coders for causes

Workshops Materials

./Innovation with a mission

Coders for Causes

Table of contents

1. About

1.1 Coders for Causes Workshops

This is where you can see all the materials for workshops that are presented to the volunteers of the project for each year.

If you would to know more about us, please visit our website at codersforcauses.org

1.2 Contributions

Hi! We are happy that you thought of contributing! If you have any suggestions or issues, please raise it here. I would be happy if you could provide pull requests, if you know how to do it here.

1.2.1 Structure

Folder Structure

The structure of this repo is as follows:

```
├── docs                    // Folders for documentation
│   ├── CNAME
│   ├── contributions.md
│   ├── deployment_and_automated_site_deployment.md
│   ├── flavoured_markdown.md
│   ├── images              // Assets
│   │   └── ...
│   │
│   ├── index.md
│   └── writing_markdown.md
├── LICENSE
├── mkdocs.yml              // MkDocs Configuration
├── overrides
│   └── partials
│       └── footer.html
├── README.md
└── requirements.txt
```

1.2.2 Installation

Python

> **Prerequisite**
>
> You need to have Python installed to be able to use `pip`. There are a few ways of installing Python. You can use a package distributor like Anaconda Or you can just install Python.

Once you have installed Python, install mkdocs requirements by opening a terminal and typing:

```
pip install -r requirements.txt
```

> **Python Environments (Optional)**
>
> however, it is good practice to use different environments for different purposes, in which case, for Anaconda, you would open a terminal and type:
>
> ```
> conda create -n mkdocstutorial python
> conda activate mkdocstutorial
> ```
>
> then enter:
>
> ```
> pip install -r requirements.txt
> ```

Docker

Just run `docker-compose up`, it should show the web server running at localhost:8000

1.2.3 Commands

- `mkdocs new [dir-name]` - Create a new project.
- `mkdocs serve` - Start the live-reloading docs server. Very helpful when you want to take a look at the docs before deploying.
- `mkdocs build` - Build the documentation site.
- `mkdocs -h` - Print help message and exit.
- `mkdocs gh-deploy` - Deploy in github pages

1.2.4 Web Documentation Configuration

For full documentation visit:

- mkdocs.org for the generic MkDocs
- PyMdown Extensions for the different extensions that are installed
- MkDocs Material for the customisation of the web server documentation.

2. 2021/2022 Summer

2.1 Coders for Causes 2021/22 Summer Workshops

This project period continues the two main projects from the winter of 2020 :

- Foodbank
- WAIS

If you have not before seen the existing progress, see this video.

These two projects have their own corresponding technology stacks being used, hence will dictate the workshops that will be held.

2.1.1 Project Technology

Foodbank

Foodbank is mainly with frontend with React + NextJS + TypeScript + TailwindCSS with Firebase and Notion CMS.

WAIS

WAIS is a full-stack application with Vue and Django. It uses Docker containerisation for both development (and production in the future).

2.1.2 Workshop Recordings

The workshop recordings will be held on our youtube channel.

2.2 Setup

This contains everything you need to know about getting setup.

2.2.1 Coders for Causes Project Team

The following access you will need to have when working on the project

- Coders for Causes Official Project Organisation
  - This includes the project repositories and CFC related long-term materials
- Coders for Causes Learning Organisation
  - This includes the templates for learning as well as the demo workshops
- Discord Channel for Project and exclusive Workshops

2.2.2 Developer Tools

These are the following tools that you need:

- Code Editor: VS Code
- Version Control: Git
- Interpreter: Nodejs
- Custom Package Manager: Yarn
- Interpreter (for WAIS): Python
- Containerisation (for WAIS): Docker

> **Optional Tools**
>
> These are tools that you may like to use, but are not required:
>
> - GUI for Git: Gitkraken / GitHub Desktop

After these installation, seek at the OS-specific tools.

Windows

These are tools specifically for Windows:

- Virtual Machine: Vbox
  - You need this if you have some trouble with windows

Warning for Windows Users

Legit, among all the OS, you will have the most frustrating time as a developer in windows (unless you're doing C#)

> **Wanna have a better developer experience**
>
> You have a couple of options:
>
> - Using WSL
> - Dual Booting
> - Virtual Machines
>
> The recommended OS to try is Ubuntu-based Linux. My personal favourite is PopOS.

Linux

These are installation specific to Linux:

- Docker Post Install
  - Lots of user forget this documentation

2.3 Project and Workshop Structure

A mission to empower the next-generation of software engineers and delivering value to the community!

