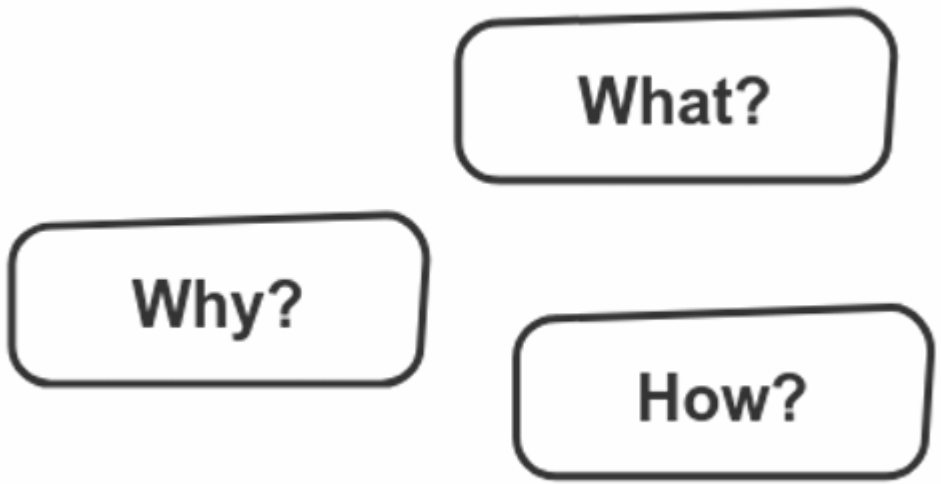
Introduction to Microfrontends


What?
Why?
How?

Mario Fernandez

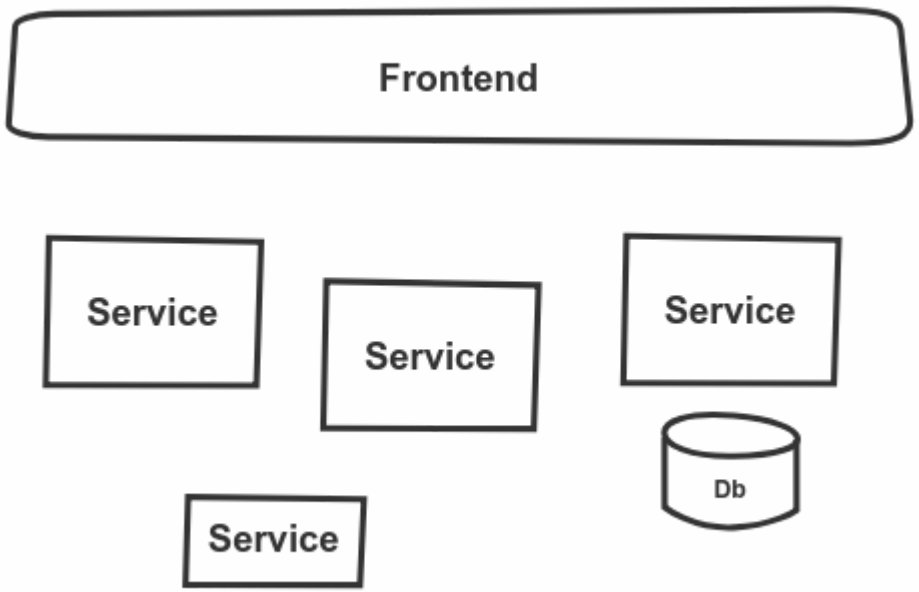
Matthias Kainer

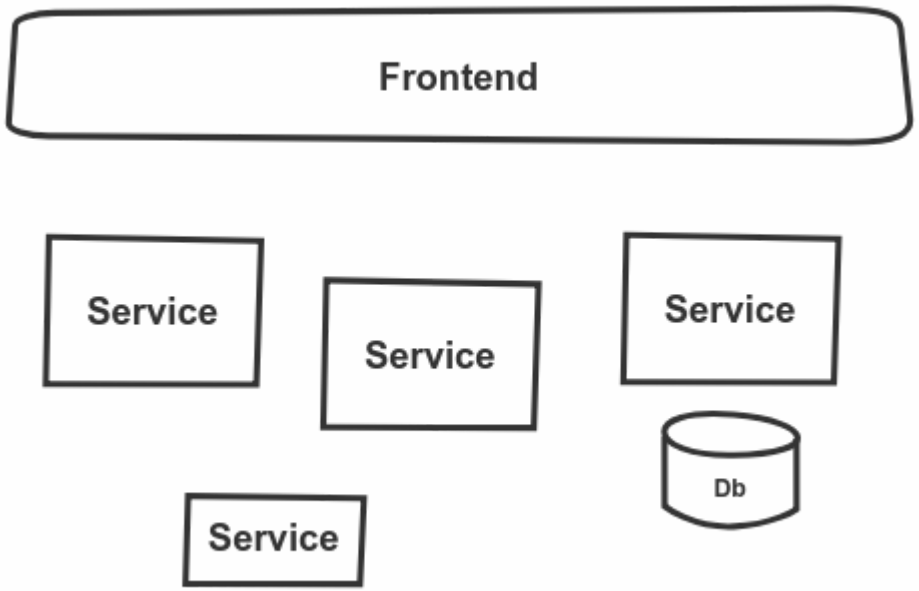
ThoughtWorks

Everybody got into the microservices train

Whether they actually needed or not

What about the frontend, though?


Frontend
Service
Service
Service
Db
Service



Release frequency

Backend: Daily

Frontend: Quarterly

Nov 19 Tech Radar

thoughtworks.com/de/radar

Nov 19 Tech Radar

> We've seen significant benefits from introducing microservices, which have allowed teams to scale the delivery of independently deployed and maintained services. Unfortunately, we've also seen many teams create a front-end monolith — a large, entangled browser application that sits on top of the back-end services — largely neutralizing the benefits of microservices.

thoughtworks.com/de/radar

Nov 19 Tech Radar

> We've seen significant benefits from introducing microservices, which have allowed teams to scale the delivery of independently deployed and maintained services. Unfortunately, we've also seen many teams create a front-end monolith — a large, entangled browser application that sits on top of the back-end services — largely neutralizing the benefits of microservices.

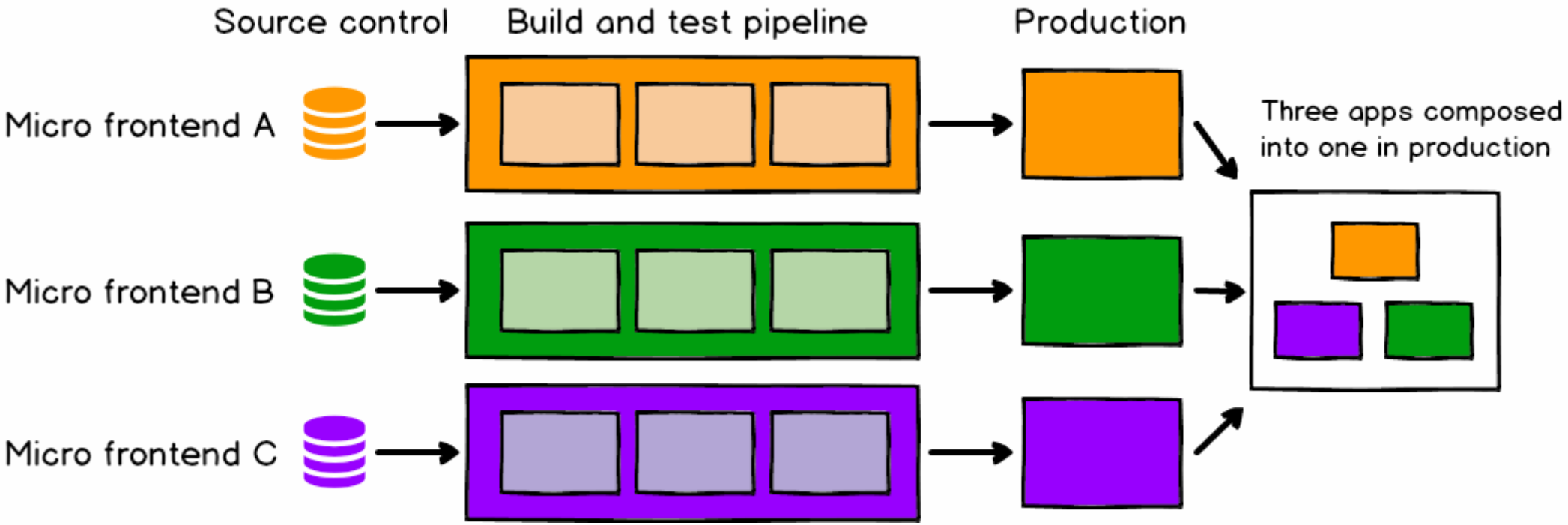
> Micro frontends have continued to gain in popularity since they were first introduced. We've seen many teams adopt some form of this architecture as a way to manage the complexity of multiple developers and teams contributing to the same user experience.

thoughtworks.com/de/radar

What are microfrontends?

