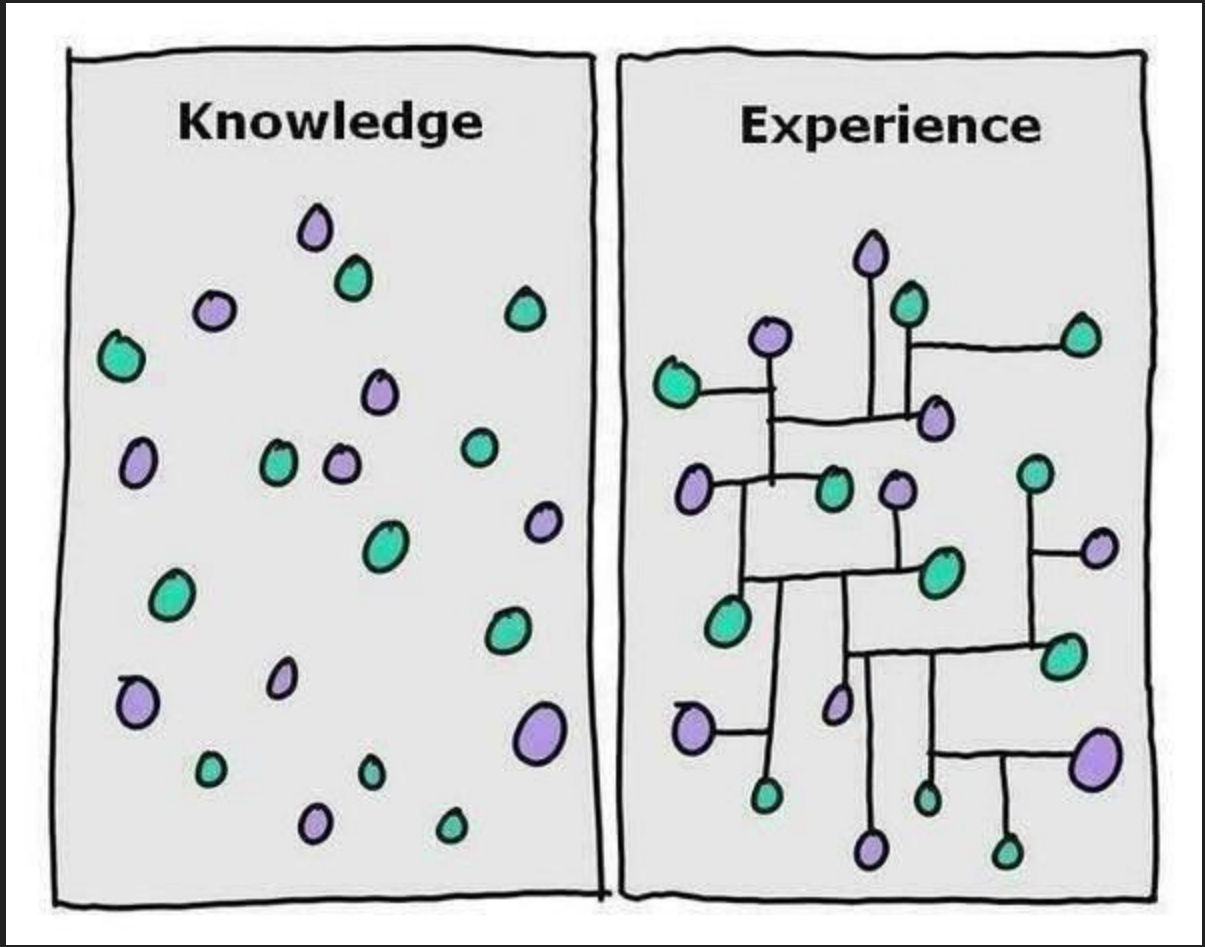
Day 6

Building up experience


Knowledge
Experience

Today

- Recap
  - Web basics - html/css/js
  - Frontend ecosystem, node vs browser runtime, npm
- Build new app
  - use ng cli
  - generate components/services
  - connect with backend
  - repeat
- Useful resources

Basic web - html/css/js

- Browsers understand html/css/js (1)
- Browsers do not understand (2):
  - Angular templates or jsx
  - scss, less, stylus…
  - All versions of EcmaScript
  - TypeScript