2.3.1 Ways of Working

- (November 27 to February 24 excluding December 19 to January 4) = ~ 11 weeks for project
- 2 meetups every week (1 online mid-week, 1 on Saturday)
- Each Saturday meeting will have ~2-3 hour workshops. Some workshops will take the whole hour, while some workshops will take 15-30 minutes.
- The other time allocated is for working on project with supervision
- Most workshops will be introductory

2.3.2 Workshop Schedule

The following are rough guideline to the workshop schedules. All workshops are optional, you can attend any of them even if you are not on the specific team dedicated to it (be mindful that there may be time conflicts if it is with the other team).

Most Workshops are introductory

They will usually cover the following idea:

- why is it useful to learn it
- what are the different aspects of it that you will need to learn

You will still have to put in the effort to learn it thoroughly.

Date	Workshop Name	Recommended Team to attend	Duration
Saturday, 27 November 2021	Introduction to CFC + Web Development Space	All	~1 hour
Wednesday, 1 December 2021	Introduction to Web Basics	All	~1-1.5 hour
Saturday, 4 December 2021	Practical Software Engineering Practices	All	~1 hour
Saturday, 4 December 2021	Introduction to Frontend Frameworks	All	~45 minutes - 1 hour
Saturday, 4 December 2021	Introduction to React	Foodbank	~45 minutes - 1 hour
Saturday, 4 December 2021	Introduction to Vue	WAIS	~45 minutes - 1 hour
Saturday, 11 December 2021	Introduction to Django	WAIS	~1 - 1.5 hours
Saturday, 11 December 2021	Introduction to Typescript	Foodbank	~30 minutes
Saturday, 18 December 2021	MVC Codebase Structure / Frontend-Backend Integration	All	~1-1.5 hour
Saturday, 18 December 2021	Typical Codebase Structure	All	~30 minutes - 1 hour
Saturday, 18 December 2021	Package Manager - JavaScript and Python	All	~1 hour
Saturday, 8 January 2022	Introduction to Docker	All	~1 hour
Saturday, 8 January 2022	Introduction to Unit Testing and CI/CD	All	~1 hour

Workshops after January 8, 2022

The workshops right here are still being decided upon. If you have an idea for a workshop that you would like to attend, please let us know either on Discord or at Github.

The one that are on consideration are:

- Deployment on Heroku, and Vercel
- Introduction to End-to-End Testing with Cypress
- Gitkraken Workshop
- Introduction to Prototyping with Figma
- Introduction to Linux and Command-Line Scripting - Bash
- Increase your productivity in VsCode
- How to write good documentation
- Honing your detective skills with Browser Developer Tools

2.3.3 Who are you?

Before continuing further, answer the following:

- What's your name?
- What's your background?
- Why you're here?

Who are the Coders for Causes

The Coders for Causes is an organisation that aims to empower the next-generation of software engineers while delivering value to the wider-community by helping charities and not-for-profit organisations.

2.4 Introduction to Web Development Space

Navigating the Deep Dark Space of Web Development

This workshop covers a brief overview of the most common tools and technologies used in web development.

2.4.1 Content

2.4.2 What and Why Web Development?

What is web development?

- Websites development
- Web applications (client-side and server-side) development

