coders
for causes

Workshops Materials

./Innovation with a mission

Coders for Causes

Copyright Coders for Causes

Table of contents

1. About 5
1.1 Coders for Causes Workshops 5
1.2 Contributions 6
1.2.1 Structure 6
1.2.2 Installation 6
1.2.3 Commands 7
1.2.4 Web Documentation Configuration 7
2. 2022/2023 Summer 8
2.1 Coders for Causes 2021/22 Summer Workshops 8
2.1.1 Project Technology 8
2.1.2 Workshop Recordings 8
2.2 Introduction to Web Development Space 9
2.2.1 Content 9
2.2.2 What and Why Web Development? 10
2.2.3 FAQs about Web Development 12
2.2.4 Basics of Web and Limitation 14
2.2.5 Server-side Applications (Backend) 19
2.2.6 Others 20
2.2.7 Word of Encouragement 29
3. 2021/2022 Summer 30
3.1 Coders for Causes 2021/22 Summer Workshops 30
3.1.1 Project Technology 30
3.1.2 Workshop Recordings 30
3.2 Setup 31
3.2.1 Coders for Causes Project Team 31
3.2.2 Developer Tools 31
3.3 Project and Workshop Structure 34
3.3.1 Ways of Working 34
3.3.2 Workshop Schedule 34
3.3.3 Who are you? 36
3.4 Introduction to Web Development Space 37
3.4.1 Content 37
3.4.2 What and Why Web Development? 37
3.4.3 FAQs about Web Development 39
3.4.4 Basics of Web and Limitation 41

Copyright Coders for Causes

3.4.5 Server-side Applications (Backend) 46
3.4.6 Others 47
3.4.7 Word of Encouragement 56
3.5 Introduction to the Web Basics - HTML, CSS & JavaScript 57
3.5.1 Content 57
3.5.2 Key Web Technologies 57
3.5.3 HTML 58
3.5.4 CSS 63
3.5.5 JavaScript 67
3.5.6 Creating a Pokemon API Webapp 71
3.6 Practical Software Engineering Practices 72
3.6.1 How do software engineers work? 72
3.6.2 Why do Software Engineers Work like this? 74
3.6.3 What is Git? 75
3.6.4 GitHub and Code Repositories 78
3.6.5 INTERACTIVE WORKSHOP TIME !!!! 78
3.6.6 Other Resources 78
3.7 Introduction to Frontend Frameworks 79
3.7.1 Introduction to Frontend Frameworks 79
3.7.2 Introduction to React 95
3.7.3 Introduction to Vue 102
3.8 Welcome to the Djangol! 103
3.8.1 How does an application communicate with backend systems? 103
3.8.2 What are APIs and REST-APIs? 104
3.8.3 What is Django? 105
3.8.4 What is Django REST Framework (DRF)? 106
3.8.5 Features of Django and DRF 106
3.8.6 Packaging a "Django App" 107
3.8.7 Django Admin 108
3.8.8 Other Tools 109
3.8.9 Demo - Live Coding 112
3.9 MVC-MVMM Architecture and Communications 115
3.9.1 What is MVC/MVVM Architecture? 115
3.9.2 Why is software architecture important? 122
3.9.3 How do frontend communicate with backend? 123
3.9.4 Demo - React to integrate with Django 125
3.9.5 Other Information 126

Copyright Coders for Causes

3.10 Package Managers 127
3.10.1 What are package managers? 127
3.10.2 Install Package Managers 130
3.10.3 Lockfiles 133
3.10.4 When to install globally? 133
3.10.5 Demo - Material UI 134
3.11 Typical Codebase Structure 136
3.11.1 Overview and Importance 136
3.11.2 Frontend 136
3.11.3 Backend 141
3.11.4 Other Resources 144
3.12 Automated Testing and CI Pipelines 145
3.12.1 What is the problem with manual testing? 145
3.12.2 Importance of Software 145
3.12.3 Software is Fragile 148
3.12.4 Consequences of Fault Softwares 149
3.12.5 Automated Testing 151
3.12.6 What is CI/CD? 153
3.12.7 Demo - Cypress Integration Test / E2E Test 155
4. 2021 Winter 156
4.1 Coders for Causes 2021 Winter Workshops 156
4.1.1 Project Technology 156
4.1.2 Where are the materials? 156

Copyright Coders for Causes

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

Copyright Coders for Causes

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

Copyright Coders for Causes

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.

Copyright Coders for Causes

2. 2022/2023 Summer

2.1 Coders for Causes 2021/22 Summer Workshops

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

- WADL
- POOPS

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

POOPS

POOPS is a web-app built the frontend with React + NextJS + TypeScript + TailwindCSS with Firebase and Notion CMS.

WADL

WADL is a full-stack application with Vue and Firebase. 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.

Copyright Coders for Causes

2.2 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.

2.2.1 Content

- Introduction to Web Development Space
- Content
- What and Why Web Development?
- What is web development?
- Why Web Development ?
- FAQs about Web Development
- Why code websites, why not use drag and drops like Wordpress, WIX?
- What is the best way to learn all these?
- Why does CFC not do mobile development as much as web development?
- If I already know a frontend framework, is it better to learn another frontend framework or to learn a backend framework?
- Basics of Web and Limitation
- Modern Frameworks
- Server-side Applications (Backend)
- How do devices communicate?
- What do server applications do?
- Databases
- Others
- Languages of the Web (the usual)
- Hosting Stuff
- CSS Frameworks
- Developer Tools
- TypeScript
- Testing
- Continuous Integration / Continuous Deployment
- Virtualisation and Containerisation
- Browsers
- Firefox Developer Tools
- Package Managers
- Version Control
- Linters and Formatters
- Teamwork
- Roadmaps
- Word of Encouragement

Copyright Coders for Causes

2.2.2 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)

Copyright Coders for Causes

Career in Web Development

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

"Half of Developers are Web Developers"

Copyright Coders for Causes

2.2.3 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

Copyright Coders for Causes

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.

Copyright Coders for Causes

2.2.4 Basics of Web and Limitation

Copyright Coders for Causes

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;
}
```


Copyright Coders for Causes

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)

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)

Copyright Coders for Causes

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.

Copyright Coders for Causes

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>
```


Copyright Coders for Causes

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)
```

2.2.5 Server-side Applications (Backend)

How do devices communicate?

- HTTP Request - Hypertext Transfer Protocol

What do server applications do?

- Serve frontends (server-side rendering)

Copyright Coders for Causes

- 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

2.2.6 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

Copyright Coders for Causes

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

Copyright Coders for Causes

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/

Copyright Coders for Causes

Frontend Developer

Copyright Coders for Causes

Copyright Coders for Causes

Backend Developer

Copyright Coders for Causes

Copyright Coders Of Causes

Dev-Ops

Copyright Coders for Causes

Copyright Codes for Causes

2.2.7 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."

Copyright Coders for Causes

3. 2021/2022 Summer

3.1 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.

3.1.1 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).

3.1.2 Workshop Recordings

The workshop recordings will be held on our youtube channel.

Copyright Coders for Causes

3.2 Setup

This contains everything you need to know about getting setup.

3.2.1 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

3.2.2 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

Copyright Coders for Causes

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#)

Copyright Coders for Causes

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 Installation Steps
- Lots of users forget this documentation
- This step is necessary if you intend to use the visual studio code docker extension
- This will also allow you to run docker without the "sudo" command
- Docker Compose Installation
- Docker compose is not installed by default with Linux installations of Docker
- Please follow this documentation
- Node installation using Node Version Manager (NVM)
- Installation steps can be found here
- NVM is useful when dealing with multiple environments that require different versions of Node
- It allows you to switch between different Node versions and environments easily
- This tool is also available if you are using Windows Subsystem for Linux

Copyright Coders for Causes

3.3 Project and Workshop Structure

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

3.3.1 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

3.3.2 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).

Copyright Coders for Causes

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

Copyright Coders for Causes

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

3.3.3 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.

Copyright Coders for Causes

3.4 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.

3.4.1 Content

- What and Why Web Development?
- FAQs about Web Development
- Basics of Web and Limitation
- Server-side Applications (Backend)
- Others
- Languages of the Web (the usual)
- Hosting Stuff
- CSS Frameworks
- Developer Tools
- TypeScript
- Testing
- Continuous Integration / Continuous Deployment
- Virtualisation and Containerisation
- Browsers
- Firefox Developer Tools
- Package Managers
- Version Control
- Linters and Formatters
- Teamwork
- Roadmaps
- Word of Encouragement

3.4.2 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)

Copyright Coders for Causes

Career in Web Development

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

"Half of Developers are Web Developers"

Copyright Coders for Causes

3.4.3 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

Copyright Coders for Causes

- 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.

Copyright Coders for Causes

3.4.4 Basics of Web and Limitation

Copyright Coders for Causes

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;
}
```


Copyright Coders for Causes

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)

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)

Copyright Coders for Causes

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.

Copyright Coders for Causes

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>
```


Copyright Coders for Causes

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)
```

3.4.5 Server-side Applications (Backend)

How do devices communicate?

- HTTP Request - Hypertext Transfer Protocol

What do server applications do?

- Serve frontends (server-side rendering)

Copyright Coders for Causes

- 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

3.4.6 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

Copyright Coders for Causes

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

Copyright Coders for Causes

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/

Copyright Coders for Causes

Frontend Developer

Copyright Coders for Causes

Copyright Coders for Causes

Backend Developer

Copyright Coders for Causes

Copyright Coders Of Causes

Dev-Ops

Copyright Coders for Causes

Copyright Codes for Causes

3.4.7 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."

Copyright Coders for Causes

3.5 Introduction to the Web Basics - HTML, CSS & JavaScript

The tools of the web

This workshop will introduce you to the basics of HTML, CSS and JavaScript, the fundamental technologies of web development. This is a very introductory workshop, and there is still so much we can learn, but this will be a good place to start.

