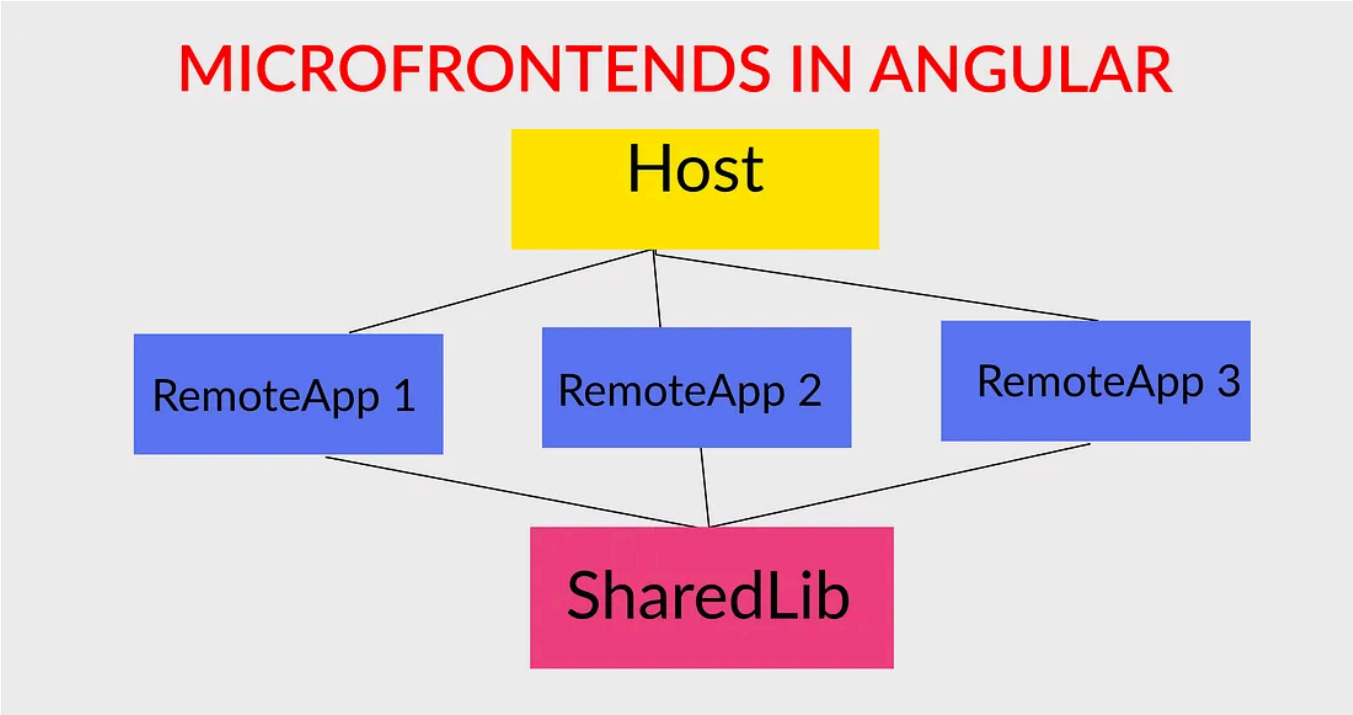
Get unlimited access to the best of Medium for less than $1/week. Become a member

Implementing MicroFrontends in Angular 15

Gurunadh Pukkalla · Follow

4 min read · Mar 1, 2023

Listen Share More



Angular MicroFrontends Architecture

What is MicroFrontends ?

MicroFrontends is an architectural style for building web applications by breaking them down into smaller, independent, and loosely coupled modules. Each module, or MicroFrontend, is developed and deployed independently, with its own set of dependencies, technologies, and teams.

Implementing MicroFrontend in Angular 15 involves breaking down the application into smaller parts that can be developed and deployed independently. Each MicroFrontend should have its own Angular application, with its own set of components, services, and modules.

Key steps to implementing MicroFrontend in Angular 15:

1. Break down the application into smaller, independent parts based on business domain or functionality.
2. Create an Angular application for each MicroFrontend, with its own set of components, services, and modules.
3. Define an API contract for communication between MicroFrontends. This could be RESTful APIs, WebSockets, or custom events.
4. Use a module federation plugin to dynamically load MicroFrontends into the main application at runtime. Module federation is a feature in Webpack that allows multiple applications to share code at runtime.
5. Implement a shared module that contains common code and components that can be used by multiple MicroFrontends.
6. Use a centralized state management system, such as Redux or NgRx, to manage application state across MicroFrontends.
7. Implement a consistent UI design system across MicroFrontends to ensure a cohesive user experience.