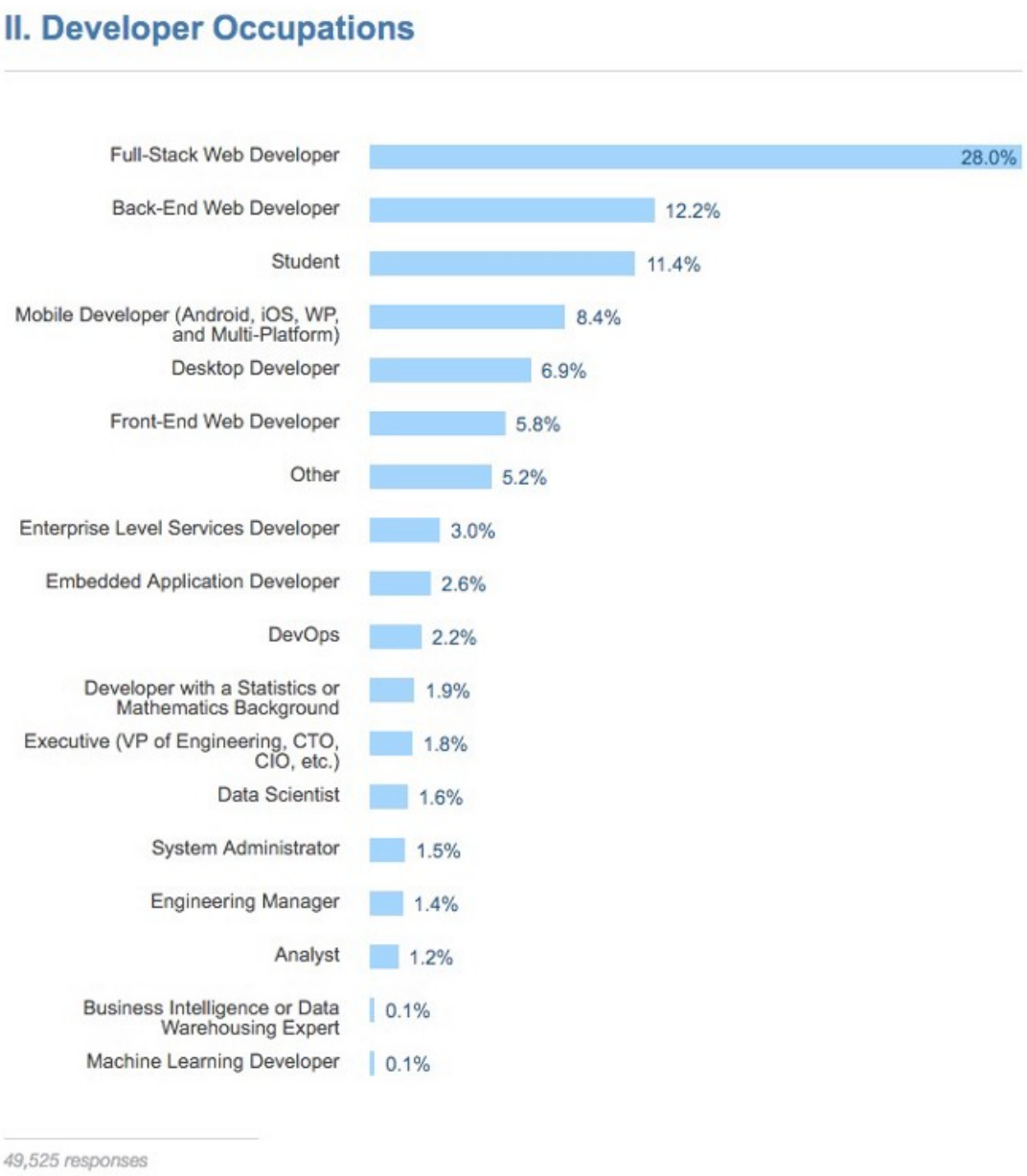
Why Web Development ?

- Accessibility and Portability
- Career and On-demand in job market
- Huge possibility to combine with other emerging technologies (e.g. IoT, Machine Learning) and industry (e.g. Health, Mining, O&G)

Career in Web Development

Source: Insights from Stack Overflow's 2016 survey of 50,000 developers

"Half of Developers are Web Developers"



2.4.3 FAQs about Web Development

Why code websites, why not use drag and drops like Wordpress, WIX?

- Content Management System (CMS)
- Limitations on theme/template used
- Difficult to extend
- Cybersecurity

> **More information**
>
> CMS are one of the application of web development, but there are plenty more such as - internet of things, custom software for a particular industrial application (eg. using Machine Learning)
>
> CMS are usually limited to the template or plugin that you use. If those plugin don't exist, then it limits your productivity very much (difficulty to extend).
>
> CMS are usually built to cater for non-technical users. This means that thye become the subject of hackers. Think about a scenario where a hacker was able to find a vulnerability in WordPress, now every other WordPress site will be vulnerable.

What is the best way to learn all these?

In summary, the best way to learn:

- Do personal projects (inspiration + motivation)
- Do team projects (get peer reviews and correct bad practices straight away)
- Watch Online Courses (to figure out what is available)

> **More Information**
>
> To be told that you have to learn "this, this, and that" before you could do things is tiresome.
>
> Often times, we want to learn to be a developer so that we can create cool things like software where thousands of people can use the app. We don't tend to be a developer for the sake of us needing to watch endless videos on different things.

Why does CFC not do mobile development as much as web development?

- App stores has a developer cost
- Easier to deal/teach web technologies

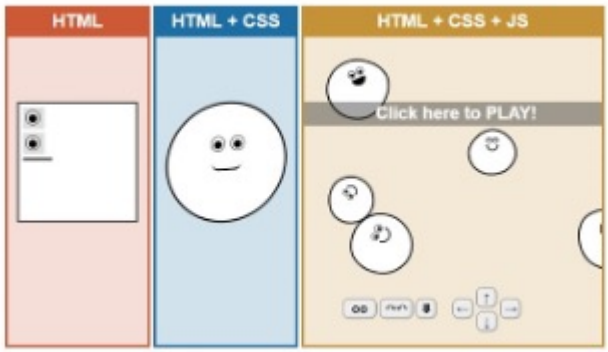
- Accessibility (mobile, sensors, tablets, laptops and PCs)
- Bigger open-source community

If I already know a frontend framework, is it better to learn another frontend framework or to learn a backend framework?

- It is better to learn a backend framework

> **Reasoning**
>
> You want to build skills that complement one another rather than be an alternative.
>
> It is much more valuable for you to learn a backend framework because that helps you build a **functional** app.