3.5.1 Content

- Key Web Technologies
- HTML
- Elements
- Attributes
- Images
- Anchors/Hyperlinks
- Forms
- Document Object Model
- CSS
- CSS Format
- Rules of Selection
- Document and External Style Sheets
- Flexboxes
- JavaScript
- The Basics of JavaScript
- Arrays
- Objects
- Functions
- Loops
- Web App Tutorial

3.5.2 Key Web Technologies

HTML, CSS and JavaScript each have a different job when it comes to creating web pages.

What do they do?

- HTML: describes the content and structure of the web page
- CSS: describes the style and appearance of the web page
- JavaScript: provides funtionality to a web page

Copyright Coders for Causes

3.5.3 HTML

Hyper Text Markup Language (HTML) is used to structure the webpage. The general structure of a webpage can be seen below.

First HTML Page

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>You see this text on your browser tab!!!</title>
    </head>
    <body>
        <!-- this is where the content goes! -->
        <h1>My First Heading</h1>
        <p>This is a paragraph</p>
    </body>
</html>
```

Elements

A webpage is made of elements, each with their own properties, that contain content to be displayed on the page. Elements are defined by tags, such as `h1`, `div` and `body`. Most elements have an open and close tab. The container and its content, together, are called an element.

```html
<h1>This is a header element.</h1>
```

The most notable, and probably the tag you will use most, is the `<div>` tag. It defines a division or section within the HTML document, and is used as a container that holds other elements.

Copyright Coders for Causes

The Seperator

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>The Web Dev Fairytale</title>
    </head>
    <body>
        <div>
            <!-- This div concerns itself with the main heading/title of the story -->
            <h1>The Dev, the Debugger and the Caffiene Addiction</h1>
        </div>
        <div>
            <!-- This div concerns itself with the first chapter it its entirety -->
            <h1>Chapter 1</h1>
            <div>
                <!-- This div concerns itself with the text of the first chapter -->
                <!-- section within a section. A subsection! -->
                <p>Once upon a time, in a castle far, far away...</p>
            </div>
        </div>
        <div>
            <h2>Chapter 2</h2>
        </div>
    </body>
</html>
```

We will go through the important ones during this workshop, but you can find the full of them here.

Attributes

Most tags have attributes that specify information or change the tag in some way. The most common being `class` and `id`.

MOST COMMON ATTRIBUTES

- `class`: used to specify one or more class names for a HTML element.
- Classes are used to group certain elements in order to give them specific features through CSS and to allow many elements to be manipulated using JavaScript.
- `id`: used to specify a unique id for an element and must be unique
- The `id` attribute is assigned to an element so that element can be exclusively specified in the style declaration and JavaScript manipulation

Below is an example of how attributes are assigned in your code.

Elements and their Attributes

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>You see this text on your browser tab!!!</title>
    </head>
    <body style="background-color: red">
        <div class="bigSection">
            <h1>BIG HEADER!</h1>
            <p id="first">This is the first paragraph of the webpage</p>
            <!-- more cool stuff goes here -->
        </div>
        <div class="smallSection">
            <h6>small header</h6>
        </div>
    </body>
</html>
```

Images

Images are added by using the `<img>` element tag. It can also be used to add gifs!

Copyright Coders for Causes

Example

```
<img src="https://i.imgur.com/SjZxZza.jpeg" alt="Three Cute Dog Mermaids">
```

Cute, right?

IMAGE ATTRIBUTES

- `src` : specifies the location of the image to be displayed
- This can either be a url, as in the example above, or the relative path of an image within the site's directory
- `alt` : an optional attribute that contains a text description.
- Is useful for accessibility or if the image does not load properly

Copyright Coders for Causes

Anchors and Hyperlinks

We can add links to other websites, or even to sections within the same page, using the `<a>` element, known as an anchor.

Adding a hyperlink to an HTML page

```
<a href="https://codersforcauses.org/" target="_blank">Coders for Causes</a>
```

They can be adding added by themself, or within text such as here.

ANCHOR ATTRIBUTES

- `href` : specifies the destination to link to
- External site: simply include the url you wish to visit
- Different page within the site: include the path to the new page, usually in the form of `"/.../index.html"`
- Different section on same page: use `#name_of_section` where `name_of_section` is the `id` of the element you wish to go to
- `target` : specifies where to open the link
- Setting the target attribute to `"_blank"` opens the link in another tab
- `download` : specifies that the linked resouces will be downloaded.
- Only needs to be included in the declaration of the element
- Optional: if value of download is set, that value will be the name of the file

Always be careul when clicking links. You never know when there is something you should not click.

IMAGES AND HYPERLINKS

Any image, or any element in general, can be turned into a link by enclosing the element within the anchor tabs.

Hyperlinked image

```
<a href="https://codersforcauses.org/" target="_blank">
    <img src="https://www.uwastudentguild.com/assets/clubs/logos/logo---coders-for-causes-1567649329.jpg">
</a>
```


Copyright Coders for Causes

Forms

Forms are used to collect user input. They are defined by the `<form>` tags and usually contain form elements such as `<input>`, `<textbox>`, etc.

Inputs can come in many different forms, such as textboxes, radio buttons, checkboxes and drop down menus. Each input field is given a `value`
Once a form is complete and filled out, we need to submit, or POST, it. This can be done through a special input of type `submit`.

The only required attribute of `<form>` is `action`. The `action` attribute speicifies the URL of the application that is to be called when the Submit button is pressed. If no action, then the attribute takes the value of an empty string and the current page is the destination.

Example

```
<p>I like:</p>
<form action="">
    <input type="radio" value="Red">
    <input type="radio" value="Green">
    <input type="radio" value="Blue">
    <input type="submit" value="Submit">
</form>
```

INPUT VALIDATION

Input validation is a very important aspect to consider when dealing with forms. Remember the famous acronym GIGO, Garbage-in Garbage-Out.
To assist you with input validation, HTML forms have in-built validation for different types of data, such as emails, numbers and dates. This can be achieved by changing the input type of a field to the respective data entry type.

Copyright Coders for Causes

Baseline input validation with HTML

```
<input type="email" name="email" value="Email">
```

Document Object Model

All the elements of a HTML page make up a document tree, called the Document Object Model (DOM).

The DOM is a platform and language-neutral interface that allows programs to dynamically access and update the content, structure and style of the HTML document. Each element in a HTML document is represented by a node on the tree.
We can then use things such as JavaScript to access and update the HTML document using the DOM. We will see more of this in the tutorial at the end.

3.5.4 CSS

Cascading Style Sheets (CSS) provides style to the web. It is used to specify the layout and style of markup languages. CSS tells the browser *how* to display the elements that are written in HTML.

We can write the CSS style rules into an element using inline CSS, where the `style` attribute of an element is modified directly in the HTML.

Inline style sheets

```
<body style="background-color: red;">
    <!-- rest of body goes here -->
</body>
```

However, this gets hard to manage and maintain, especially when the number of elements in a document grows, and when we want to change many elements that have the same style.
We use *document-level* style sheets or *external* style sheets to combat this issue.

But first, we must understand how to create these style sheets.

CSS Format

- Selector
- A value, or list of values, that specify the elements for which the following style will be applied to
- Rules for specification will be discussed a little later on, see here

Copyright Coders for Causes

- Attribute
- The attribute/property of an element you wish you change
- Some example attributes include `background-color`, `font-size` and `width`
- Effect
- The effect is the value you set each attribute to be
- This includes setting `background-color` to "red", or `font-size` to "16px"

The following CSS specifies that all `img` elements are to be centered and have a width of 50 pixels.

My First CSS

```
img{
    /* attribute: effect; */
    display: block;
    margin-left: auto;
    margin-right: auto;
    width: 50px;
}
```

How do I know what attributes exist, and what do I change to get my desired effect?

There are two simple answers to this question:
1. **Google it.** Chances are, someone has already tried to do something similar and your solution already exists.
2. **Play around**. Learn what attributes exist, and how changing thier effects changes their on screen appearance.

Rules of Selection

As mentioned earlier, selectors have a set of rules that allow you to apply styles to certain, or even very specific, groups of elements. These can range selecting elements based on the type of element they are, or what class they are in, to selecting all elements of a certain type that are immediately preceded by an element of another type.

Element

Element selectors simply apply the defined style to all elements of the same type.

```
body{
    background-color: red;
}
```

Class

Class selectors apply the defined style to all elements belonging to that class. They are similar to element selectors, except that the class name is preceded by a `.` character.

```
.narrow{
    width: 10%
}
```

We can also specify the type of element within a class we wish to apply the style to by including it before the `.` character.
The following code selects all `<p>` elements that are of the class `narrow`:

```
p.narrow{
    width:10%;
}
```


Copyright Coders for Causes

id

These selectors apply the style to an element based on its `id`. The `id` is preceded by `#` in the selector definition.

```
#bigButton{
    font-size: 16px;
}
```

Psuedo Classes

Psuedo classes are styles that apply only when a certian action occurs or a condition is met, and not all the time. Some common selectors include `hover`, `focus` and `active`. They are included after the element, class or id has been specified, and are preceded with a colon `:`.

```
button:hover{
    color: red;
}
```

The full list of them can be found here.

Group

To specify a group of elements to apply a style to, simply list the elements, ids or classes delimitted by a comma (`,`).