By following these steps, you can build a scalable, modular, and flexible web application using MicroFrontends and Angular 15.

MicroFrontends in Angular using Module Federation

HOST Application Configuration

1. Creating a Host Application

```
ng new angular-mfe-example --create-application="false"
```

The above command will create a workspace with no projects.

2. Create host app (Main Application)

It is Main Application where we are getting all remote applications and configure the routing for them.

```
ng g application host
```

3. Create a component example : admin

```
gurun@MSI MINGW64 ~/OneDrive/Desktop/Personal/poc/angular-mfe-example (master)
$ ng g c admin
CREATE projects/host/src/app/admin/admin.component.html (20 bytes)
CREATE projects/host/src/app/admin/admin.component.spec.ts (592 bytes)
CREATE projects/host/src/app/admin/admin.component.ts (272 bytes)
CREATE projects/host/src/app/admin/admin.component.scss (0 bytes)
UPDATE projects/host/src/app/app.module.ts (447 bytes)
```

creating admin component

4. Update Route to add path to admin and change AppComponent

Add Route to *app-routing.module.ts*

```
const routes: Routes = [
  {
    path: '',
    component: AdminComponent,
```

Open in app ↗

Medium Search

```
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
})
```

app-routing.module.ts

Clean up *app.component.html* and add below code

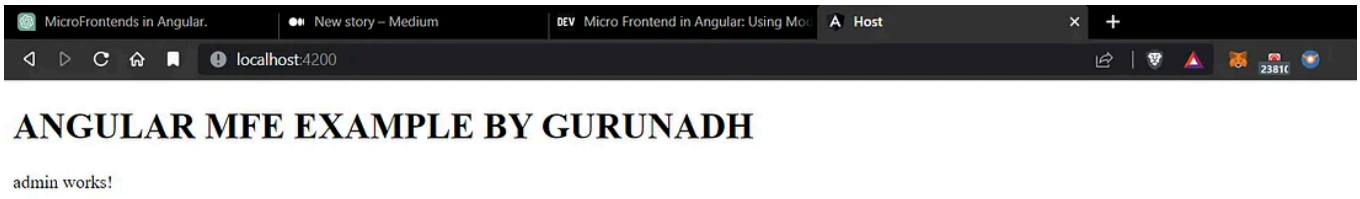
```
Go to component
1  <h1>ANGULAR MFE EXAMPLE BY GURUNADH</h1>
2  <router-outlet></router-outlet>
```

app.component.html

5. Run the application

```
ng serve host
```

Run the host app. It should run in default port 4200



Serving Host on 4200 Pic

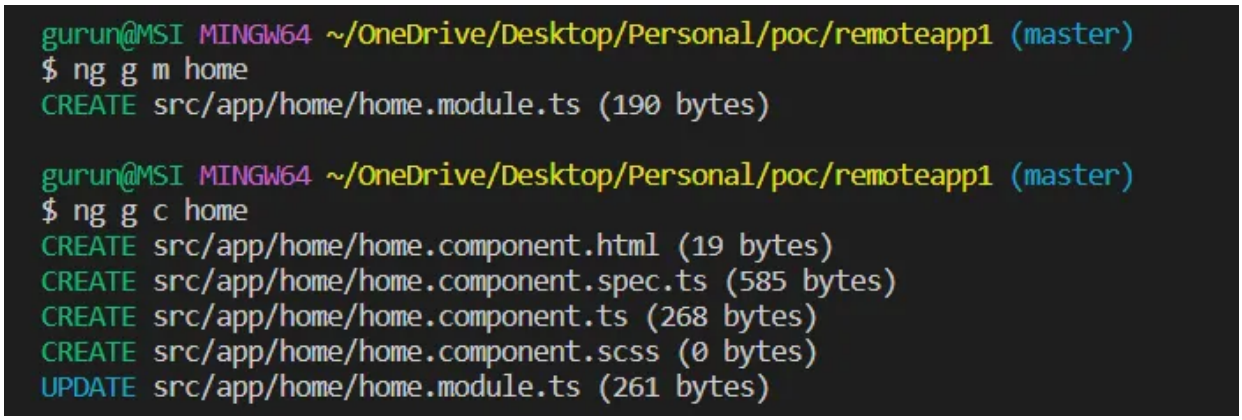
Creating a Remote Applications and Configurations

