From monolith to micro

how to break apart a frontend application

Hi, I'm Thayse Onofrio

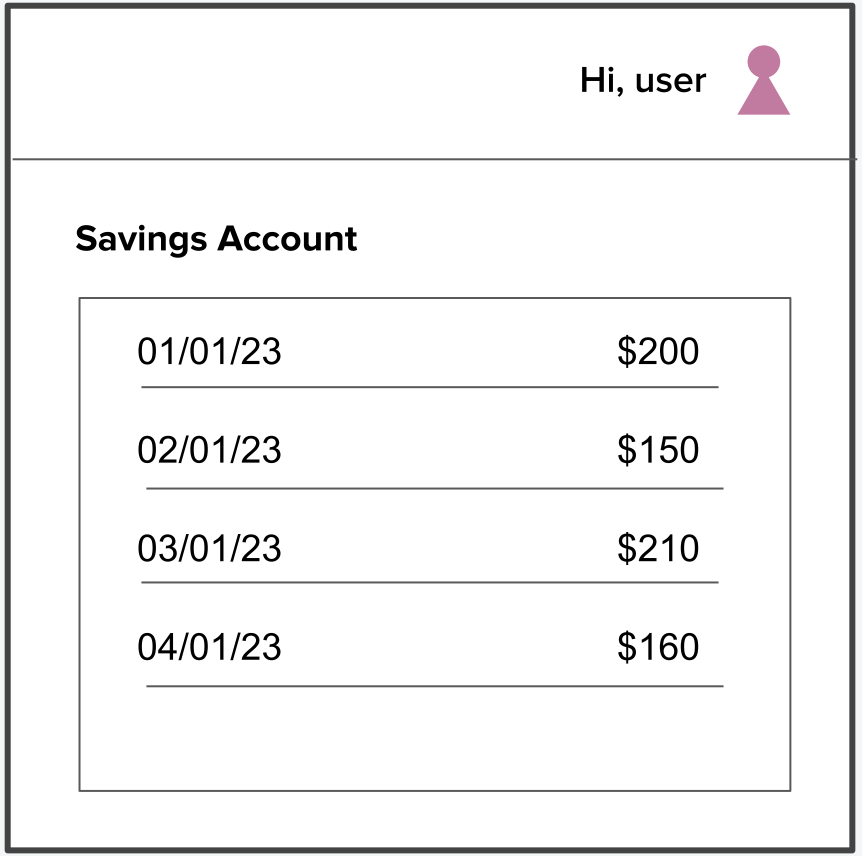
she/her

Lead Software Engineer @ Thoughtworks

thayseonofrio.com

thayseonofrio

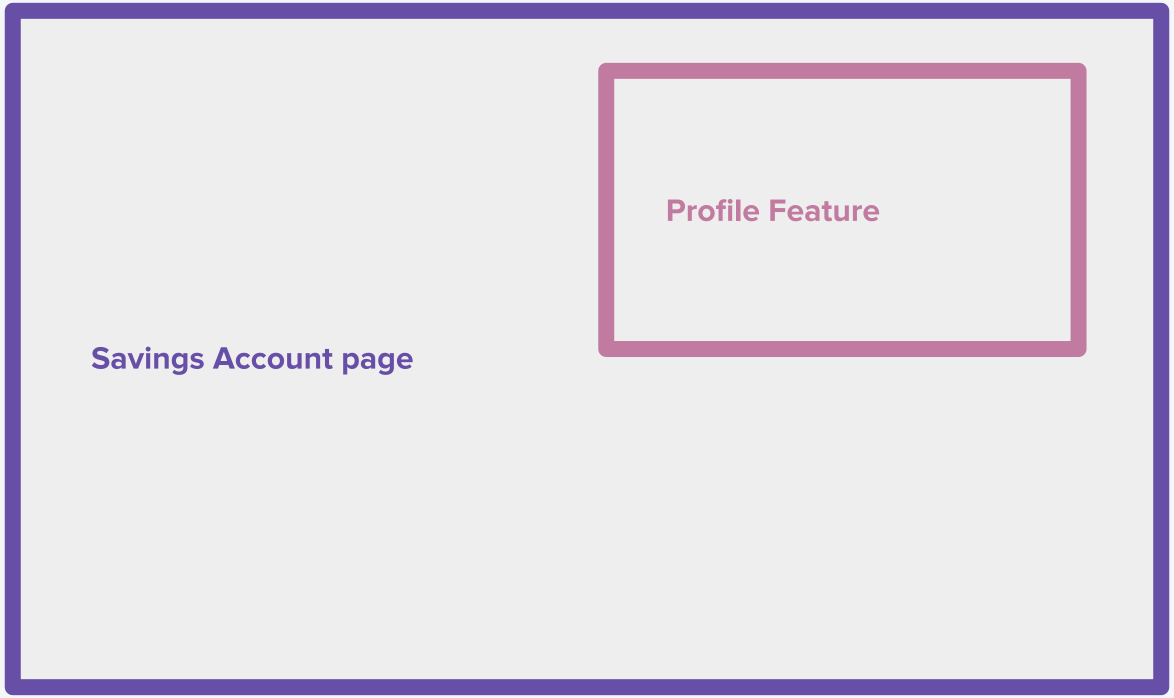
The monolith frontend

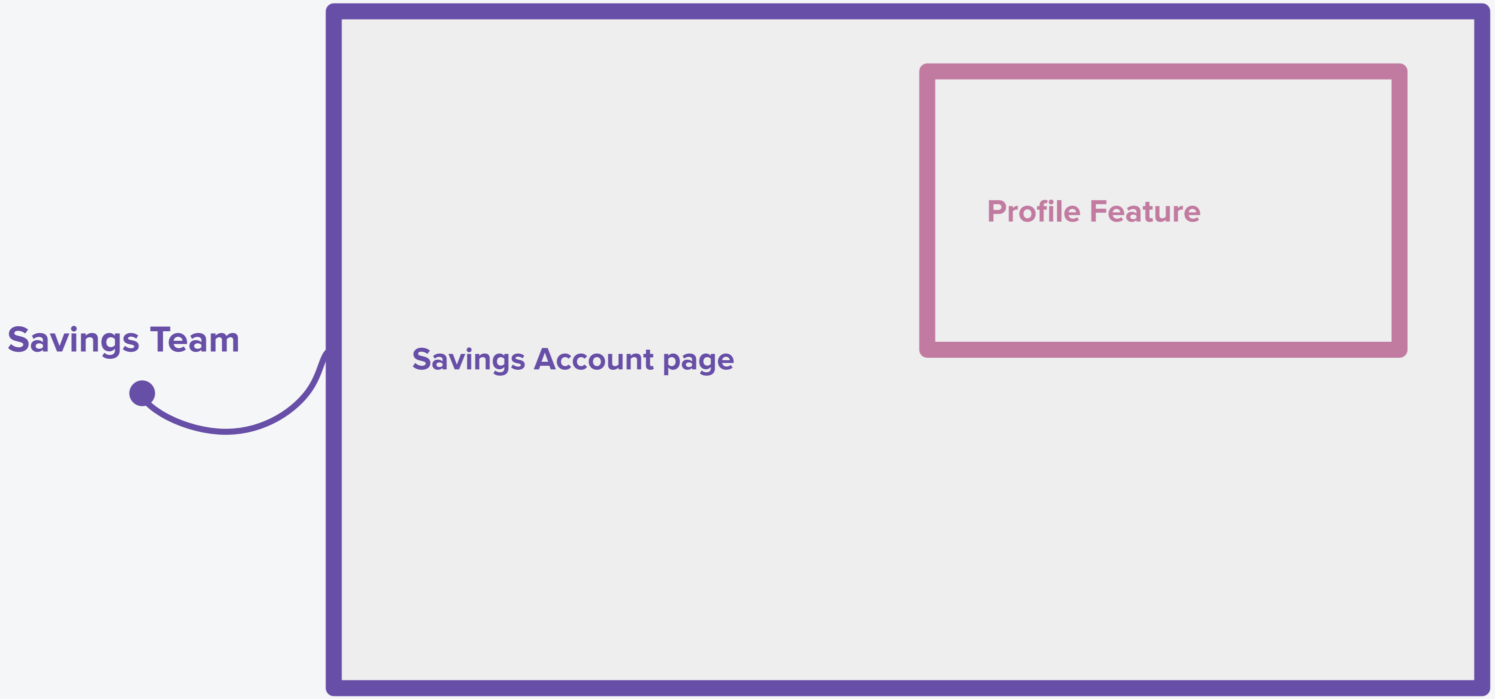


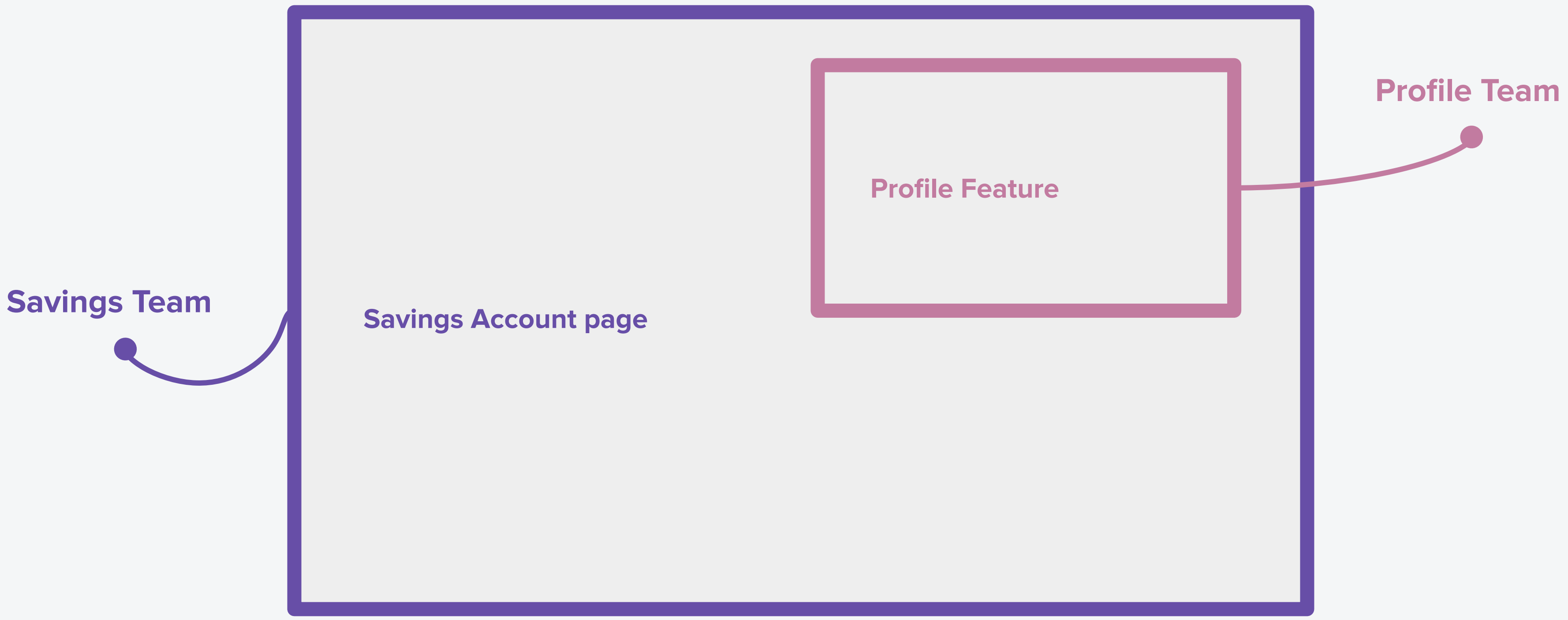
Hi, user

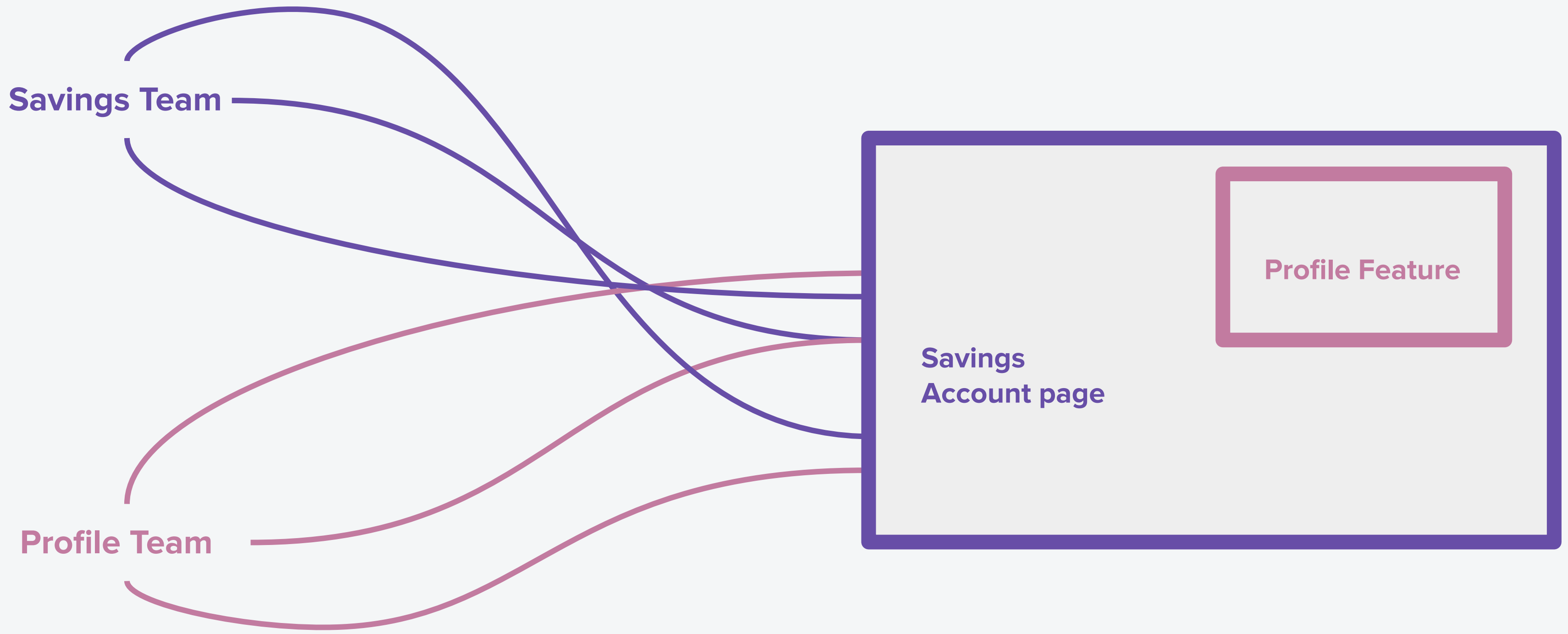