- Thus! Compiling, transpiling, converting from (2) to (1):
  - Angular compiler
  - TypeScript compiler
  - SCSS, postcss, …
- Tip: “sourcemap” is something browser can use to map html/css/js to original source when inspecting in DevTools

Frontend ecosystem

- Many EcmaScript (JS) engines:
  - **Chakra** - Internet Explorer, MS Edge
  - **SpiderMonkey** - Firefox
  - **JavaScriptCore** - Webkit, Safari
  - **V8** - Chromium, Chrome, Node.JS

- Node.JS vs Browsers runtime:
  - No DOM
  - Different security sandbox
  - …

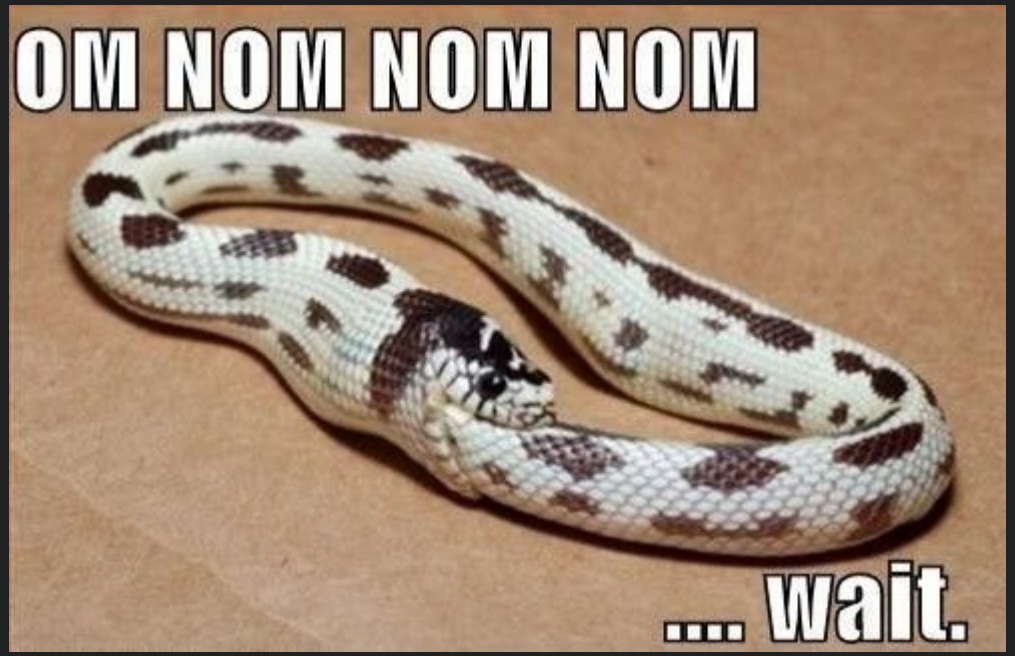
npm

- Node package manager
- Install and remove dependencies

- npm itself is:
  - JavaScript program
  - Node package

- npmjs.org:
  - Registry of node packages
  - Most popular
  - A lot of good open source solutions!

npm install –global npm@latest



Build new app

Task #1 - prepare & generate new app

- Check environment:
  - node -v
  - npm -v
- Install latest version of Angular CLI:
  - npm install -g @angular/cli
- Check if all “tools” are in place:
  - ng version

```
❯ ng version
     _                      _                 ____ _     ___
    / \   _ __   __ _ _   _| | __ _ _ __     / ___| |   |_ _|
   / △ \ | '_ \ / _` | | | | |/ _` | '__|   | |   | |    | |
  / ___ \| | | | (_| | |_| | | (_| | |      | |___| |___ | |
 /_/   \_\_| |_|\__, |\__,_|_|\__,_|_|       \____|_____|___|
                |___/


Angular CLI: 13.3.0
Node: 16.14.2
Package Manager: npm 8.5.5
OS: darwin x64
```

- Generate new Angular app:
  - ng new ita22
  - Details: https://angular.io/cli/new

- Observe and explore what was generated (ask questions)