1. Create a remote application 1 and remote application 2

```
ng new remoteapp1
ng new remoteapp2
```

2. Create a module home in both applications (remoteapp1 & remoteapp2)

```
ng g m home
ng g c home
```



adding home module and component to remoteapp1

```
gurun@MSI MINGW64 ~/OneDrive/Desktop/Personal/poc/remoteapp2 (master)
$ ng g m home
CREATE src/app/home/home.module.ts (190 bytes)

gurun@MSI MINGW64 ~/OneDrive/Desktop/Personal/poc/remoteapp2 (master)
$ ng g c home
CREATE src/app/home/home.component.html (19 bytes)
CREATE src/app/home/home.component.spec.ts (585 bytes)
CREATE src/app/home/home.component.ts (268 bytes)
CREATE src/app/home/home.component.scss (0 bytes)
UPDATE src/app/home/home.module.ts (261 bytes)
```

adding home module and component to remoteapp2

4. Now add the routing for both application in home.module.ts

```
4  import { HomeComponent } from './home.component';
5
6  const routes: Routes = [
7    {
8      path: '',
9      component: HomeComponent,
10   },
11 ];
12
13 @NgModule({
14   declarations: [HomeComponent],
15   imports: [CommonModule, RouterModule.forChild(routes)],
16 })
```

Router Config For Remote Apps

5. Now adding the Module Federation Package to the three applications

```
ng add @angular-architects/module-federation --project host --port 4200

ng add @angular-architects/module-federation --project remoteapp1 --port 5001

ng add @angular-architects/module-federation --project remoteapp2 --port 5002
```

The above command creates web pack config files and updates angular.json.

```
The package @angular-architects/module-federation@14 will be installed and executed.
Would you like to proceed? Yes
✓ Packages successfully installed.
CREATE projects/host/webpack.config.js (1599 bytes)
CREATE projects/host/webpack.prod.config.js (46 bytes)
CREATE projects/host/src/bootstrap.ts (372 bytes)
UPDATE tsconfig.json (785 bytes)
UPDATE projects/host/tsconfig.app.json (241 bytes)
UPDATE angular.json (3145 bytes)
UPDATE package.json (1241 bytes)
UPDATE projects/host/src/main.ts (58 bytes)
✓ Packages installed successfully.
```

After Adding Module Federartion

6. Now expose the modules from the Remote Applications for that open webpack.config.js in remoteapp1 and remoteapp2

```
// For remotes (please adjust)
name: "remoteapp1",
filename: "remoteEntry.js",
exposes: {
    './homeModule': './/src/app/home/home.module.ts',
},
```

remoteapp1 in webpack.config.js

```
// For remotes (please adjust)
name: "remoteapp2",
filename: "remoteEntry.js",
exposes: {
  "./homeModule": ".//src/app/home/home.module.ts",
},
```

remoteapp2 webpack.config.js

7. Catch the Exposed modules in the host application webpack.config.js

```
// For hosts (please adjust)
remotes: {
  remoteapp1: "http://localhost:5000/remoteEntry.js",
  remoteapp2: "http://localhost:5001/remoteEntry.js",
},
```

host application webpack.config.js

8.Now configure the routing in the host application app.module.ts

```
const routes: Routes = [
  {
    path: '',
    component: AdminComponent,
  },
  {
    path: 'mfe1',
    loadChildren: () =>
      import('remoteapp1/homeModule').then((module) => module.HomeModule),
  },
  {
    path: 'mfe2',
    loadChildren: () =>
      import('remoteapp2/homeModule').then((module) => module.HomeModule),
  },
];
```

9. After Configuring the routes you are getting errors like module not found. you need to create a decl.d.ts in host app folder.