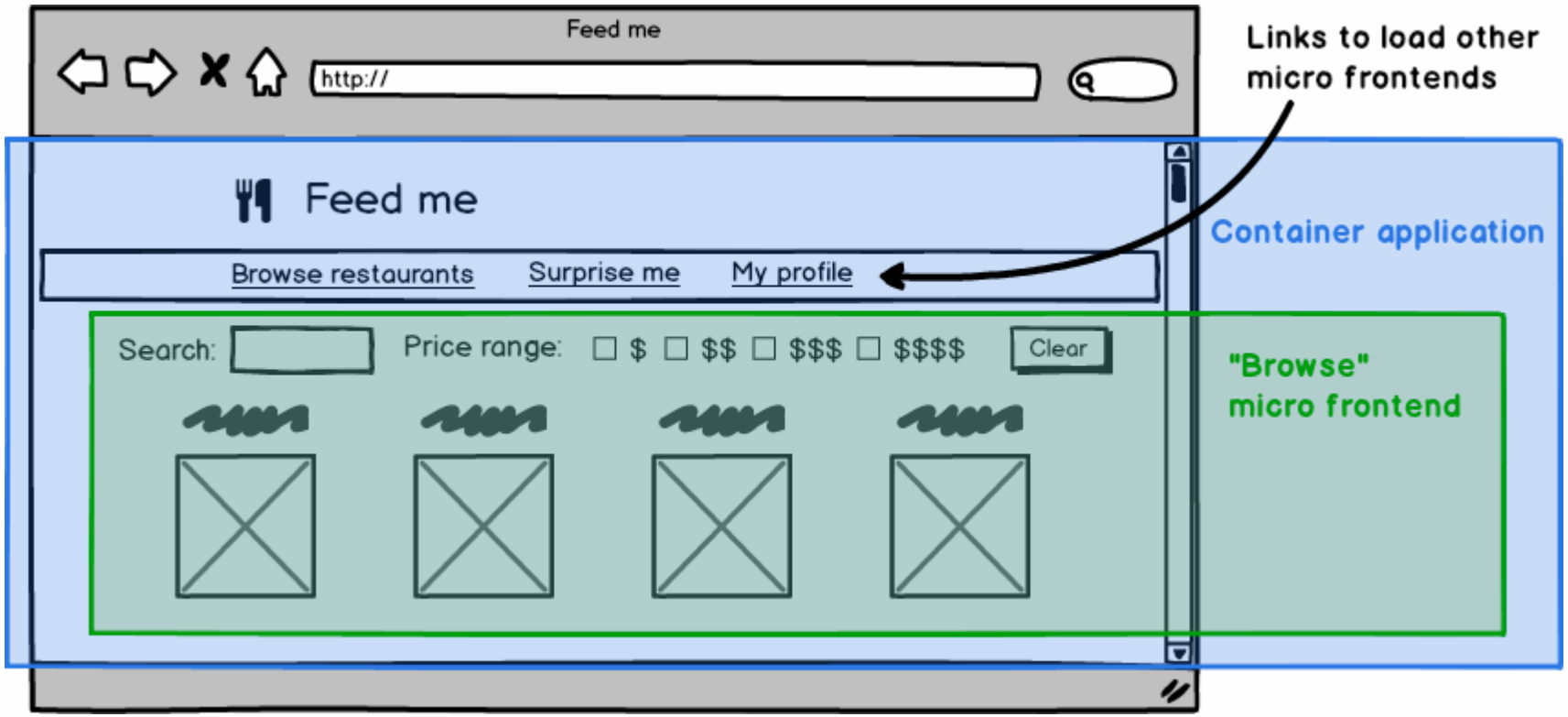
An architectural style where independently deliverable frontend applications are composed into a greater whole

From the developers perspective



microfrontends.com

From the users perspective


Feed me
http://
Feed me
Browse restaurants
Surprise me
My profile
Search:
Price range:
$
$$
$$$
$$$$
Clear
Links to load other micro frontends
Container application
"Browse" micro frontend

martinfowler.com/articles/micro-frontends.html

Why use them?

Incremental updates

Independent deployments

Autonomous teams

Should you use microfrontends?

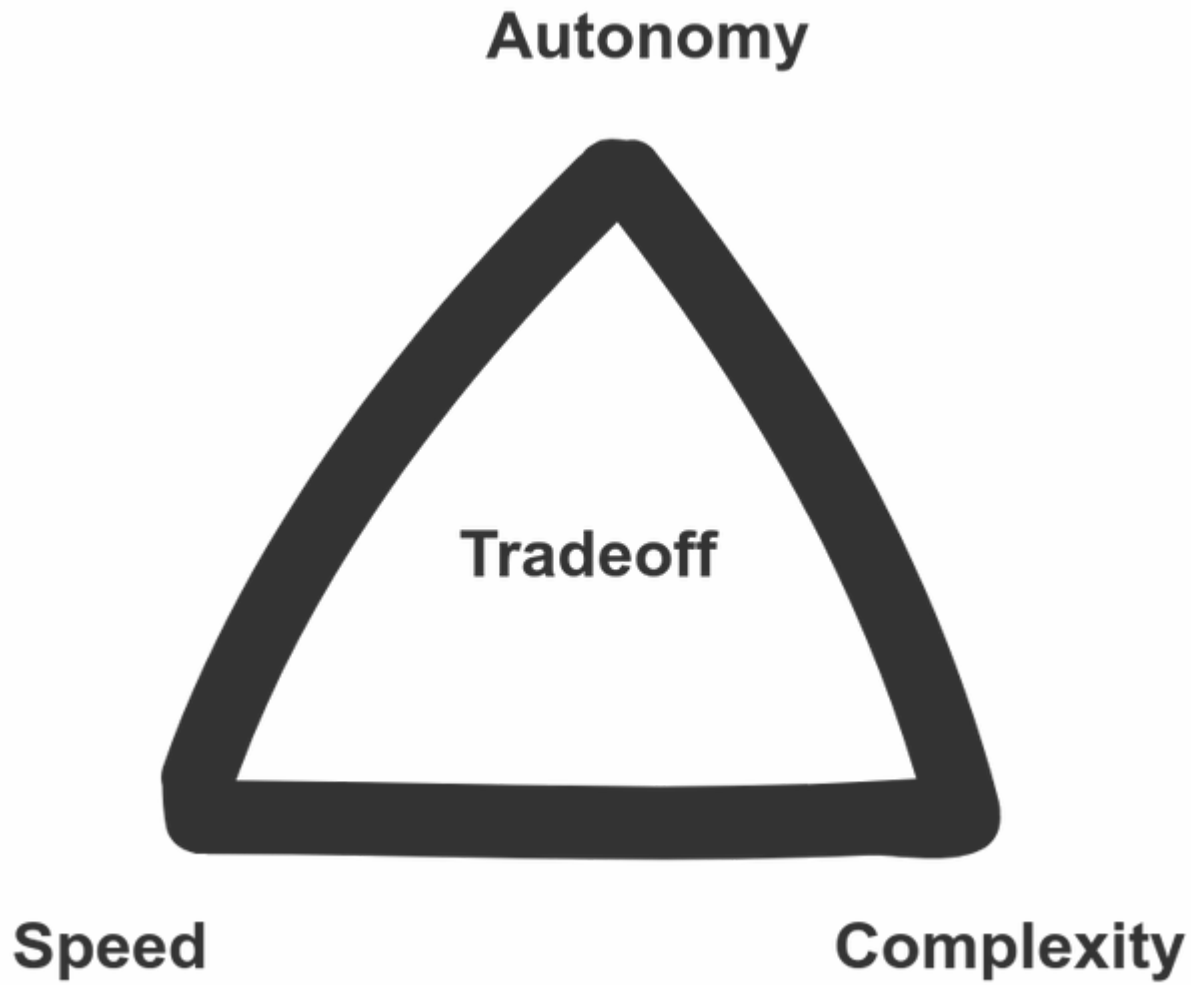
Not everybody is sold

Dan Abramov
@dan_abramov

I don’t understand micro-frontends.

