best way to do this is to pick one programming language at first and get really good at it.

LEARN HOW TO WORK WITH ALGORITHMS

The only way to pick up these programming skills is to solve problems with plain programming language. Most developers will call this a skill in "algorithms."

So, while you're learning to code, you should spend time solving coding challenges that stretch your programming knowledge (harder challenges), not just the problems that are really easy to solve. After learning how to break down complicated problems, you'll become an adaptable programmer.

In general, learning your first programming language will be the hardest. You not only need to learn the language, you also need to learn the logic of breaking down problems to make them solvable by computers. Once you're strong in one programming language, it becomes easier to pick up other programming languages.

This is why in the interview process, you'll be asked to solve problems for which there is no cookie-cutter solution. You'll have to craft your own solution. Many times, the interviewer won't even care which programming language you use, even if they don't use that programming language at the company.

LEARN DATA STRUCTURES

The term "Data Structure" generally refers to "the way a computer program can organize a bunch of stuff." There are a handful of data structures computer programs use all the time. Figuring out how to use them is critical to leveling up as a programmer. Here's a list of some of the data structures that you'll be expected to understand in a technical interview:

- Arrays
- Hash Tables (aka Dictionaries)
- Linked Lists
- Trees
- Stacks
- Queues
- Priority Queues

These concepts allow you to properly communicate like a developer. They're also often the right tools to use for many problems.

Data structures transcend all fads. Learning them is critical to becoming an adaptable programmer.

LEARN BIG OH NOTATION

Big Oh notation is something that is taught with traditional Computer Science degrees. It's a theoretical way to compare how fast one algorithm will run in comparison to other algorithms, and it comes up all the time during technical interviews.

Knowing this fuzzy theoretical stuff is incredibly important for when you write a computer program, find that it runs too slow, and want to know how to make it faster. There is a science (computer science!) that you can apply to programs to understand why they're slow, and how to make them faster.

That's why hiring managers almost always make sure you know Big Oh notation during technical interviews.

GET EXPERIENCE WORKING WITH A TEAM

The best way to show your how your value to a professional development team is to demonstrate that you've successfully built something previously while working in a team environment. After all, the best software is written by teams of programmers working together on a single application. Hiring managers want to know that you have the ability to become a productive member of a technical team.

EVENTUALLY, LEARN MULTIPLE PROGRAMMING LANGUAGES

A developer is only as good as his toolkit. So, while you want to pick one language at the beginning and use it to learn how to solve problems, you don't want to stop there.

For example, learning ruby first teaches you to solve problems using an object-oriented problem-solving approach, meaning that it teaches you to organize your code in a very specific way.

On the other hand, JavaScript focuses on functional programming, meaning it pushes you to solve problems in a different manner. JavaScript embraces first-class functions, so makes you use a different approach to solving problems.

Learning programming on the extremes of different programming paradigms will teach you to break down problems in different ways. So, once you learn the fundamental, soft-skills of programming by learning one language really well, it's time to push your thinking into a new box and learn a different language.

Now that we've explained what matters and what you need to learn to become an adaptable programmer, let's jump into what you can start doing today to put everything in motion.

10. WHAT YOU CAN START DOING TODAY

There's a wide world of code out there, and it's changing every single day. Because of that, it can be incredibly intimidating for a beginner to dive into programming full time. **It's easy to get the impression that there is simply too much information out there to learn,** and that you should give up on the idea of becoming a programmer. That's simply not true.

You shouldn't be thinking about learning the next big thing, because that's not what you need to know in order to land a job as a developer. Instead, you should **focus on what actually matters.**

Once you start coding, you learn that everything is figure-out-able. You master the art of Google searches and realize that most people have faced similar problems. You understand enough to be able to teach yourself new concepts on the fly. And you learn how to self-correct and get yourself back on track when problems arise.

That's not to say that looking into the cool new trend is always a bad idea. **What's bad is letting that feeling paralyze you and stop you from growing as a developer.**

So, don't choose the technologies you want to learn based on:

- General hype
- Number of Google search results for the language
- Average starting salary

Instead, choose the technologies that you learn based on:

- The problems they solve
- How they make your life easier
- How they enable you to add additional technologies to your toolkit later on

And remember that an adaptable programmer is capable of accepting any open programming position, regardless of the language or frameworks used. **The adaptable programmer commands a larger job market** than someone who is only open to a small percentage of positions based on the specific language they know.

It certainly makes sense to spend time learning to use the practical tools that you'll be using on the job. But **don't neglect the timeless skills.** If you learn the things that don't change, master a programming language, and figure out how to problem solve, you will become an adaptable programmer who will always be in demand.

Learn more at **thefirehoseproject.com** and reach us at ***questions@thefirehoseproject.com.*** We're happy to help.

START CODING FOR FREE TODAY

Not working? Click here.