下一步干什么？

理论上，下一步你的选择很多。自学是门手艺，你可以用它去学任何你想要掌握的其它手艺。如果，你有意在编程这个领域继续深入，那么，以下就是一些不错的线索。

当然，最先应当做的是，去检查一下自己的 “突击” 的结果，去 Pythonbasics.org 做做练习：

> https://pythonbasics.org/Exercises/

除了我在这里介绍的之外，请移步 The Hitchhiker's Guide to Python，它更为全面：

> https://docs.python-guide.org/

Python 必读书籍

无论学什么，一本书肯定不够，以下是学习 Python 的基本必读书籍：

> - The Python Tutorial
> - The Hitchhiker’s Guide to Python!
> - Think Python: How to think like a computer scientist
> - Automate the Boring Stuff with Python
> - Effective Python
> - Python Cookbook
> - Fluent Python
> - Problem Solving with Algorithms and Data Structures using Python
> - Mastering Object-oriented Python - Transform Your Approach to Python Programming

更多 Python 书籍：

> https://pythonbooks.revolunet.com

千万别觉得多，只要真的全面掌握，后面再学别的，速度上都会因此快出很多很多.....

Python Cheatsheet

你已经知道了，这种东西，肯定是自己整理的才对自己真的很有用..... 不过，你也可以把别人整理的东西当作 “用来检查自己是否有所遗漏” 的工具。

网上有无数 Python Cheatsheets，以下是 3 个我个人认为相当不错的：

> - Comprehensive Python Cheatsheet
> - Python Crash Course - Cheat Sheets
> - Pysheeet

Awesome Python

Github 上的 “居民” 现在已经养成了一个惯例，无论什么好东西，他们都会为其只做一个 “Awesome ...” 的页面，在里面齐心协力搜集相关资源。比如，你想学 Golang，那你去 Google 搜索 `Awesome Go`，一定会给你指向到一个 Github 上的 “Awesome Go” 的页面.....

以下是 Awesome Python 的链接：

> https://github.com/vinta/awesome-python

CS 专业的人都在学什么？

如果你真有兴趣把这门手艺学精，不妨看看 Computer Science 专业的人都在学什么……

下面这个链接值得认真阅读：

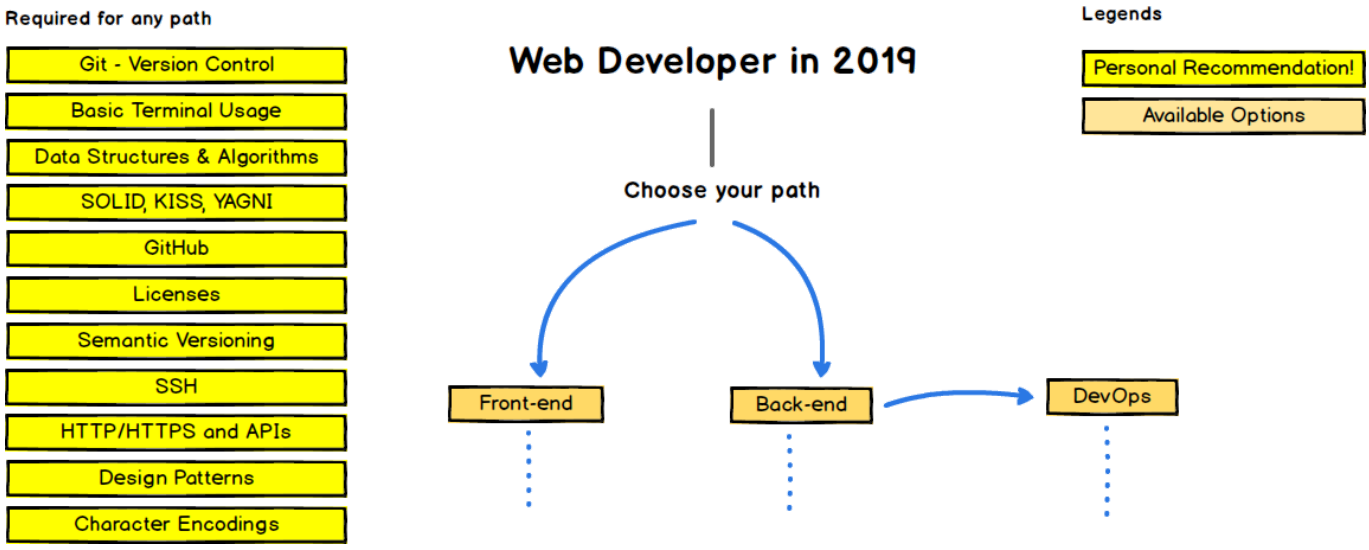
> http://matt.might.net/articles/what-cs-majors-should-know/

