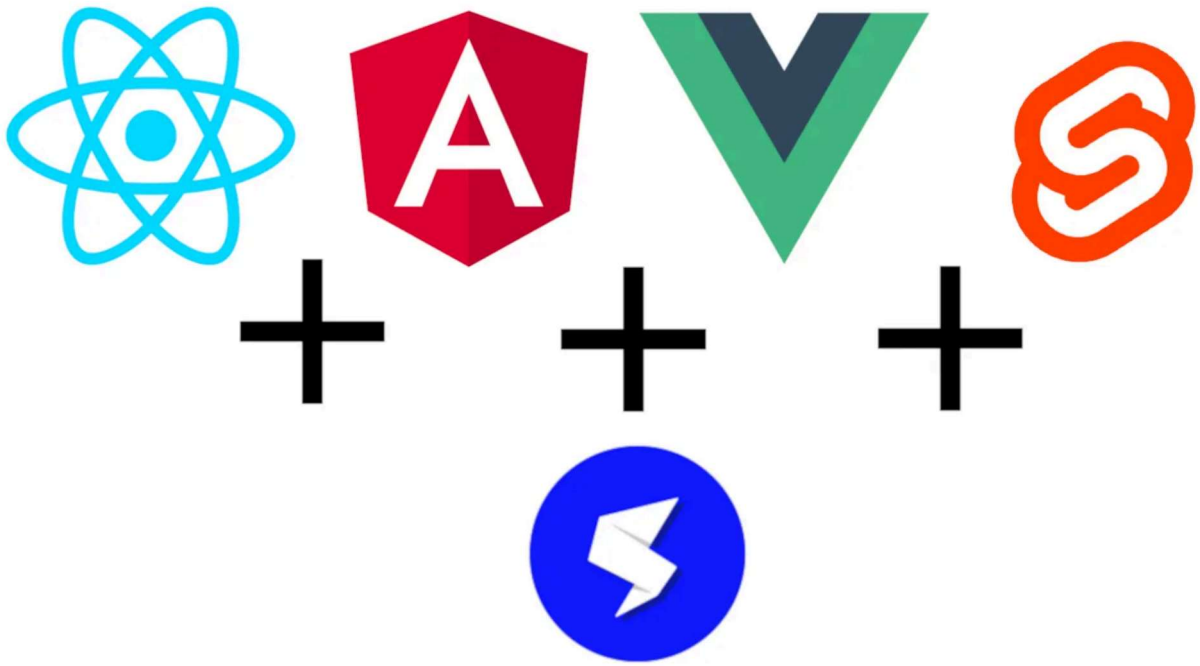
Connect Micro frontends with the Single-Spa framework. Step by step guide.