5:48 AM · May 26, 2019 · Twitter Web App

131 Retweets **855** Likes


Autonomy
Tradeoff
Speed
Complexity

Factors to consider

Are your teams...

Vertically divided

Horizontally divided

Is your priority...

Fast time to market

High UX consistency

Are you releasing your application...

Once every quarter

Once every hour

Composition approaches

Server Side

- Build time integration
- Server Side Includes

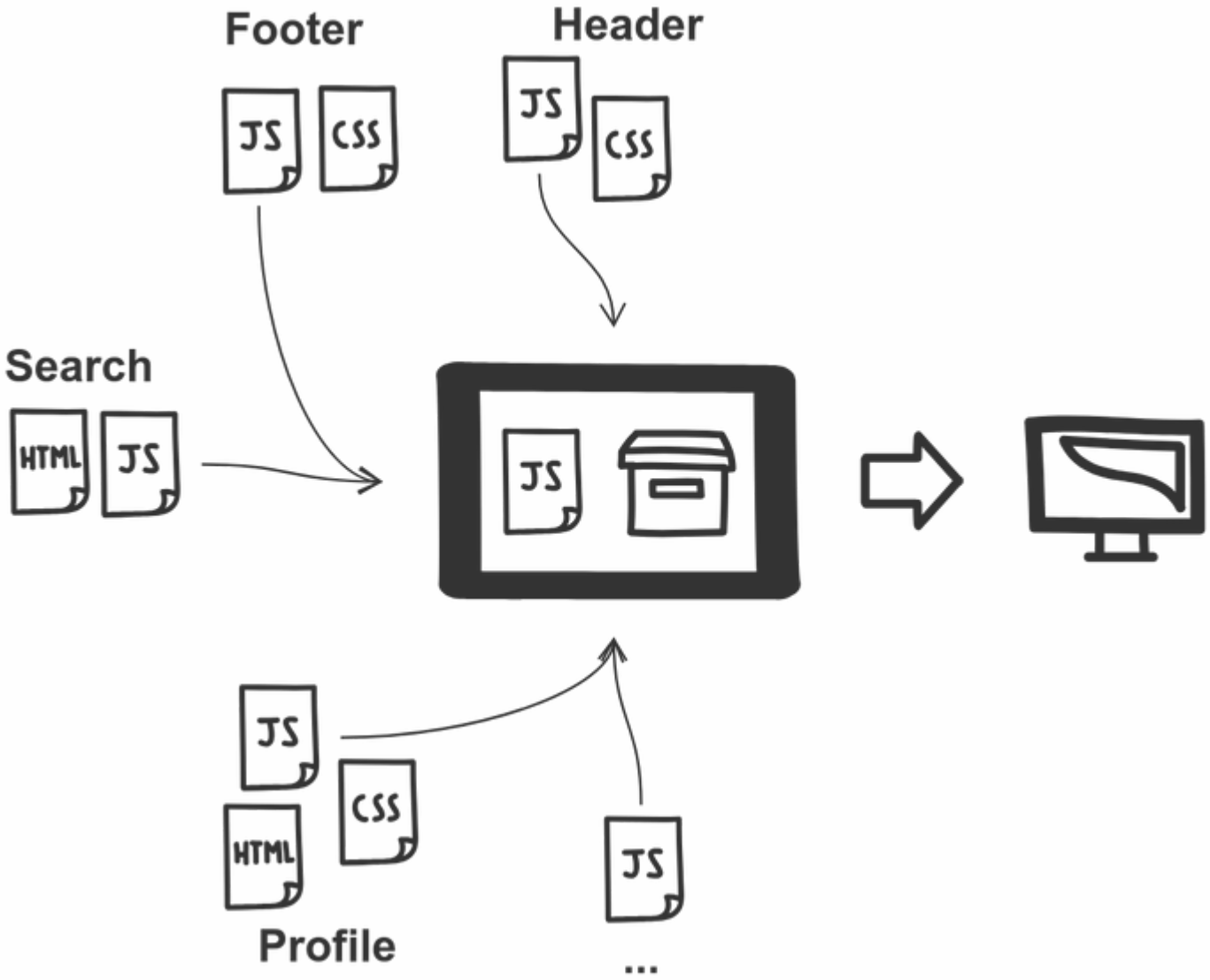
Client Side

- IFrames
- Component Based
- Web Components

Build time integration

Divide the application into separate npm packages

Deploy them as one entity


Footer
JS
CSS
Header
JS
CSS
Search
HTML
JS
JS
JS
CSS
HTML
Profile
JS
...

```
{
  "name": "@awesome-corp/container",
  "version": "1.0.0",
  "dependencies": {
    "@awesome-corp/search": "^1.2.3",
    "@awesome-corp/profile": "^7.8.9",
    "@awesome-corp/header": "^0.1.2",
    "@awesome-corp/footer": "^0.2.5"
  }
}
```

```
<BrowserRouter>
  <CssBaseline />
  <Header />

  <Box mt={12}>
    <Container component="main">
      <Switch>
        <Route path="/search" component={Search} />
        <Route path="/profile" component={Profile} />
      </Switch>
    </Container>
  </Box>

  <Footer />
</BrowserRouter>
```

Managing those packages

Lerna

github.com/lerna/lerna

```
lerna.json
package.json
packages/
  main/
  header/
  footer/
  microfrontend1/
  microfrontend2/
```

```
{
  "lerna": "3.15.0",
  "version": "independent",
  "npmClient": "yarn",
  "useWorkspaces": true,
  "packages": ["packages/*"]
}
```

✔ Simple

✔ Simple

✔ No runtime impact

❌ Independent deployments

❌ Independent deployments

❌ Autonomy

Good as a starting point

Server Side Includes

What is a SSI?

> Server Side Includes (SSI) is a simple interpreted server-side scripting language used almost exclusively for the World Wide Web. It is most useful for including the contents of one or more files into a web page on a web server, using its #include directive.

```
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>SSI Example</title>
  </head>
  <body>
    <h1>Static header</h1>
    <hr/>
    <!--# include file="$PAGE.html" -->
  </body>
</html>
```

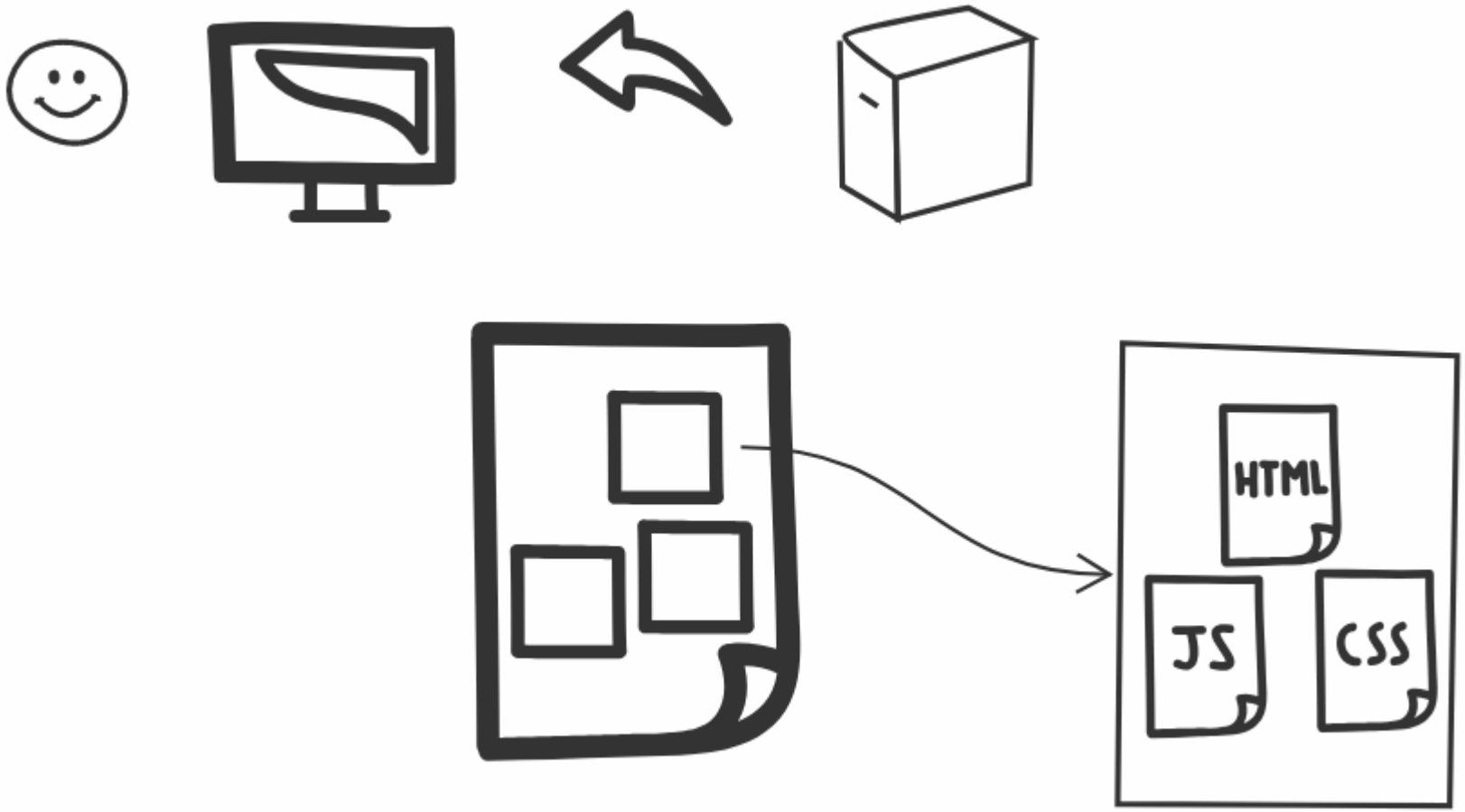
```
server {
    listen 8080;
    index index.html;
    ssi on;

    # Decide which HTML fragment to insert based on the URL
    location /browse {
        set $PAGE 'browse';
    }

    location /order {
        set $PAGE 'order';
    }

    location /profile {
        set $PAGE 'profile';
    }
}
```