全栈工程师路径图

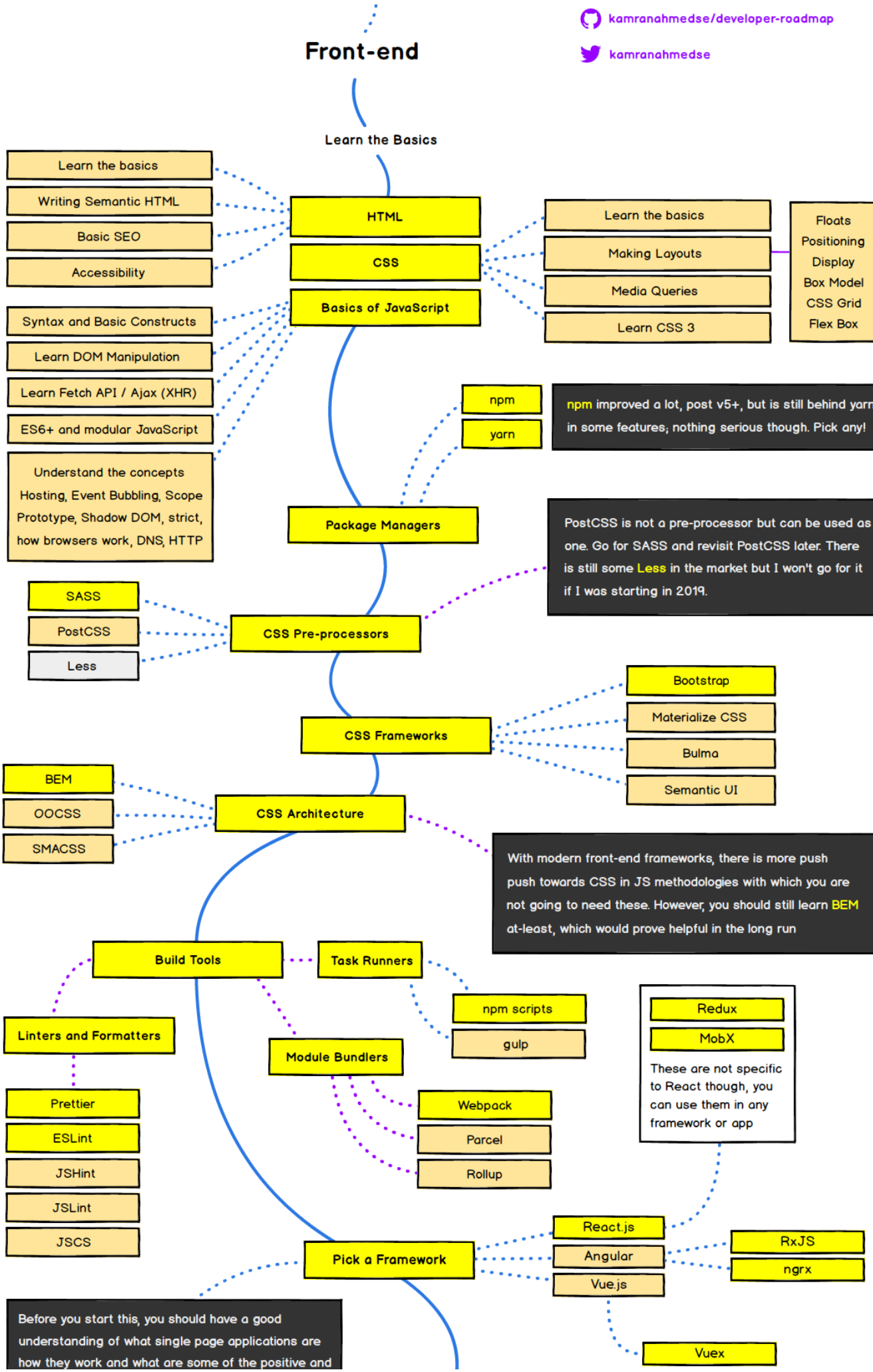
既然学了，就肯定不止 Python —— 在扎实的基础之上，学得越多学得越快。以下是一个 “全栈工程师路径图”，作者是位迪拜的帅哥 Kamran Ahmed：

