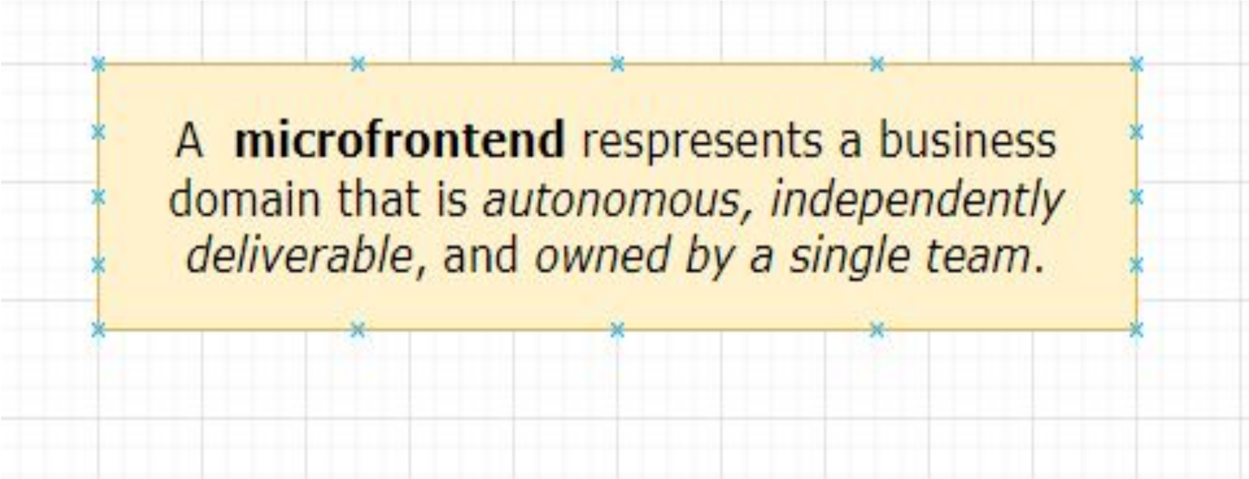
Microfrontends - An Introduction

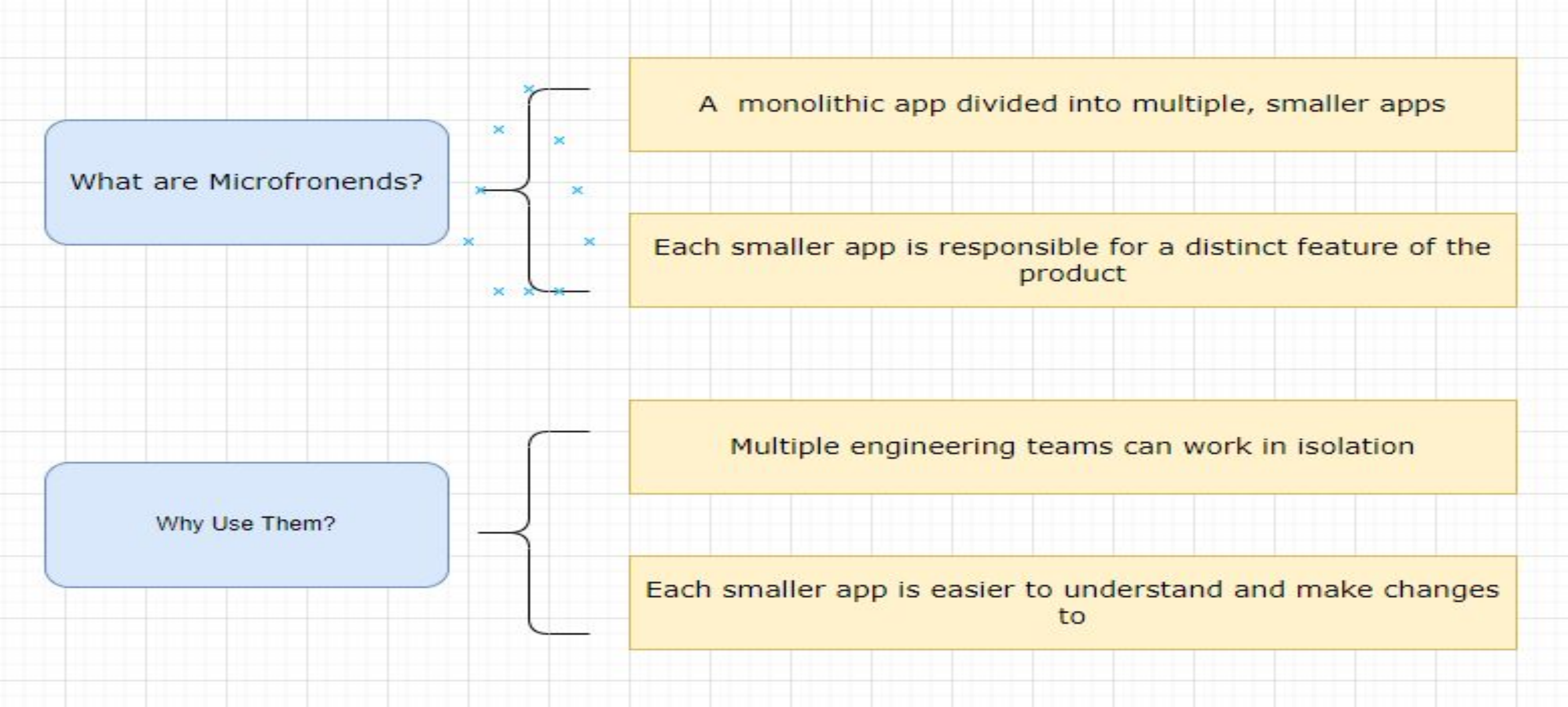
Alfio Martini

What is a Microfrontend?



A **microfrontend** respresents a business domain that is *autonomous, independently deliverable*, and *owned by a single team*.

Why Microfrontends



What are Microfronends?

