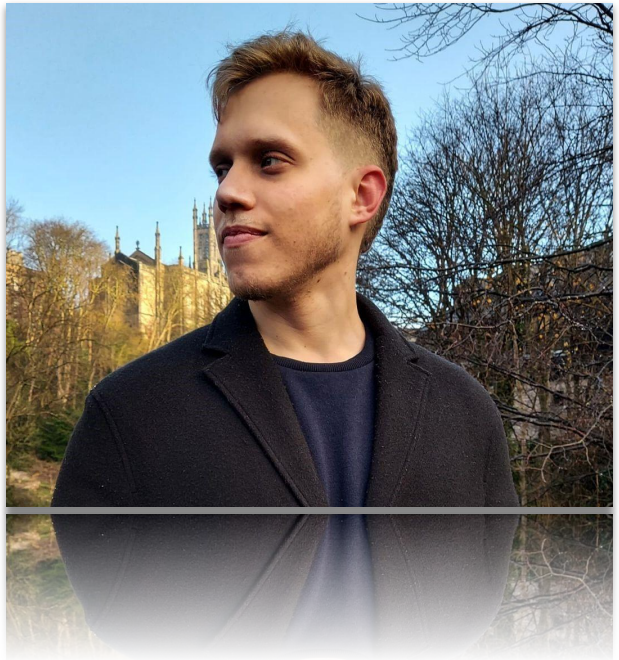

maestro

Tech Talk 1

Micro-Frontends

How to break the last Monolith



Speaker

Deliton Junior

Joined: December, 2020.

What we will cover today

- Monolith vs Micro
- Advantages and disadvantages of MFE
- Technical Implementations
- Module Federation
- Tech Demo

Topic 1

The Monolith

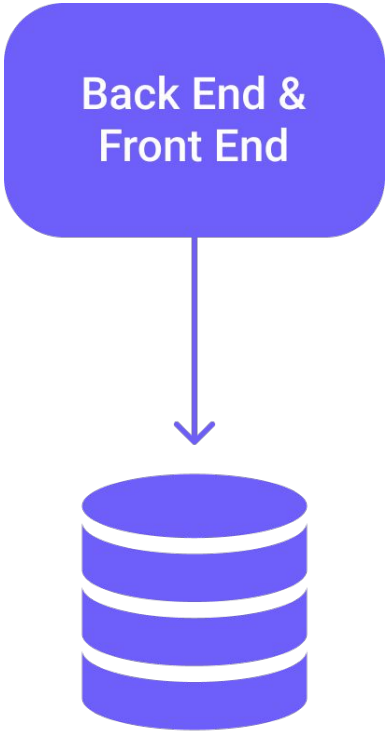
What is a monolith?

Not that Monolith…

- Single Logical Executable
- Everything needed to run a project is contained in a single codebase.

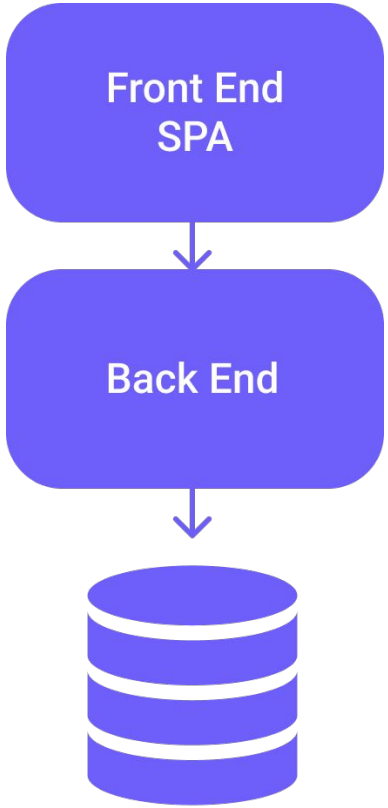
The Big Monolith

- All business logic within a single codebase
- Backend and Frontend Together
- Single Database