```
h1, h2, div{
    color: red;
}
```

Other

There exists other selectors that allow you to be more specific as to what elements are to be selected, such as contextual selectors, but we will leave those as *self-learning*.

Contextual Selectors

Contextual selectors group elements based on their position and surroundings in the document tree.

Further reading into the topic can be found through searching the web for platforms that teach you all about computer science, such as Geeks for Geeks.

Document and External Style Sheets

Now that we know what we are doing, let's create some style sheets! Document and External style sheets help us better organise and manage the styles of the document. They are located in one location, so you do not have to go far to find them and change parts of your document.

Copyright Coders for Causes

DOCUMENT STYLE SHEETS

Document style sheets are located within the `<head>` of a HTML document, under the `<style>` tag. They work in the same was as mentioned above.
See the below example of how document style sheets are implemented within a HTML page.

Document Style Sheets

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>You see this text on your browser tab!!!</title>
        <style>
            body{
                background-color: blue;
            }
            p{
                color: red;
                font-size: 16px;
            }
        </style>
    </head>
    <body>
        <!-- this is where the content goes! -->
        <h1>My First Heading</h1>
        <p>This is a paragraph</p>
    </body>
</html>
```

EXTERNAL STYLE SHEETS

What happens when our site grows, and the number of HTML pages increases, and suddenly styles have to change, and we have to change every document, but they all have to follow the same styling as the others, and we have to manage that?!
Thankfully, we can store our styles in a seperate `.css` file and then simply reference the stylesheet in our HTML document.

Styles.css

```css
body{
    background-color: blue;
}
p{
    color: red;
    font-size: 16px;
}
```

Once we have our css file completed, we can reference it in the HTML document, again in the `<head>`, so that it can apply the styles to our document.

Applying my CSS files to my HTML document

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>You see this text on your browser tab!!!</title>
        <link rel="stylesheet" href="styles.css">
    </head>
    <body>
        <!-- this is where the content goes! -->
        <h1>My First Heading</h1>
        <p>This is a paragraph</p>
    </body>
</html>
```

Flexboxes

Flexboxes are a great way to position your items within a container.

Copyright Coders for Causes

Learning is best done when having fun and getting your hands dirty (metaphorically). Flexbox Froggy is an interactive website that teaches you all about flexboxes, one step at a time. Your aim is to allign all the frogs to their corresponding lilypads. It is a much better learning tool than sitting and reading about some code I wrote.

3.5.5 JavaScript

JavaScript, also known as JS, gives a web page fucntionality and reactiveness. It allows the user to interact with the web page, and for us to make it to do things that we want it to do.
Similar to CSS, all your JS can be implemented into a HTML document by encapsulating it in the `<script>` tag within the `<head>` of the document, or even `<body>` in this case. However, we will be sticking to having our JS stored in external files for ease of managability.
See the below example of how to add a script file to your page. To add many files, simply add another `<script>` element and reference the other file.

Adding JavaScript to my HTML document

```
<script src="script.js"></script>
```

The Basics of JavaScript

We shall discuss the basic syntax of JS, such as variables and functions, just to get you started. Feel free to do some of your own learning, too. There is a lot of cool things you can do and shortcuts you can use when you dive deeper into JavaScript, such as the *ternary operator*, but we will leave these for now.

VARIABLES

Variables are named memory locations that store data.
To define a variable, we can use three different keywords, each giving the variable special properties.

Variable Declaration

- `var`
- allows the variable to be redeclared later on in the program
- gives the variable a global scope, meaning they can be accessed anywhere within the file
- `let`
- once a variable has been declared using `let`, it cannot be redeclared. It's value can still change, however.
- gives the variable *block scope*, meaning it can only be accessed within the block of code that it has been declared in.
  For example, if I declare a variable using the `let` key word, I cannot access it outside of the function.
- `const`
- Once the variable has been declared and assigned, it cannot be redeclared and the value never changes. It stays *constant*, sort of.
- It, too, gives the variable block scope.

For a full explanation on variable declaration in JS, check out w3schools' page on it.

Variables can hold different data types, such as numbers, strings, objects, functions and arrays, but JS will cover the type identification for you.

Datatypes

- There are two groups of data types in JS: primitives and structural.

Copyright Coders for Causes

- A **primitive** is data that is not an object and has no methods.
  There are seven primitive data types:
- String, Number, BigInt, undefined, null and symbol
- A **structural** data type is one where the data is in the form of an object, and that object has its own methods. The main structural data types are:
- Objects and Functions

MATH AND LOGIC

Math and logic works similar in JS to other programming languages.

Math

- `+`, `-`: addition and subtraction
- `*`, `/`: multiplication and division, respectively.
- `%`: modulo operator. Returns the remainder left over after division.
- For example, `8 % 3` returns 2.
- `**`: exponent (*x to the power of y*)
- `base ** power`

Logic

- `&&`: AND operation
- `||`: OR operation
- `!`: NOT operation
- `>`/`>=`: greater than/greater than or equal to
- `<`/`<=`: less than/less than or equal to
- `==`: equal to
- `===`: exactly equal to
- Works in a similar way to `==`, except it also checks that the datatype is the same

Difference between == and ===

```
console.log("2" == 2);  /* true */
console.log("2" === 2); /* false */
```

Arrays

Arrays are an ordered list of values. They can hold values of many datatypes. Their index starts at 0.

```
let myArray = ["a", "b", "c", 1, 2 ,3, {name: "Jared", age: 19, canRead: false}]
myArray[0] // returns "a"
myArray[4] // returns 2
```

Objects

Objects are variables that can hold more than one value. One can be seen in the previous example in Arrays.
The different values of an object are called keys. The keys can hold regular primitive values, such as numbers or strings, or can hold other objects, such as functions. Think of Objects as a list of key/value pairs.

To access a key's value within an object, you must first reference the object in question, then insert a `.` folllowed by the key you wish to get.

Copyright Coders for Causes

Objects in JavaScript

```
let person = {
    name: "Jared",
    age: 19,
    canRead: false
}
console.log(person.age) // outputs 19
```

Functions

Functions are blocks of code designed to execute a particular task.
In JS, the syntax for defining a function is as follows:

Functions in JavaScript

```
function func_name(parameter1, parameter2, ...){
    // your code goes here (optional, but highly recommended)
}
```

Functions can be called or stored in variables.

Calling and Storing Functions

```
function hello_world(){
    console.log("Hello, world!")
}

// Calling the function
helloWorld()

// Storing the function in a variable
const func_name = function (param1, param2){
    // do stuff here
}
```

Functions can return a value (after calculation, etc) or simply perform work on existing data/variables. Functions that do not return anything are normally called *procedures*.

ARROW FUNCTIONS

Arrow functions are just a compact way of writing normal functions.
They work by removing the `function` key word, and even the `return` keyword in some cases.
Arrows functions lead with their parameters, usually enclosed in normal brackets. An arrow `=>` then follows, preceding the actual block of code to be executed.

Arrow Functions: Example 1

```
// Traditional function
function addXY (x, y){
    return x + y;
}

// Arrow function
let addXY = (x, y) => x + y;
```

In single-lined functions, such as the one above, both the { braces } and the `return` can be omitted. However, when there are extra lines of processing, both must be included.

Copyright Coders for Causes

Arrow Functions: Example 2

```
// Traditional
function add10(x, y){
    x += 10
    y += 10
    return x + y;
}

// Arrow function
let add10 = (x, y) => {
    x += 10
    y += 10
    return x + y;
}
```

Loops

For

For loops repeat until a condition is met. That condition is defined in the for loop. For loops have the following structure:

For Loops

```
/*
for(int i = start; condition; increment){
    // code here
}
*/

// Loop 10 times
for(int i = 0; i < 10; i++){
    console.log(i)
}
// Output: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```

For-in

For-in loops iterate over the indexes of data in an iterable object, such as an array.

For-in Loops

```
let myArray = [2, 4, 6]
for(let j in myArray){
    console.log(j)
}
// Output: 0, 1, 2
```

For-of

For-of loops iterate over the data in the iterable object.

For-of Loops

```
let myArray = [2, 4, 6]
for(let k of myArray){
    console.log(j)
}
// Output: 2, 4, 6
```


Copyright Coders for Causes

While

While loops iterate while a condition is true. They are called a "pre-test" loop, where the condition is tested before the loop can run. The condition is included in the brackets.

While Loops

```
let b = 0
while(b < 3){
    console.log("Bon")
    b++
}
// Output: "Bon\n Bon\n Bon\n"
```

Do-while

Do-while loops are similar to while loops, except that they let the block of code run once before testing the condition. They are known as "post-test" loops, and the loop is guaranteed to execute at least once.

Do-while Loops

```
let b = 0
do {
    console.log("Bon")
    b++
} while(b > 10)
// Output: "Bon\n"
```

Loops can be broken or stopped using the `break` or `continue` statements.

BREAKIN' OUT

- `break`: execution leaves the loop completely and continues on with the next lines of code
- `continue`: disregards the rest of the code in the loop block and moves on to the next item in the loop

3.5.6 Creating a Pokemon API Webapp

Now that we know a little bit about the tools of the web, let's build a simple web app that uses the skills we have learnt in this workshop, as well as some other skills we will learn along the way, to create an app that can do something cool.

For this tutorial, we will pay homage to the recent releases of Pokemon Brilliant Diamond and Shining Pearl, of which I have spent an embarrassing number of hours on since they came out about a week ago, and create a web app that uses the PokeAPI to display images and information about any Pokemon we want.

This tutorial will take place at 4:30pm AWST on the 31st of November, 2021 via the CFC Discord server. The recording of that tutorial will be here once it has finished.

Back to Top

Copyright Coders for Causes

3.6 Practical Software Engineering Practices

How software engineer works collaboratively!

3.6.1 How do software engineers work?

They use **issue/ticket management system** to track their work.

Copyright Coders for Causes

Github Issues and Jira

The examples are Github Issues and Jira.

Listing of issues can be displayed in **kanban boards**.

Copyright Coders for Causes

Kanban Boards

3.6.2 Why do Software Engineers Work like this?

Copyright Coders for Causes

Scrums

One example of "agile" ways of working and it looks like this.