A monolithic app divided into multiple, smaller apps

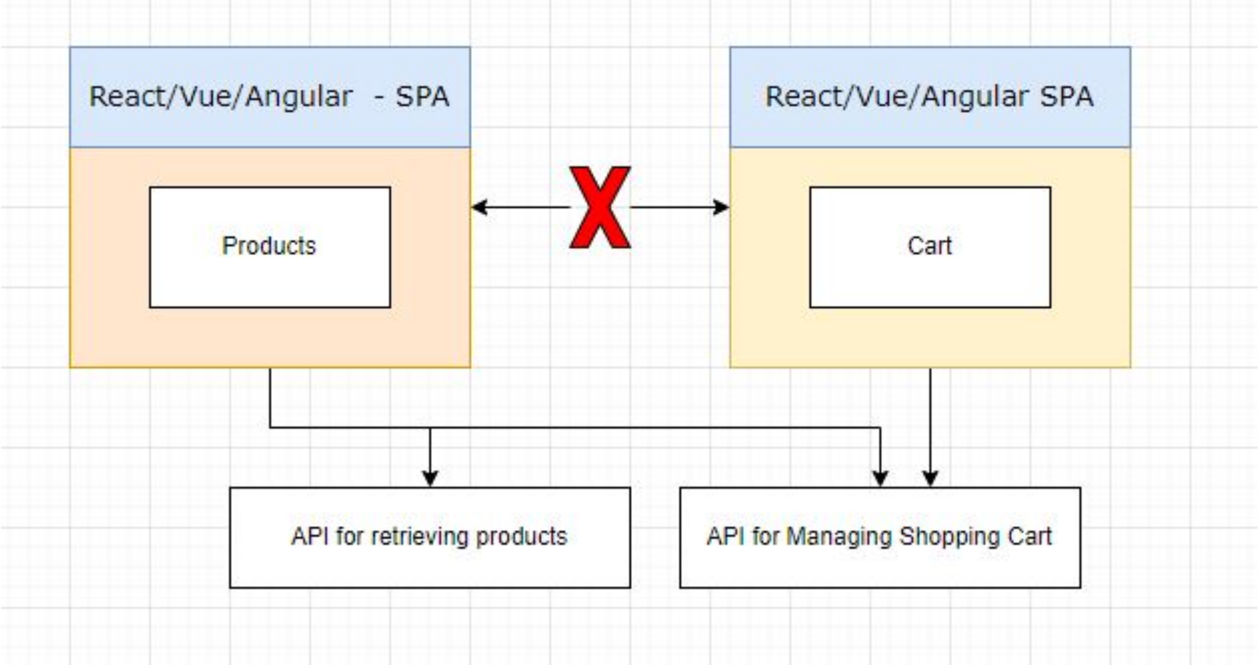
Each smaller app is responsible for a distinct feature of the product

Why Use Them?

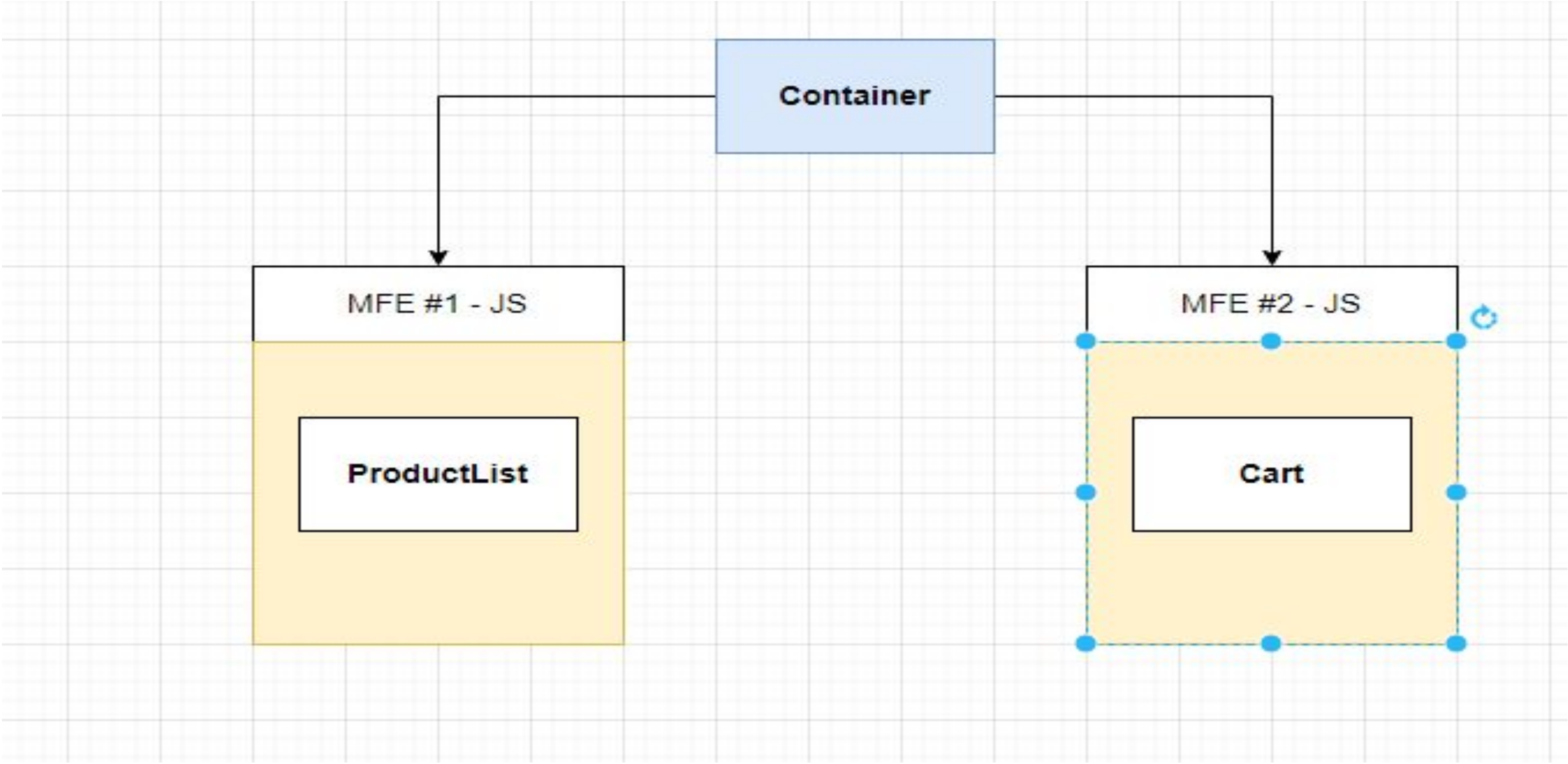
Multiple engineering teams can work in isolation