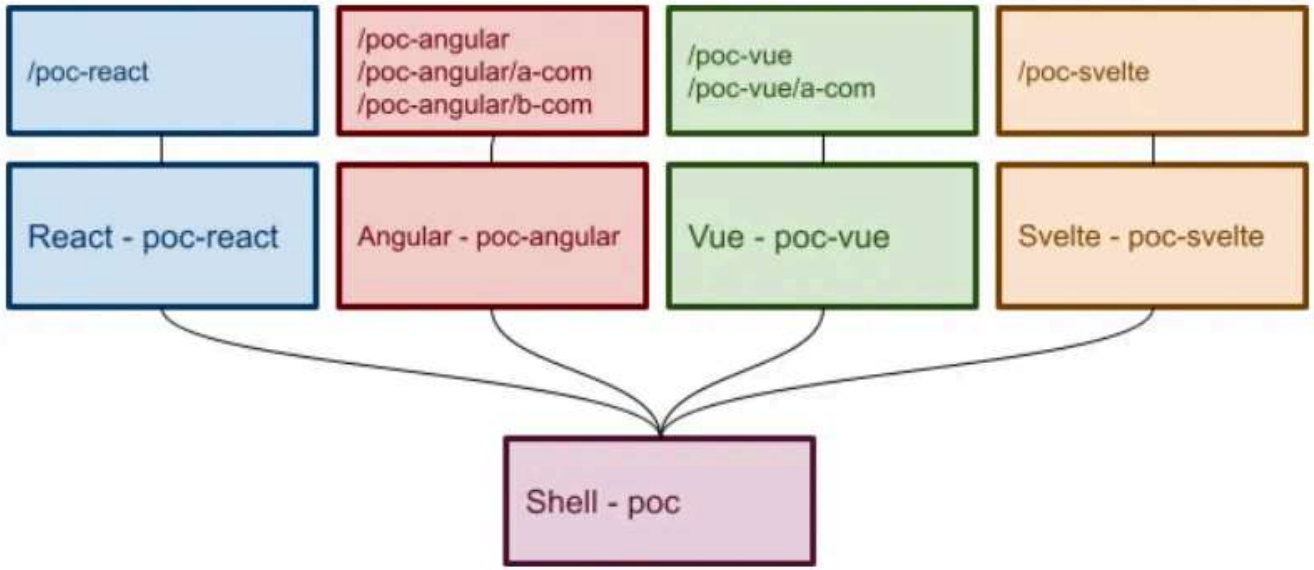
Oleh Baranovskyi · Follow

7 min read · Aug 13, 2021



In this article, we are going to set up the shell/layout project which will be using the Single-Spa as a glue to connect multiple frameworks, such as React, Angular, Vue, and Svelte. We will try to keep it simple as much as it's possible, to focus rather on the composition part than on the styling.

Our projects architecture is going to have the following structure:



Let's dig right into implementation.

Setup Shell (layout) app

First, we need to create a shell app that will be the glue for the rest. This app will be responsible for connecting all the frameworks that we will use in the current article.

1. Create a folder for the shell app:

```
mkdir poc && cd poc
```

2. Setup project:

```
create-single-spa --layout
```

and choose:

- Directory for new project — `.`
- Select type to generate — `single-spa root config`
- Which package manager do you want to use? — `npm`
- Will this project use TyeScript — `y`
- Organization name — `obaranovskyi`

3. Install dependencies:

```
npm install
```

4. Open ***src/microfrontend-layout.html***, and remove the following line:

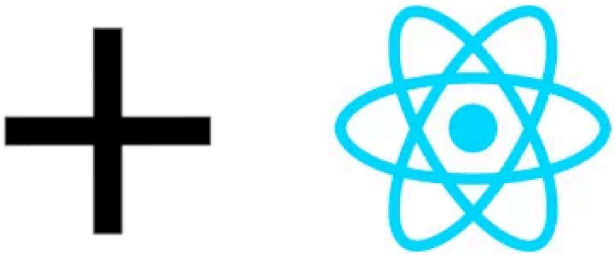
```
<application name="@single-spa/welcome"></application>
```

5. Open *src/index.ejs*, and remove this line:

```
"@single-spa/welcome": "https://unpkg.com/single-spa-
welcome/dist/single-spa-welcome.js",
```

6. Start the project, which will run on *http://localhost:9000*:

```
npm start
```



1. Create a folder for the React application:

```
mkdir poc-react && cd poc-react
```

2. Setup React project with the Single-Spa CLI:

```
create-single-spa --framework react
```

and choose:

- Directory for new project — `.`
- Which package manager do you want to use? — `npm`
- Will this project use TypeScript — `y`
- Organization name — `obaranovskyi`
- Project name — `poc-react`

3. Install dependencies:

```
npm install
```

4. Start the project:

```
npm start -- --port 8500
```

The Shell project updates:

5. Update *src/index.ejs* file import map:

```
...
<script type="systemjs-importmap">
    {
      "imports": {
        "react":
"https://cdn.jsdelivr.net/npm/react@16.13.1/umd/react.production.min.
js",
        "react-dom": "https://cdn.jsdelivr.net/npm/react-
dom@16.13.1/umd/react-dom.production.min.js",
        "@obaranovskyi/root-config": "//localhost:9000/obaranovskyi-
root-config.js",
        "@obaranovskyi/poc-react": "//localhost:8500/obaranovskyi-
poc-react.js"
      }
    }
</script>
...
```

6. Update *src/obaranovskyi-root-config.ts:*

```
...
registerApplication(
  "@obaranovskyi/poc-react",
  () => System.import("@obaranovskyi/poc-react"),
  (location) => location.pathname === "/poc-react"
);

...
```

7. Update *src/microfrontend-layout.html*:

```
...

<main>
    <route default>
      <ul>
          <li>
            <a href="/">Home</a>
          </li>
          <li>
            <a href="./poc-react">React</a>
          </li>
      </ul>
    </route>
</main>

...
```

Here we will also add child routes to prove that everything works as expected, including the lazy loading module.