3.6.3 What is Git?

Git is a

- **version control system**
- tools for **coordinating work for different changes**

Main Parts of Git

Main Parts of git:

- Change Node (commit node)
- Branch

Copyright Coders for Causes

Copyright Coders for Causes

Git Different Versions

Software Versions

Copyright Coders for Causes

Nodes - Representation of a change (commits + merge)

Main/Master: - Branch usually as production or deployed

Development Branch - Branch used for development Used for compiling features for a release

Feature Branch - Branch used for development for a particular issue

3.6.4 GitHub and Code Repositories

GitHub - Place to hold code repositories

Repositories - Place where codes are stored with different versions

3.6.5 INTERACTIVE WORKSHOP TIME !!!!

1. Do the following from codersforcauses/software-engineering-practices-demo: Demonstration of software engineering practices (github.com):
2. Look at the GitHub issues
3. Assign yourself an issue
4. Clone repo with `git clone https://github.com/codersforcauses/software-engineering-practices-demo-2021-summer.git`
5. Create a branch with the following format s{Issue Number}-{Issue Name}. `git checkout -b s{Issue Number}-{Issue Name}`
6. Create the change to satisfy or complete the github issue
7. Commit (package) the change `git commit -m "{insert message here}"` and push git push.
8. Create a pull request and request for a reviewer from CFC peeps

3.6.6 Other Resources

- Git and Github

Copyright Coders for Causes

3.7 Introduction to Frontend Frameworks

3.7.1 Introduction to Frontend Frameworks

An intuitive explanation of why frontend frameworks are used

Frontend Engineering

- Frontend Engineering, one of the most important aspect of web dev
- Feared by lots of developers (ehem ehem CSS)

Copyright Coders for Causes

Memes

Copyright Coders for Causes

Job Market and Trends

StackOverflow 2020

Can be seen that there is a huge portion that will needs frontend.

Note: Developers can identify as many types

Copyright Coders for Causes

Startups

GITHUB COPILOT AND DRAG AND DROPS

- Getting replaced by AIs, Drag and drops?
- Far from reality?

Copyright Coders for Causes

hub Copilot and Squarespace

Problems

CROSS-BROWSER SUPPORT

- Need to support multiple browsers

HTML IS WET!

Not DRY = WET

D – Don't

R – Repeat

Y - Yourself

Copyright Coders for Causes

Github Tags System

INEFFICIENCIES OF RELOADING

- Most parts of website don't change

Copyright Coders for Causes

Youtube

- What about with notifications and realtime updates?

Notifications

Copyright Coders for Causes

STATE MANAGEMENT AND RERENDERING

- How do you handle changes in the state of the application?

Example "Spotimania"

Realtime Multiplayer Song Guessing Game https://spotimania.herokuapp.com/

Information

- Song picture
- Notifications when someone joined or scored
- Song title + artist

How do we handle when we "Submit Guess"?

HTML Format Submit Approach

1. Reloads = loses current state
2. How do you coordinate with players?

Copyright Coders for Causes

AJAX Calls

1. From frontend, use JS to do AJAX Calls
2. After receiving AJAX call, modify everything that needs the data (Find element by DOM, do some janky HTML inject)

```
const newData = await fetch(…)
const image = document.querySelector('img .music')
image.outerText = `<img src={newData.image} / >`
```

Cross-site Scripting

Now, you have to implement your own strategy to do this.

```
newData = {… , image: "<script>hack()</script>", ...}
```

When you try to do render by that approach:

- slower (rerendering things that should not render)
- tedious (find element and update)
- security
- code complexity
- hard to keep track of states (especially if different portion of the UI relies on a certain data)

How do we fix it?

CROSS-BROWSER SUPPORT

Babel: Transpiles / Translates New JavaScript to old JavaScript that is supported

Copyright Coders for Causes

Webpack: Bundles / Packs Bundles the JavaScript modules into a single minified file

SCSS Compilation

Translates concise and programatic version of CSS to something that can be understood by the browser

```
.example {
    display: grid;
    transition: all .5s;
    user-select: none;
    background: linear-gradient(to bottom, white,
black);
}
```

```
.example {
    display: -ms-grid;
    display: grid;
    -webkit-transition: all .5s;
    -o-transition: all .5s;
    transition: all .5s;
    -webkit-user-select: none;
       -moz-user-select: none;
        -ms-user-select: none;
            user-select: none;
    background: -webkit-gradient(linear, left top,
left bottom, from(white), to(black));
    background: -webkit-linear-gradient(top, white,
black);
    background: -o-linear-gradient(top, white,
black);
    background: linear-gradient(to bottom, white,
black);
}
```


Copyright Coders for Causes

Minification

Makes files smaller by removing unnecessary characters

```
var gapi=window.gapi=window.gapi||{};gapi._bs=new Date(
","")};return[encodeURIComponent(a.g).replace(/%2C/g,","
b[d].split("="),l=decodeURIComponent(g[0]),k=decodeURIC
ga=/\/cb=/g,fa=/\/\//g,la=function(){var a=F();if(!a)th
g&&Object.prototype.hasOwnProperty.call(c,e)&&g(c[e],a,
z){var v=c.callback;v&&(c.callback=function(){0==--z&&v
((D||{}).config||{}).update;d?d(e):e&&w(E,"cu",[]).push
I,k);k.push.apply(k,p);N("ml0",p,v);b.sync||f.___gapisy
gapi.load("",{callback:window["gapi_onload"],_c:{"jsl":
```

VERSUS

```
angular.module('memryMain')
  .directive('defPanel', ["defModel", "$timeout", funct
    'use strict';
    return {
      replace: true,
      restrict: 'E',
      templateUrl: 'views/def-panel.html',
      scope: true,
      link: function (scope, element) {
```


Copyright Coders for Causes

WHAT IF HTML IS NOT WET? WHAT IF HTML IS A PROGRAMMING LANGUAGE?

Copyright Coders for Causes

Memes about HTML as a programming language

Copyright Coders for Causes

Copyright Coders for Causes

What if HTML has variables?

- perform computations to be displayed in the interface (solves **State Management and Rerendering**)

What if HTML has loops?

- Eliminate repetitive code (solves **wet HTML**)

What if HTML has functions?

- Divide interface into modular components (solves **wet HTML**)
- load only components that are not loaded (solves **Inefficiencies of Reloading**)
- interface determined by parameters
- Maybe we can write a function that will automatically update HTML when data changes? (solves **State Management and Rerendering**)

What if HTML has condition?

- Conditional rendering (solves Reloading to show different view)

Copyright Coders for Causes

What's next?

The next two workshop content are:

- React
- Vue

Copyright Coders for Causes

3.7.2 Introduction to React

Sparking the REACTion of User Interface and Experience

Copyright Coders for Causes

Decomposing the HTML User Interface

Complete code

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
        integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous" />
    <link rel="stylesheet" href="style.css" />
    <title>My Home Page - Frinze Lapuz</title>
</head>

<body>
    <nav class="navbar sticky-top navbar-dark bg-dark">
        <div class="container nav">
            <a class="navbar-brand" href="#">Royal Homepage</a>
        </div>
    </nav>
    <div class="container">
        <div class="row">
            <div class="col">Here Some Dog For You. Cool Stuff right?</div>
            <div class="col">
                <!-- <img class="img-fluid img-thumbnail" src="dog.jpg" alt="dog" /> -->
                <div class="carouselWrapper">
                    <div id="carouselDoggo" class="carousel slide" data-ride="carousel">
                        <ol class="carousel-indicators">
                            <li data-target="#carouselDoggo" data-slide-to="0" class="active"></li>
                            <li data-target="#carouselDoggo" data-slide-to="1"></li>
                            <li data-target="#carouselDoggo" data-slide-to="2"></li>
                        </ol>
                        <div class="carousel-inner">
                            <div class="carousel-item active">
                                <img src="dog.jpg" class="d-block w-100" alt="..." />
                                <div class="carousel-caption d-none d-md-block">
                                    <h5>First Dog</h5>
                                    <p>Cutie :)</p>
                                </div>
                            </div>
                            <div class="carousel-item">
                                <img src="dog2.jpg" class="d-block w-100" alt="..." />
                                <div class="carousel-caption d-none d-md-block">
                                    <h5>Second Dog</h5>
                                    <p>Good Doggo!</p>
                                </div>
                            </div>
                            <div class="carousel-item">
                                <img src="dog3.jpg" class="d-block w-100" alt="..." />
                                <div class="carousel-caption d-none d-md-block">
                                    <h5>Third Dog</h5>
                                    <p>Walkie Barkie Cutie!</p>
                                </div>
                            </div>
                        </div>
                        <a class="carousel-control-prev" href="#carouselDoggo" data-slide="prev">
                            <span class="carousel-control-prev-icon"></span>
                            <span class="sr-only">Previous</span>
                        </a>
                        <a class="carousel-control-next" href="#carouselDoggo" data-slide="next">
                            <span class="carousel-control-next-icon"></span>
                            <span class="sr-only">Next</span>
                        </a>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
        integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
        crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
        integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
        crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
        integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
        crossorigin="anonymous"></script>
</body>