Savings Account

01/01/23	$200
02/01/23	$150
03/01/23	$210
04/01/23	$160


Profile Feature
Savings Account page


Savings Team
Savings Account page
Profile Feature


Profile Team
Profile Feature
Savings Team
Savings Account page

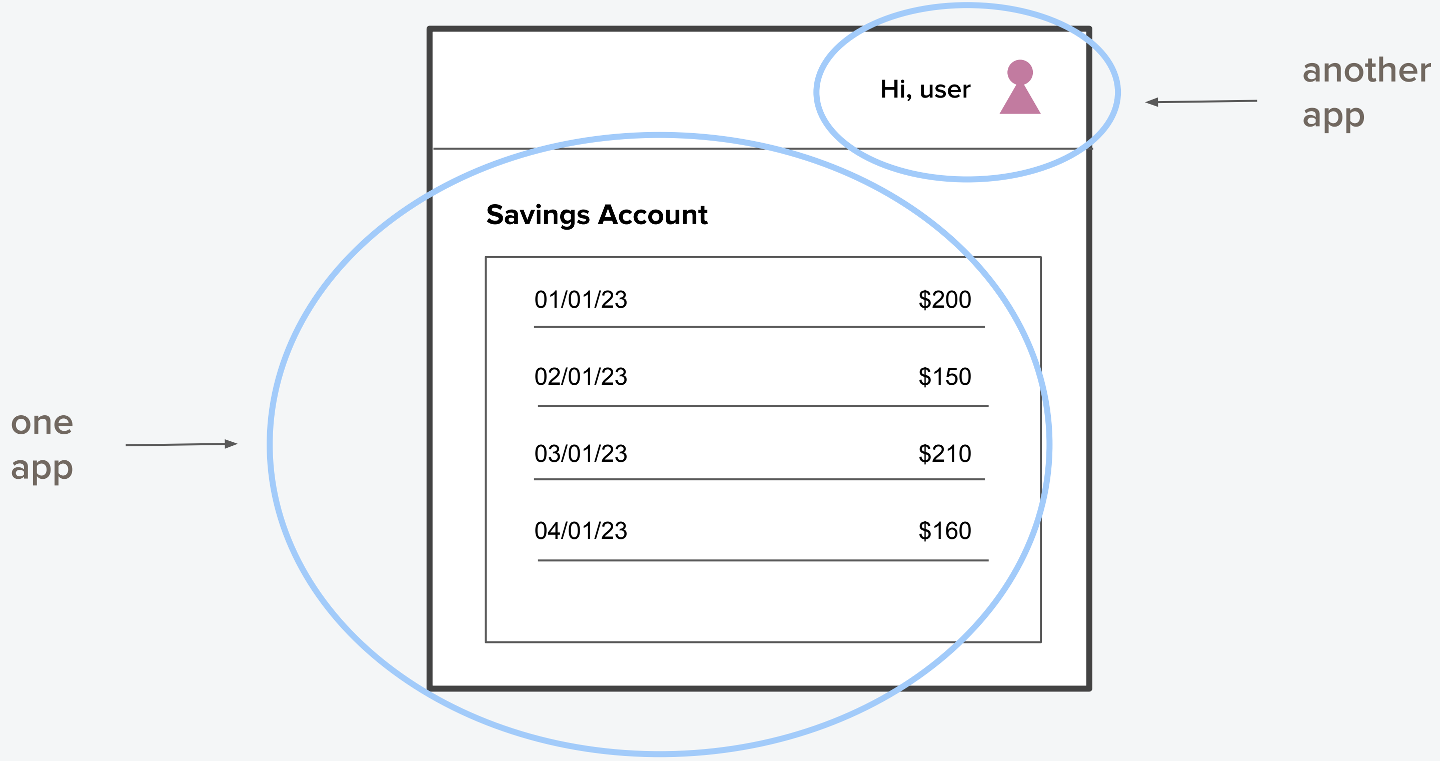

Savings Team
Profile Team
Profile Feature
Savings Account page

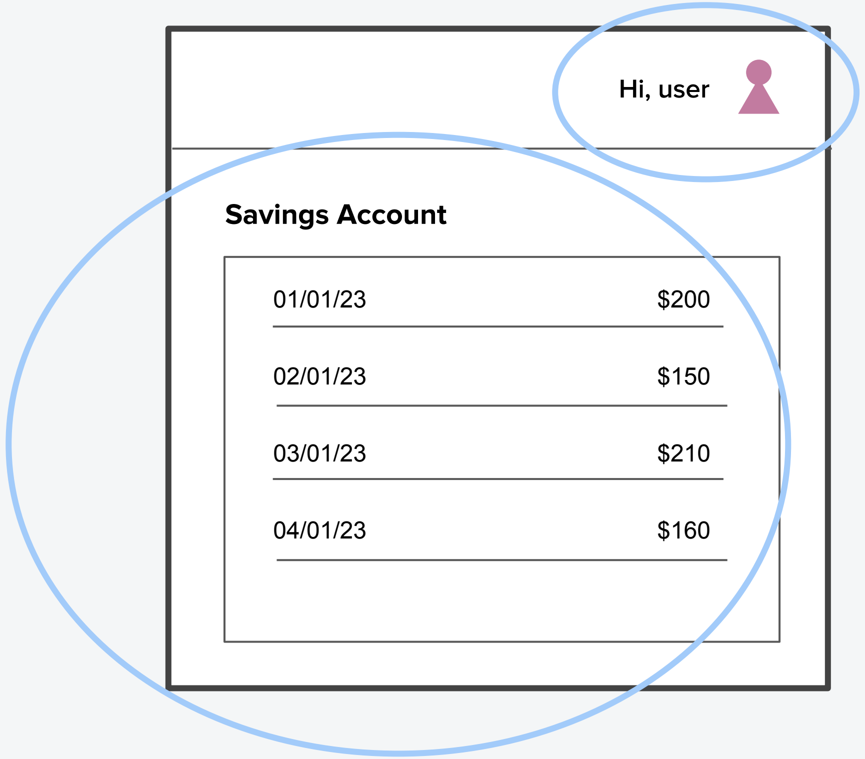
What are the problems with it?

- Teams have different code standards and processes
- Lack of communication
- Busy pipelines
- A lot of conflicts
- Dependency on deployments

It's time to start disintegrating

- architecture composed of smaller frontend applications
- **developed** and **deployed independently**
- displaying a **unified user experience.**


Hi, user
Savings Account
01/01/23
$200
02/01/23
$150
03/01/23
$210
04/01/23
$160
one app
another app



