
Micro-frontends

2025-02



2020-10



2021-08

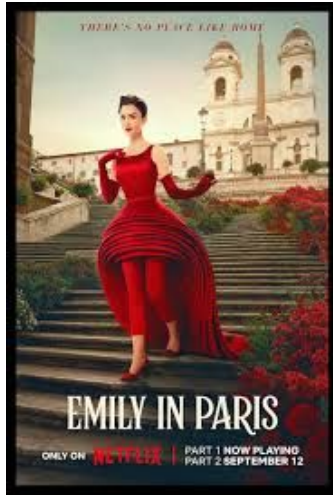


2015-09

Speaker



I am Renato Matos, a full-stack developer with a focus on .NET. In my free time, I enjoy watching Netflix series, reading books, and spending quality time with my two dogs and my lovely child.



What are Micro-frontends?

The term **Micro-frontends** first appeared in [ThoughtWorks Technology Radar](#) at the end of 2016

NOV
2016

*"We saw an approach emerge that our teams call **micro-frontends**. In this approach, a web application is broken into pages and features, **with each feature being owned end-to-end by a single team**"*

NOV
2017

*"This approach continued, the web application was broken down into its features, **with each feature being owned, from frontend to backend, by a different team**"*

NOV
2019

Good frontend development is hard. Scaling frontend development so that many teams can work simultaneously on a large and complex product is even harder.

<https://martinfowler.com/articles/micro-frontends.html>



Cam Jackson

Core ideas behind Micro-frontends

1

The idea behind Micro-frontends is to think about a website or web app as a **composition of features** which are owned by **independent teams**

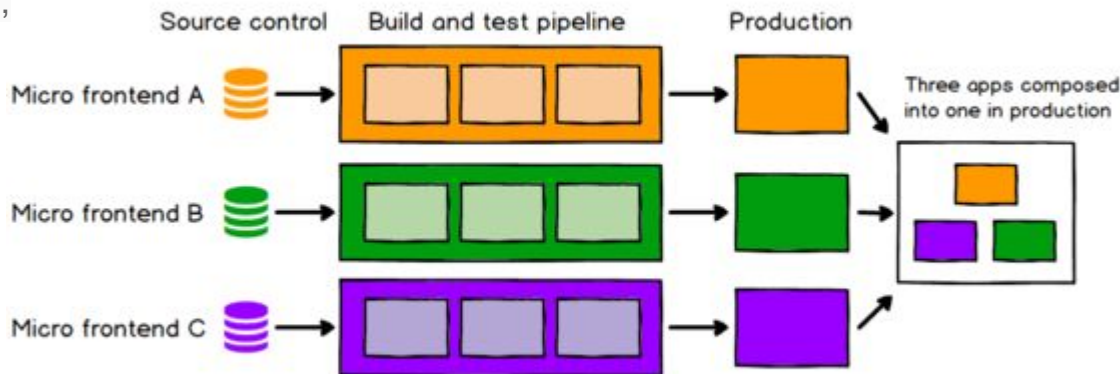
Each team has a **distinct area of business** or **mission** it cares about and specialises in.

2

3

A team is **cross functional** and develops its features **end-to-end**, from database to user interface.

“The idea of Self-contained Systems (SCS) emerged as part of the evolution of microservices architecture in the early 2010s. While microservices themselves were being explored throughout the late 2000s”



Core ideas behind Micro-frontends



1

Be Technology Agnostic

2

Don't share a runtime, even if all teams use the same framework. Build independent apps that are self contained. Don't rely on shared state or global variables.

2

Isolate Team Code

3

Agree on naming conventions where isolation is not possible yet: Namespace CSS, local storage, Events, Cookies

3

Establish Team Prefixes

5

Your feature should be useful, even if JavaScript failed or hasn't executed yet.

4

Favor Native Browser Features over Custom APIs

1

Each team should be able to choose and upgrade their stack without having to coordinate with other teams.

5

Build a Resilient Site

4

Use [Browser Events for communication](#) instead of building a global PubSub system. If you really have to build a cross-team API, try keeping it as simple as possible.

Module Federation

<https://module-federation.io>



Module Federation is an architectural pattern for the decentralization of JavaScript applications. It allows us to share code and resources among multiple JavaScript applications (or micro-frontends).