Each smaller app is easier to understand and make changes to

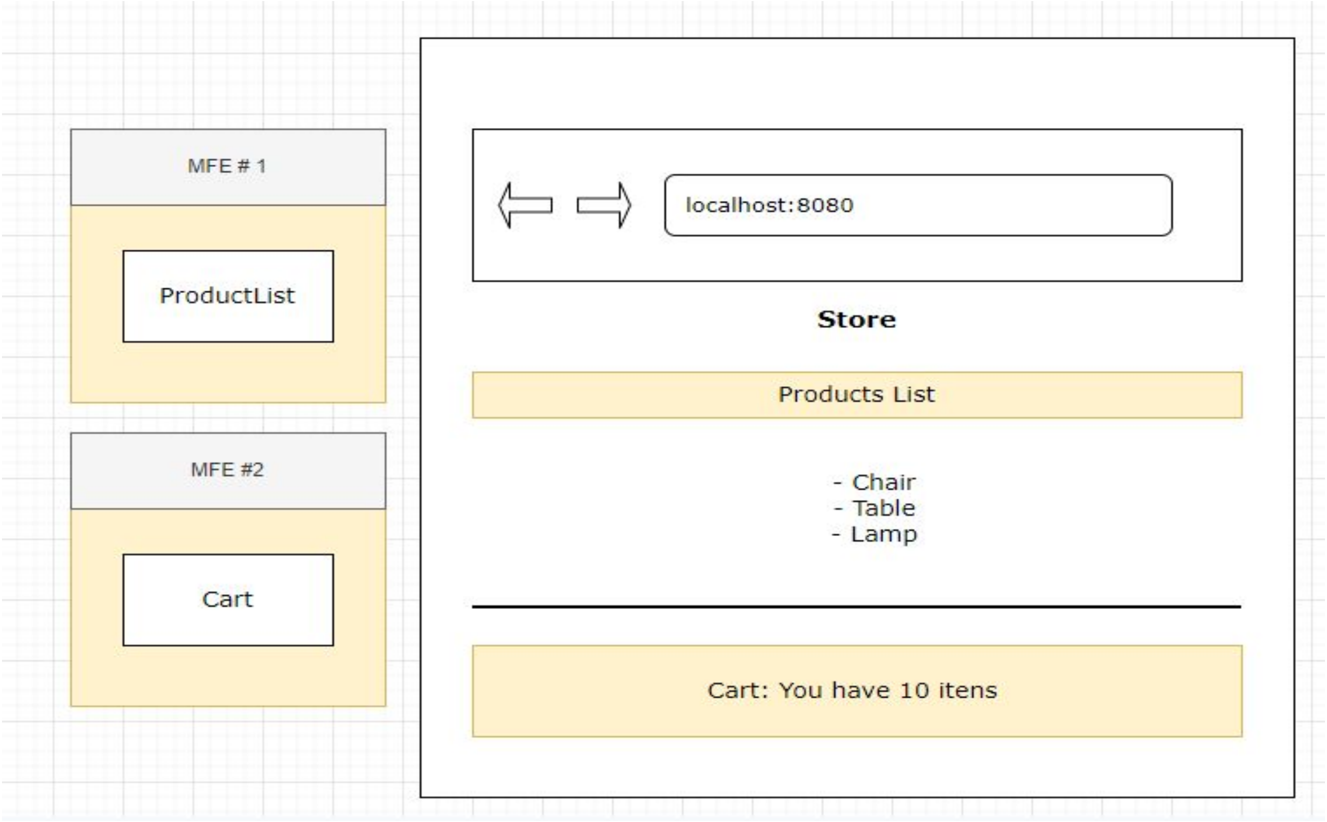
Independent (Single Page) Applications



First Project - Products + Cart (JavaScript Only)