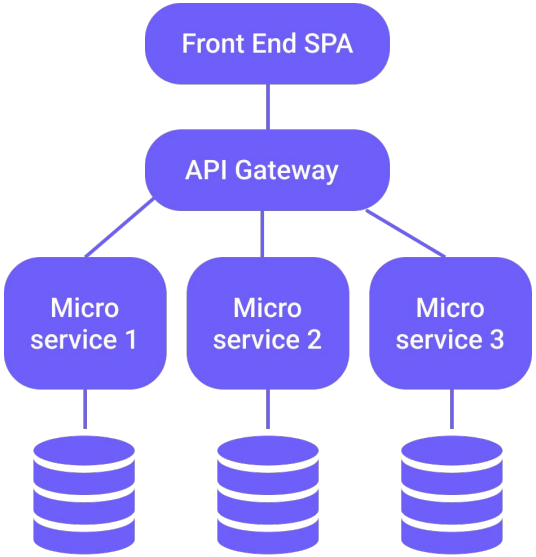
A wild pokemon appears: SPAs

- Big monolith was broken into 2 smaller monoliths
- Introduction of SPA frameworks like React, Vue and Angular.



Backend Monolith is broken

- Backend was broken into smaller units, also known as microservices.
- Single services can be replicated for better performance and availability.