2.4.4 Basics of Web and Limitation



HTML

What is it?

- Hypertext Markup Language
- Describes the structure of a web page

Limitation

- Doesn’t handle repeated content well
- No variables or calculation

HTML Syntax

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    Hello World!
</body>
</html>
```

CSS

What is it?

- Cascading Style Sheets
- Describes the presentation of a web page

Limitation

- Most css is quite similar (Handled by CSS Libraries)
- Not very dynamic (Handled by CSS Frameworks)

CSS Syntax

```css
body {
    background-color: #f0f0f0;
    font-family: sans-serif;
}

.container{
    width: 80%;
    margin: 0 auto;
}
```

JS

What is it?

- JavaScript
- Used to program complex features on a web page

Limitation

- Has the capability to modify the user interface, but becomes really tedious to modify interface (more about this in another workshop)

JS Syntax

```
const bodySelector = document.querySelector('body');
const myFunction = () => {
    bodySelector.innerHTML = 'Hello World 2!';
    // Add a container class to space out
    bodySelector.classList.add('container');
}
bodySelector.onclick = myFunction;
```

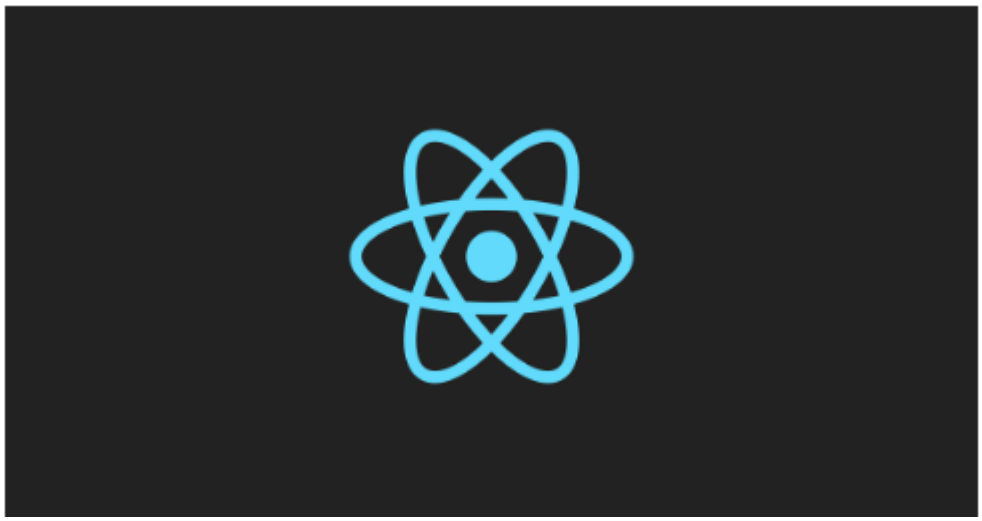
Modern Frameworks

- Websites can be much more... they can be web applications
- “App” in a website (client-side rendering)

Modern Frameworks

React.js

- More mature and used more in industry



Vue.js

- Growing fast in popularity and use.

General Information

- Both are good to use and learn.
- Knowledge is transferable between the two frameworks.

Comparison between HTML and JSX

Highlighted portions are starting chunk of distinct code.

HTML