1. Create an Angular app using angular CLI:

```
ng new poc-angular
```

and choose:

- Do you want to enforce stricter type checking and stricter bundle budgets in the workspace? — `yes`
- Would you like to add angular routing? — `y`
- Which stylesheet format would you like to use? — `scss`

2. Install dependencies:

```
cd poc-angular && npm install
```

3. Add Single-Spa to the project with all configurations:

```
ng add single-spa-angular
```

and choose:

- Does your application use angular routing? — `y`
- Does your application use the BrowserAnimationModule — `y`

4. Generate component B:

```
ng generate component b-com
```


5. Generate component A with the module:

```
ng generate component a-com
ng generate module a-com
```

6. Add *src/app/a-com/a-com-routing.module.ts*:

```
import { Routes, RouterModule } from '@angular/router';
import { AComComponent } from './a-com.component';
const routes: Routes = [
  {
    path: '',
    component: AComComponent
  },
];
export const AComRoutingModule = RouterModule.forChild(routes);
```

7. Update *src/app/a-com/a-com.module.ts*:

```
import { NgModule } from '@angular/core';

import { AComRoutingModule } from './a-com-routing.module';
import { AComComponent } from './a-com.component';

@NgModule({
  imports: [AComRoutingModule],
  declarations: [AComComponent],
})
export class AComModule {}
```

8. Update *src/app/app-routing.module.ts*:

```
import { RouterModule, Routes } from '@angular/router';

import { BComComponent } from './b-com/b-com.component';

const routes: Routes = [
  {
    path: 'b-com',
    component: BComComponent,
  },
  {
    path: 'a-com',
    loadChildren: () =>
      import('./a-com/a-com.module').then((m) => m.AComModule),
  },
];
```

```
export const AppRoutingModule = RouterModule.forRoot(routes, {
  relativeLinkResolution: 'legacy',
});
```

9. Update *src/app/app.module.html*:

```
Angular works!

<ul>
    <li>
        <a href="" routerLink="./a-com">A component</a>
    </li>
    <li>
        <a href="" routerLink="./b-com">B component</a>
    </li>
</ul>

<router-outlet></router-outlet>
```

10. Update *src/app/app.module.ts*:

```
import { APP_BASE_HREF } from '@angular/common';
import { NgModule } from '@angular/core';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
```

```
@NgModule({
  declarations: [AppComponent],
  imports: [BrowserAnimationsModule, AppRoutingModule],
  providers: [{ provide: APP_BASE_HREF, useValue: '/poc-angular' }],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

11. Remove *src/app/empty-route* folder;

12. Install `systemjs-webpack-interop` module:

```
npm install systemjs-webpack-interop -S
```

13. Add *src/set-public-path.js*:

```
import { setPublicPath } from "systemjs-webpack-interop";

setPublicPath("@obaranovskyi/poc-angular");
```

14. In *src/main.single-spa.ts* add the line bellow as the first line:

```
import './set-public-path';
```

15. Install dependencies (again):

```
npm install
```

16. Start the project:

```
npm run serve:single-spa:poc-angular
```

The Shell project updates:

16. Add *zone.js* script to the ***src/index.ejs***:

```
...
<script src="https://unpkg.com/zone.js"></script>
...
```

17. Update import map in the ***src/index.ejs***:

```
...
<script type="systemjs-importmap">
    {
      "imports": {
        "react":
"https://cdn.jsdelivr.net/npm/react@16.13.1/umd/react.production.min.
js",
        "react-dom": "https://cdn.jsdelivr.net/npm/react-
dom@16.13.1/umd/react-dom.production.min.js",
        "@obaranovskyi/root-config": "//localhost:9000/obaranovskyi-
root-config.js",
        "@obaranovskyi/poc-react": "//localhost:8500/obaranovskyi-
poc-react.js",
        "@obaranovskyi/poc-angular": "//localhost:4200/main.js"
      }
    }
  </script>
...
```

18. Register the app in the *src/obaranovskyi-root-config.ts*:

```
...
registerApplication(
  "@obaranovskyi/poc-angular",
  () => System.import("@obaranovskyi/poc-angular"),
  (location) => {
    return location.pathname.startsWith("/poc-angular");
  }
);
...
```

19. Update menu in the *src/microfrontend-layout.html*:

```
...
<main>
    <route default>
      <ul>
          <li>
            <a href="/">Home</a>
          </li>
          <li>
            <a href="/poc-react">React</a>
          </li>
          <li>
            <a href="/poc-angular">Angular</a>
          </li>
      </ul>
    </route>
  </main>
...
```

1. Create a folder for the project:

```
mkdir poc-vue && cd poc-vue
```

2. Setup using Single-Spa CLI:

```
create-single-spa --framework vue
```

and choose:

- Directory for new project — .
- Organization name — `obaranovskyi`
- Target directory .. already exists. Pick an action — `Overwrite`
- Please pick a preset — `Default ([Vue 3] babel, eslint)`

3. Rename ***src/components/HelloWorld.vue*** to ***src/components/ACom.vue***:

4. Update **src/components/ACom.vue**:

```
<template>
  <div>
      A component works!
  </div>
</template>

<script>
export default {
  name: 'ACom'
}
</script>

<style scoped>
</style>
```

5. Add *src/router/index.js*:

```
import { createRouter, createWebHistory } from "vue-router";
import ACom from "../components/ACom.vue";

const routes = [
  {
    path: "/poc-vue/a-com",
    name: "ACom",
    component: ACom
  }
];

const router = createRouter({
  history: createWebHistory(process.env.BASE_URL),
  routes
});

export default router;
```

6. Update **src/main.js**:

```
import "./set-public-path";
import { h, createApp } from "vue";
import singleSpaVue from "single-spa-vue";

import App from "./App.vue";
import router from "./router";

const vueLifecycles = singleSpaVue({
  createApp,
  appOptions: {
```

```
    render() {
      return h(App, {
        name: this.name,
        mountParcel: this.mountParcel,
        singleSpa: this.singleSpa
      });
    }
  },
  handleInstance: app => {
    app.use(router);
  }
});

export const bootstrap = vueLifecycles.bootstrap;
export const mount = vueLifecycles.mount;
export const unmount = vueLifecycles.unmount;
```

7. Update **src/App.vue**:

```
<template>
  <div id="nav">
    <router-link to="/poc-vue">Home</router-link> |
    <router-link to="/poc-vue/a-com">A component</router-link>
  </div>
  <router-view/>
</template>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
```

```
  color: #2c3e50;
}

#nav {
  padding: 30px;
}

#nav a {
  font-weight: bold;
  color: #2c3e50;
}

#nav a.router-link-exact-active {
  color: #42b983;
}
</style>
```

8. Add **src/set-public-path.js**:

```
import { setPublicPath } from "systemjs-webpack-interop";

setPublicPath("@obaranovskyi/poc-vue");
```

9. Install `systemjs-webpack-interop` module:

```
npm install systemjs-webpack-interop -S
```

10. Install `vue-router`, (we need `^4.0.0-0` and greater)

```
npm install vue-router
```

Note: alternatively, you can put `"vue-router": "^4.0.0-0",` into your *dependencies* in the ***package.json*** and run in the command line `npm install`.

11. Start the project:

```
npm run serve
```

The Shell project updates:

12. Update import map in the ***src/index.ejs:***

```
...
<script type="systemjs-importmap">
    {
      "imports": {
        "react":
"https://cdn.jsdelivr.net/npm/react@16.13.1/umd/react.production.min.
js",
        "react-dom": "https://cdn.jsdelivr.net/npm/react-
```