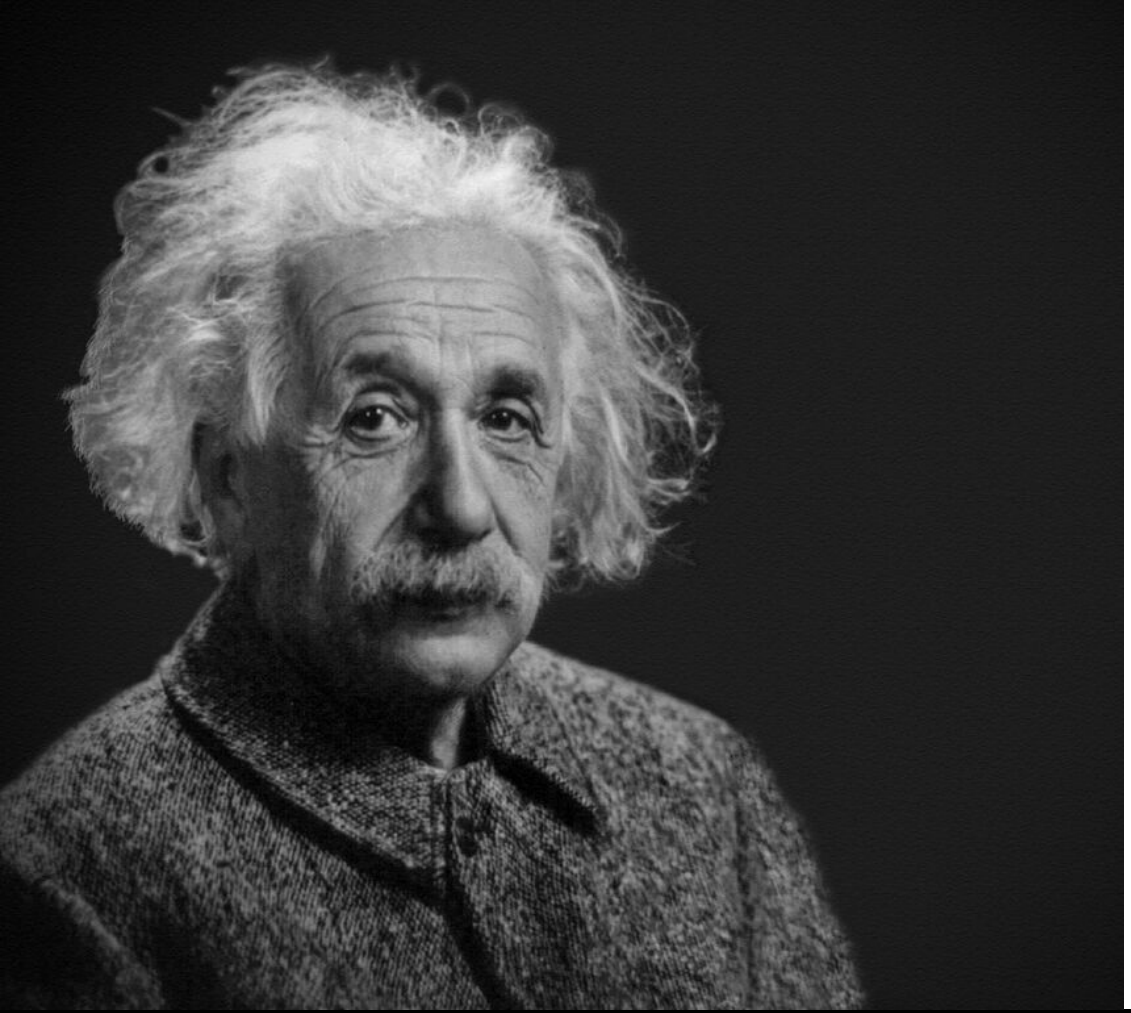


But there’s one last Monolith

Breaking the last Monolith

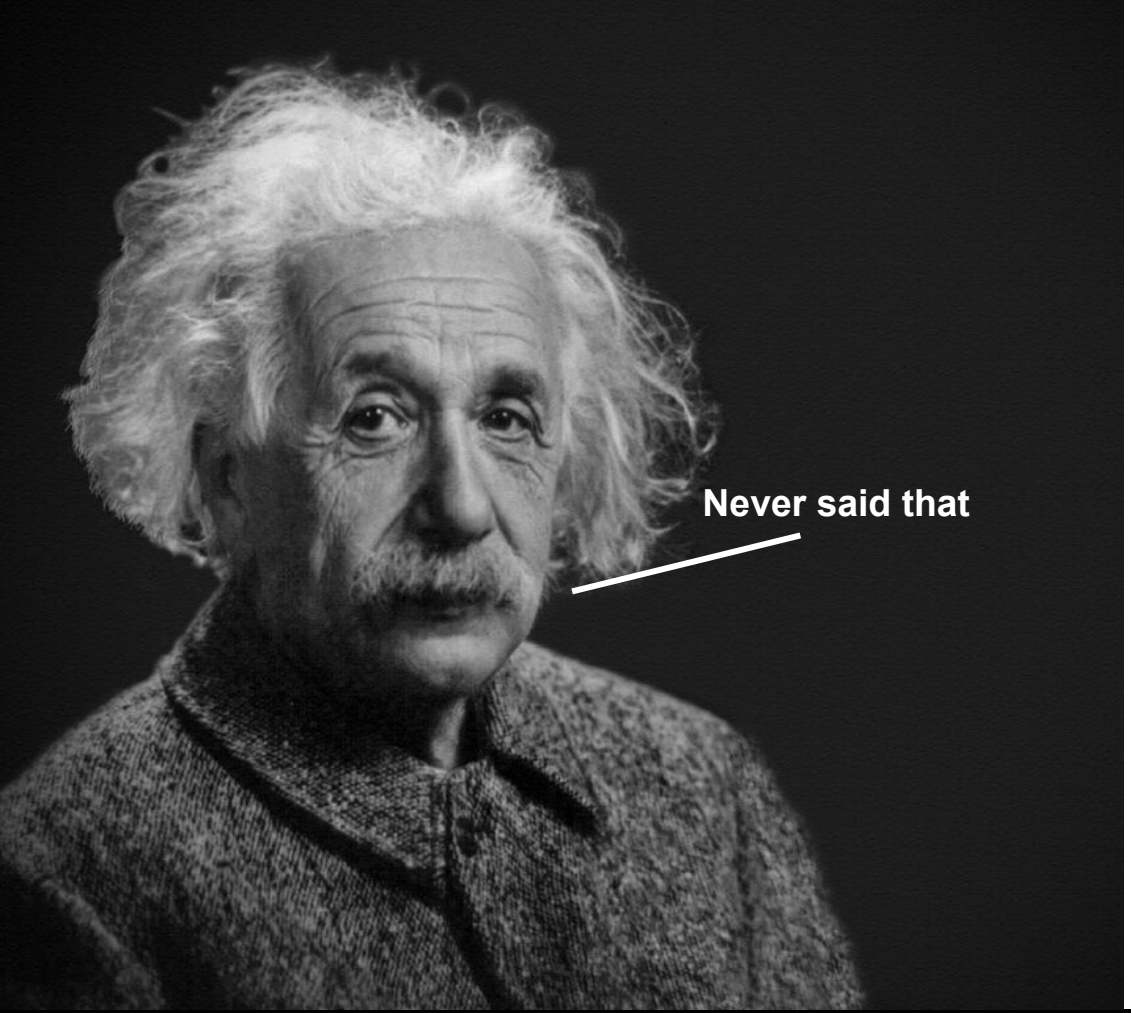
- The more the frontend grows, it gets harder and harder to add new features.
- What to do if your framework of choice becomes deprecated?

What’s the most simple way of solving those problems?