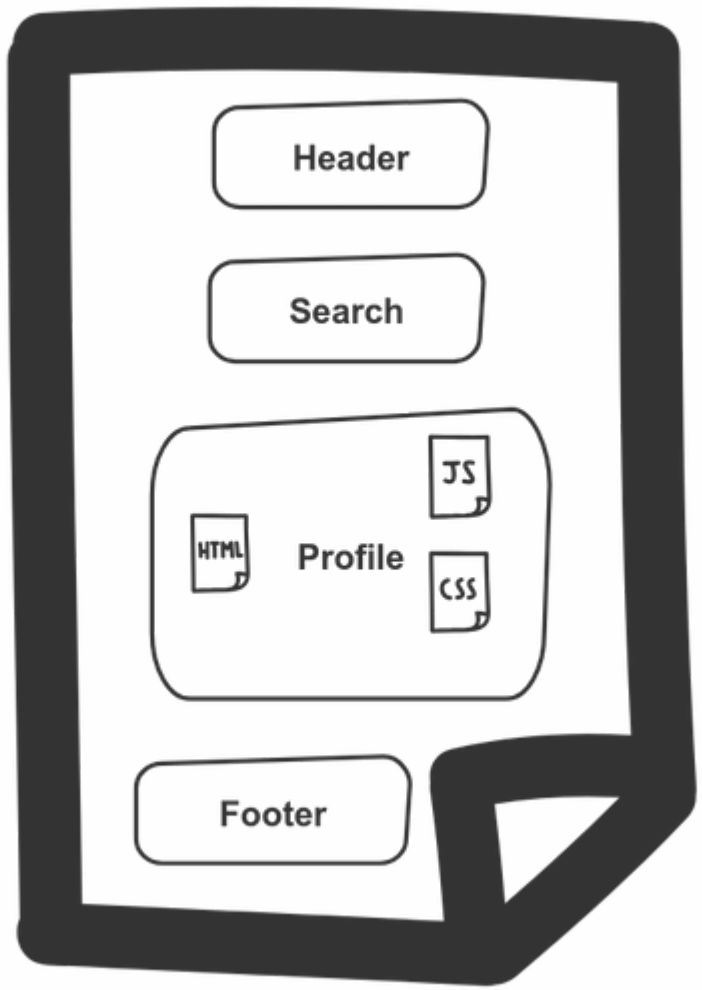
Where is the microfrontend here?


HTML
JS
CSS

```
<html lang="en" dir="ltr">
 <body>
    <!--# include file="$HEADER.html" -->
    <main>
      <!--# include file="$PAGE.html" -->
    </main>

    <aside>
      <!--# include file="$SIDEBAR.html" -->
    </aside>

    <!--# include file="$FOOTER.html" -->
  </body>
</html>
```


Header
Search
JS
HTML
Profile
CSS
Footer

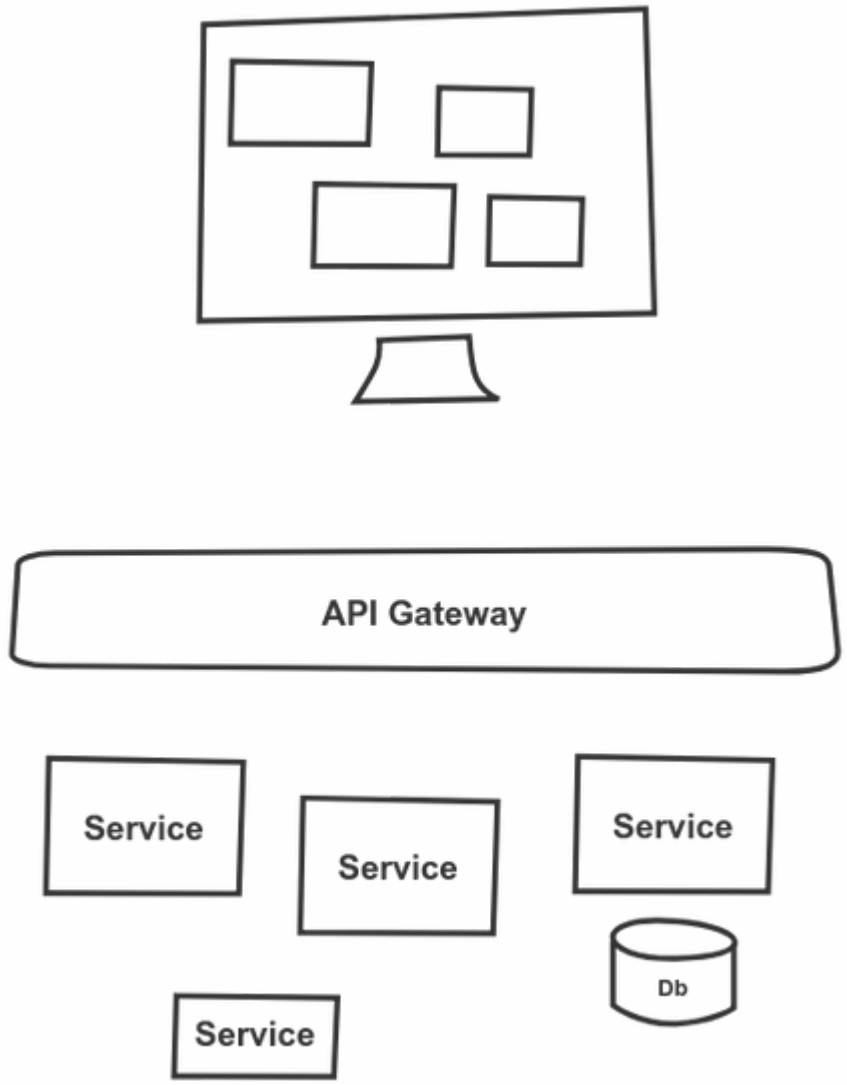
✔ (Still) Simple

❌ Managing assets

thoughtworks.com/talks/a-high-performmance-solution-to-microservice-ui-composition