Rendering Products & Cart in the Container



MFE # 1

ProductList

MFE #2

Cart

localhost:8080

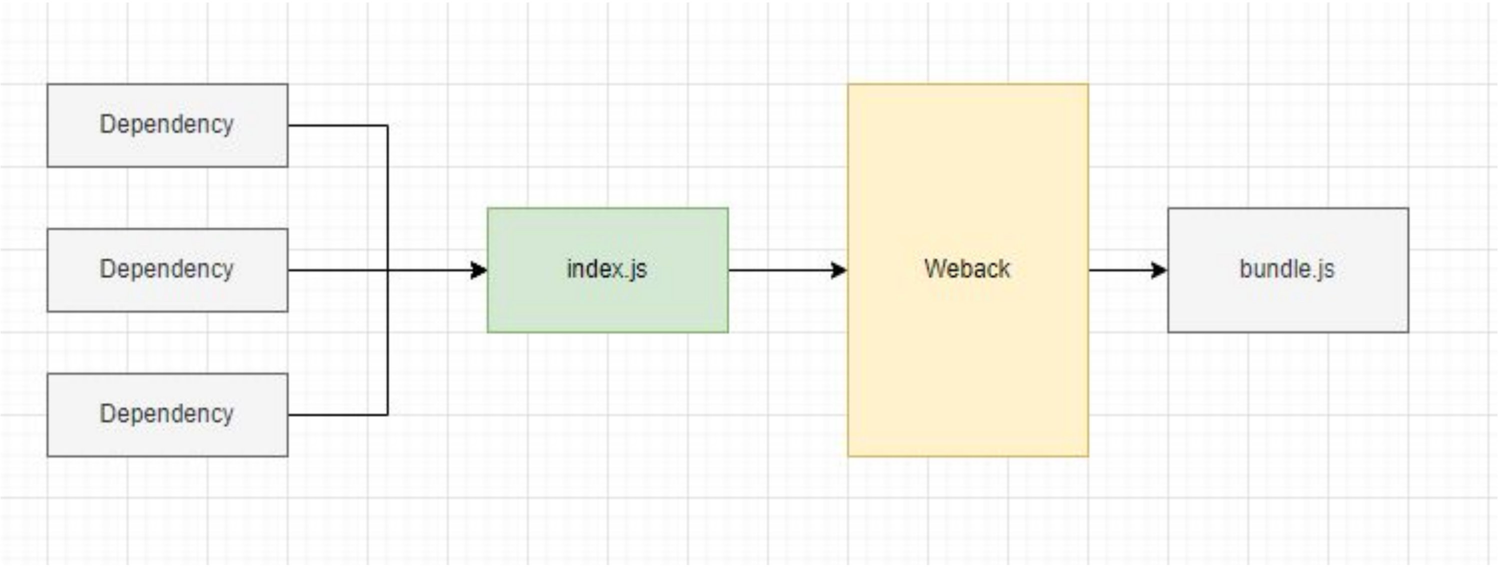
Store

Products List

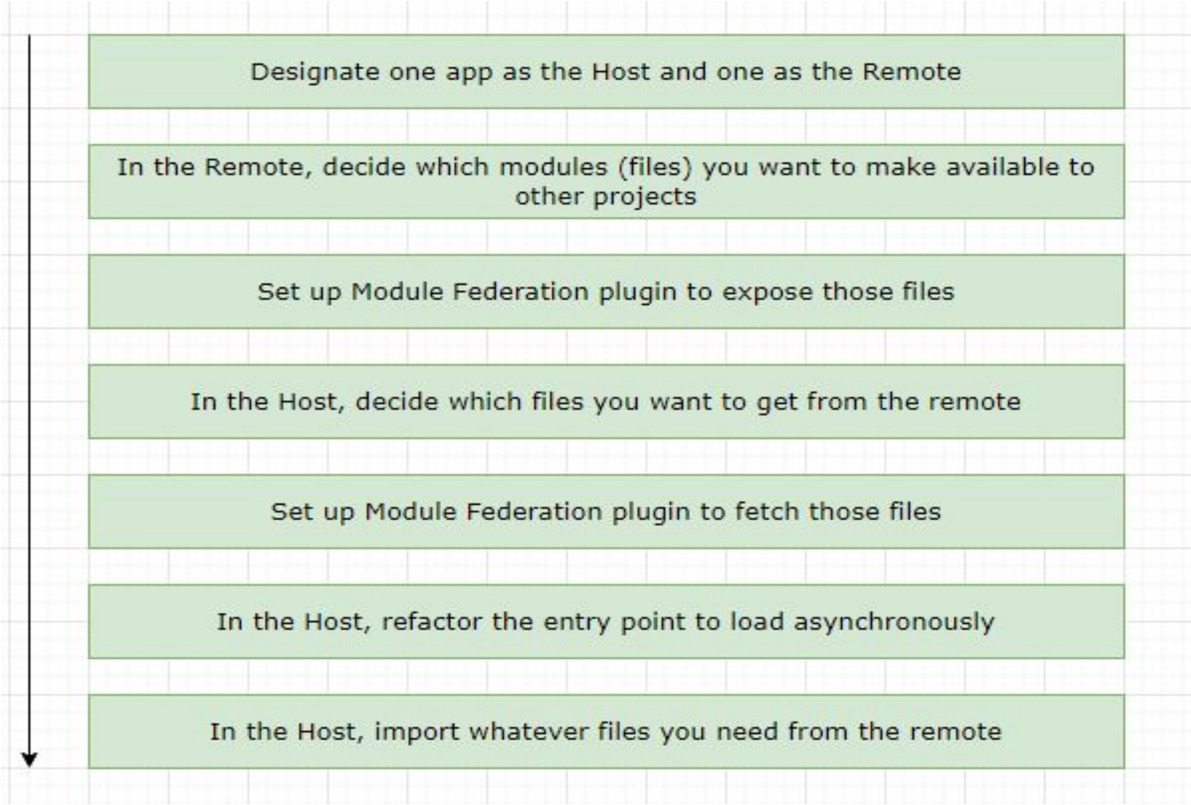
- Chair
- Table
- Lamp

Cart: You have 10 itens

Webpack Basics



Webpack - Module Federation Basics



Designate one app as the Host and one as the Remote

In the Remote, decide which modules (files) you want to make available to other projects

Set up Module Federation plugin to expose those files

In the Host, decide which files you want to get from the remote

Set up Module Federation plugin to fetch those files

In the Host, refactor the entry point to load asynchronously

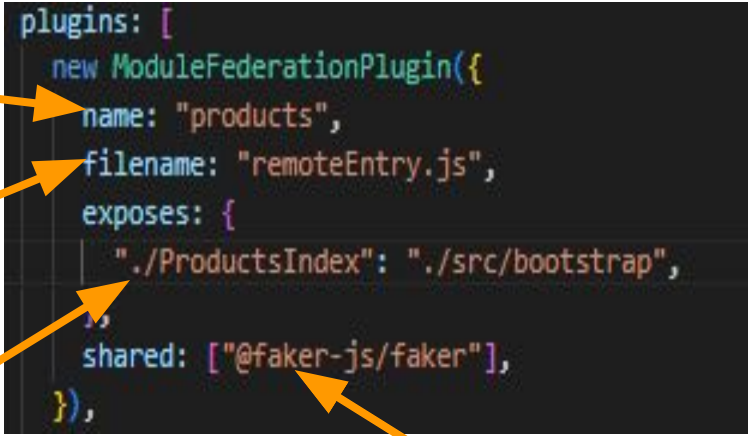
In the Host, import whatever files you need from the remote

Module Federation Plugin - Products

It works as an ‘ID’ for the container to access Product’s exposed resources

Sets the name of the manifest file. Usually ‘remoteEntry.js’