“Nothing is particularly hard if you divide it into small parts”

-Albert Einstein



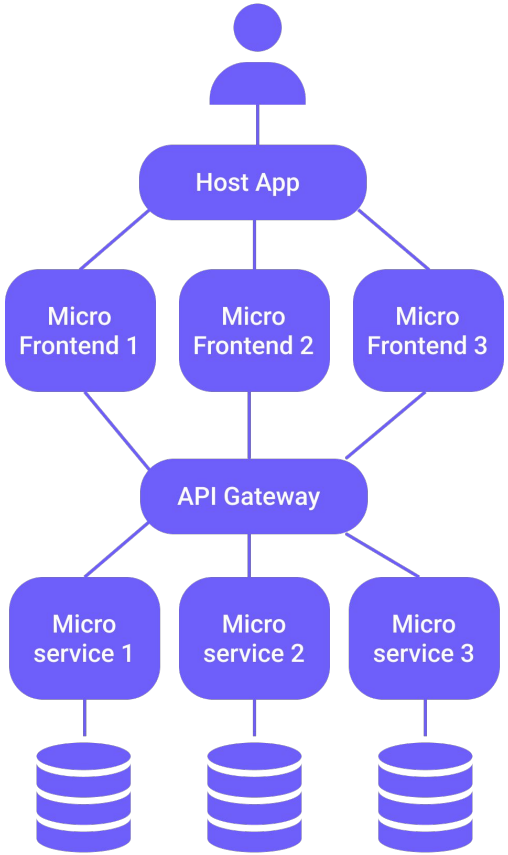
- Breaking problems into smaller parts has been applied to everything for centuries.
- Why not simplify the frontend into smaller parts?
- Why not break the last monolith?
- That’s why micro frontends were created

What is a Micro Frontend

- An approach to develop web apps as a composition of small frontend apps
- MFEs are self-contained and can be developed and deployed independently.

How the Architecture looks like?

- Small units of frontends, tied together by a Host App, Server or Reverse Proxy.



Topic 2

Advantages of MFEs

Why bother?

Autonomy

- Autonomous
- Loosely coupled
- Opens space to innovation

All about the teams

- Teams can have ownership of the codebase.
- Each time can specialize in one domain of the business

Faster time to market

- Individual features can be shipped faster.
- While moving faster, teams can iterate faster.

Feature Flags

- We can easily implement performant feature flags.
- Same with A/B Testing

Single Responsibility

- Easier to track bugs, you know where to attack.
- Specialized frontends.

Technology Agnostic

- It’s possible to stitch together all frameworks in the same app.
- It’s possible to use the best tool for each new feature.

Faster and Independent Deployments

- We can ship one frontend, and the whole app will update.
- Build times are faster.

Topic 3

Disadvantages of MFEs

Everybody has flaws

Organizing Code Standards

- If each team choose its own standard, it might become a mess.
- Teams have to adopt clear naming conventions
- Won’t save you from bad code.

With small projects, it’s not worth the struggle

- MFEs might be a good idea for big projects and businesses, but for small ones it’s just not worth it.
- Good old monolith works, just be careful.

Additional DevOps effort

- Since we have independent deployments, it’s harder to get it all setup.
- Things get more scary with multiple environments.

Topic 4

How to break an App

The Micro way

Divide it by pages

- Each page is a separate app.
- Share common components
- It’s daunting in most cases.