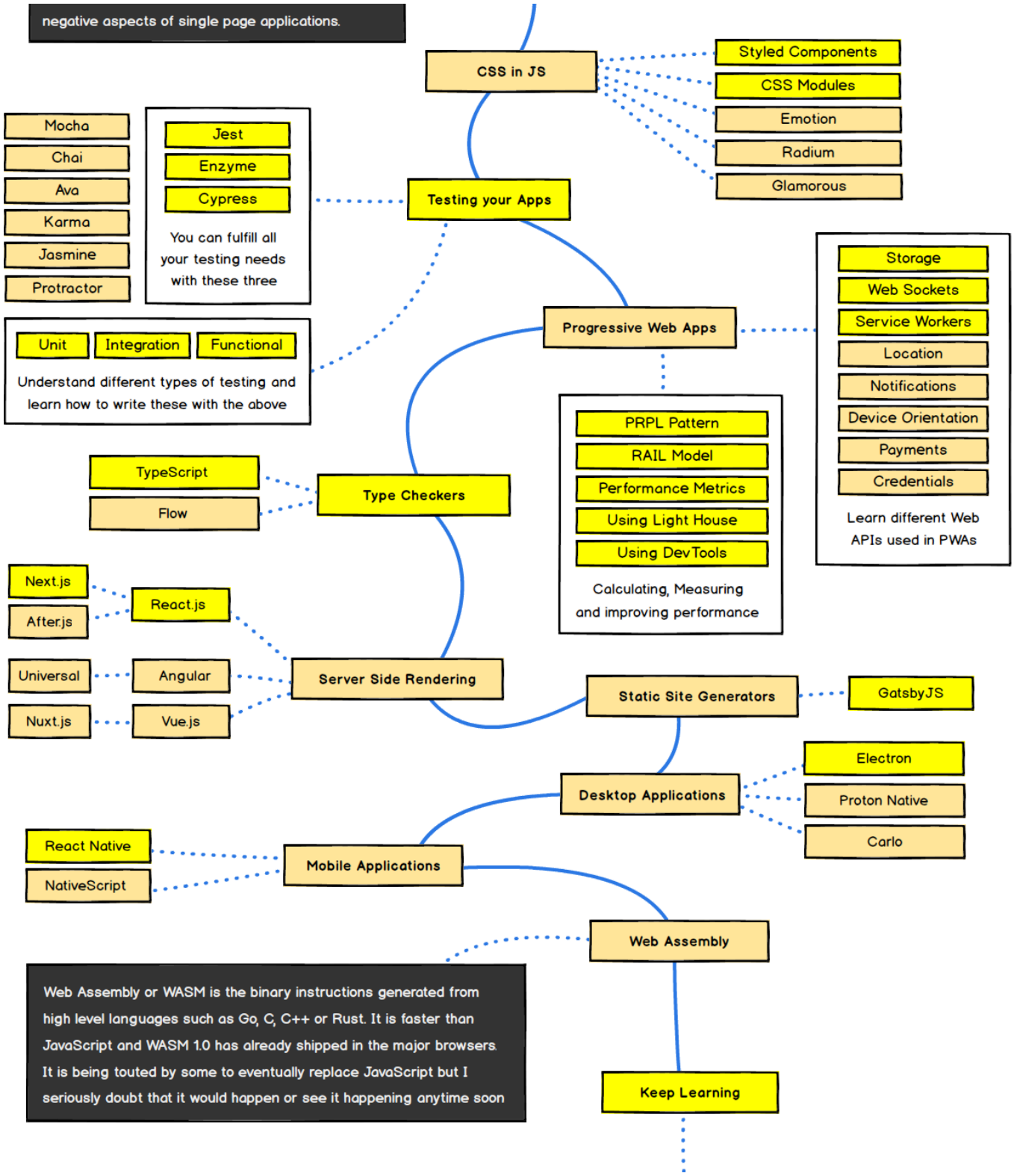
> https://github.com/kamranahmedse/developer-roadmap
>
> Below you find a set of charts demonstrating the paths that you can take and the technologies that you would want to adopt in order to become a frontend, backend or a devops. I made these charts for an old professor of mine who wanted something to share with his college students to give them a perspective; sharing them here to help the community.
>
> Check out my blog and say "hi" on Twitter.
>
> **Disclaimer**
>
> The purpose of these roadmaps is to give you an idea about the landscape and to guide you if you are confused about what to learn next and not to encourage you to pick what is hip and trendy. You should grow some understanding of why one tool would better suited for some cases than the other and remember hip and trendy never means best suited for the job