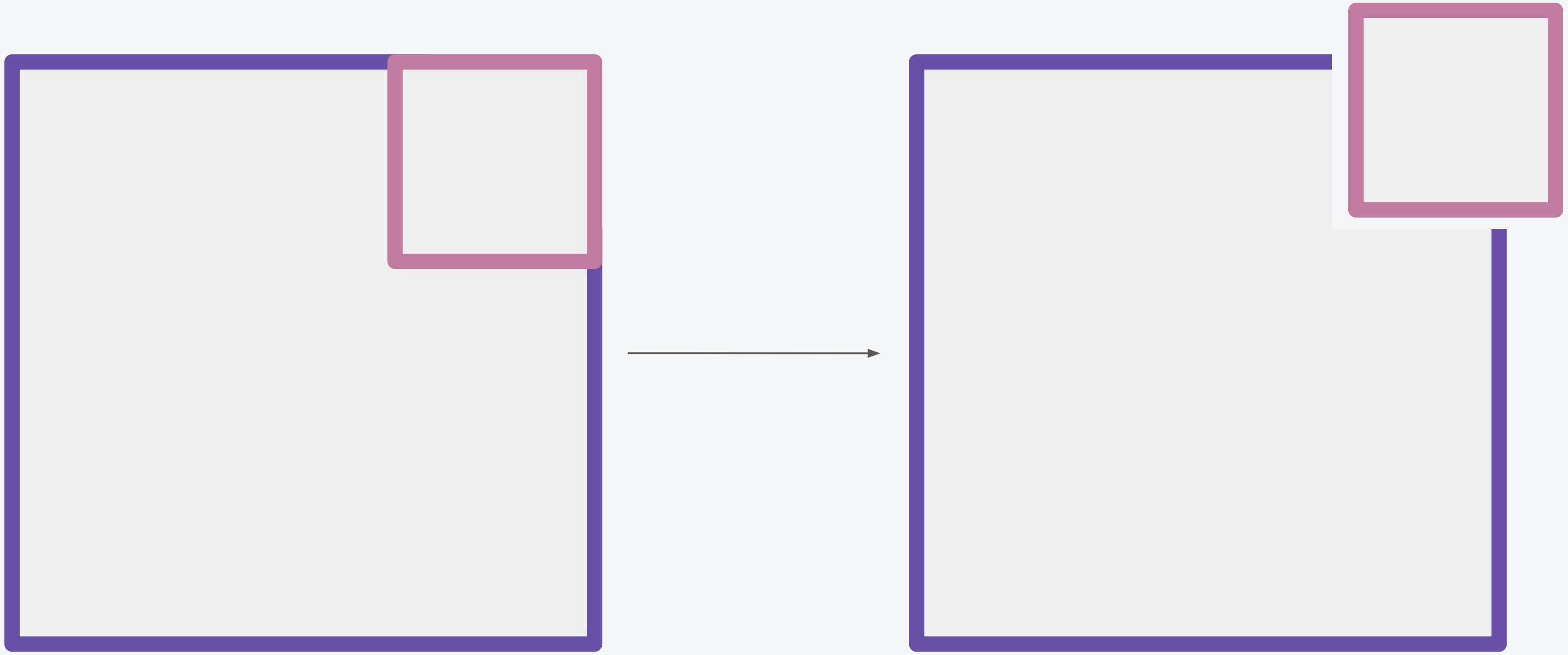
Developed and deployed independently

Developed and deployed independently

Other benefits

- Scalable
- Maintainable
- Easier to adopt new tools and tech

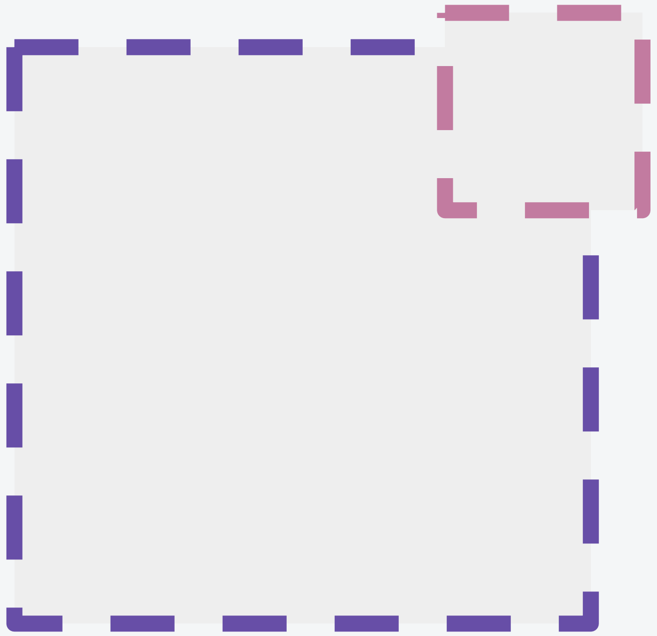
Cool, but how to we get to it?



start from scratch

vs evolve what we have

Start defining **boundaries**



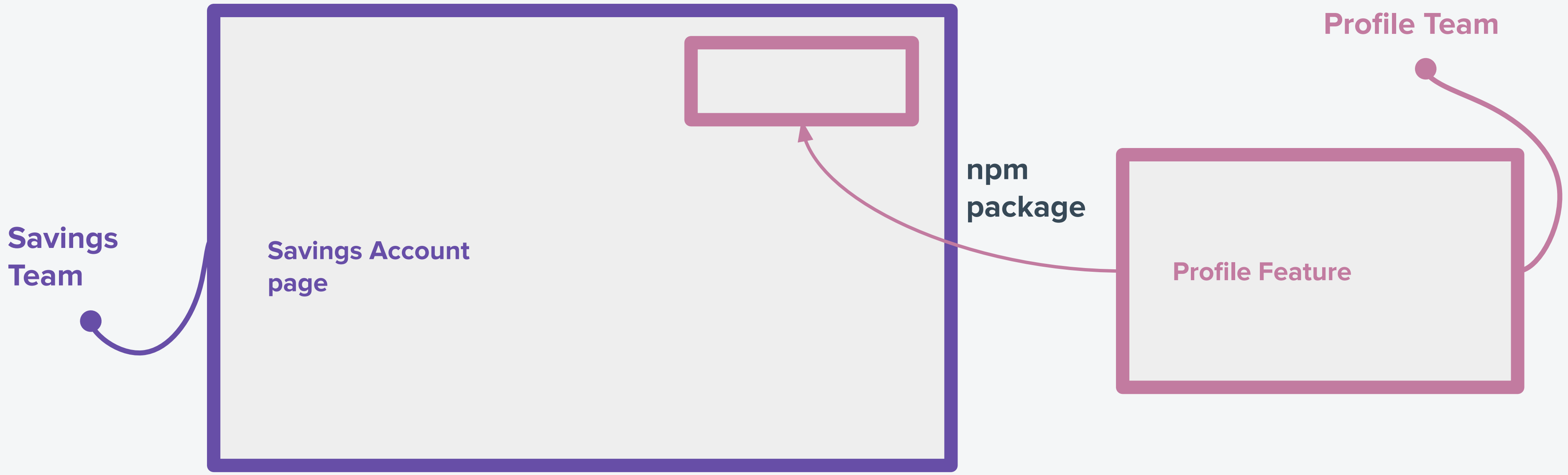
think about **Bounded Contexts**

and what parts belong together

- implemented as an **individual service**
- **evolved independently** from the other bounded contexts
- maintained and owned by **individual teams**

How to actually do it?

Profile feature as a package



What are the problems with it?

- ✓ Teams have different code standards and processes
- ✓ Lack of communication
- ✓ Busy pipelines
- ✓ A lot of conflicts
- ● Dependency on deployments

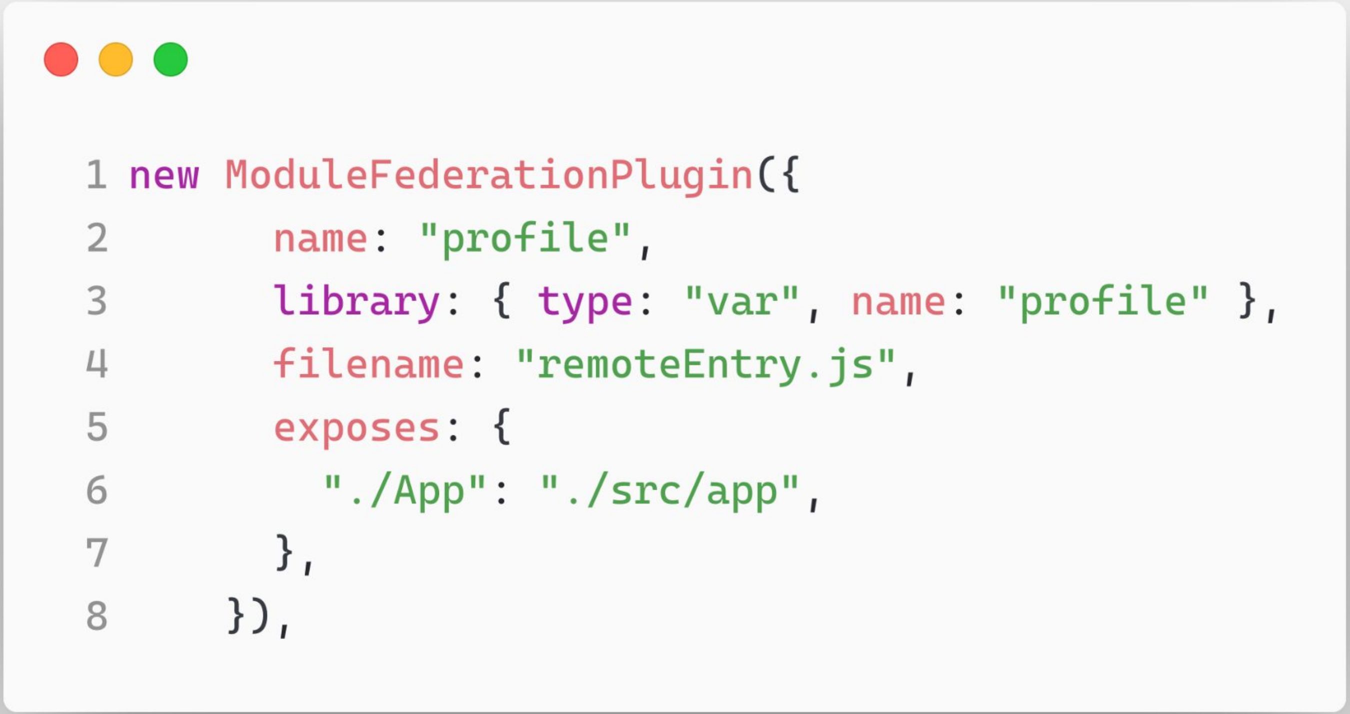
What are the problems with it?