Backend interaction

What to avoid


API Gateway
Service
Service
Service
Service
Db

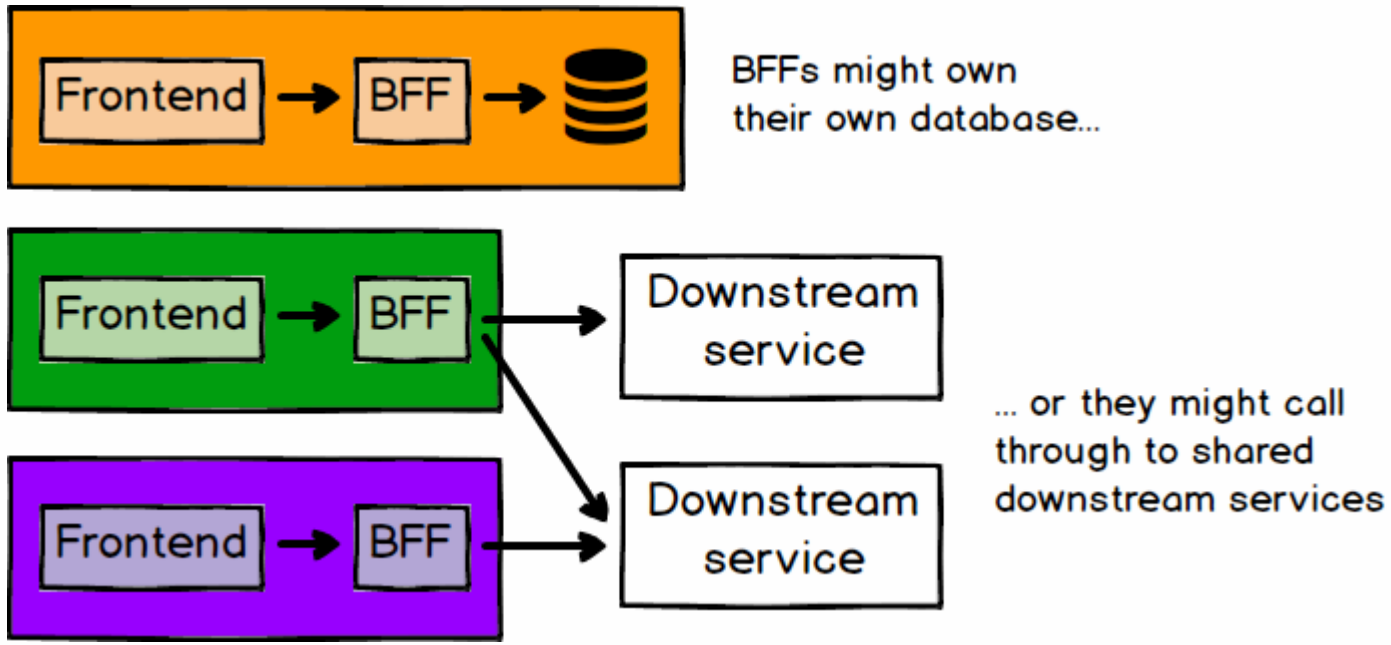
Backend for frontend

samnewman.io/patterns/architectural/bff/

A custom backend for a particular client

Built together with the frontend

Owned by the same team


Frontend
BFF
BFFs might own
their own database...
Frontend
BFF
Downstream
service
... or they might call
through to shared
downstream services
Frontend
BFF
Downstream
service

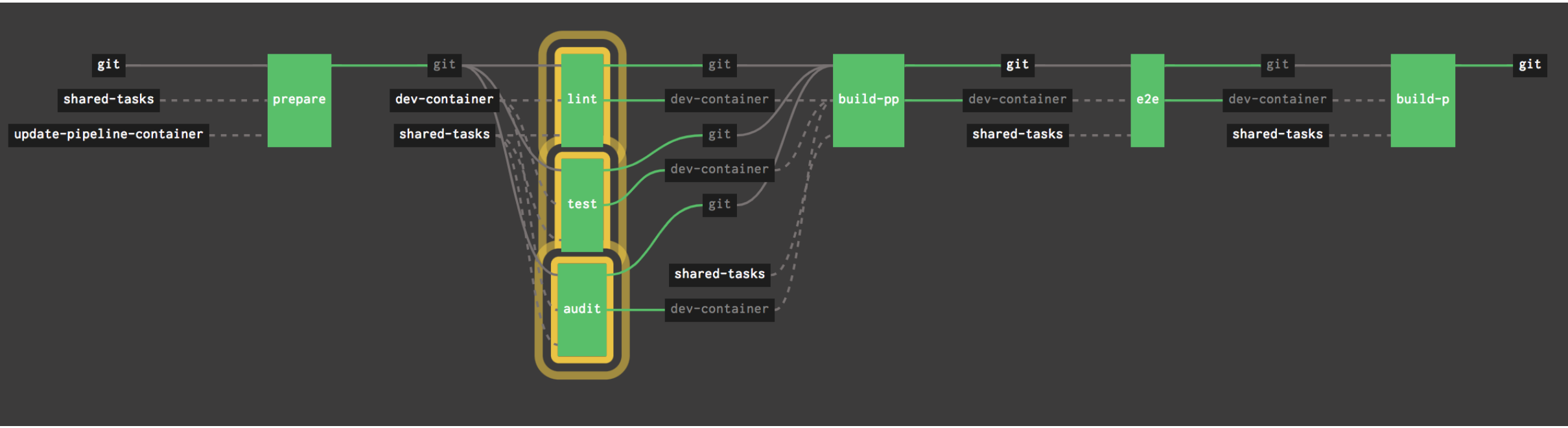
Microfrontends talk to their own BFF

Avoids coupling

Chatty applications make your life harder

CI/CD

Smaller clients lead to Smaller deployments


git
shared-tasks
update-pipeline-container
prepare
git
dev-container
shared-tasks
lint
test
audit
git
dev-container
git
dev-container
git
shared-tasks
dev-container
build-pp
git
dev-container
shared-tasks
e2e
git
dev-container
shared-tasks
build-p
git

github.com/sirech/talks/raw/master/2019-04-tw-build_pipelines.pdf

THEORY

Not everything on a website can be put behind a shared nginx

Not everything on a website can be put behind a shared nginx

Assumptions

Not everything on a website can be put behind a shared nginx

Assumptions

1. There exists a setup where multiple teams create content for one website

Not everything on a website can be put behind a shared nginx

Assumptions

1. There exists a setup where multiple teams create content for one website
2. There exist at least two teams in the world that don't share the same tool/infrastructure

Not everything on a website can be put behind a shared nginx

Assumptions

1. There exists a setup where multiple teams create content for one website
2. There exist at least two teams in the world that don't share the same tool/infrastructure

Statement

If 1. and 2. are true there exists a number `x` of websites that are build by teams not sharing the same tools/infrastructure, where `x > 0`

Not everything on a website can be put behind a shared nginx

Assumptions

1. There exists a setup where multiple teams create content for one website
2. There exist at least two teams in the world that don't share the same tool/infrastructure

Statement

If 1. and 2. are true there exists a number `x` of websites that are build by teams not sharing the same tools/infrastructure, where `x > 0`

Proof

Not everything on a website can be put behind a shared nginx

Assumptions

1. There exists a setup where multiple teams create content for one website
2. There exist at least two teams in the world that don't share the same tool/infrastructure

Statement

If 1. and 2. are true there exists a number `x` of websites that are build by teams not sharing the same tools/infrastructure, where `x > 0`