```
<div class="row">
    <div class="m-0 col-12 col-md-6 col-lg-3">
        <div class="text-center border-0 bg-transparent card">
        <div class="px-0 card-body">
            <i class="material-icons-sharp md-lg">devices</i>
            <p class="mt-4 font-weight-bold text-monospace text-larger">
            Applications
            </p>
            <p class="mb-0">
            Build custom web and mobile applications to engage with your audience
            </p>
        </div>
        </div>
    </div>
    <div class="m-0 col-12 col-md-6 col-lg-3">
        <div class="text-center border-0 bg-transparent card">
        <div class="px-0 card-body">
            <i class="material-icons-sharp md-lg">web</i>
            <p class="mt-4 font-weight-bold text-monospace text-larger">Websites</p>
            <p class="mb-0">
            Build new websites or optimise existing pages to improve online
            visibility
            </p>
        </div>
        </div>
    </div>
    <div class="m-0 col-12 col-md-6 col-lg-3">
        <div class="text-center border-0 bg-transparent card">
        <div class="px-0 card-body">
            <i class="material-icons-sharp md-lg">storage</i>
            <p class="mt-4 font-weight-bold text-monospace text-larger">
            Data Storage
            </p>
            <p class="mb-0">
            Design and create databases for efficient information storage and
            retrieval
            </p>
        </div>
        </div>
    </div>
    <div class="m-0 col-12 col-md-6 col-lg-3">
        <div class="text-center border-0 bg-transparent card">
        <div class="px-0 card-body">
            <i class="material-icons-sharp md-lg">how_to_reg</i>
            <p class="mt-4 font-weight-bold text-monospace text-larger">
            Consulting
            </p>
            <p class="mb-0">
            Empower your organisation through technical knowledge and advice
            </p>
        </div>
        </div>
    </div>
</div>
```

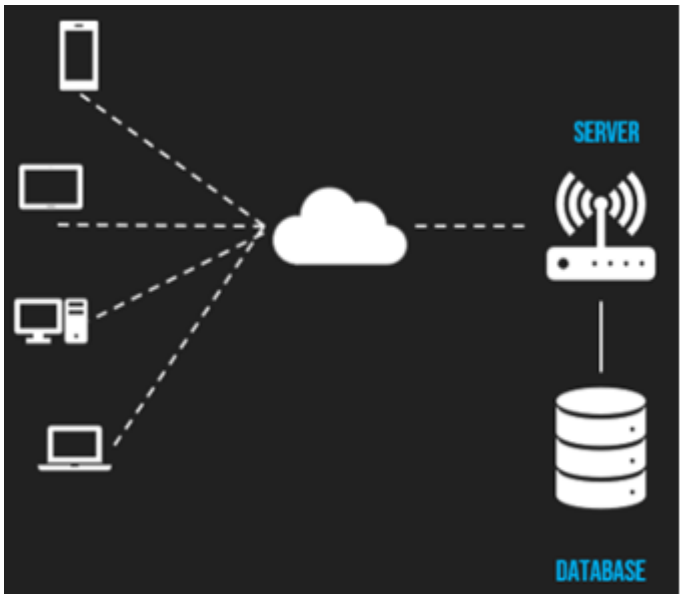
JSX

```jsx
import { memo } from 'react'
import { Row, Col, Card, CardBody } from 'reactstrap'
import services from 'data/services.json'

const Service = (props: {
icon: string
title: string
description: string
}) => (
<Card className='text-center border-0 bg-transparent'>
    <CardBody className='px-0'>
    <i className='material-icons-sharp md-lg'>{props.icon}</i>
    <p className='mt-4 font-weight-bold text-monospace text-larger'>
        {props.title}
    </p>
    <p className='mb-0'>{props.description}</p>
    </CardBody>
</Card>
)

const Services = () => (
<Row>
    {services.map(service => (
    <Col
        xs={12}
        md={6}
        lg={12 / services.length}
        className='m-0'
        key={service.title}
    >
        <Service {...service} />
    </Col>
    ))}
</Row>
)