```
dom@16.13.1/umd/react-dom.production.min.js",
        "vue":
"https://cdn.jsdelivr.net/npm/vue@2.6.11/dist/vue.min.js",
        "vue-router": "https://cdn.jsdelivr.net/npm/vue-
router@3.1.6/dist/vue-router.min.js",
        "@obaranovskyi/root-config": "//localhost:9000/obaranovskyi-
root-config.js",
        "@obaranovskyi/poc-react": "//localhost:8500/obaranovskyi-
poc-react.js",
        "@obaranovskyi/poc-angular": "//localhost:4200/main.js",
        "@obaranovskyi/poc-vue": "//localhost:8080/js/app.js"
      }
    }
</script>
...
```

13. Register app in the *src/obaranovskyi-root-config.ts*:

```
...
registerApplication(
  "@obaranovskyi/poc-vue",
  () => System.import("@obaranovskyi/poc-vue"),
  (location) => {
    return location.pathname.startsWith("/poc-vue");
  }
);
...
```

14. Update *src/microfrontend-layout-.html*:

```
...
<main>
    <route default>
      <ul>
          <li>
            <a href="/">Home</a>
          </li>
          <li>
            <a href="/poc-react">React</a>
          </li>
          <li>
            <a href="/poc-angular">Angular</a>
          </li>
          <li>
            <a href="/poc-vue">Vue</a>
          </li>
      </ul>
    </route>
</main>
...
```

1. Create a folder for the project:

```
mkdir poc-svelte && cd poc-svelte
```

2. Setup project using Single-Spa CLI:

```
create-single-spa --framework svelte
```

and choose:

- Directory for new project — `.`
- Which package manager do you want to use? — `npm`
- Organization name — `obaranovskyi`
- Project name — `poc-svelte`

3. Install dependencies:

```
npm install
```

4. Start the project:

```
npm start
```

The Shell project updates:

5. Update import maps in the *src/index.ejs*:

```
...
<script type="systemjs-importmap">
    {
      "imports": {
        "react":
"https://cdn.jsdelivr.net/npm/react@16.13.1/umd/react.production.min.
js",
        "react-dom": "https://cdn.jsdelivr.net/npm/react-
dom@16.13.1/umd/react-dom.production.min.js",
        "vue":
"https://cdn.jsdelivr.net/npm/vue@2.6.11/dist/vue.min.js",
        "vue-router": "https://cdn.jsdelivr.net/npm/vue-
router@3.1.6/dist/vue-router.min.js",
        "@obaranovskyi/root-config": "//localhost:9000/obaranovskyi-
root-config.js",
        "@obaranovskyi/poc-react": "//localhost:8500/obaranovskyi-
poc-react.js",
        "@obaranovskyi/poc-angular": "//localhost:4200/main.js",
        "@obaranovskyi/poc-vue": "//localhost:8080/js/app.js",
        "@obaranovskyi/poc-svelte": "//localhost:5000/obaranovskyi-
poc-svelte.js"
      }
    }
</script>
...
```

6. Register app in the *src/obaranovskyi-root-config.ts*:

```
...
registerApplication(
  "@obaranovskyi/poc-svelte",
  () => System.import("@obaranovskyi/poc-svelte"),
```

```
  (location) => {
    return location.pathname.startsWith("/poc-svelte");
  }
);
...
```

7. Update menu content in the *src/microfrontend-layout.html*:

```
...
<main>
    <route default>
      <ul>
          <li>
            <a href="/">Home</a>
          </li>
          <li>
            <a href="/poc-react">React</a>
          </li>
          <li>
            <a href="/poc-angular">Angular</a>
          </li>
          <li>
            <a href="/poc-vue">Vue</a>
          </li>
          <li>
            <a href="/poc-svelte">Svelte</a>
          </li>
      </ul>
    </route>
  </main>
...
```

Conclusion

Thank you guys for reading. I hope you enjoyed it and learned some new stuff related to JavaScript. Please subscribe and press the ‘Clap’ button if you like this article.

Single Spa | Angular | Micro Frontends | React | Vue

Written by Oleh Baranovskyi

646 Followers · 74 Following

Frontend Lead & Architect | Web community manager https://obaranovskyi.com/

Follow

Responses (2)

What are your thoughts?

Respond

Nohemi Martinez
almost 2 years ago