</html>
```


Copyright Coders for Causes

Relevant Code for Discussion

```
<nav class="navbar sticky-top navbar-dark bg-dark"> <!--(1)-->
    <div class="container nav">
        <a class="navbar-brand" href="#">Royal Homepage</a>
    </div>
</nav>
<div class="container">
    <div class="row">
        <div class="col">Here Some Dog For You. Cool Stuff right?</div> <!--(2)-->
        <div class="col">
            <div class="carouselWrapper"> <!--(3)-->
                <div id="carouselDoggo" class="carousel slide" data-ride="carousel">
                    <ol class="carousel-indicators">
                        <li data-target="#carouselDoggo" data-slide-to="0" class="active"></li>
                        <li data-target="#carouselDoggo" data-slide-to="1"></li>
                        <li data-target="#carouselDoggo" data-slide-to="2"></li>
                    </ol>
                    <div class="carousel-inner">
                        <div class="carousel-item active">
                            <img src="dog.jpg" class="d-block w-100" alt="..." />
                            <div class="carousel-caption d-none d-md-block">
                                <h5>First Dog</h5>
                                <p>Cutie :)</p>
                            </div>
                        </div>
                        <div class="carousel-item">
                            <img src="dog2.jpg" class="d-block w-100" alt="..." />
                            <div class="carousel-caption d-none d-md-block">
                                <h5>Second Dog</h5>
                                <p>Good Doggo!</p>
                            </div>
                        </div>
                        <div class="carousel-item">
                            <img src="dog3.jpg" class="d-block w-100" alt="..." />
                            <div class="carousel-caption d-none d-md-block">
                                <h5>Third Dog</h5>
                                <p>Walkie Barkie Cutie!</p>
                            </div>
                        </div>
                    </div>
                    <a class="carousel-control-prev" href="#carouselDoggo" data-slide="prev">
                        <span class="carousel-control-prev-icon"></span>
                        <span class="sr-only">Previous</span>
                    </a>
                    <a class="carousel-control-next" href="#carouselDoggo" data-slide="next">
                        <span class="carousel-control-next-icon"></span>
                        <span class="sr-only">Next</span>
                    </a>
                </div>
            </div>
        </div>
    </div>
</div>
```

1. Navigation Bar
2. Some Text
3. Carousel

HTML STORED IN VARIABLES

```
const Nav =
    `<nav class="navbar sticky-top navbar-dark bg-dark">
        <div class="container nav">
            <a class="navbar-brand" href="#">Royal Homepage</a>
        </div>
    </nav>
    `
```

HTML STORED IN FUNCTIONS

```
const Nav = (title) =>
    `
    <nav class="navbar sticky-top navbar-dark bg-dark">
        <div class="container nav">
            <a class="navbar-brand" href="#">${title}</a>
        </div>
    </nav>
    `
```


Copyright Coders for Causes

Carousel Decomposition

The highlighted ones are the key-distinct data driven elements.

```
<div class="carouselWrapper">
    <div id="carouselDoggo" class="carousel slide" data-ride="carousel">
        <ol class="carousel-indicators">  <!--(1)-->
            <li data-target="#carouselDoggo" data-slide-to="0" class="active"></li>
            <li data-target="#carouselDoggo" data-slide-to="1"></li>
            <li data-target="#carouselDoggo" data-slide-to="2"></li>
        </ol>
        <div class="carousel-inner">
            <div class="carousel-item active"> <!--(2)-->
                <img src="dog.jpg" class="d-block w-100" alt="..." />
                <div class="carousel-caption d-none d-md-block">
                    <h5>First Dog</h5>
                    <p>Cutie :)</p>
                </div>
            </div>
            <div class="carousel-item">
                <img src="dog2.jpg" class="d-block w-100" alt="..." />
                <div class="carousel-caption d-none d-md-block">
                    <h5>Second Dog</h5>
                    <p>Good Doggo!</p>
                </div>
            </div>
            <div class="carousel-item">
                <img src="dog3.jpg" class="d-block w-100" alt="..." />
                <div class="carousel-caption d-none d-md-block">
                    <h5>Third Dog</h5>
                    <p>Walkie Barkie Cutie!</p>
                </div>
            </div>
        </div>
        <a class="carousel-control-prev" href="#carouselDoggo" data-slide="prev">
            <span class="carousel-control-prev-icon"></span>
            <span class="sr-only">Previous</span>
        </a>
        <a class="carousel-control-next" href="#carouselDoggo" data-slide="next">
            <span class="carousel-control-next-icon"></span>
            <span class="sr-only">Next</span>
        </a>
    </div>
</div>
```

1. Carousel Indicators
2. Carousel Items

Now, what if we have a variable called `dogsData` an array of objects (may come from `fetch`).

```
dogsData = [
    ...
    {
        imgSrc: "dog.jpg",
        title: "First Dog",
        caption: "Cutie :)"
    }
    ...
]
```


Copyright Coders for Causes

```
const Carousel = (dogsData) =>
    `
    <div class="carouselWrapper"> <!--(1)-->
        <div id="carouselDoggo" class="carousel slide" data-ride="carousel">
            <ol class="carousel-indicators">
                ${dogsData.map((dog, index) => `<li data-target="#carouselDoggo" data-slide-to="${index}" class="${index === 0 ? 'active' : ''}"></
li>`).join('')}
            </ol>
            <div class="carousel-inner">
                ${dogsData.map((dog, index) => `
                    <div class="carousel-item ${index === 0 ? 'active' : ''}">
                        <img src="${dog.imgSrc}" class="d-block w-100" alt="..." />
                        <div class="carousel-caption d-none d-md-block">
                            <h5>${dog.title}</h5>
                            <p>${dog.caption}</p>
                        </div>
                    </div>
                `).join('')}
            </div>
            <a class="carousel-control-prev" href="#carouselDoggo" data-slide="prev">
                <span class="carousel-control-prev-icon"></span>
                <span class="sr-only">Previous</span>
            </a>
            <a class="carousel-control-next" href="#carouselDoggo" data-slide="next">
                <span class="carousel-control-next-icon"></span>
                <span class="sr-only">Next</span>
            </a>
        </div>
    </div>
    `
```

ASSEMBLE THE COMPONENTS

```
{Nav("Royal Homepage")}
<div class="container">
    <div class="row">
        <div class="col-md-12">
            {Carousel(dogsData)}
        </div>
    </div>
```

or with JSX

```
<Nav title="Royal Homepage"></Nav>
<div class="container">
    <div class="row">
        <div class="col-md-12">
            <Carousel dogsData={dogsData}></Carousel>
        </div>
    </div>
```

Intro to React

Copyright Coders for Causes

Complete Modularised Code

```
const Nav = () => <nav class="navbar sticky-top navbar-dark bg-dark">
    <div class="container nav">
        <a class="navbar-brand" href="#">Royal Homepage</a>
    </div>
</nav>

const Carousel = (dogsData) =>
    <div class="carouselWrapper">
        <div id="carouselDoggo" class="carousel slide" data-ride="carousel">
            <ol class="carousel-indicators">
                ${dogsData.map((dog, index) => `<li data-target="#carouselDoggo" data-slide-to="${index}" class="${index === 0 ? 'active' : ''}"></
li>`).join('')}
            </ol>
            <div class="carousel-inner">
                ${dogsData.map((dog, index) => `
                    <div class="carousel-item ${index === 0 ? 'active' : ''}">
                        <img src="${dog.imgSrc}" class="d-block w-100" alt="..." />
                        <div class="carousel-caption d-none d-md-block">
                            <h5>${dog.title}</h5>
                            <p>${dog.caption}</p>
                        </div>
                    </div>
                `).join('')}
            </div>
            <a class="carousel-control-prev" href="#carouselDoggo" data-slide="prev">
                <span class="carousel-control-prev-icon"></span>
                <span class="sr-only">Previous</span>
            </a>
            <a class="carousel-control-next" href="#carouselDoggo" data-slide="next">
                <span class="carousel-control-next-icon"></span>
                <span class="sr-only">Next</span>
            </a>
        </div>
    </div>

const Homepage = () =>{
    const dogsData = fetch(...) // Get data from API
    return (
        <>
            <Nav title="Royal Homepage"></Nav>
            <div class="container">
                <div class="row">
                    <div class="col-md-12">
                        <Carousel dogsData={dogsData}></Carousel>
                    </div>
            </div>
        </>
    )
}
```

Getting Started with React

Getting Started with the app

To get started with react, use the following commands

`npx create-react-app my-app` where my-app is the name of the folder for where the react app is going to be.

`cd my-app`

Notice that it created a couple of folders. In a couple of sentences:

`src/index.js` - injects the user interface in the public/index.html (the front part of the build)

`public` - folder containing the static assets as well as the index.html

`package.json` - dependencies and scripts for the app

type `yarn start` to run the development version of React (it will run `react-scripts` start within the node_modules … so running `react-scripts` start will not exactly be the same). This will serve up React in a port

To be left as a live coding exercise. Live coding demonstration:

Demo

- Passing Array props
- show “rafce”

Copyright Coders for Causes

- useState when adding new item

Copyright Coders for Causes

3.7.3 Introduction to Vue

A Vuetiful Introduction to VueJS

Due to time constraints, we had to reuse the previous Winter Materials.

As always, see the youtube video playlist

Copyright Coders for Causes

3.8 Welcome to the Djangol!

We've got puns 'n' data, We've got everything you request

3.8.1 How does an application communicate with backend systems?

JavaScript Object Notation

Copyright Coders for Causes

JSON

Link

```
[
{
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    "zipcode": "92998-3874",
    "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
    }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
    }
},
...
{
    "id": 10,
    "name": "Clementina DuBuque",
    "username": "Moriah.Stanton",
    "email": "Rey.Padberg@karina.biz",
    "address": {
    "street": "Kattie Turnpike",
    "suite": "Suite 198",
    "city": "Lebsackbury",
    "zipcode": "31428-2261",
    "geo": {
        "lat": "-38.2386",
        "lng": "57.2232"
    }
    },
    "phone": "024-648-3804",
    "website": "ambrose.net",
    "company": {
    "name": "Hoeger LLC",
    "catchPhrase": "Centralized empowering task-force",
    "bs": "target end-to-end models"
    }
}
]
```

Query this endpoint using JavaScript