export default memo(Services)
```

2.4.5 Server-side Applications (Backend)

How do devices communicate?

- HTTP Request - Hypertext Transfer Protocol



What do server applications do?

- Serve frontends (server-side rendering)

- Web API (Application Programming Interface)
  - Serve data (usually from a database)
  - Process Request (Sending emails or SMS, Machine Learning)

Databases

Place to store the data

Mongodb

Allows for database design to be modified without complex migration or data loss

SQL

Typically faster and better for large amounts of data or systems that need data consistency and reliability

2.4.6 Others

Languages of the Web (the usual)

- Python (Django, Flask)
- JavaScript (Node.js, Express)
- Ruby, Go, Rust, C

Hosting Stuff

Many ways - Own a server - Use a 3rd party platform

CSS Frameworks

Frontend (JS)

- Vue, React
- Nuxt.js, Next.js

Frontend (CSS)

- MaterialUI, Vuetify
- Bootstrap

Developer Tools

TYPESCRIPT

- Type checking is super useful for complex apps
- Allows for way better javascript developer tooling
- Can be annoying if you’re new at it

TESTING

- Selenium, Cypress
  - End to end automated testing tools
- Jest, Mocha, Pytest
  - Unit testing
- Testing is vital to software projects

CONTINUOUS INTEGRATION / CONTINUOUS DEPLOYMENT

- Automated Testing
- Event-driven scripts
- E.g. Github Action, Bitbucket Pipelines

VIRTUALISATION AND CONTAINERISATION

- Allows execution of services in a virtual environment
- eg. Docker (Containerisation), Vagrant (Virtualisation)

BROWSERS

- Standard browsers
  - Google Chrome, Firefox, Edge, etc.
- Backwards compatibility
  - Internet Explorer
- Other
  - Mobile - Responsive