Introduction

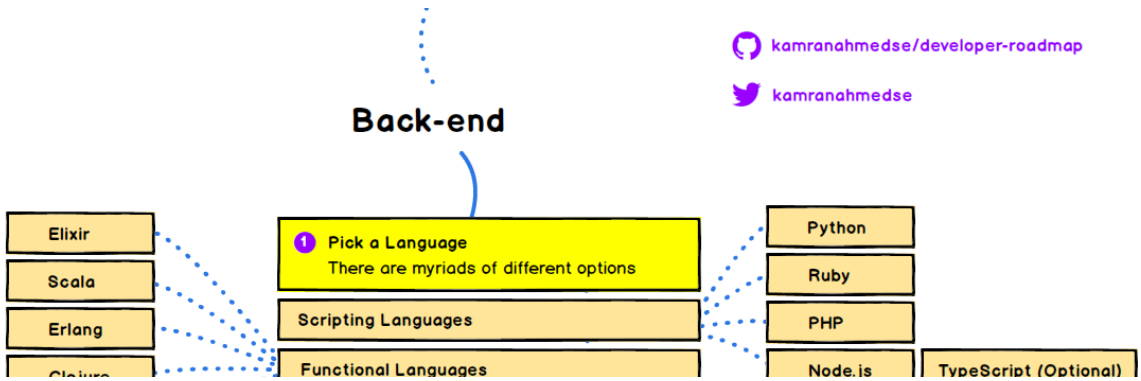


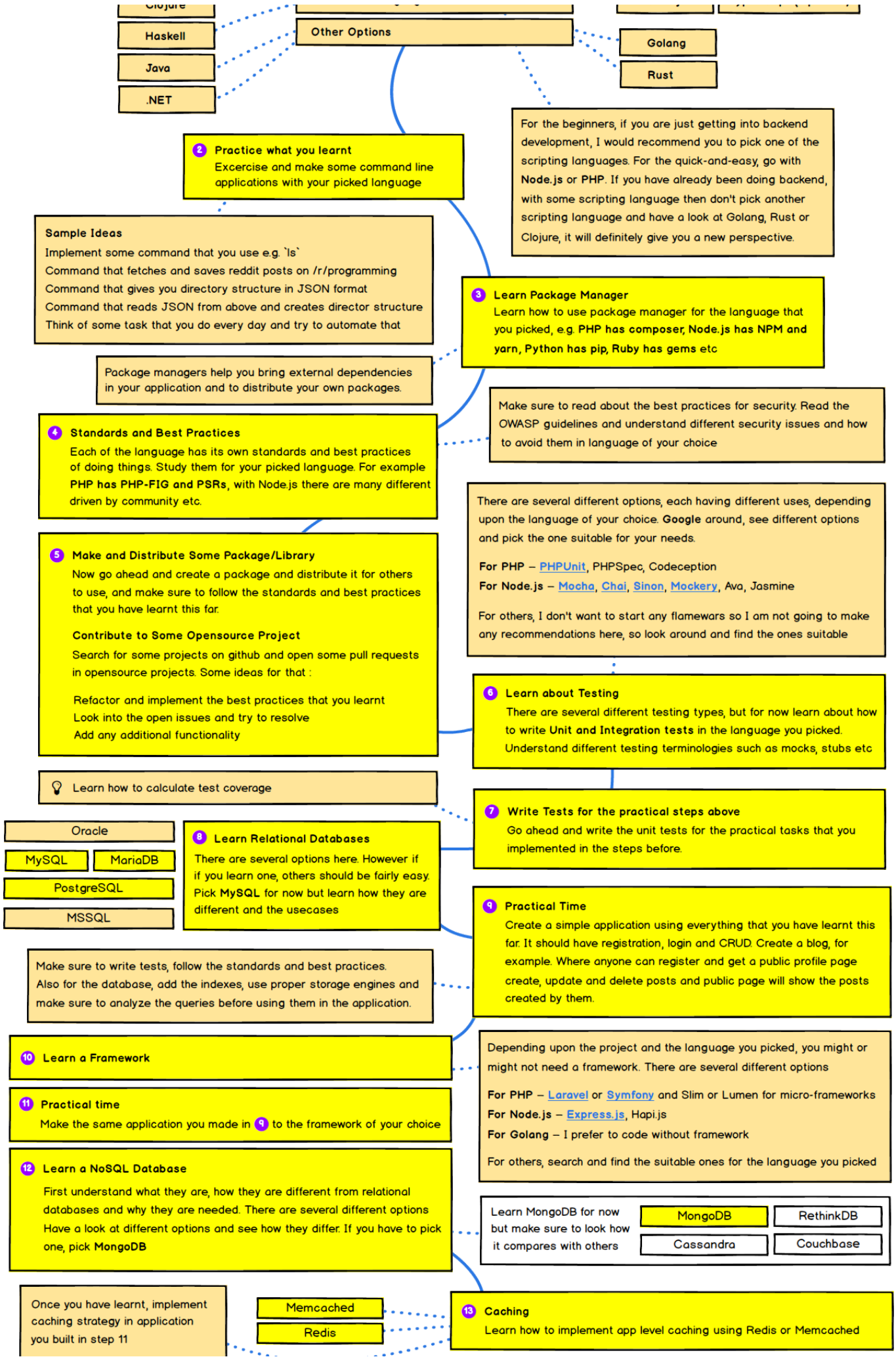
Frontend Roadmap

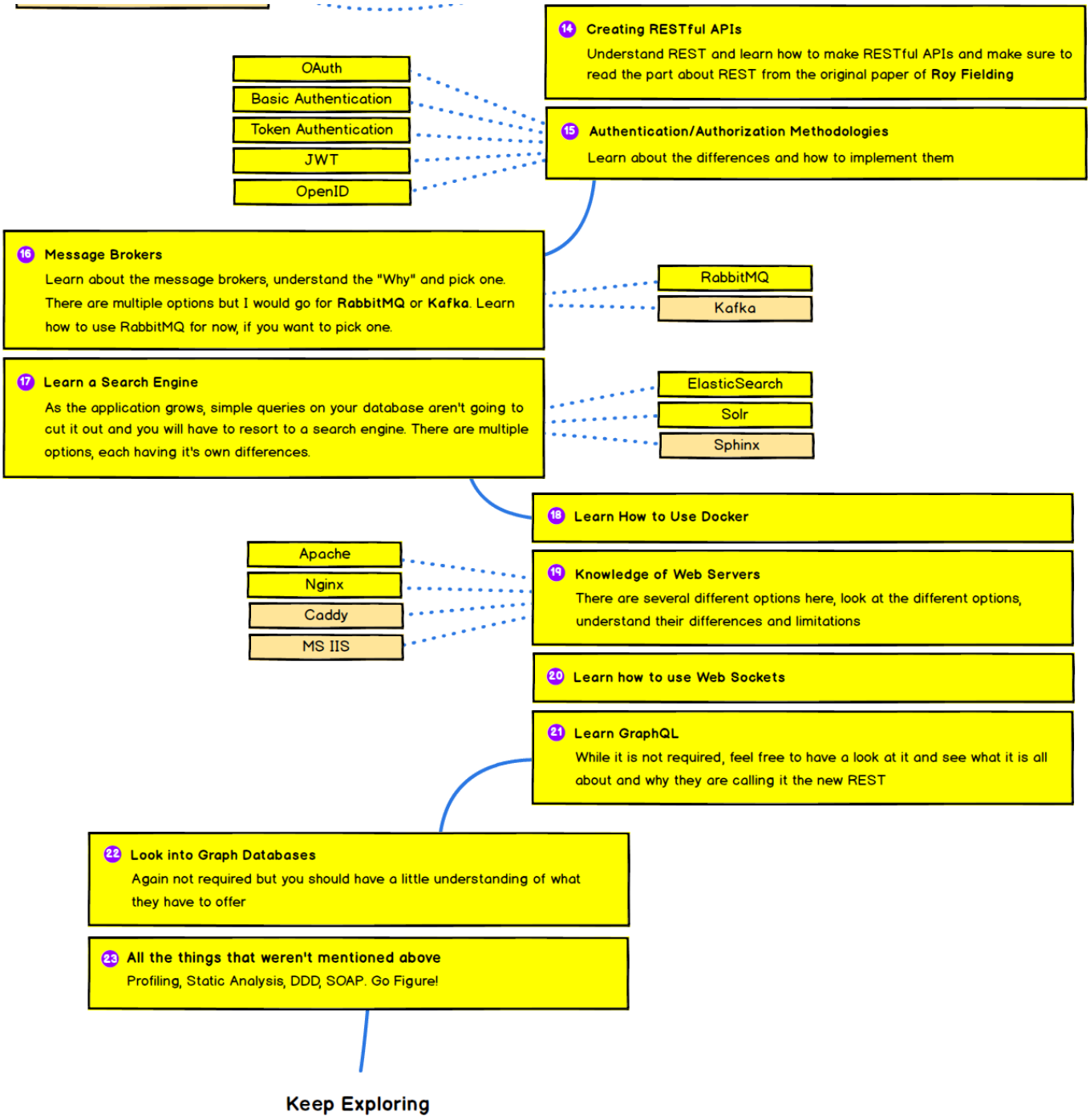

Front-end
kamranahmedse/developer-roadmap
kamranahmedse
Learn the Basics
Learn the basics
Writing Semantic HTML
Basic SEO
Accessibility
HTML
CSS
Basics of JavaScript
Learn the basics
Making Layouts
Media Queries
Learn CSS 3
Floats
Positioning
Display
Box Model
CSS Grid
Flex Box
Syntax and Basic Constructs
Learn DOM Manipulation
Learn Fetch API / Ajax (XHR)
ES6+ and modular JavaScript
Understand the concepts
Hosting, Event Bubbling, Scope
Prototype, Shadow DOM, strict,
how browsers work, DNS, HTTP
npm
yarn
npm improved a lot, post v5+, but is still behind yarn in some features; nothing serious though. Pick any!
Package Managers
PostCSS is not a pre-processor but can be used as one. Go for SASS and revisit PostCSS later. There is still some Less in the market but I won't go for it if I was starting in 2019.
SASS
PostCSS
Less
CSS Pre-processors
Bootstrap
Materialize CSS
Bulma
Semantic UI
CSS Frameworks
BEM
OOCSS
SMACSS
CSS Architecture
With modern front-end frameworks, there is more push push towards CSS in JS methodologies with which you are not going to need these. However, you should still learn BEM at-least, which would prove helpful in the long run
Build Tools
Task Runners
npm scripts
gulp
Linters and Formatters
Module Bundlers
Redux
MobX
These are not specific to React though, you can use them in any framework or app
Prettier
ESLint
JSHint
JSLint
JSCS
Webpack
Parcel
Rollup
React.js
Angular
Vue.js
RxJS
ngrx
Pick a Framework
Vuex
Before you start this, you should have a good understanding of what single page applications are how they work and what are some of the positive and