- ✓ Teams have different code standards and processes
- ✓ Lack of communication
- ✓ Busy pipelines
- ✓ A lot of conflicts
- ● Dependency on deployments
- ● Dependency to upgrade package version

We want to achieve **Independent Deploys**

Module Federation

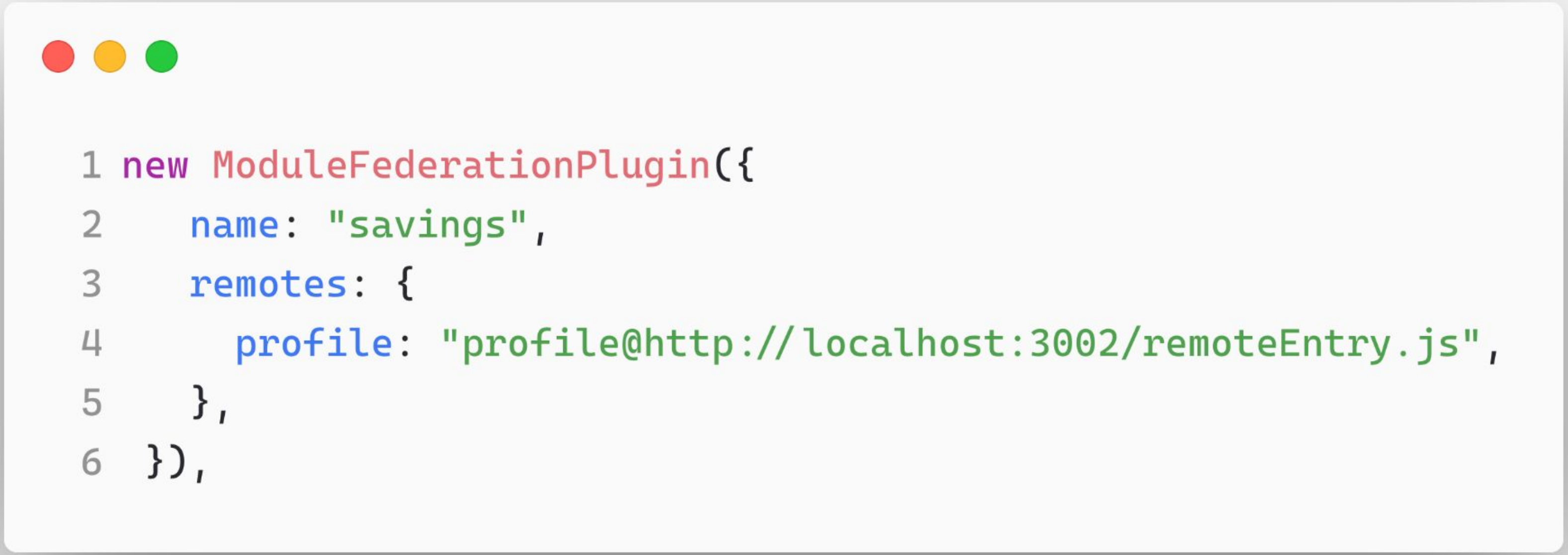
Profile App



```
new ModuleFederationPlugin({
      name: "profile",
      library: { type: "var", name: "profile" },
      filename: "remoteEntry.js",
      exposes: {
        "./App": "./src/app",
      },
    }),
```

https://github.com/thayseonofrio/mod-fed-example/tree/basic-webpack-setup

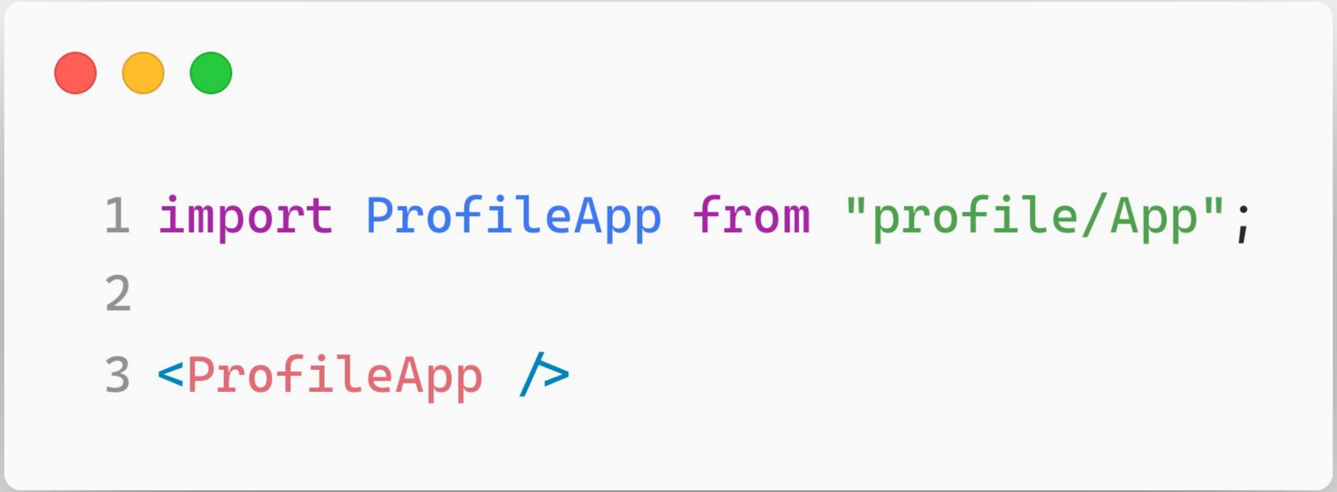
Savings App



```
new ModuleFederationPlugin({
   name: "savings",
   remotes: {
     profile: "profile@http://localhost:3002/remoteEntry.js",
   },
}),
```

https://github.com/thayseonofrio/mod-fed-example/tree/basic-webpack-setup

Savings App



```jsx
import ProfileApp from "profile/App";

<ProfileApp />
```

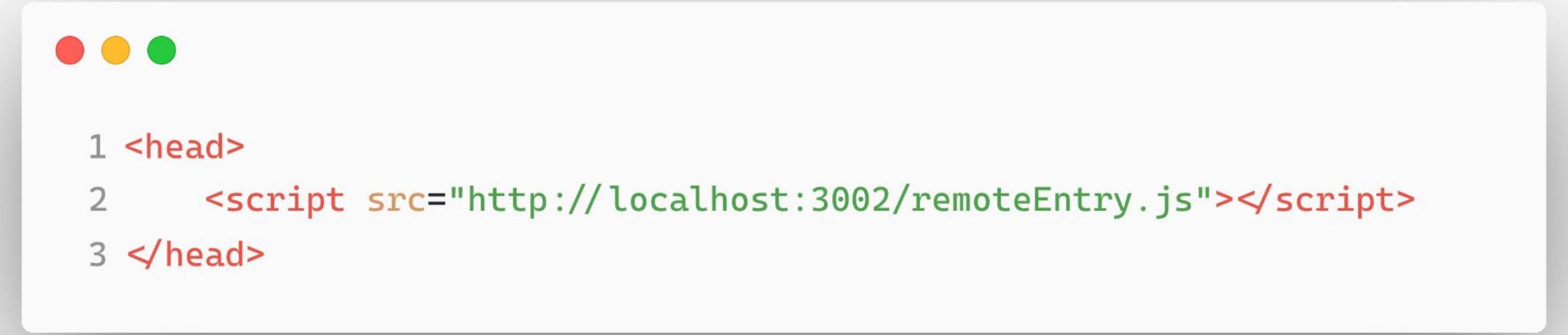
https://github.com/thayseonofrio/mod-fed-example/tree/basic-webpack-setup

What are the problems with it?

- ✓ Teams have different code standards and processes
- ✓ Lack of communication
- ✓ Busy pipelines
- ✓ A lot of conflicts
- ✓ Dependency on deployments
- ✓ Dependency to upgrade package version
- ● What if we want to load the Profile app under a condition?