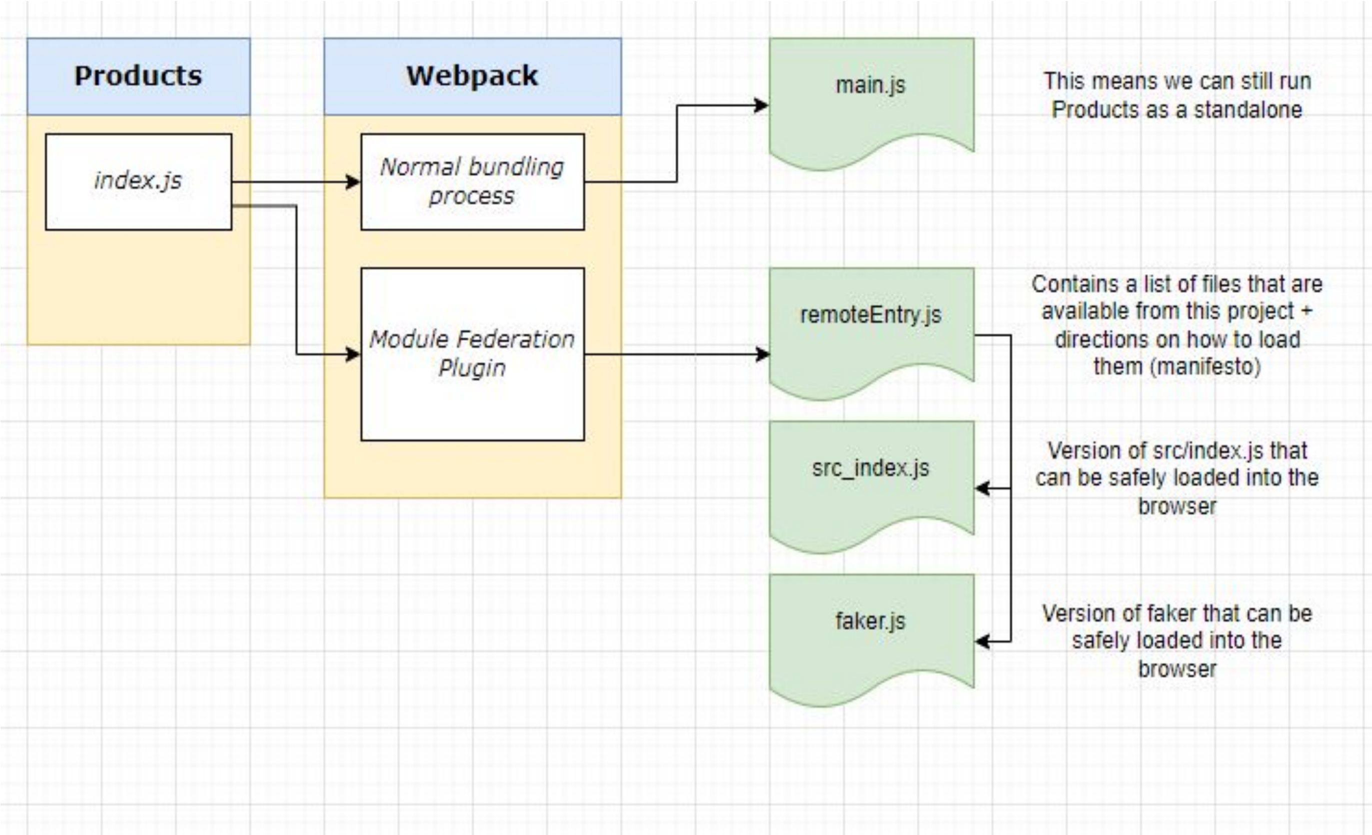
Aliases for filenames



```
plugins: [
  new ModuleFederationPlugin({
    name: "products",
    filename: "remoteEntry.js",
    exposes: {
      "./ProductsIndex": "./src/bootstrap",
    },
    shared: ["@faker-js/faker"],
  }),
```

Avoids duplication of resources

Webpack - Products



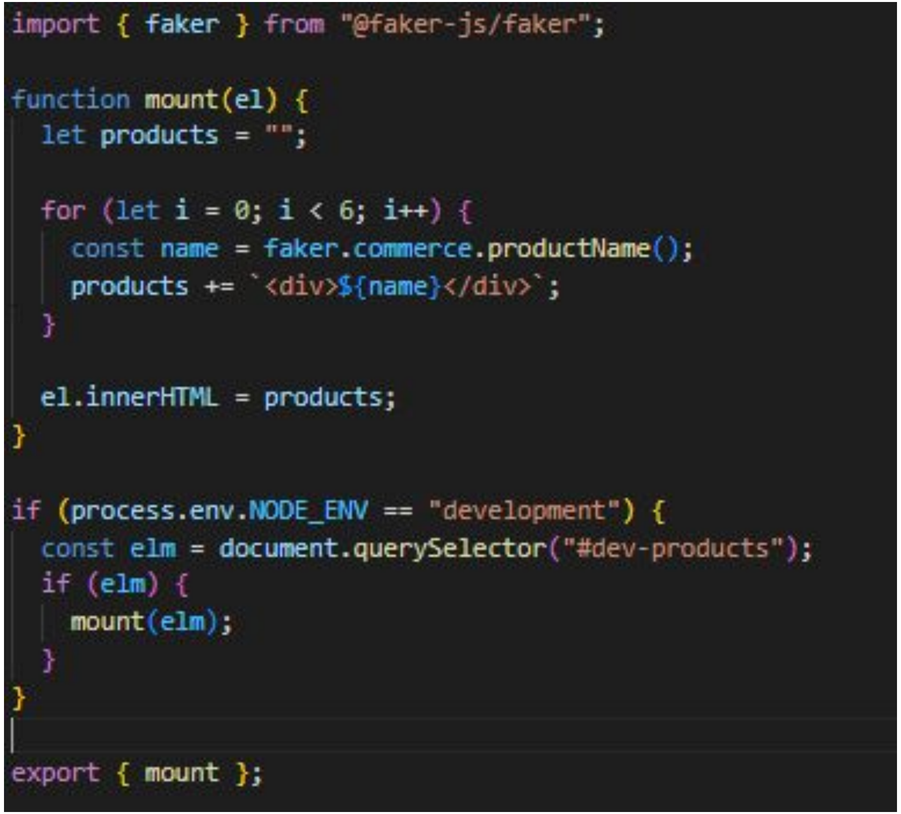
This means we can still run Products as a standalone

Contains a list of files that are available from this project + directions on how to load them (manifesto)