```
fetch("https://jsonplaceholder.typicode.com/users")
    .then(response => response.json())
    .then(data => console.log(data))
```

Other ways

JSON is not the only way. There are things like SOAP, RPC, but nevermind them for now as you will most likely have to deal with JSON for the most parts.

3.8.2 What are APIs and REST-APIs?

Application Programming Interface

Analogy

Who interacts with the user interface? - the user Who interacts with the application programming interface ? - the application program (eg. the browser)

Copyright Coders for Causes

Representational State Transfer Application Programming Interface

- backend architectural pattern that follows the GET/POST/PUT/PATCH/DELETE
- Can be represented in Swagger/ Open API specification

Swagger/ Open API specification

3.8.3 What is Django?

Copyright Coders for Causes

Django

- Python web framework for creating server-side application

Follows MVC:

- Model - database
- View – Interface (API or User Interface)
- Controller – URLs + routes

See Documentation

3.8.4 What is Django REST Framework (DRF)?

- library for creating REST-API
- just makes it easier develop REST-API

In:

- Authentication + Permission
- Generic API Views
- Serialisers (payload validation and format)

See Documentation

3.8.5 Features of Django and DRF

Serializers

- converter to and from Python Data types to JSON/XML
- can be used to do validation

More info here

Models

Models

- representation of database connection or schema Example: Todo Django + React

MODELS WITH SERIALIZERS

- Model already defined the structure of the data

Configuration file `settings.py`

One of the most important file - determines the app configurations and default

Example of Items you can configure

- timezones
- start of URL
- installed Django Apps
- middlewares
- usually becomes the "default" configs after environment variables
- database configuration

Copyright Coders for Causes

Route with URLs

Determines the URL route of the API

- starts with the `urls.py` in the config (settings.py)

3.8.6 Packaging a "Django App"

"Django Apps" are plugin system of Django. Django codebase prefers packaging each endpoint feature to a "Django App"

Authenticate with Microsoft

For example: django-saml2-auth-lw (used for Microsoft SAML SSO)

More documentation can be seen here

You can start creating your own Django app by following the steps:

1. `python manage.py startapp <app_name>`. This will bootstrap commonly used files for any "app".
2. Add the app in the `INSTALLED_APPS` in the settings.py
3. `python manage.py makemigrations` to create the migration scripts
4. `python manage.py migrate` to run the migration script

Views

The interface at which any application program interacts with to get something done in the backend.

Use DRF instead of Base Django

This is where it gets a little tricky. Django accomodates both frontend and backend, but where it shines is with DRF. So always refer to documentation regarding write it from the documentation.

If you plan to use Django as an entire full-stack option, then you may not need DRF at all. Just use the generic views.

Copyright Coders for Causes

FUNCTION-BASED VIEWS AND CLASS-BASED VIEWS

Codes

Summary: In DRF, there's a lot more advantage to use class-based views. However, if you want the flexibility and writing things on your own, then you might want to use function-based view.

Function-Based View

Link:

```
from django.shortcuts import render
from django.http import JsonResponse

from rest_framework.decorators import api_view
from rest_framework.response import Response
from .serializers import TaskSerializer

from .models import Task

@api_view(['GET'])
def apiOverview(request):
    api_urls = {
        'List':'/task-list/',
        'Detail View':'/task-detail/<str:pk>/',
        'Create':'/task-create/',
        'Update':'/task-update/<str:pk>/',
        'Delete':'/task-delete/<str:pk>/',
        }

    return Response(api_urls)

@api_view(['GET'])
def taskList(request):
    tasks = Task.objects.all().order_by('-id')
    serializer = TaskSerializer(tasks, many=True)
    return Response(serializer.data)

@api_view(['GET'])
def taskDetail(request, pk):
    tasks = Task.objects.get(id=pk)
    serializer = TaskSerializer(tasks, many=False)
    return Response(serializer.data)


@api_view(['POST'])
def taskCreate(request):
    serializer = TaskSerializer(data=request.data)

    if serializer.is_valid():
        serializer.save()

    return Response(serializer.data)

@api_view(['POST'])
def taskUpdate(request, pk):
    task = Task.objects.get(id=pk)
    serializer = TaskSerializer(instance=task, data=request.data)

    if serializer.is_valid():
        serializer.save()

    return Response(serializer.data)


@api_view(['DELETE'])
def taskDelete(request, pk):
    task = Task.objects.get(id=pk)
    task.delete()

    return Response('Item succsesfully delete!')
```

Class-Based View

To be seen in in the live demos

3.8.7 Django Admin

Django comes with a built-in user-interface for managing "Django Apps" with models.py By default it in the path `/admin`

Copyright Coders for Causes

3.8.8 Other Tools

DRF Frontend for Consuming API

When you use DRF for developing API, and use the generic API views and serializers, it will automatically create this for you.

Copyright Coders for Causes

DRF Frontend

Postman

Development tool for interacting with APIs:

- environment variables
- collection of possible endpoints
- send different JSON body

Copyright Coders for Causes

Postman

Python Debugger (pdb)

When you run the django `python manage.py runserver`, you can add `breakpoint()` to anywhere piece of your code. This will stop the terminal line to the breakpoint if it hits it. From here you can do debugging.

Copyright Coders for Causes

3.8.9 Demo - Live Coding

Demonstrate how to create a backend for a Todo app.

Copyright Coders for Causes

1. Setup Python environment `python -m venv venv` then activate with `source venv/bin/activate`
2. Install Django `pip install Django`
3. Setup django codebase `django-admin startproject todo`
4. Install DRF

```
# As per https://www.django-rest-framework.org/#installation
pip install djangorestframework
pip install markdown       # Markdown support for the browsable API.
pip install django-filter  # Filtering support
pip install pytz           # Timezone support
```

5. Freeze requirements `pip freeze > requirements.txt`
6. Do a python sqlite migration `python manage.py migrate` to initialise the Django database for the apps
7. Create a super user `python manage.py createsuperuser`
8. Run the django app with `python manage.py runserver`
9. Login with the super user and show Django Admin
10. Setup django app `python manage.py startapp api`
11. Create a model `models.py`

```
from django.db import models

# Todo Model
class Todo(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField()
    completed = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.title
```

12. Do a python sqlite migration `python manage.py makemigrations` and `python manage.py migrate`
13. Show sqlite
14. Create serialiser `serializers.py`

```
from api.models import Todo
from rest_framework import serializers

class TodoSerializer(serializers.ModelSerializer):
    """
    Serializer for Todo model
    """
    class Meta:
        model = Todo
        # All fields
        fields = '__all__'
```


Copyright Coders for Causes

15. Create Class-based view `views.py` and link back to the `urls.py`

```python
# api/views.py
from django.shortcuts import render
from rest_framework import viewsets, permissions
from api.serializers import TodoSerializer
from api.models import Todo

# Class Model Viewset
class TodoModelViewSet(viewsets.ModelViewSet):
    # Define the serializer class
    serializer_class = TodoSerializer
    # Define the queryset
    queryset = Todo.objects.all()

    # Permissions (left to your own exercise)
    # permission_classes = [permissions.IsAuthenticated]

    # Define the list of allowed HTTP methods (by default if you didn't define it, it will just enable all)
    http_method_names = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace']
```

```python
# api/urls.py
from django.urls import path, include
from rest_framework import routers
from api.views import TodoModelViewSet

router = routers.DefaultRouter(trailing_slash=False)
router.register(r'todos', TodoModelViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

```python
# urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api-auth/', include('rest_framework.urls')),
    path('api/', include('api.urls')),
]
```

16. Manually test with DRF Frontend or postman

Final code can be seen here.

Read through the Viewset Documentation

Viewsets are one of the most powerful feature ever. In just a couple of lines of code, you can create a CRUD API for your models.

As one of my favourite developer once said,

> **Quote**
>
> Frinze, Viewset is the future! - J. Morales

Copyright Coders for Causes

3.9 MVC-MVMM Architecture and Communications

Separate the concerns with Design Patterns!

3.9.1 What is MVC/MVVM Architecture?

MVC (Model-View-Controller)

- Model – data representation layer (how app communicates to db – Raw or ORM)
- View – interface representation (with HTML, CSS and JS)
- Controller – request relays (processes HTTP requests)

MVC

MVVM (Model-ViewModel-View)

ViewModel – data-binding relays (processes the HTTP requests)

Copyright Coders for Causes

MVVM

Some inaccuracy

I acknowledge that this may not be correct. If you search MVC/MVMM, you will see lots of conflicting details. This portion of the slide is explained to make frontend-backend integration easier to explain.

Copyright Coders for Causes

Examples of Technologies with the Architecture

MVC - Django

Copyright Coders for Causes

MVVM - Django + React

Copyright Coders for Causes

MVC - Flask

Copyright Coders for Causes

MVVM - Flask + React

Copyright Coders for Causes

MVC - Express

Copyright Coders for Causes

MVVM - Express + React

3.9.2 Why is software architecture important?

Architecture shows how an entire software works in a high-level.

Important for:

- maintainability and extensibility
- seperation of concerns
- choosing different pieces of the stack

Copyright Coders for Causes

Just some architecture diagrams

3.9.3 How do frontend communicate with backend?

Via HTTP Requests responds with:

- HTML, CSS, JS
- JSON/XML

Copyright Coders for Causes

Server-side Rendering

Via - HTML, CSS, JS

Data Flow

Client-side Rendering and AJAX

Via HTTP Requests responds with: - blank HTML with JS (at first) - JSON/XML

Data Flow

Copyright Coders for Causes

CRUD to HTTP Verb Matching for JSON standard communications with REST-APIs

Matches HTTP verbs with action

AJAX – Asynchronous JavaScript and ~~XML~~ JSON

JSON are strings

JSON (JavaScript Object Notation) Note: JSONs are String... you have to serialise JSON to become an object

```
{} sample.json > [ ] data
1  {
2      "data": [
3          {
4              "type": "articles",
5              "id": "1",
6              "attributes": {
7                  "title": "Working with JSON Data in python",
8                  "description": "This article explains the various ways to work with JSON data in python.",
9                  "created": "2020-12-28T14:56:29.000Z",
10                 "updated": "2020-12-28T14:56:28.000Z"
11             },
12             "author": {
13                 "id": "1",
14                 "name": "Aveek Das"
15             }
16         }
17     ]
18 }
```

3.9.4 Demo - React to integrate with Django

This follows up the demo with React to consume a standard REST-API Django Todo App demonstrated in the previous workshop.