Dynamic Remote Container

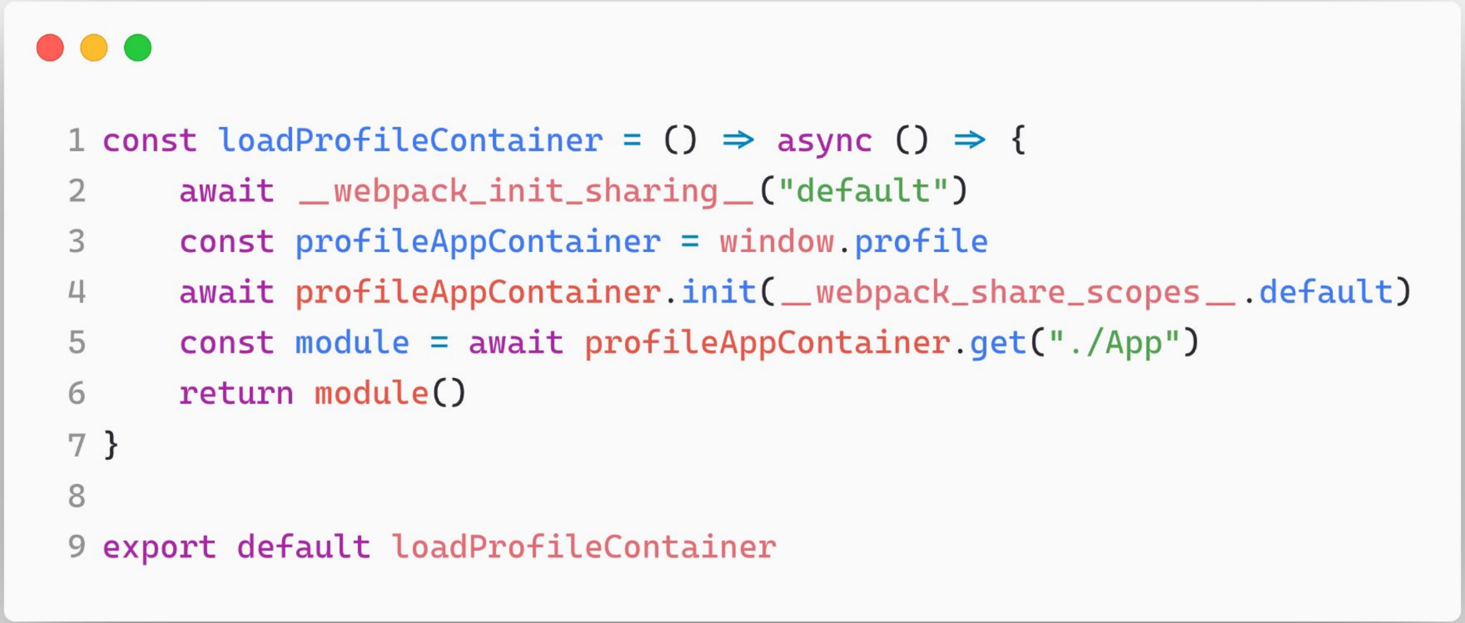
Savings App



```html
<head>
    <script src="http://localhost:3002/remoteEntry.js"></script>
</head>
```

https://github.com/thayseonofrio/mod-fed-example/tree/dynamic-remote-container

Savings App

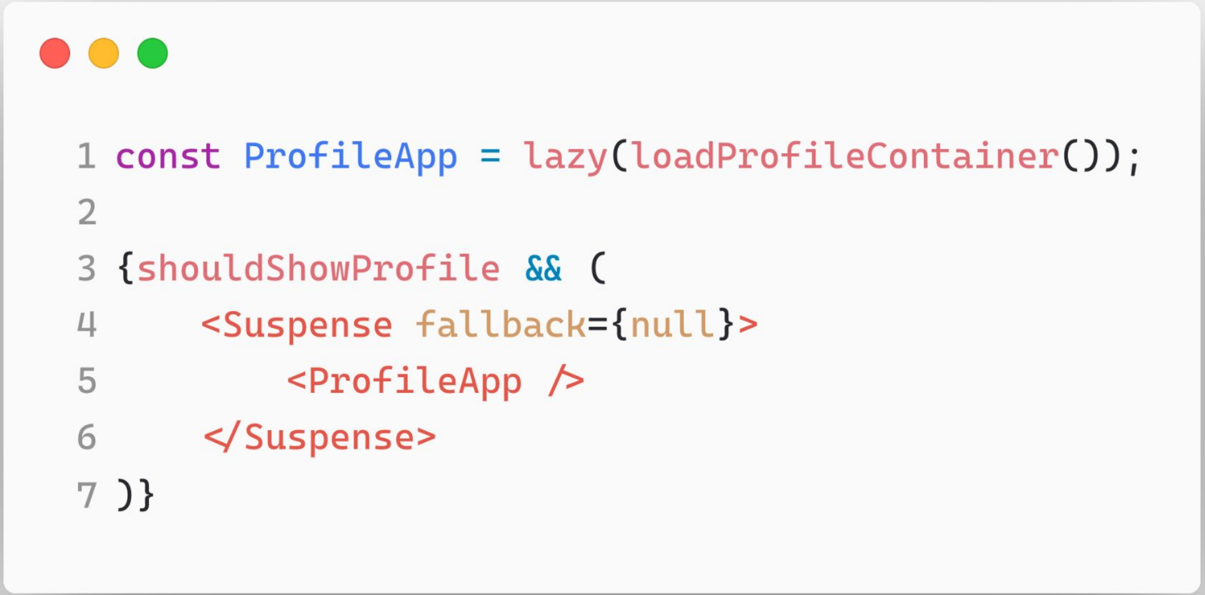


```
const loadProfileContainer = () => async () => {
    await __webpack_init_sharing__("default")
    const profileAppContainer = window.profile
    await profileAppContainer.init(__webpack_share_scopes__.default)
    const module = await profileAppContainer.get("./App")
    return module()
}

export default loadProfileContainer
```

https://github.com/thayseonofrio/mod-fed-example/tree/dynamic-remote-container

Savings App



```
const ProfileApp = lazy(loadProfileContainer());

{shouldShowProfile && (
    <Suspense fallback={null}>
        <ProfileApp />
    </Suspense>
)}
```

https://github.com/thayseonofrio/mod-fed-example/tree/dynamic-remote-container

What are the problems with it?

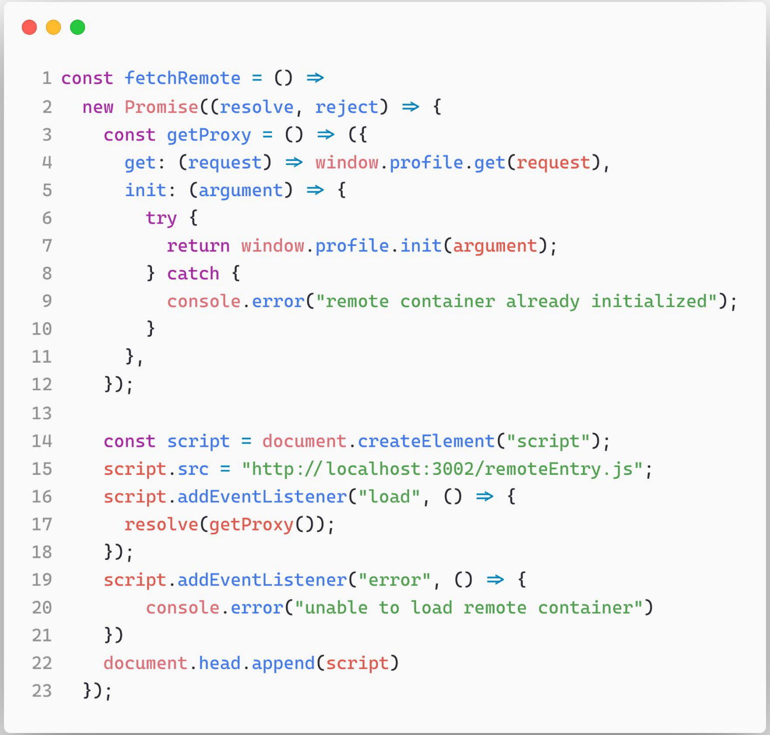
- ✓ Teams have different code standards and processes
- ✓ Lack of communication
- ✓ Busy pipelines
- ✓ A lot of conflicts
- ✓ Dependency on deployments
- ✓ Dependency to upgrade package version
- ✓ What if we want to load the Profile app under a condition?

What are the problems with it?