  - Screen readers - Accessibility

FIREFOX DEVELOPER TOOLS

- Page Inspector
  - Visualise page aspects
  - Grid layout
- Web Console
  - `console.log("Hello World")`
- Responsive Design Mode
  - View from POV of different screen sizes such as mobile, tablets, etc.

> **Some more tools**
>
> - JavaScript Debugger
> - Network Monitor
> - Performance Tools
> - Rulers
> - Colour Pickers Learn more at: https://developer.mozilla.org/en-US/docs/Tools

PACKAGE MANAGERS

- Installs libraries that can be used
- Also has code shortcuts (e.g. npm run start)

(More about `package.json` and `poetry.toml` in the projects and Package Manager Workshop)

VERSION CONTROL

- Essential for developer teams and complex software development
- Git

LINTERS AND FORMATTERS

- Makes code formatting consistent (following standard)
- Useful with version controls to avoid pointless change

eg.ESLint, Prettier

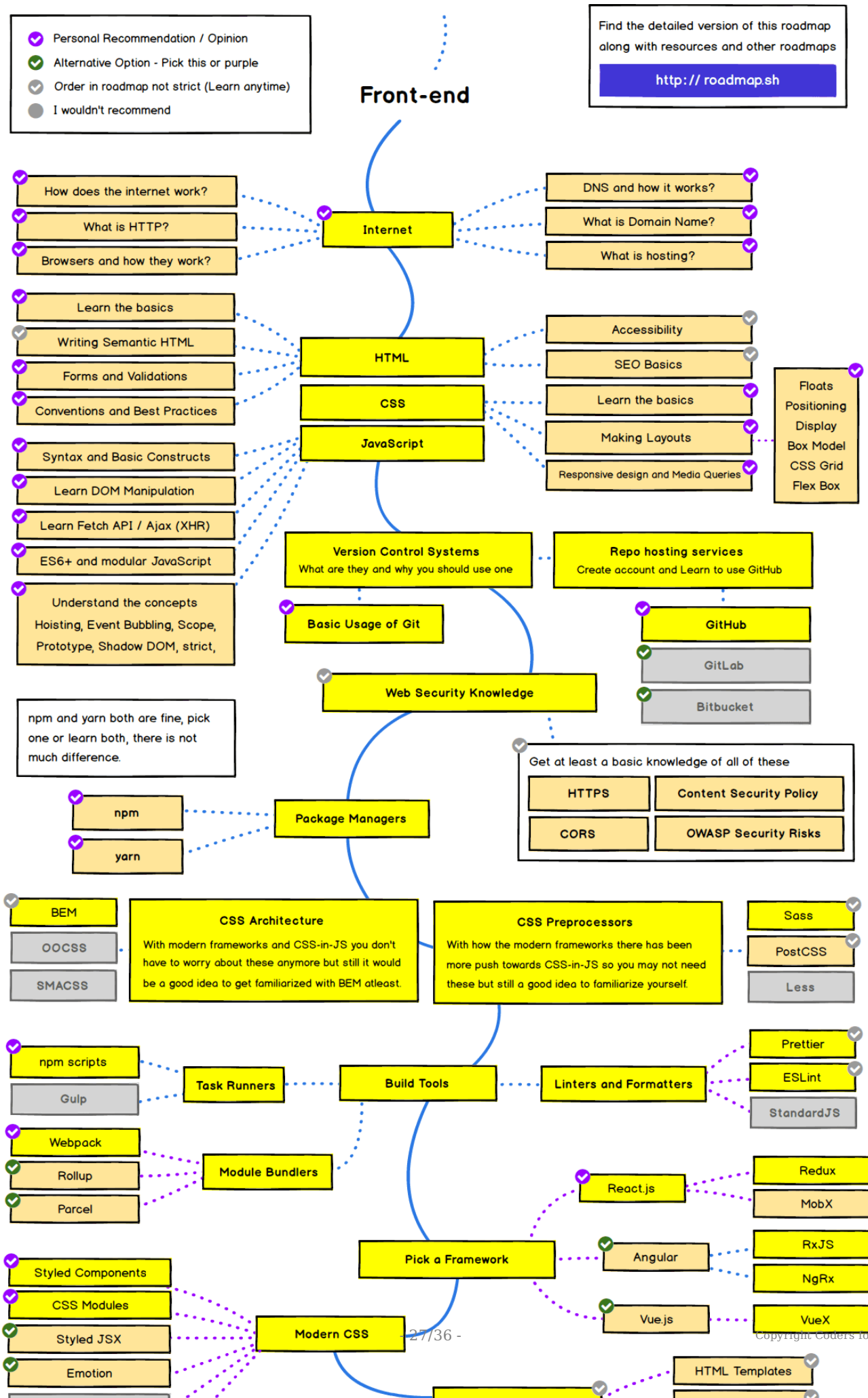
TEAMWORK

- Many tools out there
- Used to stop teams from stepping on each others toes
- Github Issues + Pull Requests

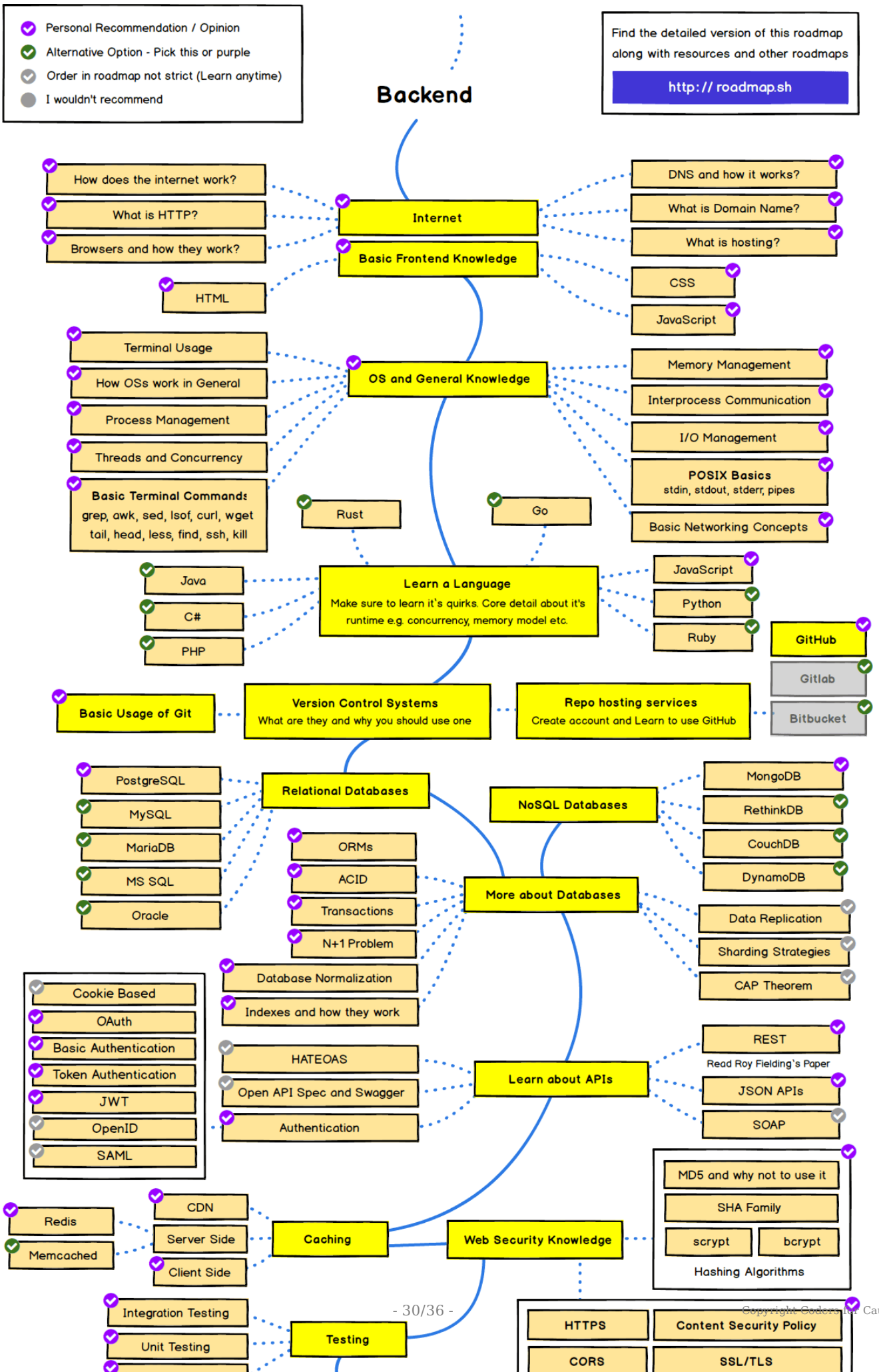
Roadmaps

There's an open-source community that maintains a learning roadmap for developers. See https://roadmap.sh/

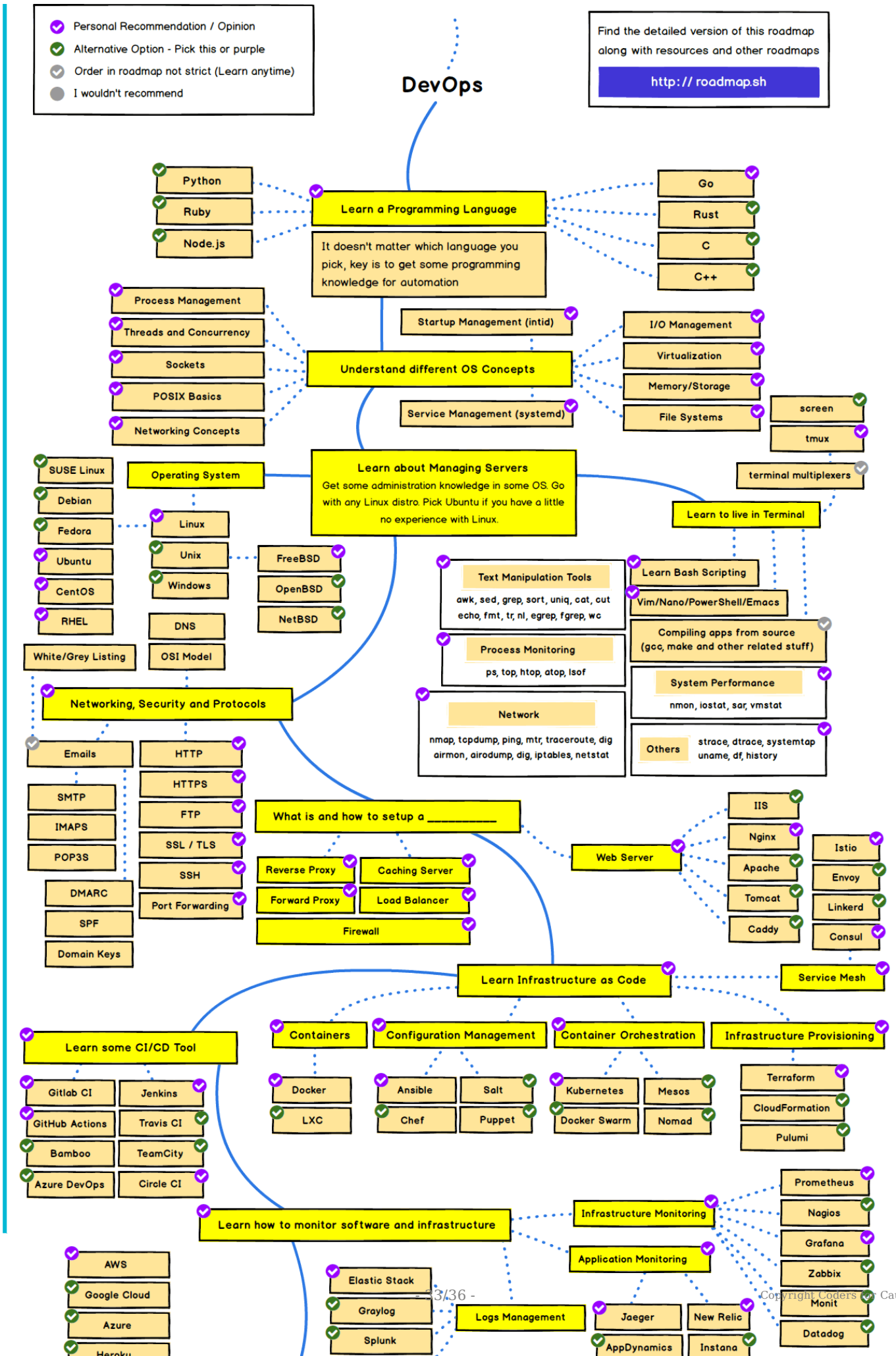
Frontend Developer



Backend Developer



Dev-Ops



2.4.7 Word of Encouragement

> **Encouragement from the Tech Lead**

"I can admit that this journey of learning will be difficult, and can sometimes be overwhelming and demotivating. Please, if at any point of this project, you feel that you don't know enough, or you're feeling lost, please reach out! We are all in this journey together! Nobody is born talented, skills are honed with determination and willingness to learn."