Version of src/index.js that can be safely loaded into the browser

Version of faker that can be safely loaded into the browser

Products - Bootstrap



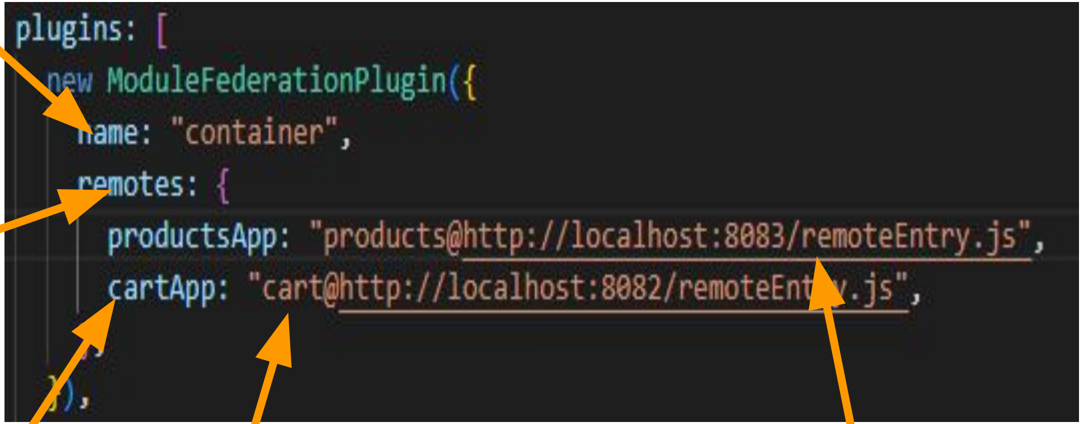
```
import { faker } from "@faker-js/faker";

function mount(el) {
  let products = "";

  for (let i = 0; i < 6; i++) {
    const name = faker.commerce.productName();
    products += `<div>${name}</div>`;
  }

  el.innerHTML = products;
}

if (process.env.NODE_ENV == "development") {
  const elm = document.querySelector("#dev-products");
  if (elm) {
    mount(elm);
  }
}

export { mount };
```

Module Federation Plugin - Container

Not used. Added for clarity. Needed for remotes

List projects that Container can search to get additional code

Load the file at the listed URL if anything in Container has an import like: import abc from ‘cartApp’

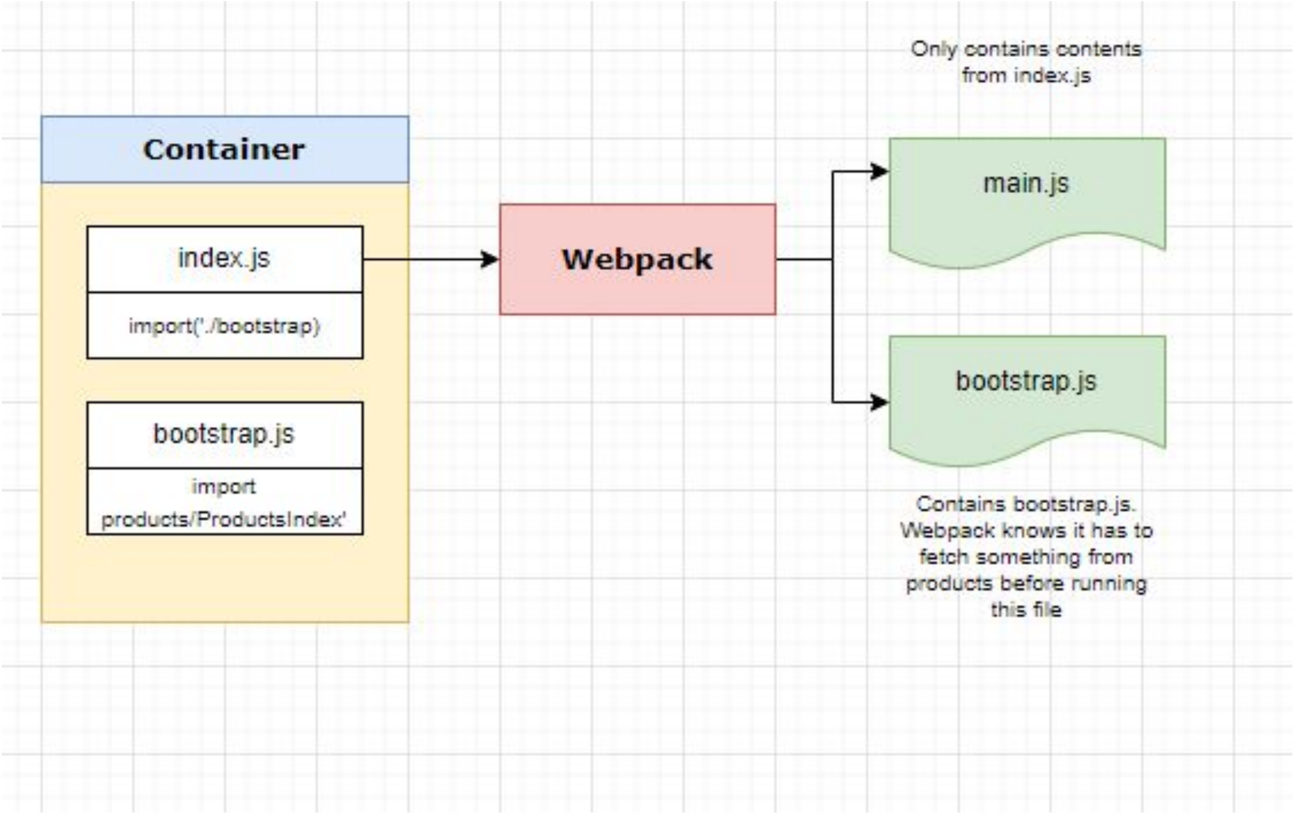


```
plugins: [
  new ModuleFederationPlugin({
    name: "container",
    remotes: {
      productsApp: "products@http://localhost:8083/remoteEntry.js",
      cartApp: "cart@http://localhost:8082/remoteEntry.js",
    },
  }),
```

Related to the ‘name property in the Cart webpack config file

URL for the ‘remoteEntry.js’ file

Webpack - Container



Container

index.js

import('./bootstrap)

bootstrap.js

import products/ProductsIndex'