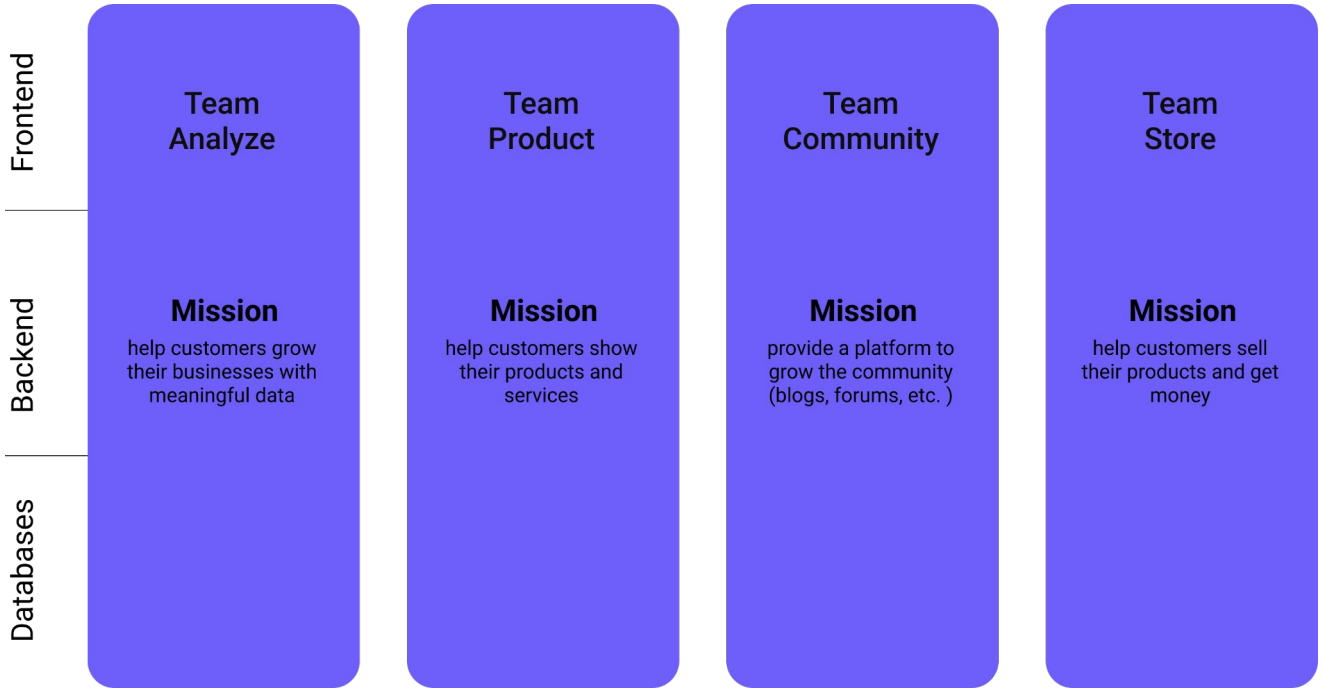
Divide it by sections

- We have separate apps for different sections of a website.
- Share common components
- Grouped pages

Divide it by experiences

- Understand the different experiences a frontend app provides.
- Product oriented on experiences.
- Organize self-sufficient teams for self-sufficient front ends.

All about the teams



Topic 5

Best Practices

Single Responsibility

- Avoid coupling of code
- Each micro frontend should be a specialist.
- Avoid sharing code and state.

CI/CD All the way

- Build and Deployment automation is a must.
- Test automation

Don’t overuse micro frontends

- Try not to over fragment your app and create things that don’t have real value.
- It doesn’t make sense to fragment everything.
- Think ahead, figure out what parts might grow over time.

Find the right size

- Needs to be just right.
- If MFEs are too big, they’re too coupled.
- If MFEs are too small, over-fragmented.
- There’s no Golden Rule, do what you and your team think feels best for the long term.

Topic 6

Technical Implementations

The IFrame Approach

- IFrames are HTML embedded into other HTML pages.
- Good isolation.
- Easy to maintain.
- One of the easiest ways to implement micro frontends

The IFrame Approach

```
<!DOCTYPE html>
<html>
<body>

<iframe src="http://localhost:4200" width="400" height="500">
  <p>Your browser does not support iframes.</p>
</iframe>

<iframe src="http://localhost:4201" width="400" height="500">
  <p>Your browser does not support iframes.</p>
</iframe>

</body>
</html>
```

iframe.html hosted with ❤ by **GitHub** **view raw**

The IFrame Approach

- Widely used by Spotify.
- Apps communicate with each other via an Event Bus, in the DOM.

The IFrame Approach Disadvantages

- Hard to make responsive.
- Difficult integration between different apps.
- Routing, History and Deep Link, forget it.

The Web Component Approach

- Very close to IFrame Approach
- We wrap different apps (even from different frameworks) as Web Components and mount them together.

Runtime Composition Approach (Webpack)

- Loads different apps on demand at runtime.
- Also makes use of a container app (host) that tie other MFEs together.
- It’s possible to share state between apps, with things like Redux or Context API.