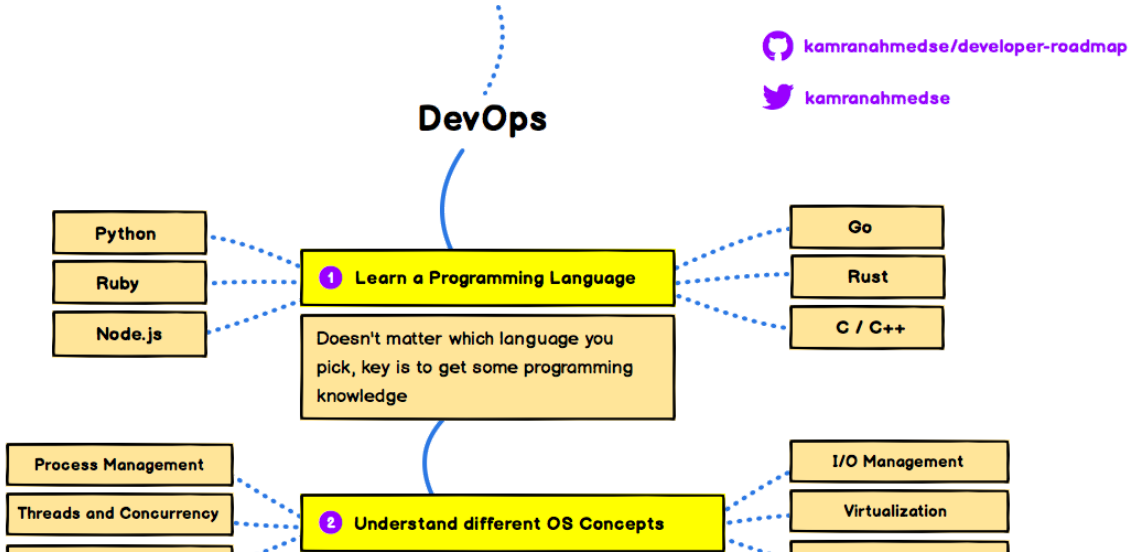
Back-end Roadmap

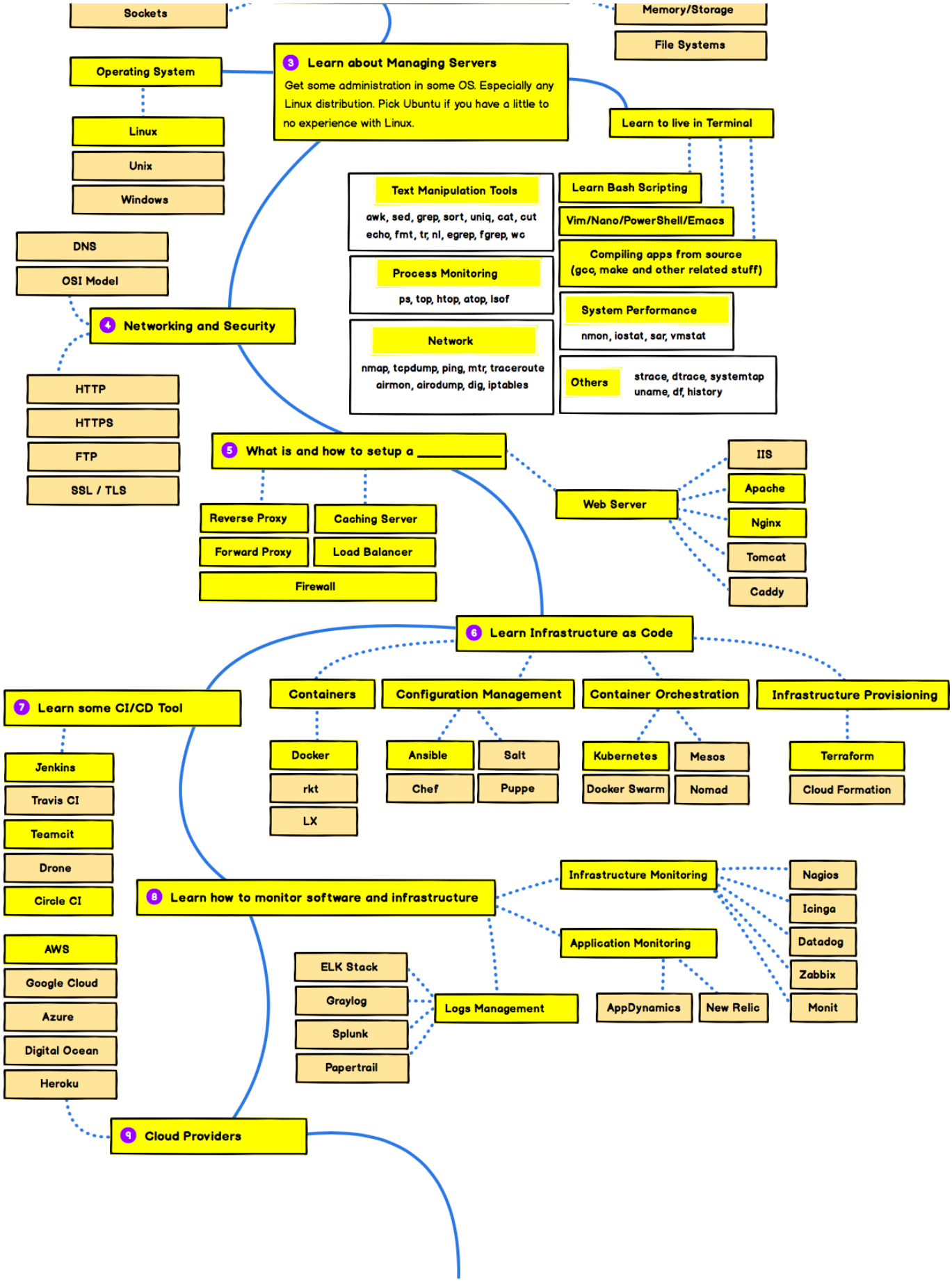



Clojure
Haskell
Java
.NET
Other Options
Golang
Rust
For the beginners, if you are just getting into backend development, I would recommend you to pick one of the scripting languages. For the quick-and-easy, go with Node.js or PHP. If you have already been doing backend, with some scripting language then don't pick another scripting language and have a look at Golang, Rust or Clojure, it will definitely give you a new perspective.
2 Practice what you learnt
Excercise and make some command line applications with your picked language
Sample Ideas
Implement some command that you use e.g. `ls`
Command that fetches and saves reddit posts on /r/programming
Command that gives you directory structure in JSON format
Command that reads JSON from above and creates director structure
Think of some task that you do every day and try to automate that
3 Learn Package Manager
Learn how to use package manager for the language that you picked, e.g. PHP has composer, Node.js has NPM and yarn, Python has pip, Ruby has gems etc
Package managers help you bring external dependencies in your application and to distribute your own packages.
4 Standards and Best Practices
Each of the language has its own standards and best practices of doing things. Study them for your picked language. For example PHP has PHP-FIG and PSRs, with Node.js there are many different driven by community etc.
Make sure to read about the best practices for security. Read the OWASP guidelines and understand different security issues and how to avoid them in language of your choice
5 Make and Distribute Some Package/Library
Now go ahead and create a package and distribute it for others to use, and make sure to follow the standards and best practices that you have learnt this far.
Contribute to Some Opensource Project
Search for some projects on github and open some pull requests in opensource projects. Some ideas for that :
Refactor and implement the best practices that you learnt
Look into the open issues and try to resolve
Add any additional functionality
There are several different options, each having different uses, depending upon the language of your choice. Google around, see different options and pick the one suitable for your needs.
For PHP – PHPUnit, PHPSpec, Codeception