- ✓ Teams have different code standards and processes
- ✓ Lack of communication
- ✓ Busy pipelines
- ✓ A lot of conflicts
- ✓ Dependency on deployments
- ✓ Dependency to upgrade package version
- ✓ What if we want to load the Profile app under a condition?
- ● Strong coupling

Promise Based Dynamic Remotes

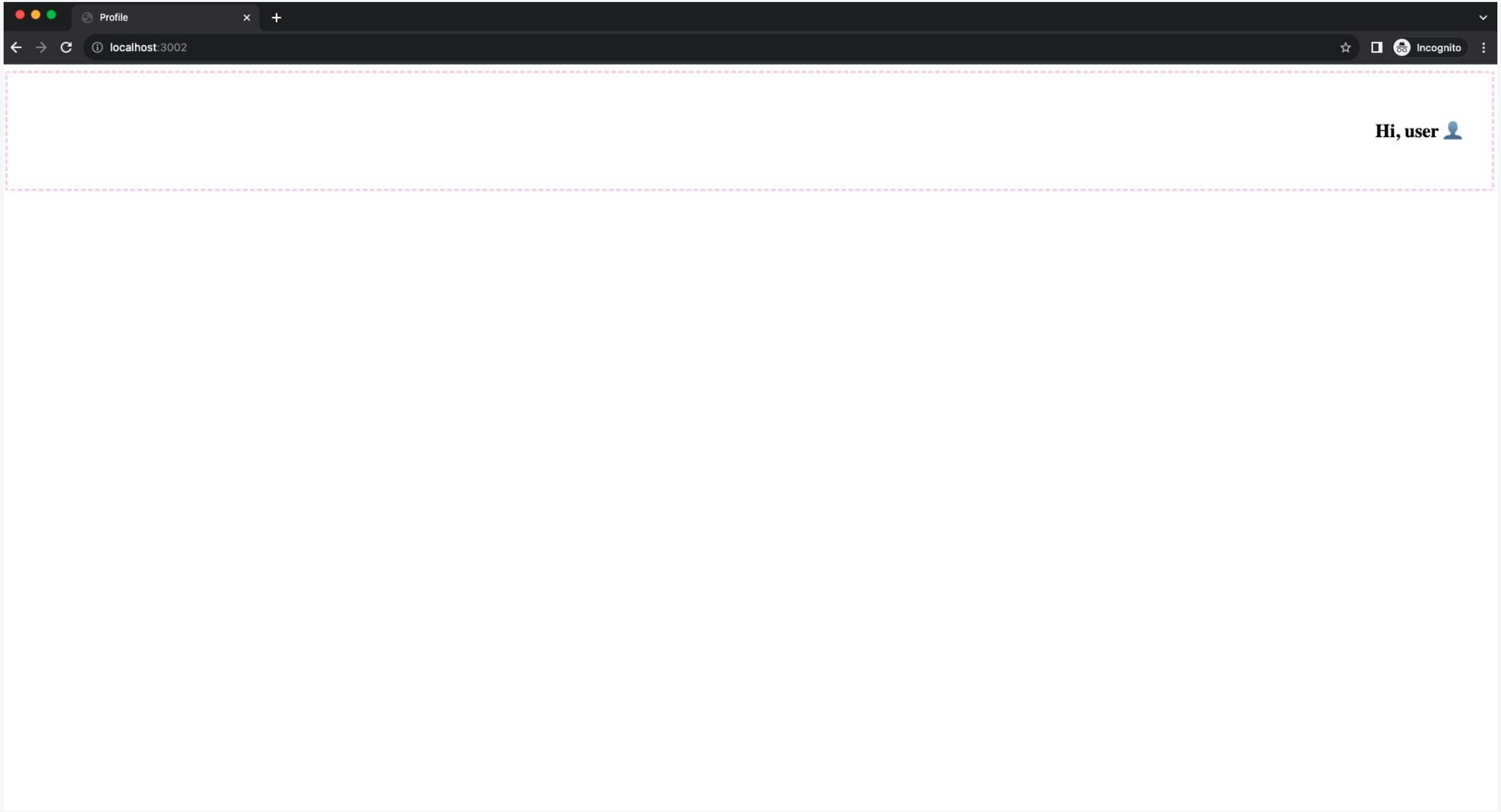
Savings App



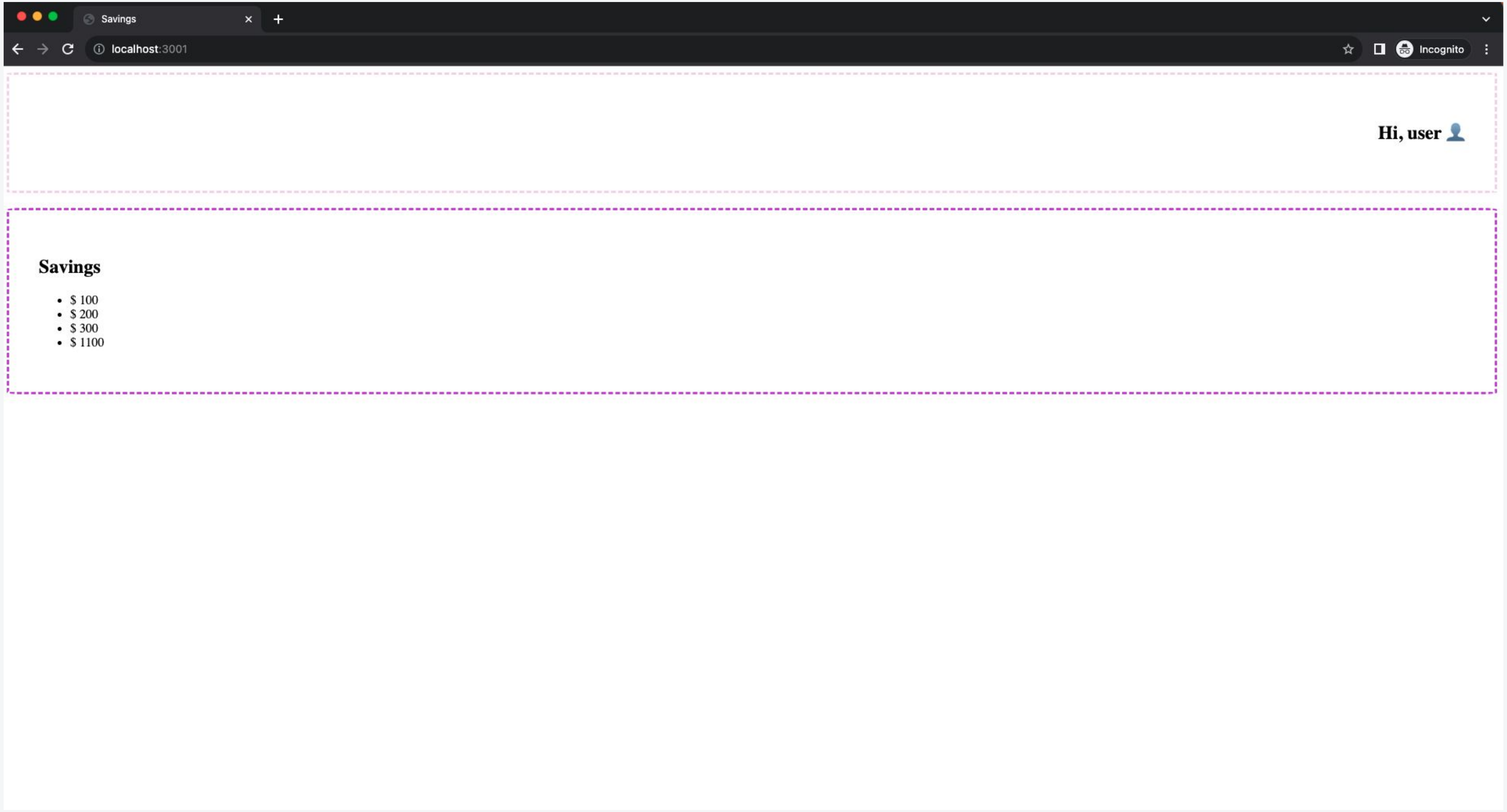
```
const fetchRemote = () =>
  new Promise((resolve, reject) => {
    const getProxy = () => ({
      get: (request) => window.profile.get(request),
      init: (argument) => {
        try {
          return window.profile.init(argument);
        } catch {
          console.error("remote container already initialized");
        }
      },
    });

    const script = document.createElement("script");
    script.src = "http://localhost:3002/remoteEntry.js";
    script.addEventListener("load", () => {
      resolve(getProxy());
    });
    script.addEventListener("error", () => {
        console.error("unable to load remote container")
    })
    document.head.append(script)
  });
```

https://github.com/thayseonofrio/mod-fed-example/tree/promise-based-dynamic-remotes