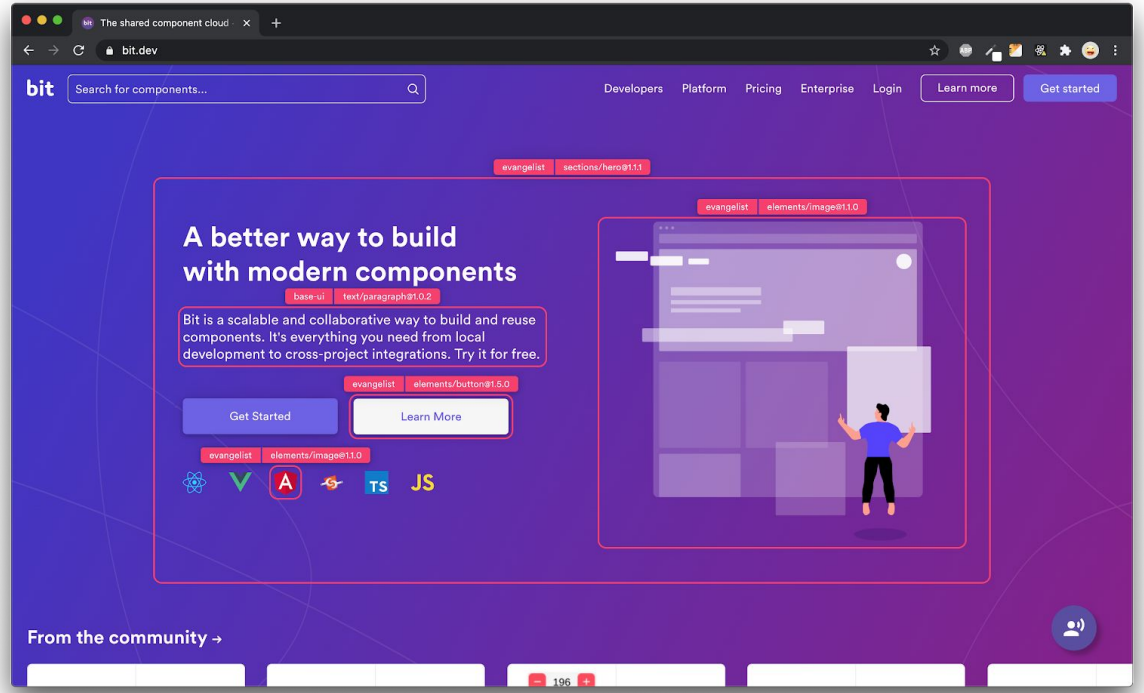
Build Time Integration

- Downloads the other MFEs at build time, assemble them together and deploy.
- Gets deployed just like monoliths.

Build Time Integration

- With traditional implementations, if a developer makes a change, the entire root package needs to be rebuilt.
- There are modern solutions that can address this and make this approach very intuitive.

Build Time Integration (Bit)



Server side Approach

- Mount the Front end in the server.
- Use CDNs and cached pages.
- Very flexible and excellent SEO performance.

Server side Approach

- Used by AirBNB
- Develop their own framework to do this: Hypernova.

Server side Approach

- Is by FAR the most expensive!

Topic 6

Communication

How MFEs talk to each other

Communication

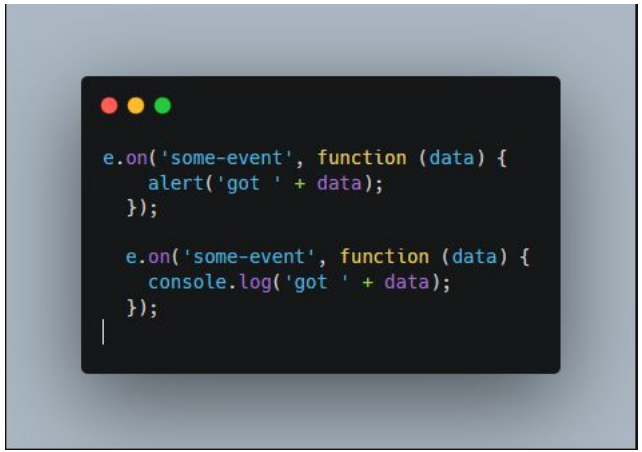
- Communication between different MFEs is not encouraged, but it’s inevitable in some cases.
- How does one MFE knows when to show up?

It depends

- Communication can be done by having a root app that handles state and events.
- Or that can be handled by the server.

Event Bus

- For Execution time approaches, we can use Event Buses that publish/subscribe to events.



Topic 7

Module Federation

With Webpack 5

Module Federation

- Allows Javascript apps to download code from another app.
- Also shares dependencies, if those are missing.
- Probably a game changer.

Topic 8

Tech Demo

Let’s see this working

Thank You