Task #2 - running & building app

- Change directory
  - cd ita22
- Start serving application locally (dev environment)
  - ng serve --open
  - Details: https://angular.io/cli/serve
- Build production version of application
  - ng build
  - Details: https://angular.io/cli/build
- Serve “production” build with http-server
  - npx http-server ./dist/ita22
    - npx - “npm exec” downloads/installs node package and executes it
- Try production build with sourcemaps
  - ng build --source-map=true
  - npx http-server ./dist/ita22
- Observe “source” availability/readability in DevTools
- Run tests
  - ng test
  - Details: https://angular.io/cli/test


ONE DOES NOT SIMPLY
RUN NG SERVE IN PRODUCTION
imgflip.com

Grab: Task #1 & #2

- Clone from GitHub
  - git clone https://github.com/minijus/ita22.git
- Change directory
- Checkout branch
  - git checkout task-1-2
- Install dependencies
  - npm ci

Task #3 - mocked server, contract based development

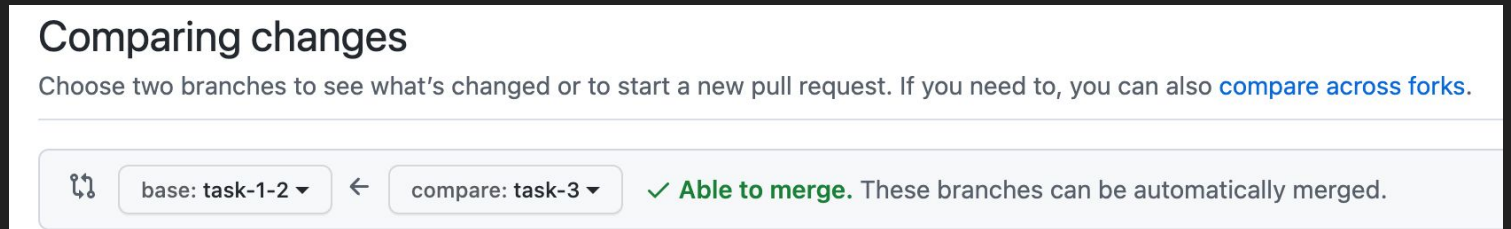
- Create mocked data
  - New file data.json from https://pastebin.com/raw/YCNXhWEq
  - curl https://pastebin.com/raw/YCNXhWEq -o data.json
- Try [json-server]() with npx
  - npx json-server data.json
- Add json-server as project devDependency
  - npm install json-server --save-dev
- Add npm script in package.json to start data server
  - *"start:backend": "json-server --watch data.json"*
  - npm run start:backend

A *contract* is only as strong as its weakest link.

Grab: Task #3

- Clone from GitHub
  - git clone https://github.com/minijus/ita22.git
- Change directory
- Checkout branch
  - git checkout task-3
- Install dependencies
  - npm ci

- Tip! Use GitHub compare between branches:
  - https://github.com/minijus/ita22/compare/task-1-2...task-3



Task #4 - transactions module, lazy loading

- Generate new transactions module
  - ng generate module transactions
  - Details https://angular.io/cli/generate#module-command
- Observe and remove changes
  - Use --dry-run
- Generate new transactions module with routing
  - ng g m transactions --module=app.module.ts --routing --route=transactions --dry-run
  - Remove --dry-run when output looks ok

Task #4

- Clean AppComponent template:
  - Add menu with routerLink to /transactions
  - Leave <router-outlet>
- Observe lazy loading of Transactions module

- Repeat with new modules for:
  - Categories
  - Accounts
  - Payees

Task #4+ - adding prettier

- Install devDependency prettier
  - npm i prettier -D
- Create *.prettierignore*
- Create *.prettierrc*
- Add npm script
  - *"format": "prettier --write ."*
- Run it:
  - npm run format
- Setup IDE to format on save

.prettierignore

```
.angular
.vscode
dist
node_modules
```

.prettierrc

```
{
  "printWidth": 120
}
```

Grab: Task #4

- Clone from GitHub
  - git clone https://github.com/minijus/ita22.git
- Change directory
- Checkout branch
  - git checkout task-4
- Install dependencies
  - npm ci

Task #5 - router, SCAM

- Display transactions on default route
  - Add redirect from “/” to “/transactions”
- Handle Not Found cases
  - Generate new NotFoundModule
  - ng g m not-found -m app.module.ts --routing --route=’**’
  - Make sure catch all path ‘**’ open not found component