Proof

Ads

Microfrontends in the Browser

Microfrontends in the Browser are really old

Mehr zum Thema Covid-19 →

Coronavirus

668 bestätigte Fälle in Deutschland

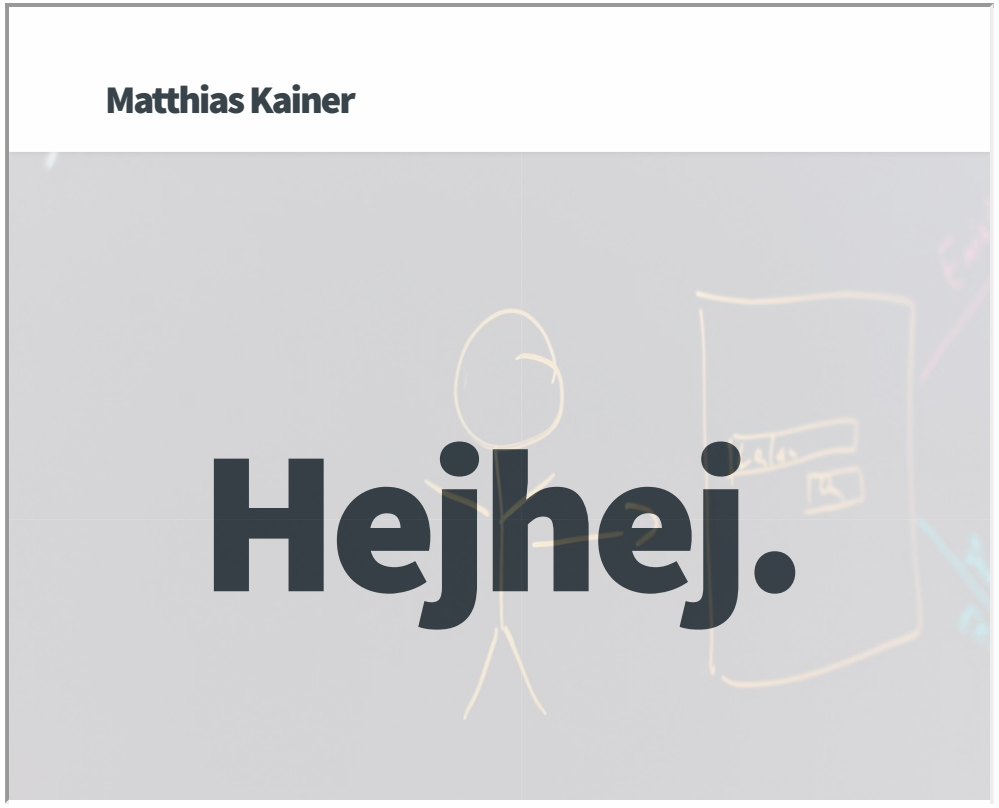
Mehr zum Thema Coronavirus →

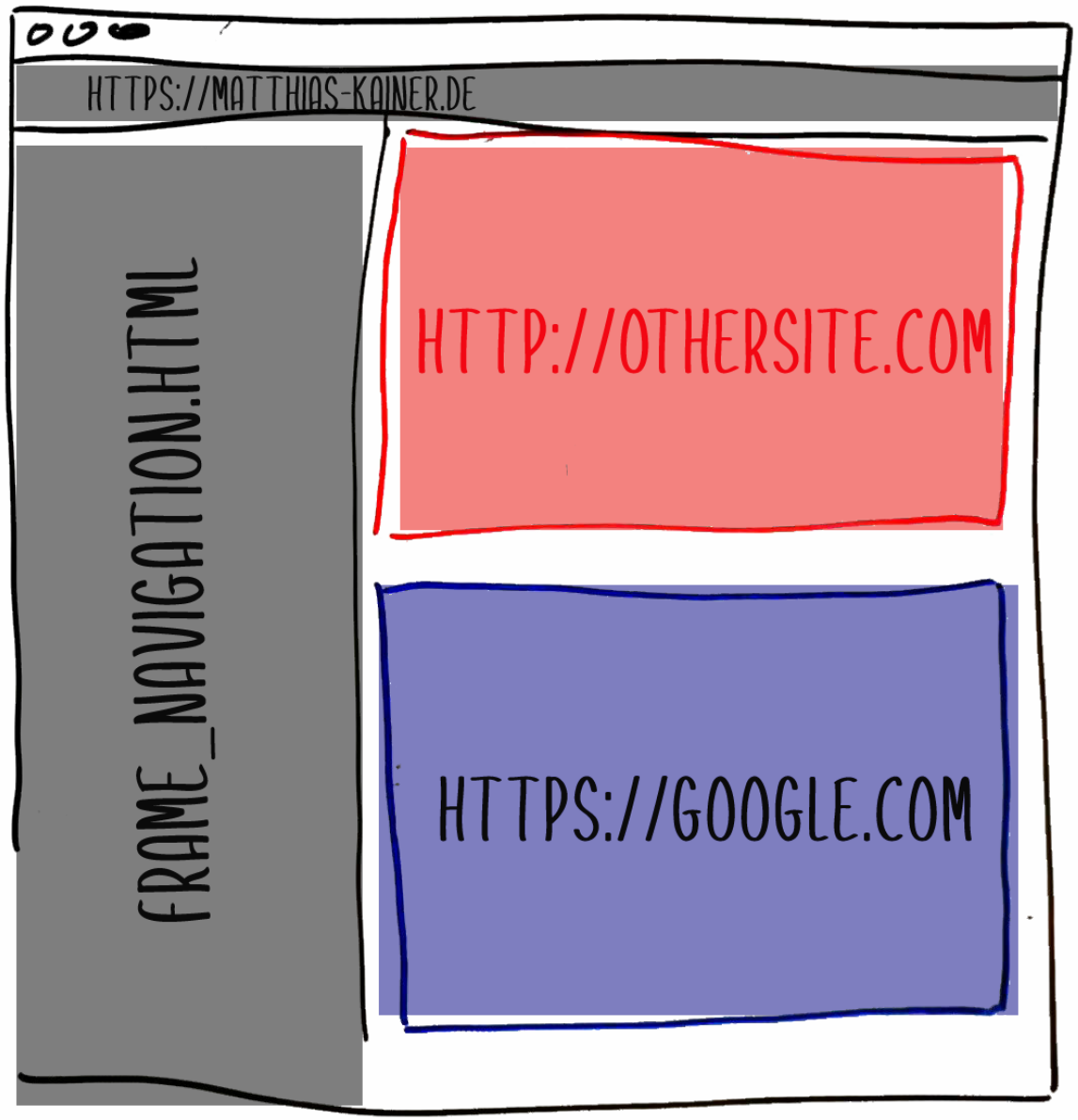
Coronavirus

668 bestätigte Fälle in Deutschland

Microfrontends in the Browser are even older than that

The wonderful world of (i)Frames




HTTPS://MATTHIAS-KAINER.DE
FRAME_NAVIGATION.HTML
HTTP://OTHERSITE.COM
HTTPS://GOOGLE.COM

```
<body>
  <header><iframe src="/parts/shopping-card"></iframe></header>
  <nav><!-- shared navigation --></nav>
  <iframe id="micro-frontend-container"></iframe>

  <script type="text/javascript">
    const {search, pathname} = window.location;
    const microFrontendsByRoute = {
      '/': '/parts/search' + search,
      '/product': '/parts/product' + search
    };
    document.getElementById('micro-frontend-container')
      .src = microFrontendsByRoute[pathname];
  </script>
</body>
```

✔ Very Simple

✔ Very Simple

✔ Isolated

❌ Very Hard To Do Responsive

❌ Very Hard To Do Responsive

❌ Routing can be tricky

❌ Very Hard To Do Responsive

❌ Routing can be tricky

❌ SEO is "interesting"

❌ Very Hard To Do Responsive

❌ Routing can be tricky

❌ SEO is "interesting"

❌ Very Isolated (`window.postMessage` can help)

Detour: Communication in the Browser

```
window.addEventListener("message", (evt) => {
  if (evt.source === window && evt.origin === "https://matthias-kainer.d
    console.log("Doing things with: ", evt.data)
  } else {
    console.log("not trusted")
  }
}, false);

window.postMessage("The secret world of secrets is secret", "https://mat
```

✔ Immune to network lags!

✔ Immune to network lags!

✔ Can be made pretty secure (also against extensions)

✔ Immune to network lags!

✔ Can be made pretty secure (also against extensions)

✔ Very lightweight and can help you to think in events

❌ **Needs some extra love from you to support things like queuing/delivery guarantees**

❌ Needs some extra love from you to support things like queuing/delivery guarantees