```
registerApplication(
"@obaranovskyi/poc-vue",
() => System.import("@obaranovskyi/poc-vue"),
(location) => {
return location.pathname.startsWith("/poc-vue");
}
);
```

Hi this line shows me an error) => System.import("@obaranovskyi/poc-vue"),

Argument of type '() => Promise<System.Module>' is not assignable to parameter of type 'Application<{}>'.

Type '() => Promise<System.Module>' is not assignable to type.....

Read More

1 1 reply Reply

Hamza Zaidi
about 2 years ago

Single spa is great but there is lack of Angular support

Reply

More from Oleh Baranovskyi

In JavaScript in Plain English by Oleh Baranovskyi

How do you clean RxJS Subjects?

Reset cache with the ResettableSubject

Dec 16, 2021 481

Oleh Baranovskyi

This is the simplest way to reload data using RxJS

Most of the time, we have to load data from the server. To perform the action client...

Dec 1, 2021 536 3

Oleh Baranovskyi

The ultimate guide to Observables and/vs Promises (+RxJS7).

Oleh Baranovskyi

Top 8 functions you'll ever need to work with Enums in TypeScript

During this article, you'll get a full insight into how promises differ with regard to...

Dec 11, 2021 445

Discover helper functions to simplify work with enums

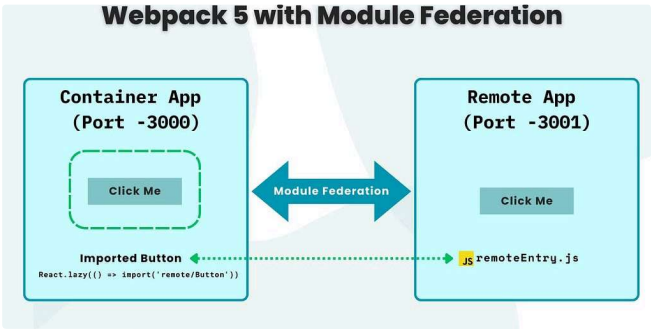
Jan 6, 2022 484 5

See all from Oleh Baranovskyi

Recommended from Medium

Priyabrata Saha

Mastering Angular: Best Practices to Code Like a Pro in 2025



In The Syntax Diaries by Amaresh Adak

Micro-Frontends with Webpack 5: A Complete Guide

As Angular continues evolving, coding practices must adapt to align with its latest...

Dec 10 89 3

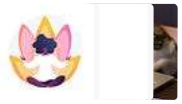
Scale and Simplify Your App with Micro-Frontends and Webpack

Oct 4 572

Lists

General Coding Knowledge

20 stories · 1825 saves



Stories to Help You Grow as a Software Developer

19 stories · 1527 saves

Medium's Huge List of Publications Accepting...

377 stories · 4171 saves

Natural Language Processing

1871 stories · 1496 saves

In Dev Genius by Karthik Seenuvasan

In ITNEXT by Maksim Dolgikh

Angular 19. Trying to stay afloat

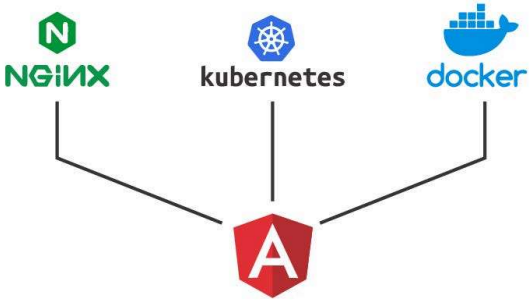
Build and Deploy Using Bitbucket Pipelines ⭐

Before you begin setting up Bitbucket pipelines, you have to ensure that you alread...

Sep 11 10

The story of how, in the pursuit of all things Angular has already stopped realizing what i...

Dec 8 1.1K 11

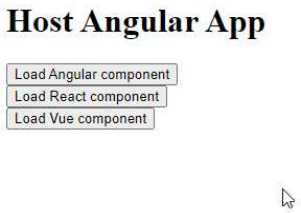


Priyabrata Saha

Deploying Angular Apps: From Development to Production

Deploying an Angular application is an essential step that transforms your...

Dec 2 52



Joe Moceri

Using Module Federation to work with Angular, React, and Vue as...

Links

Jul 2 3

See more recommendations