For Node.js – Mocha, Chai, Sinon, Mockery, Ava, Jasmine
For others, I don't want to start any flamewars so I am not going to make any recommendations here, so look around and find the ones suitable
6 Learn about Testing
There are several different testing types, but for now learn about how to write Unit and Integration tests in the language you picked. Understand different testing terminologies such as mocks, stubs etc
Learn how to calculate test coverage
7 Write Tests for the practical steps above
Go ahead and write the unit tests for the practical tasks that you implemented in the steps before.
Oracle
MySQL
MariaDB
PostgreSQL
MSSQL
8 Learn Relational Databases
There are several options here. However if if you learn one, others should be fairly easy. Pick MySQL for now but learn how they are different and the usecases
9 Practical Time
Create a simple application using everything that you have learnt this far. It should have registration, login and CRUD. Create a blog, for example. Where anyone can register and get a public profile page create, update and delete posts and public page will show the posts created by them.
Make sure to write tests, follow the standards and best practices.
Also for the database, add the indexes, use proper storage engines and make sure to analyze the queries before using them in the application.
10 Learn a Framework
Depending upon the project and the language you picked, you might or might not need a framework. There are several different options
For PHP – Laravel or Symfony and Slim or Lumen for micro-frameworks
For Node.js – Express.js, Hapi.js
For Golang – I prefer to code without framework
For others, search and find the suitable ones for the language you picked
11 Practical time
Make the same application you made in 9 to the framework of your choice
12 Learn a NoSQL Database
First understand what they are, how they are different from relational databases and why they are needed. There are several different options Have a look at different options and see how they differ. If you have to pick one, pick MongoDB
Learn MongoDB for now but make sure to look how it compares with others
MongoDB
RethinkDB
Cassandra
Couchbase
Once you have learnt, implement caching strategy in application you built in step 11
Memcached
Redis
13 Caching
Learn how to implement app level caching using Redis or Memcached



DevOps Roadmap





路漫漫其修远兮.....

但多有意思啊？这完全就是一场闯关游戏。