❌ If you can't trust yourself to not forget checks, then don't

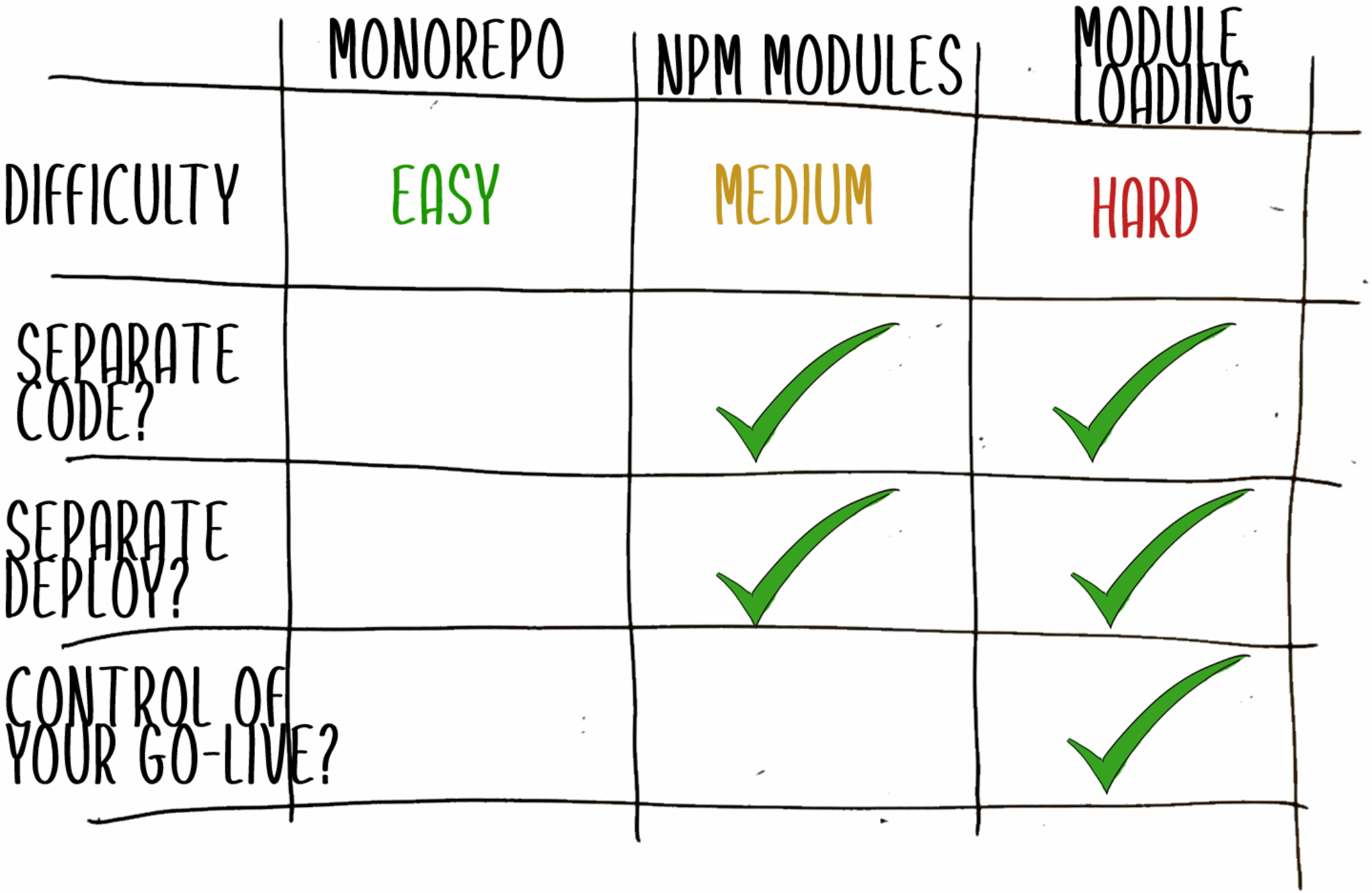
❌ Needs some extra love from you to support things like queuing/delivery guarantees

❌ If you can't trust yourself to not forget checks, then don't

❌ If you can't trust your user, don't rely solely on it

Module Loading

different approaches


MONOREPO
NPM MODULES
MODULE LOADING
DIFFICULTY
EASY
MEDIUM
HARD
SEPARATE CODE?
SEPARATE DEPLOY?
CONTROL OF YOUR GO-LIVE?

Pinning the version for a dependency?

It's a TECH DEBT, friend

single-spa

It's only one example, there are more

DEMO TIME

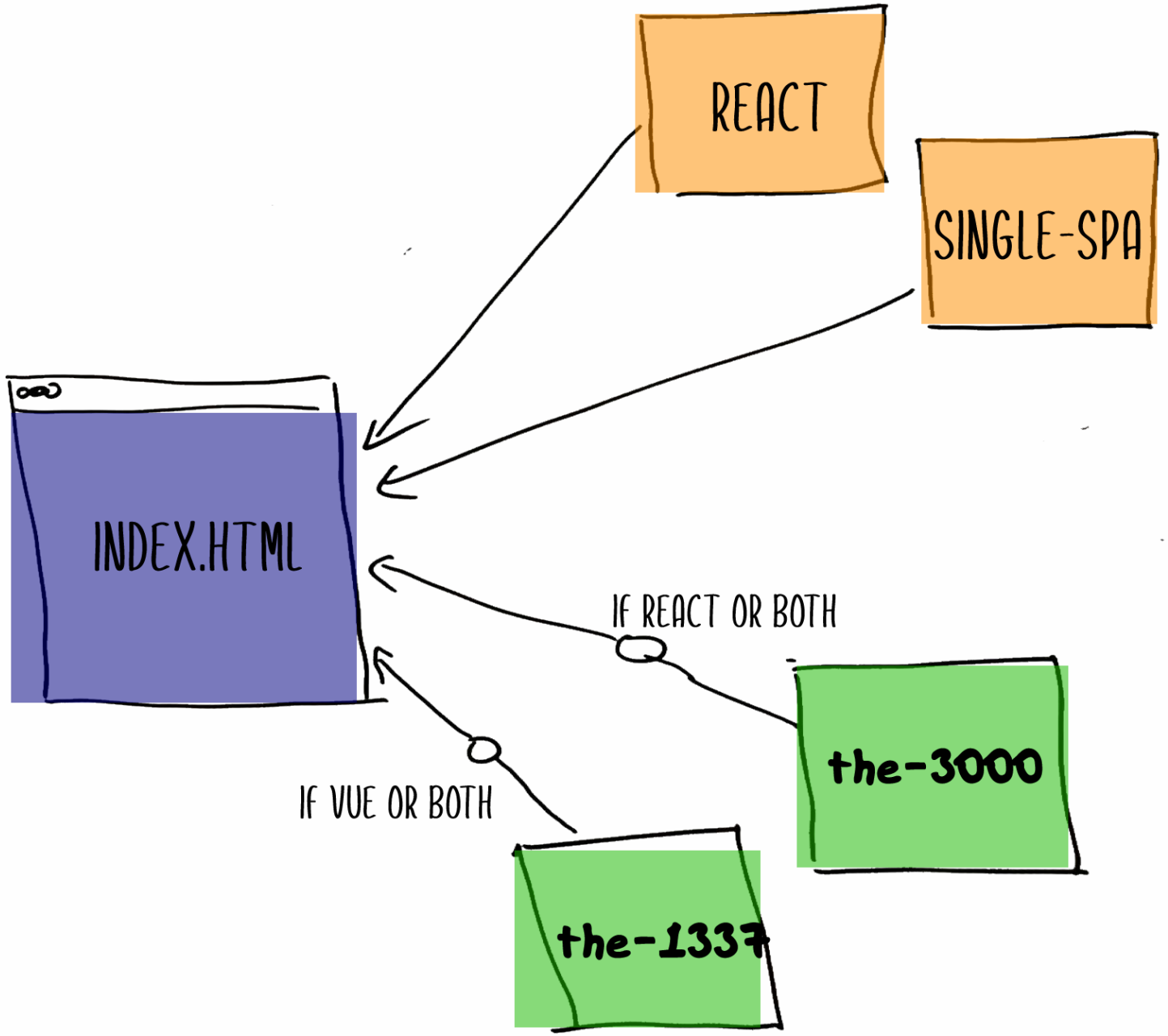
```
System.import("single-spa").then(function(singleSpa) {
  singleSpa.registerApplication(
    "@matthias-kainer/the-3000",
    () => System.import("@matthias-kainer/the-3000"),
    location => location.pathname.startsWith("/react")
      || location.pathname.startsWith("/both")
  );
  singleSpa.registerApplication(
    '@matthias-kainer/the-1337',
    () => System.import('@matthias-kainer/the-1337'),
    location => location.pathname.startsWith('/vue')
      || location.pathname.startsWith("/both")
  );
  singleSpa.start();
});
```