- SCAM - Single Component Angular Module

Grab: Task #5

- Clone from GitHub
  - git clone https://github.com/minijus/ita22.git
- Change directory
- Checkout branch
  - git checkout task-5
- Install dependencies
  - npm ci

Task #6 - real SCAM, MainMenuModule

- Generate new main-menu module
  - ng g m main-menu -m app.module.ts
- Generate new main-menu component within MainMenuModule
  - ng generate component main-menu/main-menu --export --flat
  - Details: https://angular.io/cli/generate#component-command
- Move main-menu section from AppComponent to MainMenuComponent
- Use MainMenuComponent inside AppComponent

- What's missing?
  - ng doc routerLink
  - Import RouterModule

Grab: Task #6

- Clone from GitHub
  - git clone https://github.com/minijus/ita22.git
- Change directory
- Checkout branch
  - git checkout task-6
- Install dependencies
  - npm ci

Task #7 - fetching data, HttpClient, proxyConfig, CORS

- Generate new TransactionsService within TransactionsModule scope
  - ng generate service transactions/services/transactions
  - Details https://angular.io/cli/generate#service-command
- Generate new Transaction interface within TransactionsModule scope
  - ng generate interface transactions/interfaces/transaction
  - Details https://angular.io/cli/generate#interface-command
- Implement getTransactions method in TransactionsService
  - Import HttpClient
  - Use ‘get’ for /api/transactions
  - Add typings using Transaction interface
- Render transactions$ in TransactionComponent:
  - {{transactions$ | async | json}}

- What’s missing?
  - ng doc HttpClient
  - Import HttpClientModule:
    - https://angular.io/guide/http
    - It says most apps import to root AppModule


WHY YOU NO WORK?
WHY?

Task #7 - fetching data, HttpClient, proxyConfig

- Create proxyConfig file:
  - ng doc proxyConfig
- Reference proxy config in angular.json for “serve”
  - “proxyConfig": "src/proxy.conf.json"
- Restart server
  - Stop current instance CTRL/CMD + C
  - ng serve
- Do not forget to start backend!
  - npm run start:backend

```
{
  "/api": {
    "target": "http://localhost:3000",
    "secure": false,
    "pathRewrite": {
      "^/api": ""
    },
    "logLevel": "debug"
  }
}
```

Task #7 - fetching data, HttpClient, proxyConfig

- proxyConfig is local
  - Works only in development environment (ng serve)

- Production will need web server configuration
- Or possible CORS issues
  - Cross-Origin Request Sharing
  - All you need to know about CORS
  - https://jakearchibald.com/2021/cors/

Task #7 - fetching data, HttpClient, proxyConfig

- Using environment files
  - Add “apiUrl” to environment.ts
  - apiUrl: '/api'
  - Add “apiUrl” to environment.prod.ts
  - apiUrl: 'http://localhost:3000'
- Why/how it works?

- Use environment.apiUrl in getTransactions method
- Verify production build using http-server
  - ng build
  - npx http-server ./dist/ita22 -o

Task #8 - building transactions page

- Generate TransactionsPageComponent
  - ng g c transactions/components/transactions-page
  - Move contents of TransactionsComponent inside new component
  - Remove TransactionsComponent
- Render list of Transactions with routerLink to transaction details
  - /transactions/:id
  - Configure new route, reuse TransactionsPageComponent
  - Mark selected transaction using routerLinkActive
- Using ActivatedRoute
  - Check if Transactions is selected and show details/edit
  - If no Transaction is selected show form to create new transaction

Useful resources

Bible (aka MDN)

- MDN Web Docs
- https://developer.mozilla.org/

Git + code review practices

- https://learngitbranching.js.org/
- Google Engineering Practices Documentation
  - https://google.github.io/eng-practices/

CSS

- https://flexboxfroggy.com/
- https://cssgridgarden.com/

iLoveCoding

CSS Cheatsheet

Comprehensive video course on HTML & CSS
https://ilovecoding.org/courses/htmlcss

1 Syntax to write CSS

Selectors
The element(s) on which the style should be applied

Property and its value
This is the actual style to be applied to the element(s)

```
selectors {
....property: value;
}
```

2 3 places to write CSS

(A) Inline styles

```
<element style="property: value;">
```

(B) In the <style> element

```
<head>
....<style>
........ selectors { property: value; }
....</style>
</head>
```