"When I was a first year student entering on the CFC winter project, I didn't feel like I was good enough. I couldn't create a good looking interface, I didn't know how to use npm and all sorts of those things. I was just like many of you! if I gave up just because of all those things I didn't know, of all those self-doubts, then I wouldn't be here today. I admit that I was lucky because I was in CFC, I had connections where I can just ask questions instead of feeling lost of not knowing. So please do leverage that opportunity to reach out"

"You being in this project not only gives you the opportunity to raise your talents, but you also unlock one of the biggest factor of the growth of your career, and that is the connections with your fellow software engineers."

3. 2021 Winter

3.1 Coders for Causes 2021 Winter Workshops

This project period there are two main projects:

- Foodbank
- WAIS

If you have not before seen the existing progress, see this video.

These two projects have their own corresponding technology stacks being used, hence will dictate the workshops that will be held.

3.1.1 Project Technology

Foodbank

Foodbank is mainly with frontend with React + TailwindCSS with Firebase and Notion CMS.

WAIS

WAIS is a full-stack application with Vue and Django. It uses Docker containerisation for both development (and production in the future).

3.1.2 Where are the materials?

This website has only been created prior to 2021/22 Summer workshops. However, you can find the videos in our Youtube channel, and workshops slides in google drive.

https://github.com/codersforcauses/workshops/