

JS guild: Monorepos

Juho Härme 19.4.2024

```
git clone https://github.com/knowit-  
finland-javascript-guild/turborepo.git
```

```
|— non-mono  
|— mono
```

```
cd non-mono
```

```
cd component-library
```

```
npm install
```

```
cd ../front
```

```
npm install
```

```
cd ../back
```

```
npm install
```

```
# OR just
```

```
cd non-mono/
```

```
for pkg in */; do cd $pkg && npm install && cd ../done
```

```
cd non-mono/front
```

```
npm run dev
```

Intro

- Mono? Poly? Monolith? Microservices? Packages?
- Starting with a monorepo vs converting something into a monorepo
 - a) converting a monolith into a monorepo
 - b) converting a bunch of microservices + packages into a monorepo

My monorepo story

- over 10 individual repositories
- common ui elements from 2 different packages
- general shared code from 2 different packages
- types from a separate package...

The problem with packages in polyrepo

Hands-on 1

Make a change to a component imported from an internal package

Please make the button hot pink 😊

Best wishes, UX team



C'mon, such a small change, it'll be up in a minute?



```
cd non-mono/front  
npm run dev
```

.....

```
|— component-library  
|   |— ...  
|   |— index.ts  
|— front  
|   |— ...  
|   |— src  
|       |— counter.ts  
|       |— main.ts
```

```
cd non-mono/component-library/
```

```
# make changes to button at index.ts
```

```
npm version patch
```

```
npm pack
```

```
cd ../front
```

```
npm install ../component-library/component-library-1.0.1.tg
```

**How could a
monorepo make
this easier?**

Prerequisites for the next hands-on exercise

```
cd mono
npm install
```

Some important concepts

- workspace vs root

```
root
├─ package.json
├─ package-lock.json
├─ node_modules
├─ workspace 1
│   └─ package.json
├─ workspace 2
│   └─ package.json
```

- internal vs external
- published vs non-published
- versioning
- deploying an individual microservice?

Hands-on 2

using shared packages in a monorepo

Just kidding, make it light blue instead! 🤔

Warmest wishes, UX



We might want to change this again in the future...



```
cd mono/apps/front  
npm run dev
```

.....

```
|— apps  
|   |— front  
|       |— src  
|       |   |— main.ts  
|— packages  
|   |— component-library  
|       |— index.ts
```

```
cd mono/packages/component-library  
  
# make changes to button at index.ts
```

How does this work?

```
# ./mono
```

```
npm ls
```

```
mono@0.0.0 /Users/juhhar/guilds/monorepo/mono
├─ component-library@1.0.0 → ./packages/component-libra
├─ typescript@5.4.5
├─ front@0.0.0 → ./apps/front
├─ component-library@1.0.0 deduped → ./packages/compo
├─ typescript@5.4.5 deduped
└─ vite@5.2.8
```

```
├─ node_modules
├─ apps
├─   └─ front
├─ packages
├─   └─ component-library
```

```
#./node_modules
```

```
front@ → ../apps/front
```

```
component-library@ → ../packages/component-library
```

Running all the microservices in the monorepo

```
# Open a new shell
```

```
cd mono/apps/back  
npm run dev
```

```
# Open another new shell
```

```
cd mono/packages/utility-library  
npm run dev
```

What do we have?

1. UI running at `mono/apps/front`
2. API running at `mono/apps/back`
3. Utility library running at `mono/packages/utility-library`

Turbo? Nx? Lerna?

Hands-on 3

Managing tasks in a monorepo

Task: Use turbo to run all the dev tasks with one command

1. Create a file called `turbo.json` at the root level:

```
{  
  "pipeline": {  
    "dev": {}  
  }  
}
```

2. run `npx turbo run dev`

Task: Install a new package

```
# ./mono
```

```
npm install --workspace front cowsay
```

- alter the code at `mono/apps/front/src/main.ts` to use

```
import {say} from 'cowsay'
```

```
// ...
```

```
say({text: "text here"})
```

instead of just the simple text

? What happens if you install without `--workspace` ?

Recap

- Multiple microservices
- Multiple shared internal packages used by the services
- Npm managing the package installation process
- Turbo (etc) orchestrating the different tasks

Next steps

- How to deploy?
- Publishing the internal packages
- Sharing configs across the packages
- Pnpm? Yarn? Bun?

Resources

- <https://github.com/vercel/turbo/tree/main/examples>
 - <https://github.com/vercel/turbo/tree/main/examples/with-docker>
- <https://docs.npmjs.com/cli/v10/using-npm/workspaces>