(C) In a dedicated file (style.css)
& refer that file via the <link> element

```
<head>
....<link rel="stylesheet"
........ href="style.css" />
</head>
```

3 Selectors and their syntax

Basic Selectors	Combinators		Pseudo Selectors
elementname	selectorA + selectorB	Adjacent sibling	:active
.classname	selectorA ~ selectorB	General sibling	:hover
#idname	parent > child	Direct child	:visited
[attr=value]	parent descendent	Descendent	:focus
*			

4 Common CSS properties (by group)

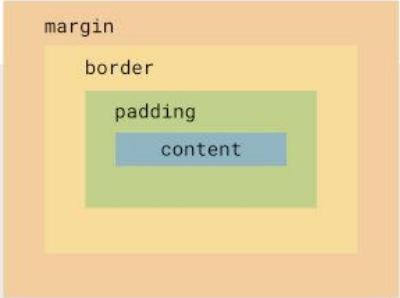
TEXT:
color
font
font-family
font-size
font-weight
letter-spacing
line-height
text-align
text-decoration
text-indent
text-transform
vertical-align

LIST:
list-style
list-style-image
list-style-position
list-style-type

BACKGROUND:
background
background-attachment
background-color
background-image
background-position
background-repeat

DISPLAY:
display
float
clear
overflow
visibility

OTHER:
cursor

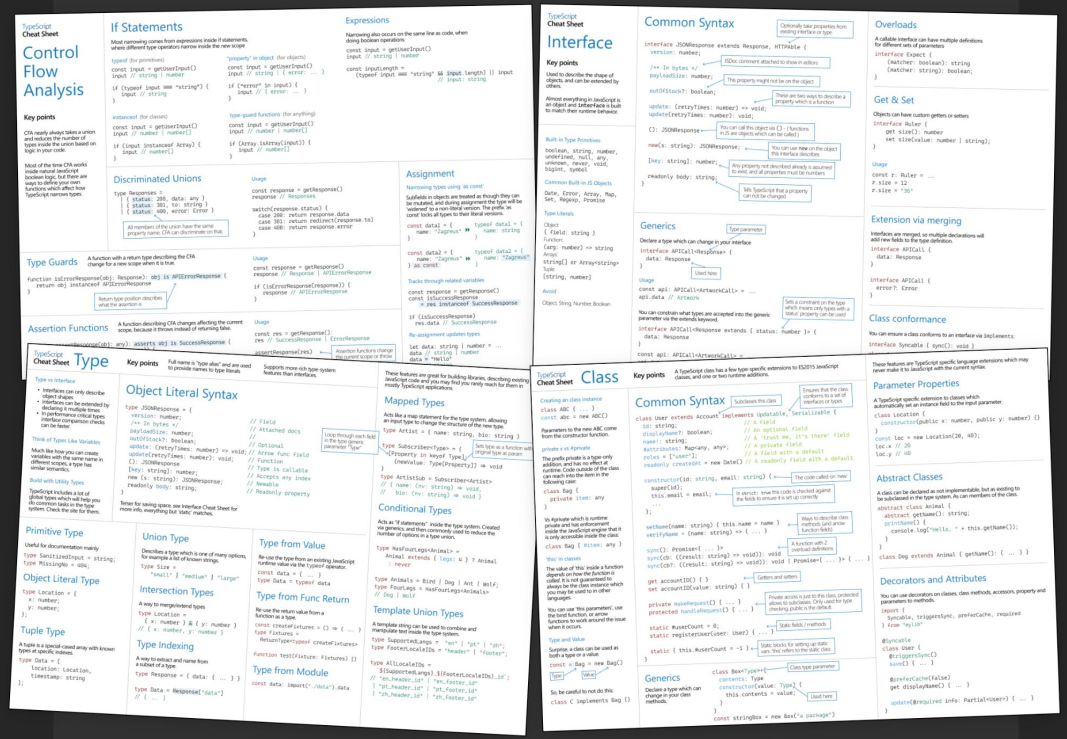


BOX:
border
border-color
border-style
border-width
height
margin
padding
width
box-sizing

POSITION:
position
top
bottom
left
right
z-index

TypeScript

- https://www.typescriptlang.org/cheatsheets



RegExr

- https://regexr.com/
- https://regexper.com/