Copyright Coders for Causes

3.9.5 Other Information

- Traversy Media MVC
- React Query
- Axios (a better Fetch?) Travery Media
- Async/Await JavaScript

Copyright Coders for Causes

3.10 Package Managers

Unlock the power of open source

3.10.1 What are package managers?

- Developer tools that allows access to libraries

amples

- JavaScript (NPM/Yarn)
- Python(Pip/Poetry)

agram of the Package Manager Data Flow

Copyright Coders for Causes

Modern Infrastructure Dependency

Examples of Actual Libraries and their dependencies

See this tool to visualise dependency.

Copyright Coders for Causes

MUI Dependency

Copyright Coders for Causes

Express Dependency

3.10.2 Install Package Managers

- NPM Download | Node.js (nodejs.org)
- Yarn Installation | Yarn (yarnpkg.com)
- Pip Download Python | Python.org
- Poetry Poetry - Python dependency management and packaging made easy (python-poetry.org)

Copyright Coders for Causes

Actual Examples of Real-World Projects and their dependencies

Copyright Coders for Causes

foodbank `package.json`

```
{
    "name": "foodbank", // (1)
    "version": "0.1.0",
    "private": true,
    "browserslist": [
        ">0.3%",
        "not ie 11",
        "not dead",
        "not op_mini all"
    ],
    "repository": {
        "type": "git",
        "url": "git+https://github.com/codersforcauses/foodbank.git"
    },
    "author": "Coders for Causes",
    "license": "MIT",
    "bugs": {
        "url": "https://github.com/codersforcauses/foodbank/issues"
    },
    "homepage": "https://github.com/codersforcauses/foodbank#readme",
    "scripts": { // (2)
        "dev": "next dev",
        "build": "next build",
        "start": "next start",
        "lint": "next lint",
        "prepare": "husky install"
    },
    "dependencies": { // (3)
        "@headlessui/react": "1.4.2",
        "critters": "0.0.15",
        "keen-slider": "6.0.5",
        "next": "12.0.4",
        "react": "17.0.2",
        "react-dom": "17.0.2",
        "react-hook-form": "7.20.5"
    },
    "devDependencies": { // (4)
        "@types/react": "17.0.37",
        "autoprefixer": "10.4.0",
        "eslint": "7.29.0",
        "eslint-config-next": "12.0.4",
        "eslint-config-prettier": "8.3.0",
        "eslint-plugin-jsx-a11y": "6.5.1",
        "eslint-plugin-prettier": "4.0.0",
        "husky": "7.0.4",
        "postcss": "8.4.4",
        "prettier": "2.5.0",
        "tailwindcss": "2.2.19",
        "typescript": "4.5.2"
    }
}
```


Copyright Coders for Causes

1. Package Headers and Meta Information (line 2-20)

   Contains some meta information about the project. Eg. used for publication in npmjs.org

2. Scripts

   Alias of commands to run using the modules in the dependency

   ```
   yarn start // yarn [script-name]
   ```

   or

   ```
   npm run start // npm run [script-name]
   ```

3. Dependencies

   Libraries that are used in the project

   To add a dependency, run

   ```
   yarn add formik // yarn add [library-name]
   ```

   or

   ```
   npm install formik // npm install [library-name]
   ```

4. Dev-Dependencies

   Libraries that are used in the project on development environments Typically: linters, and testers

   ```
   yarn add -D prettier // yarn add -D [library-name]
   ```

   or

   ```
   npm install -D prettier // npm install -D [library-name]
   ```

3.10.3 Lockfiles

It locks the dependencies and sub dependencies (dependency of dependency) of your application the last time it installed correctly.

Files: - `package-lock.json` - `Yarn-lock.json` - `Poetry.lock`

E.g. wais/yarn.lock at main · codersforcauses/wais (github.com)

3.10.4 When to install globally?

Whenever installing package, it is recommended to install it for a specific project The only thing you should really install globally are CLI tools. Even then, you have npx

```
yarn global add [dependency]
```

or

```
npm install –g [dependency]
```


Copyright Coders for Causes

Firebase CLI

```
npm install -g firebase-tools
```

3.10.5 Demo - Material UI

1. Create a react-app `npx create-react-app mui-demo`
2. Move directory `cd mui-demo`
3. Install react-spring `yarn add @mui/material @mui/icons-material`
4. Start the app `yarn start`

Copyright Coders for Causes

5. Show Bottom Navigation

```
import * as React from 'react';
import BottomNavigation from '@mui/material/BottomNavigation';
import BottomNavigationAction from '@mui/material/BottomNavigationAction';
import FolderIcon from '@mui/icons-material/Folder';
import RestoreIcon from '@mui/icons-material/Restore';
import FavoriteIcon from '@mui/icons-material/Favorite';
import LocationOnIcon from '@mui/icons-material/LocationOn';

export default function LabelBottomNavigation() {
  const [value, setValue] = React.useState('recents');

  const handleChange = (event, newValue) => {
    setValue(newValue);
  };

  return (
    <BottomNavigation sx={{ width: 500 }} value={value} onChange={handleChange}>
      <BottomNavigationAction
        label="Recents"
        value="recents"
        icon={<RestoreIcon />}
      />
      <BottomNavigationAction
        label="Favorites"
        value="favorites"
        icon={<FavoriteIcon />}
      />
      <BottomNavigationAction
        label="Nearby"
        value="nearby"
        icon={<LocationOnIcon />}
      />
      <BottomNavigationAction label="Folder" value="folder" icon={<FolderIcon />} />
    </BottomNavigation>
  );
}
```


Copyright Coders for Causes

3.11 Typical Codebase Structure

Codebase Jigsaw: How it all fits together!

Info

This presentation is only intended to be a brief overview of the typical codebase structure

If you want more information about anything mentioned feel free to reach out to CFC committee and project helpers!

3.11.1 Overview and Importance

- Most codebase will follow almost the same pattern (if they change, it will only change with little quirks)
- Structure helps every software engineer understand codebase and where to put it certain codes (better code readability = better collaboration = better profit!)

Profit

3.11.2 Frontend

Example: About Us - Coders for Causes Website

Copyright Coders for Causes

Different Folders and Purpose

COMPONENTS

Components are individual parts/pieces of user interfaces

Copyright Coders for Causes

PAGES

Pages are the direct components that are related specific to a route or URL

PUBLIC / ASSETS

Assets are usually elements that are referenced E.g. Images, Videos

Copyright Coders for Causes

UTILITIES / HELPERS

Pieces of code that are used as utilities or helpers to other codes

Copyright Coders for Causes

TESTS

- Scripts that tests that code works
- Usually E2E frontend

Comparison with Two Codebases

Slight difference, look at software documentation or ask other engineers

Copyright Coders for Causes

3.11.3 Backend

Context - Role and Subject

In these example, when I mean "role", I mean the general role of file. "subject" means the feature or group functionality

Method of Organisation

SUBJECT-ORIENTED DRIVEN STRUCTURE

As it can be seen here, both **different folders** will have the **same filenames based on its role**. App grows, those files can itself become folders as well.

Example: erdem/DRF-TDD-example: An example Django REST framework project for test driven development. (github.com)

Copyright Coders for Causes

ROLE-ORIENTED DRIVEN STRUCTURE

As it can be seen here, both **both different folders** will have the same **filenames based on its subject**

Example: our-lab-training/olt-backend: Our Lab Training - Feathers Backend (github.com)

Different Folders and Purpose

MODELS / DATABASE CONNECTION

Object-Relational Model (ORM). Links the database into your app into objects

Copyright Coders for Causes

master olt-backend / src / models / notifications.model.js / Jump to

mrfrase3 initial notifications and events

1 contributor

27 lines (24 sloc) | 710 Bytes

```javascript
// notifications-model.js - A mongoose model
//
// See http://mongoosejs.com/docs/models.html
// for more of what you can do here.

const DefaultSchema = require('../types/default.schema');
const ObjectIdType = require('../types/objectId.type');

module.exports = function (app) {
  const mongooseClient = app.get('mongooseClient');
  const notifications = DefaultSchema(app);
  notifications.add({
    ntId: ObjectIdType('notify-templates', app),
    userIds: [ObjectIdType('users', app)],
    sendOn: {
      type: Date,
      required: true,
    },
    sent: {
      type: Boolean,
      required: true,
      default: false,
    },
  });

  return mongooseClient.model('notifications', notifications);
};
```

master DRF-TDD-example / todoapp / todos / models.py / Jump to

rigaspapas Support Django 2 ...