Hi, user 👤



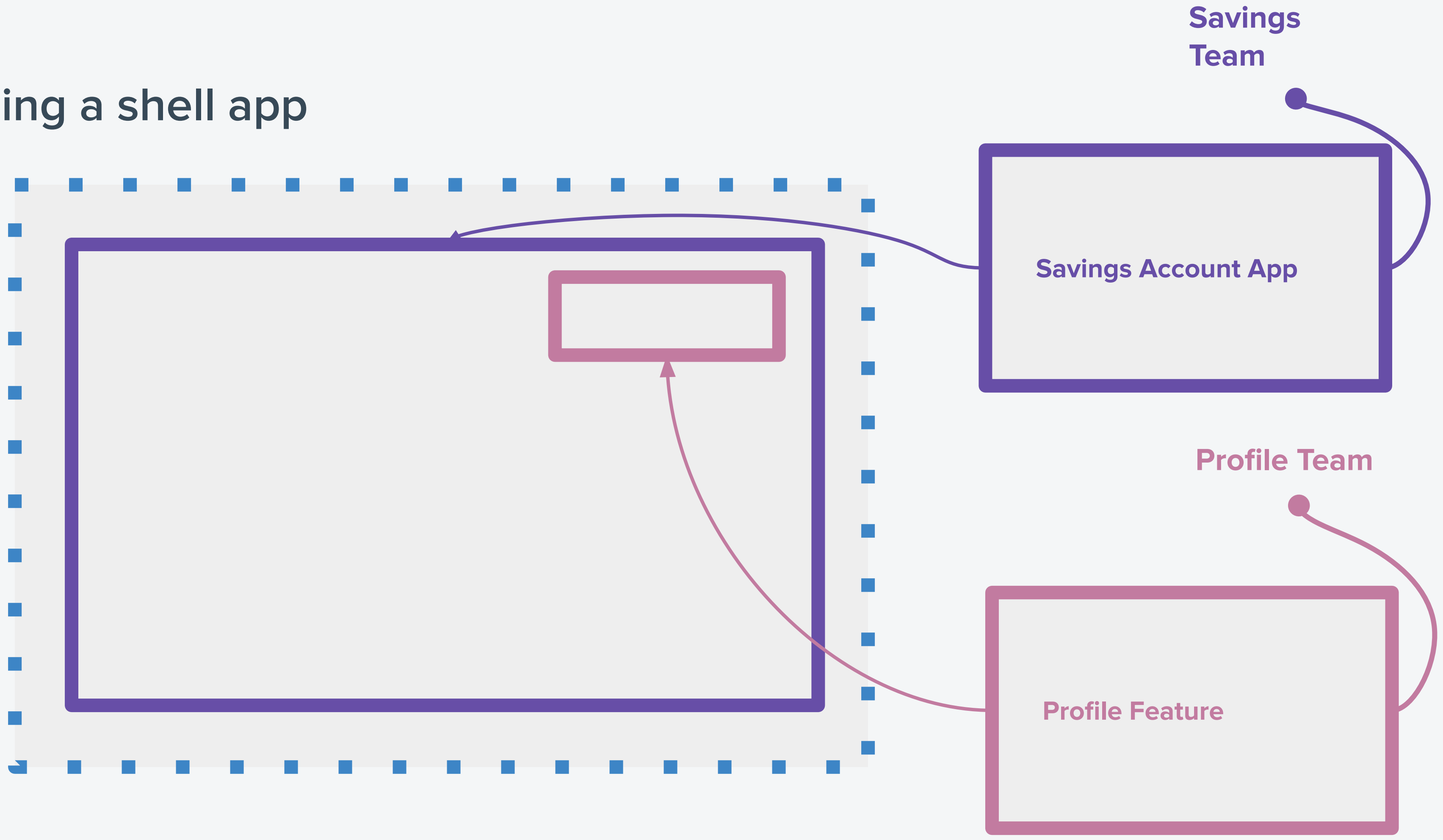
Hi, user 👤

Savings

- $ 100
- $ 200
- $ 300
- $ 1100

[redacted]

Using a shell app



What are the problems with it?

- ✓ Teams have different code standards and processes
- ✓ Lack of communication
- ✓ Busy pipelines
- ✓ A lot of conflicts
- ✓ Dependency on deployments
- ✓ Dependency to upgrade package version
- ✓ What if we want to load the Profile app under a condition?
- ✓ Strong coupling

New things to worry about:

- Sharing Data

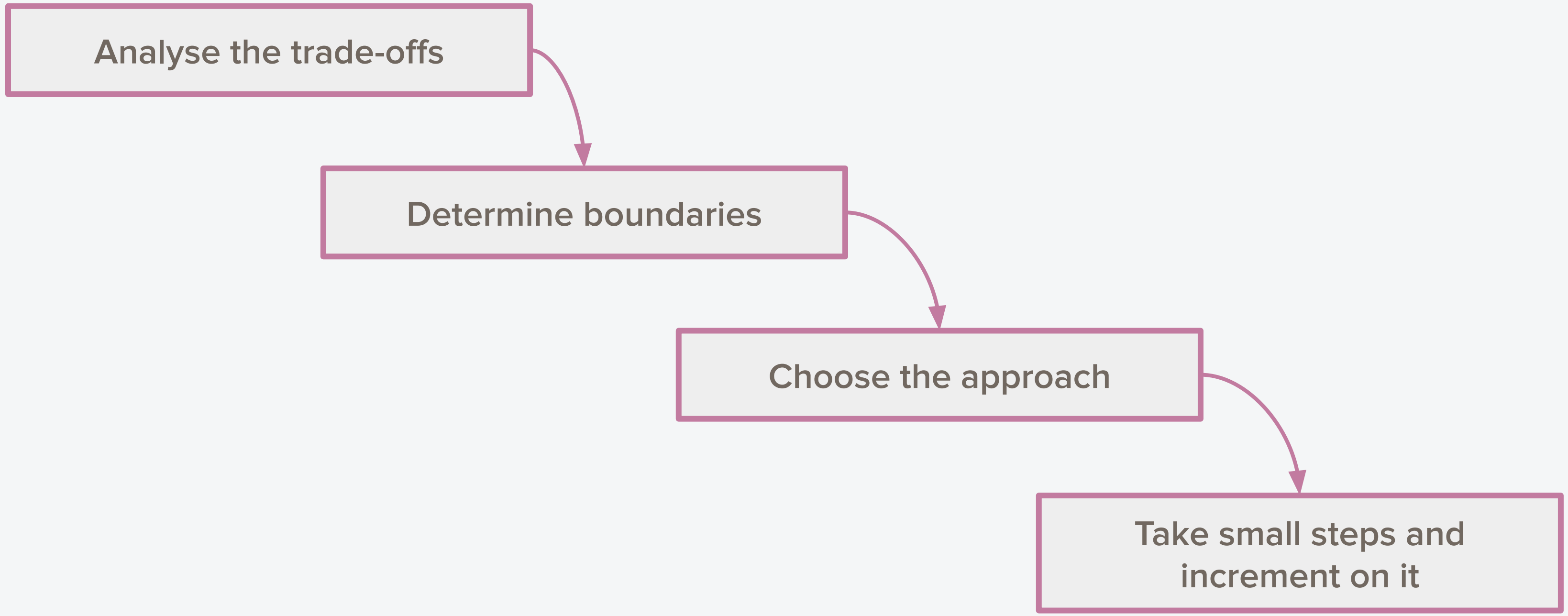
New things to worry about:

- Sharing Data
- Duplicate Dependencies

New things to worry about:

- Sharing Data
- Duplicate Dependencies
- Testing

How to disintegrate a frontend



Micro frontends won't solve all your problems

Thanks

Thayse Onofrio
thayseonofrio.com