Webpack

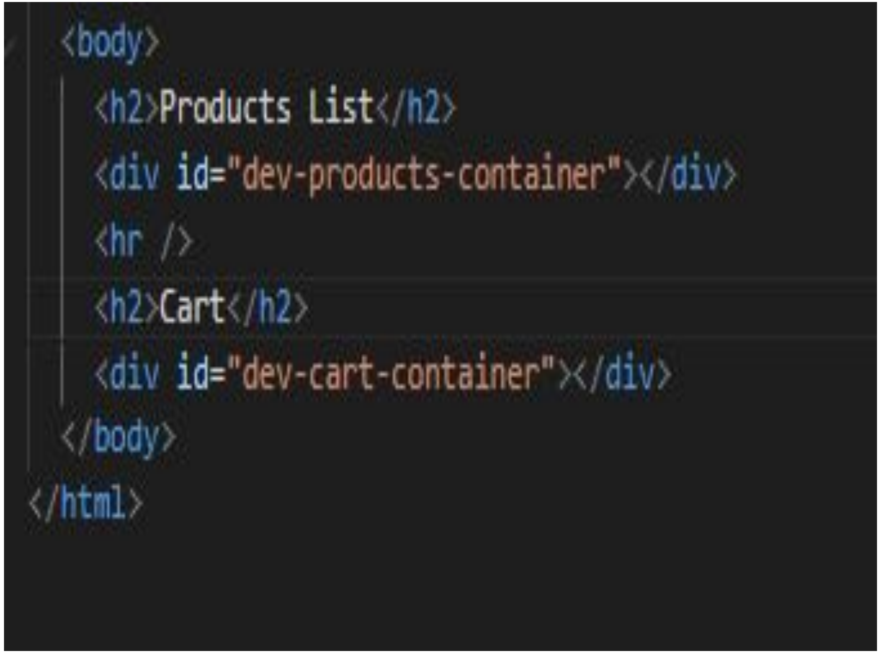
Only contains contents from index.js

main.js

bootstrap.js

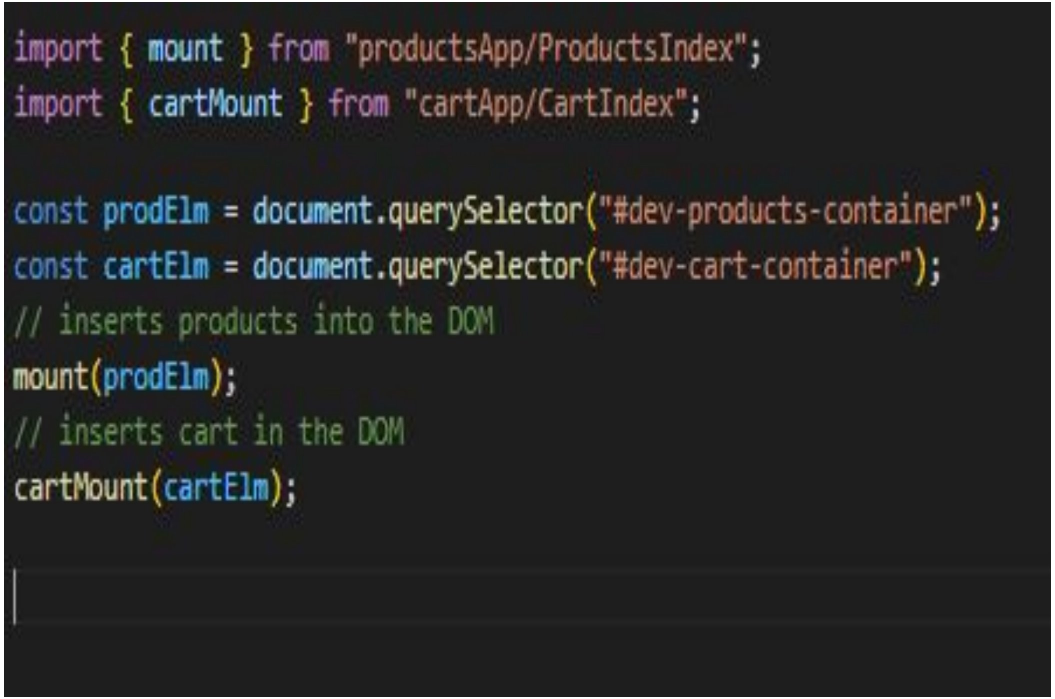
Contains bootstrap.js. Webpack knows it has to fetch something from products before running this file

Container - Renderização (index.html)



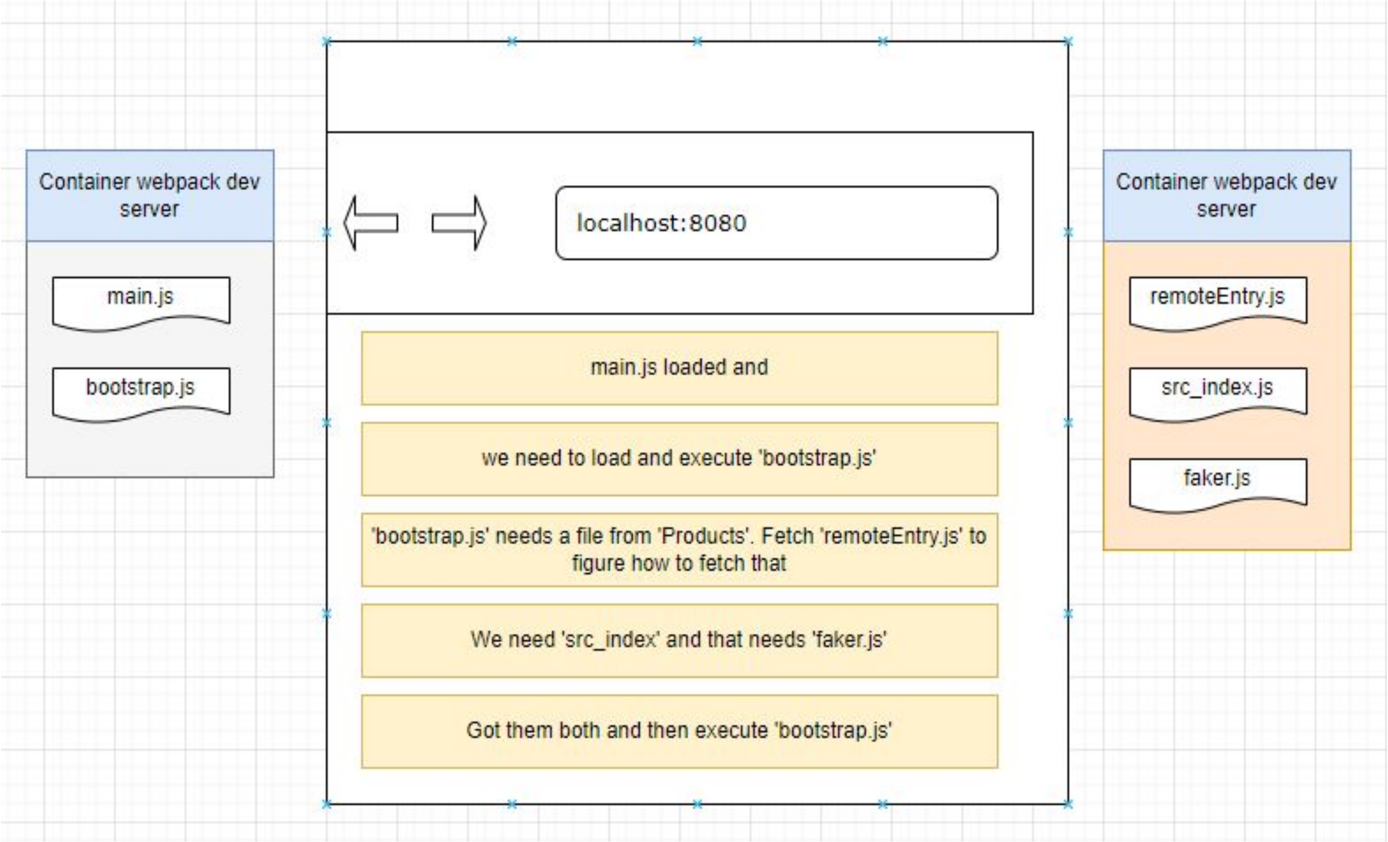
```
  <body>
    <h2>Products List</h2>
    <div id="dev-products-container"></div>
    <hr />
    <h2>Cart</h2>
    <div id="dev-cart-container"></div>
  </body>
</html>
```