2 contributors

20 lines (15 sloc) | 679 Bytes

```python
from __future__ import unicode_literals
from django.conf import settings

from django.db import models
from django.utils.encoding import smart_text as smart_unicode
from django.utils.translation import ugettext_lazy as _


class Todo(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    name = models.CharField(_("Name"), max_length=255)
    done = models.BooleanField(_("Done"), default=False)
    date_created = models.DateTimeField(_("Date Created"), auto_now_add=True)

    class Meta:
        verbose_name = _("Todo")
        verbose_name_plural = _("Todos")

    def __unicode__(self):
        return smart_unicode(self.name)
```

ROUTING

Determines the rules for the URL pattern

master olt-backend / src / services / notifications / notifications.service.js / Jump to

mrfrase3 plugin notifications timeout and events listeners

1 contributor

32 lines (26 sloc) | 1.03 KB

```javascript
// Initializes the `notifications` service on path `/notifications`
const createService = require('feathers-mongoose');
const createModel = require('../../models/notifications.model');
const hooks = require('./notifications.hooks');
const timeout = require('./notifications.timeout');
const notifme = require('@feathers-nuxt/feathers-notifme');
const { disallow } = require('feathers-hooks-common');

module.exports = function (app) {
  timeout(app);
  const Model = createModel(app);
  const paginate = app.get('paginate');

  const options = {
    Model,
    paginate
  };

  // Initialize our service with any options it requires
  app.use('/notifications', createService(options));

  // Get our initialized service so that we can register hooks
  const service = app.service('notifications');

  service.hooks(hooks);

  app.use('/notifme', notifme({
    useNotificationCatcher: process.env.NODE_ENV !== 'production',
    channels: app.get('notifyChannels'),
  }));
  app.service('notifme').hooks({ before: { all: [disallow('external')] } });
};
```

master DRF-TDD-example / todoapp / todos / urls.py / Jump to

ulgens Use Django2 style URLs

3 contributors

9 lines (7 sloc) | 260 Bytes

```python
from django.urls import path
from todos.views import TodoListCreateAPIView, TodoDetailAPIView

app_name = 'todos'

urlpatterns = [
    path('', TodoListCreateAPIView.as_view(), name="list"),
    path('<int:pk>/', TodoDetailAPIView.as_view(), name="detail"),
]
```

VIEWS

- In REST API, Views are your main “view” of the API

Copyright Coders for Causes

master DRF-TDD-example / todoapp / todos / views.py / Jump to

erdem implemented queryset method for todo list view

1 contributor

24 lines (15 sloc) | 719 Bytes

```python
from rest_framework.generics import ListCreateAPIView, RetrieveUpdateDestroyAPIView
from rest_framework.permissions import IsAuthenticated

from todos.models import Todo
from todos.permissions import UserIsOwnerTodo
from todos.serializers import TodoSerializer


class TodoListCreateAPIView(ListCreateAPIView):
    serializer_class = TodoSerializer

    def get_queryset(self):
        return Todo.objects.filter(user=self.request.user)

    def perform_create(self, serializer):
        serializer.save(user=self.request.user)


class TodoDetailAPIView(RetrieveUpdateDestroyAPIView):
    serializer_class = TodoSerializer
    queryset = Todo.objects.all()
    permission_classes = (IsAuthenticated, UserIsOwnerTodo)

```

develop IndEAA / server / src / services / course-evaluation / course-evaluation.class.js /

frinzekt course evaluation backend service init

1 contributor

We found potential security vulnerabilities in your dependencies.
You can see this message because you have been granted access to Dependabot alerts for this repository.

5 lines (3 sloc) | 125 Bytes

```python
const { Service } = require('feathers-mongoose');

exports.CourseEvaluation = class CourseEvaluation extends Service {

};
```

TESTS

- Contains unit testing (+ possibly e2e testing)
- Most common candidate for transitioning from file to a folder containing more files

3.11.4 Other Resources

- Writing your first Django app, part 1 | Django documentation | Django (djangoproject.com)
- File Structure – React (reactjs.org)
- Best practice for Django project working directory structure - Stack Overflow

Copyright Coders for Causes

3.12 Automated Testing and CI Pipelines

3.12.1 What is the problem with manual testing?

- tedious
- some scenarious are hard to repeat
- eg. network issues, database randomly crashes
- repetitive
- growning number of scenarios
- time consuming

Cost Savings

Scenarios that are hard to repeat - Network Requests

```python
import requests
def make_network_request():
try:
    response = requests.get("https://www.google.com")
    if response.status_code == 200:
        return response.text
    else:
        return None
exception requests.exceptions.ConnectionError:
    return None
```

This is the way to simulate an exception after a network request.

```python
import pytest
def test_make_network_request_exception(mocker):
    mocker.patch("requests.get", side_effect=requests.exceptions.ConnectionError)
    assert make_network_request() is None
```

3.12.2 Importance of Software

Before you can truly understand why testing is important, we first have to understand the importance of software.

Copyright Coders for Causes

portance of software in different parts

Control Systems

Healthcare

Copyright Coders for Causes

Transportations

Daily life

Office Work

Copyright Coders for Causes

Education

3.12.3 Software is Fragile

> **Quote**
>
> It only takes a character difference to prevent a software from compiling or running.

Copyright Coders for Causes

```
: prin("hi")

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-091467a1e20e> in <module>()
----> 1 prin("hi")

NameError: name 'prin' is not defined
```

```
    self._get_fundamentals(proxy)
  File "C:\PythonA\lib\site-packages\yfinance\base.py", line 284, in _get_fundamentals
    holders = _pd.read_html(url)
  File "C:\PythonA\lib\site-packages\pandas\io\html.py", line 1105, in read_html
    displayed_only=displayed_only,
  File "C:\PythonA\lib\site-packages\pandas\io\html.py", line 912, in _parse
    raise_with_traceback(retained)
  File "C:\PythonA\lib\site-packages\pandas\compat\__init__.py", line 46, in raise_with_traceback
    raise exc.with_traceback(traceback)
ValueError: No tables found

Process finished with exit code 1
```

3.12.4 Consequences of Fault Softwares

Huge Compute Charges

Copyright Coders for Causes

Embarrassment

Disruptions

disruptions

User Frustations

Copyright Coders for Causes

user-frustations

Client loss

Death

death

3.12.5 Automated Testing

What is involved manual testing usually?

To be able to automate testing, we need to understand what is involved in manual testing.

- Open your own environment and tools
- Git checkout/pull/fetch
- Install dependency
- Setup the environment
- Do you need to clear the database?
- Do you need to enter a couple of data?
- Do you need to disconnect the internet while server is running?
- Perform the steps to test
- Assert the test correction?
- Did you make a mistake? (repeat the entire thing)
- Now do a different scenario!
- Hang on a sec? Your coworker forgot to push? Now do all scenarios again

How does automated testing work?

- Write scripts to evaluate code

Copyright Coders for Causes

- Write script to run scripts when certain events happen (more so later)
- Update Scripts whenever change happens

Example from the workshop with github actions - Software Engineering Practices

```yaml
name: Python package

on:
pull_request:
workflow_dispatch:
    inputs:
    logLevel:
        description: "Log level"
        required: false
        default: "warning"

jobs:
flake8_py3:
    name: "Linting with Flake8"
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2

    - name: "Set up Python"
        uses: actions/setup-python@v2
        with:
        python-version: 3.x

    - name: "Install flake8"
        run: pip install flake8

    - name: "Run flake8"
        uses: liskin/gh-problem-matcher-wrap@v1
        with:
        linters: flake8
        run: flake8 --max-line-length 150 .

test:
    runs-on: ubuntu-latest
    strategy:
    matrix:
        python-version: [3.8] # you can add more python version to test it in

    steps:
    - uses: actions/checkout@v2
    - name: Set up Python ${{ matrix.python-version }}
    uses: actions/setup-python@v2
    with:
        python-version: ${{ matrix.python-version }}
    - name: Install dependencies
    run: |
        python -m pip install --upgrade pip
        if [ -f requirements.txt ]; then pip install -r requirements.txt; fi
    - name: Test with pytest
    run: |
        pytest --cov-report html --cov=.
    - uses: actions/upload-artifact@v1
    if: always()
    with:
        name: coverage-report
        path: htmlcov
```

Test-Driven Development

Define test/specification first, then write functionality. Works best if you know way ahead of time what you want.

Note: This does not always work (eg. Frontend)

Simple Example

```python
def test_add_two_numbers():
assert add_two_numbers(1, 2) == 3
```

```python
def add_two_numbers(a, b):
    return a + b
```


Copyright Coders for Causes

More complex example, but demonstrates how it can save time

Let say you have to develop a function that sends email depending on the number of users in the database

```
from django.core.mail import send_mail
# Not guaranteed to work code
def send_email_to_admin():
    number_of_users = User.objects.count()
    send_mail("Number of users", f"There are {number_of_users} users in the database", "no-reply@system.com",["admin@system.com"])
```

```
from django.core import mail
from module import send_email_to_admin
class EmailTest(TestCase):
    def setup():
        # Create users
        User.objects.create(username="admin" ...)

    def test_send_email_to_admin(self):
        send_email_to_admin()

        # Test that one message has been sent.
        self.assertEqual(len(mail.outbox), 1)

        self.assertEqual(mail.outbox[0].subject, "Number of users")
        self.assertTrue("1 users in the database" in mail.outbox[0].body)
        ...
```

Usual workflow:

1. Prepare your tools - mail client and smtp server
2. Prepare your database
3. Do manual testing + Write code (repeat)
4. Write Automated testing

Test Driven development workflow (also known as red-green-refactor):

1. Write the Automated Testing - when this is run it will fail
2. Write code until test pass
3. Refactor + improve code

GIVEN-WHEN-THEN

Acceptance of features can be divided into: - GIVEN – what are the assumptions - WHEN – occurence - THEN - effect or observation

3.12.6 What is CI/CD?

Scripts that runs for certain events (such as pull request, a commit) to run test or deploy

Providers

- Github Actions
- Circle CI
- Bitbucket Pipelines
- Travis CI

What's usually in CI?

- Linting (code consistency + undefined variables)
- Tests
- Builds + Deploy

Copyright Coders for Causes

Others: - bot to auto tag issue - Security checks - Code Autosuggestion (kinda like linting, but it suggests stuff)

Testing Artifacts

Files given when testing

Coverage Report

coverage

Cypress Screenshots + Videos

Copyright Coders for Causes

3.12.7 Demo - Cypress Integration Test / E2E Test

Cypress is a tool that simulates user interaction by programatically performing action against a browser.

Get started at Cypress.

Copyright Coders for Causes

4. 2021 Winter

4.1 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.

4.1.1 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).

4.1.2 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.

Copyright Coders for Causes

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

Copyright Coders for Causes