Code Sharing



Modularization



Rspack

<https://rspack.dev/>



Development Experience



Flexibility



Plugin System

Module Federation Plugins

<https://module-federation.io>



ModuleFederationPlugin

ContainerReferencePlugin

(with some remotes)

App1: ContainerPlugin (exposing)

App2: ContainerPlugin (exposing)

App3: ContainerPlugin (exposing)

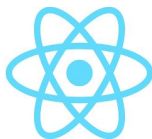


02:08

<https://youtu.be/8yt0isRJGBc?t=128>

<https://github.com/renatomatos79/livro-webpack>

Micro-frontends Demo



03:06

https://youtu.be/36q4Vz_GsmM?t=186

<https://github.com/renatomatos79/livro-webpack>

Single-SPA

<https://single-spa.js.org/>



Single-spa is a framework for bringing together multiple JavaScript Micro-frontends in a frontend application.

A [single-spa root config](#), renders the HTML page *and* the JavaScript that registers applications.

Each application is registered with three things:

1. A name
2. A function to load the application's code
3. A function that determines when the application is active/inactive

[Applications](#) which can be thought of as single-page applications packaged up into modules. Each application must know how to bootstrap, mount, and unmount itself from the DOM

<https://github.com/renatomatos79/livro-webpack>

Single-SPA

<https://single-spa.js.org/docs/create-single-spa/>



Different ways of doing the same thing

```
create-single-spa --framework react
npm init single-spa --framework react
npx create-single-spa --framework react
yarn create single-spa --framework react
```

--moduleType

```
create-single-spa --moduleType root-config
create-single-spa --moduleType app-parcel
create-single-spa --moduleType util-module
```

--moduleFormat

```
create-single-spa --moduleFormat esm
create-single-spa --moduleFormat systemjs
```

--framework

```
create-single-spa --framework react
create-single-spa --framework vue
create-single-spa --framework angular
```

--layout (For root config)

<https://single-spa.js.org/docs/layout-overview/>

<https://github.com/renatomatos79/livro-webpack>

Single-SPA

<https://single-spa.js.org/docs/ecosystem-react/>

Single-spa is a framework for bringing together multiple JavaScript Micro-frontends in a frontend application.

- [single-spa-react](#)
- [single-spa-vue](#)
- [single-spa-angular](#)
- [single-spa-angularjs](#)
- [single-spa-cycle](#)
- [single-spa-ember](#)
- [single-spa-inferno](#)
- [single-spa-preact](#)
- [single-spa-svelte](#)
- [single-spa-riot](#)
- [single-spa-backbone](#)
- [single-spa-dojo](#)
- [single-spa-alpinejs](#)



<https://www.youtube.com/watch?v=L4jqow7NTVg>

<https://github.com/renatomatos79/livro-webpack>

Single-SPA Setup



In order to build a single-spa project that supports micro-frontends, please follow this tutorial right below:

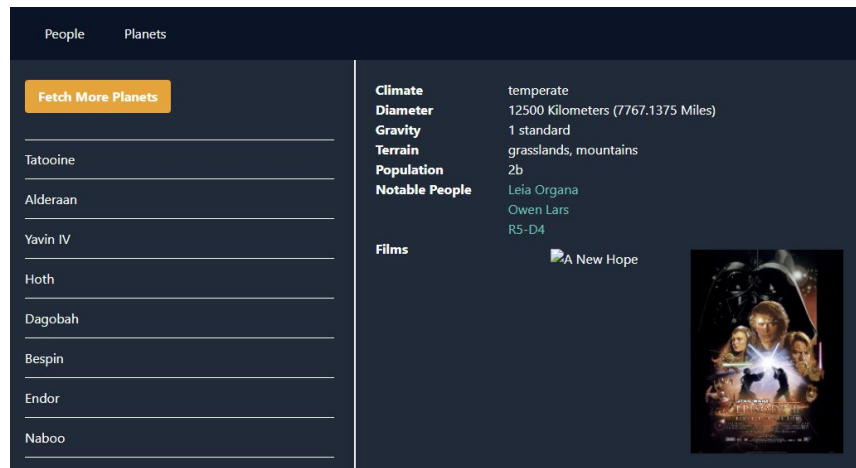
- <https://single-spa.js.org/docs/getting-started-overview/>
- <https://single-spa.js.org/docs/recommended-setup/>

And here we can find the repository related to these projects:

- <https://github.com/orgs/react-microfrontends/repositories>

And finally, the demo project loading every module together SPA

- <https://react.microfrontends.app>





<http://github.com/renatomatos79/livro-webpack>



<https://www.udemy.com/course/microfrontend-na-pratica>