Container - Bootstrap



```
import { mount } from "productsApp/ProductsIndex";
import { cartMount } from "cartApp/CartIndex";

const prodElm = document.querySelector("#dev-products-container");
const cartElm = document.querySelector("#dev-cart-container");
// inserts products into the DOM
mount(prodElm);
// inserts cart in the DOM
cartMount(cartElm);
```

Container - Webpack development server



main.js loaded and

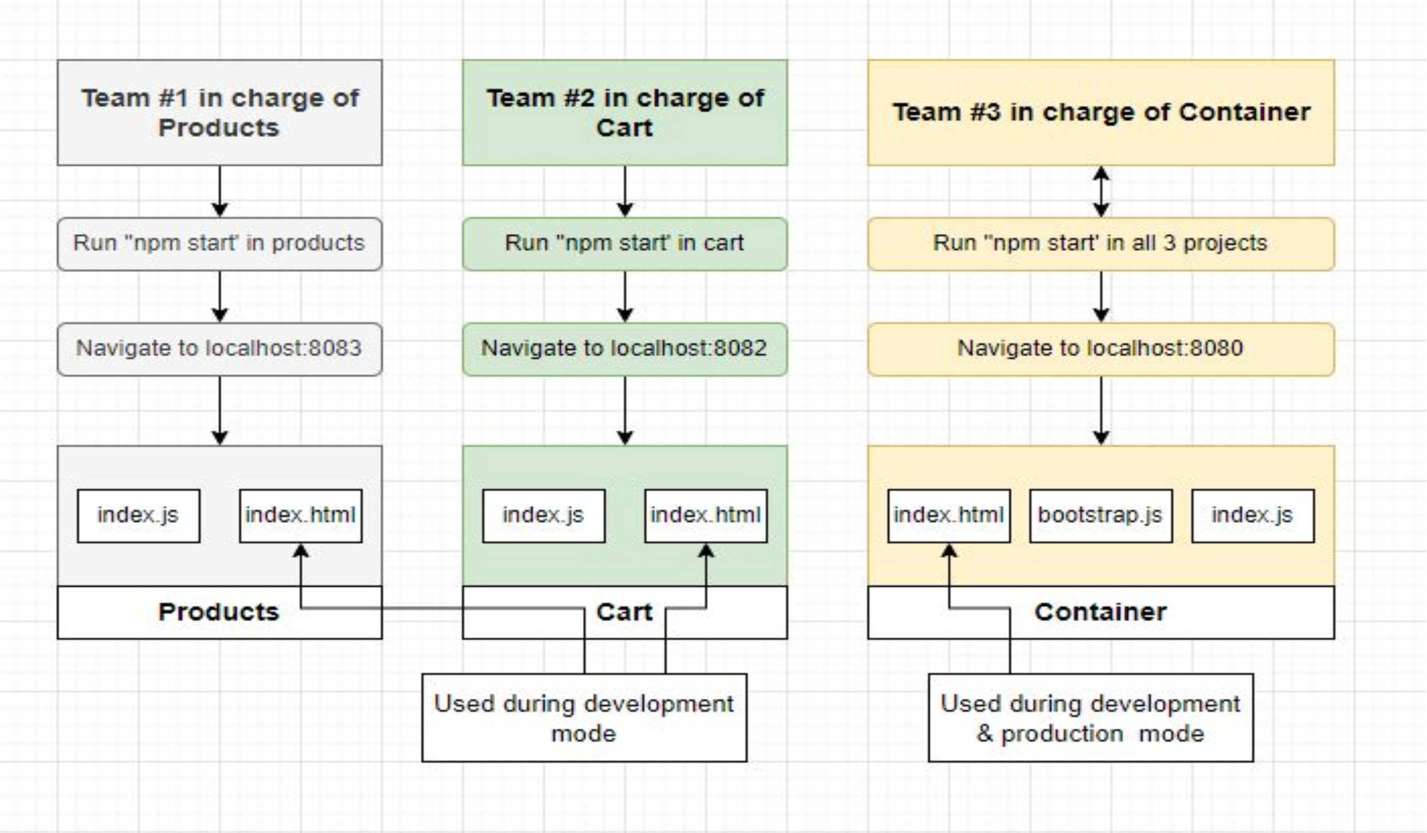
we need to load and execute 'bootstrap.js'

'bootstrap.js' needs a file from 'Products'. Fetch 'remoteEntry.js' to figure how to fetch that

We need 'src_index' and that needs 'faker.js'

Got them both and then execute 'bootstrap.js'

Summary - Development Process



Team #1 in charge of Products

Run "npm start' in products

Navigate to localhost:8083

index.js

index.html

Products

Team #2 in charge of Cart

Run "npm start' in cart

Navigate to localhost:8082

index.js

index.html

Cart

Team #3 in charge of Container

Run "npm start' in all 3 projects

Navigate to localhost:8080

index.html

bootstrap.js

index.js

Container

Used during development mode

Used during development & production mode

Technical Issues - To be continued…

- *CSS conflicts between container and child apps*
- *Synchronization between host routing and local routing*
- *…*