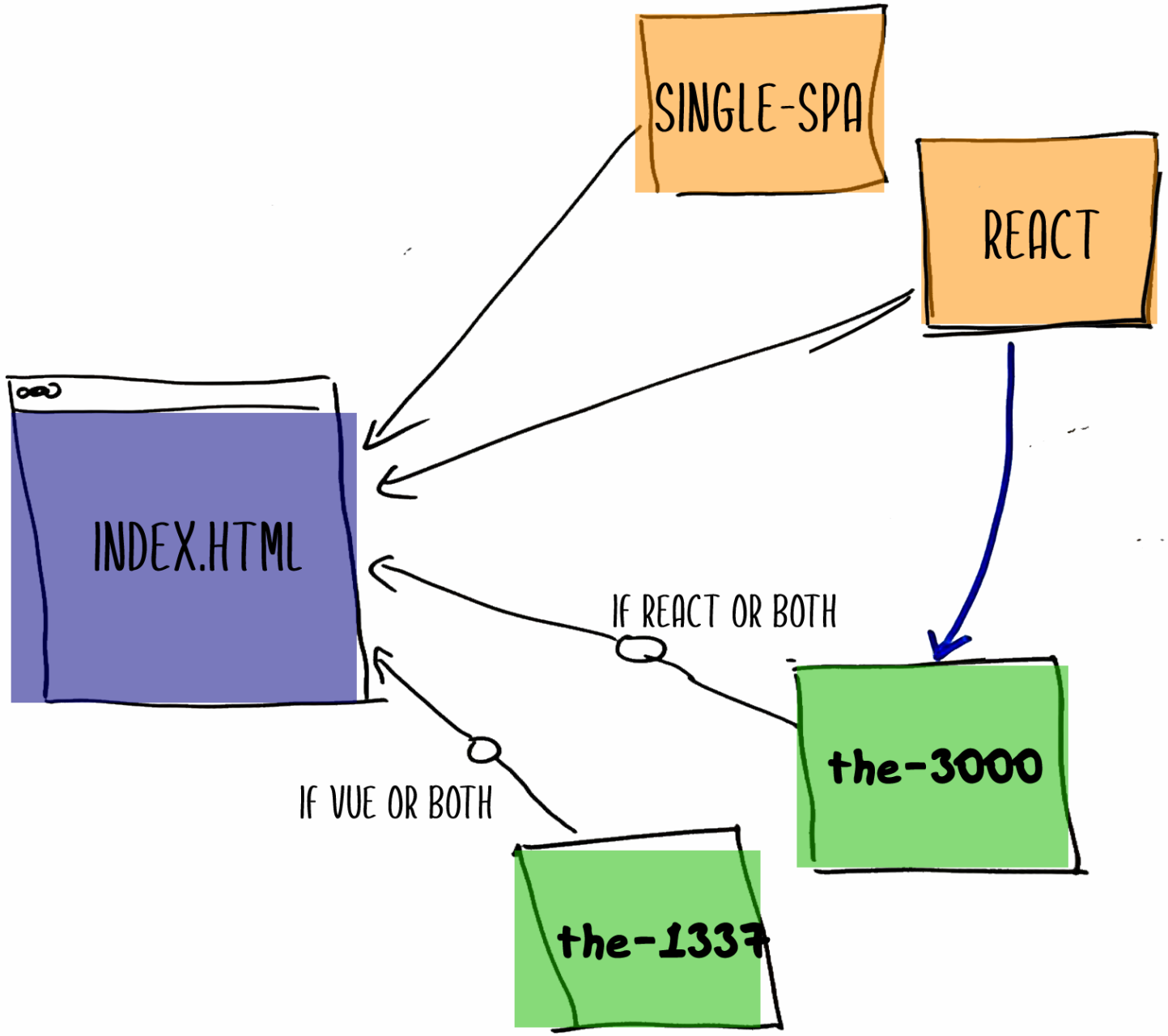
Wait, how does it import?!

Detour: Managing your imports in the browser with import-maps

```
<script type="systemjs-importmap">
  {
    "imports": {
      "react": "https://cdn.jsdelivr.net/npm/react@16.12.0/umd/react.pro
      "react-dom": "https://cdn.jsdelivr.net/npm/react-dom@16.12.0/umd/r
      "single-spa": "https://cdn.jsdelivr.net/npm/single-spa@4.4.1/lib/s
      "@matthias-kainer/the-1337": "https://localhost:1337/js/app.js",
      "@matthias-kainer/the-3000": "https://localhost:3000/matthias-kain
    }
  }
</script>
```

https://wicg.github.io/import-maps/


REACT
SINGLE-SPA
INDEX.HTML
IF REACT OR BOTH
the-3000
IF VUE OR BOTH
the-1337


SINGLE-SPA
REACT
INDEX.HTML
IF REACT OR BOTH
IF VUE OR BOTH
the-3000
the-1337

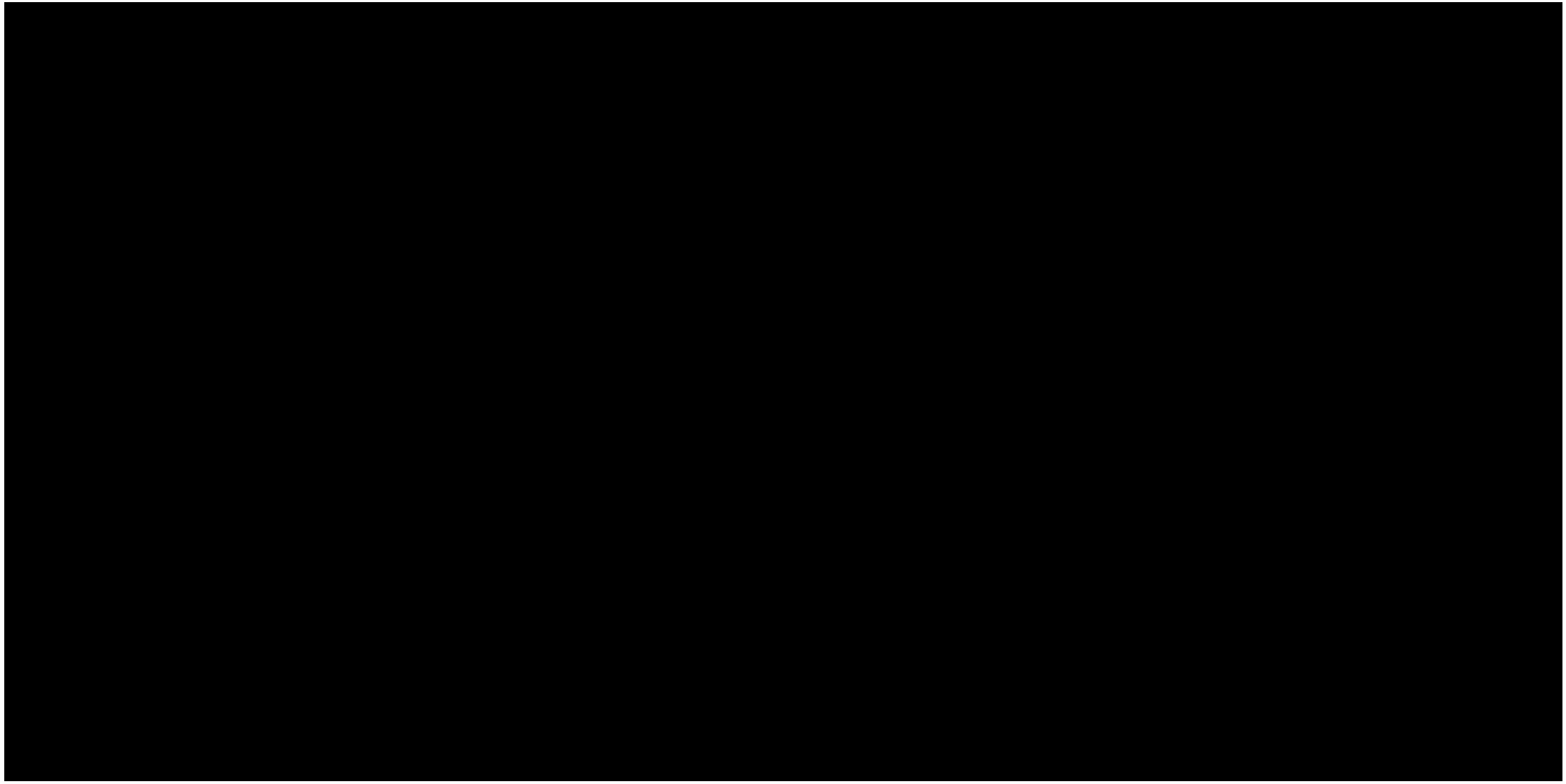
webpack

wait for it...

https://github.com/webpack/webpack/issues/10352

WebComponents

Browser-supported Microfrontend-fragments



```
class ClockDigit extends HTMLElement {
    constructor() { super(); }

    get digit() { return this.getAttribute("digit"); }

    set digit(value) {
        this.setAttribute("digit", value); this._showDigit()
    }

    connectedCallback() { this._showDigit(); }

    _showDigit() {
        this.innerText = this.getAttribute("digit");
    }
}

window.customElements.define(`digital-clock-character`, ClockDigit);
```

✔ Simple and lib-less

✔ Simple and lib-less

✔ As isolated as you need it

✔ Simple and lib-less

✔ As isolated as you need it

✔ Browser support is getting strong

❌ Chatty class-based inflexible structure

❌ Chatty class-based inflexible structure

❌ Tracking attribute values can be hard

❌ Chatty class-based inflexible structure

❌ Tracking attribute values can be hard

❌ A lot of things that would make your life easier (i.e. css styling tweaks, Constructible Stylesheets) still not widely supported or feels heavy/complicated

So...

how do you have it with microfrontends?


YOU MUST BE THIS
TALL TO RIDE
TO DO
MICROFRONTENDS
imgflip.com