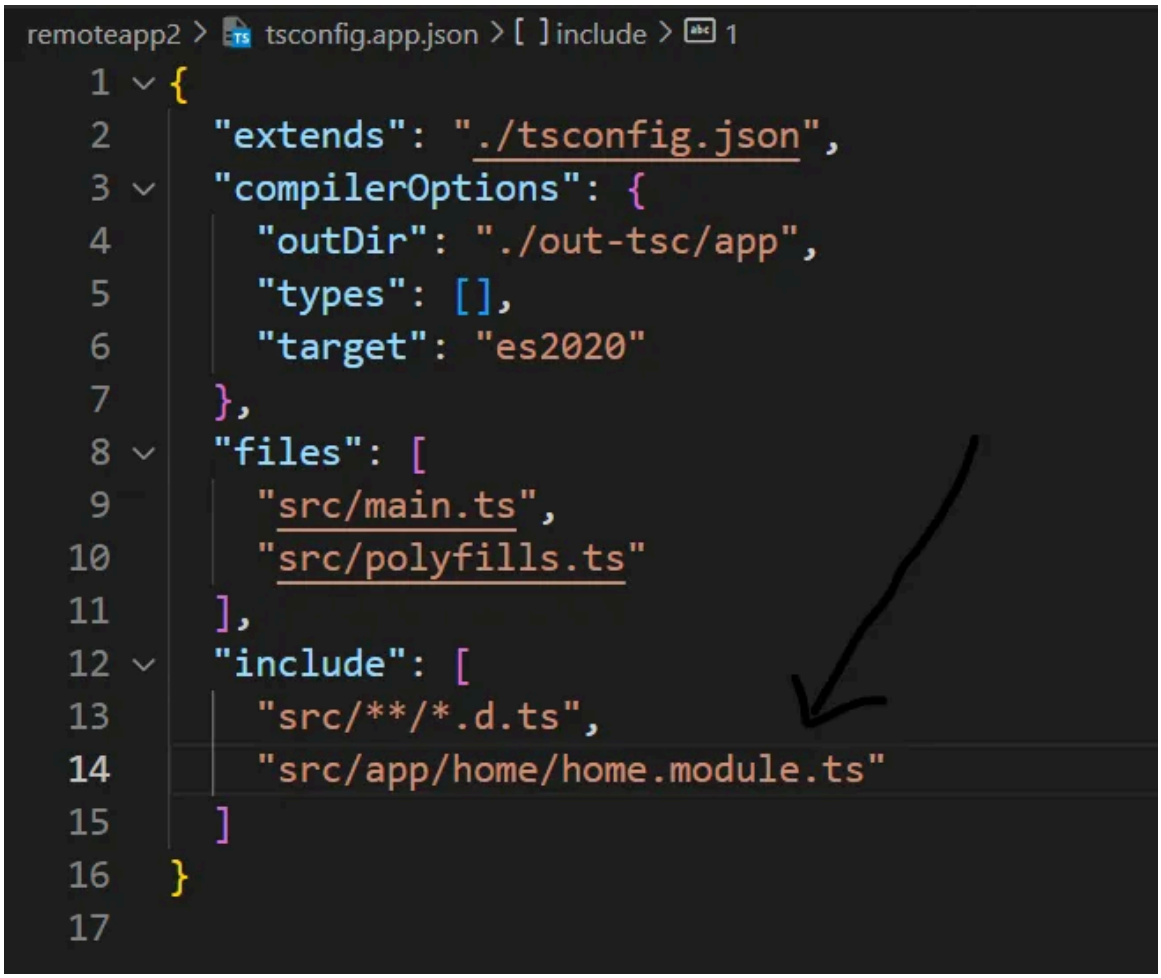
```
declare module 'remoteapp1/homeModule';
declare module 'remoteapp2/homeModule';
```

decl.d.ts in host folder

10. Then if you get the below error in remoteapp1 and remoteapp2 . then add the home.module.ts in tsconfig (files).

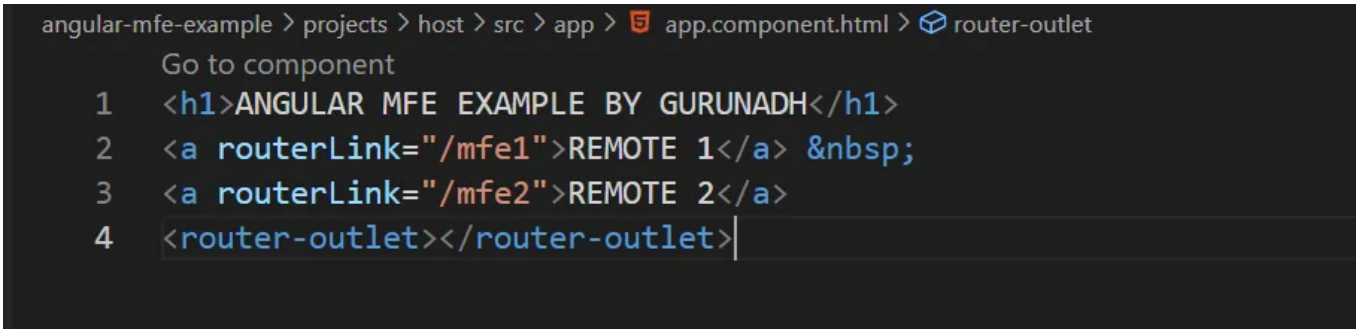
```
./src/app/home/home.module.ts - Error: Module build failed (from ./node_modules/@ngtools/webpack/src/ivy/index.js):
Error: C:\Users\gurun\OneDrive\Desktop\Personal\poc\remoteapp1\src\app\home\home.module.ts is missing from the TypeScript compilation. Please m
ake sure it is in your tsconfig via the 'files' or 'include' property.
```

error while serving remote app 1 and 2



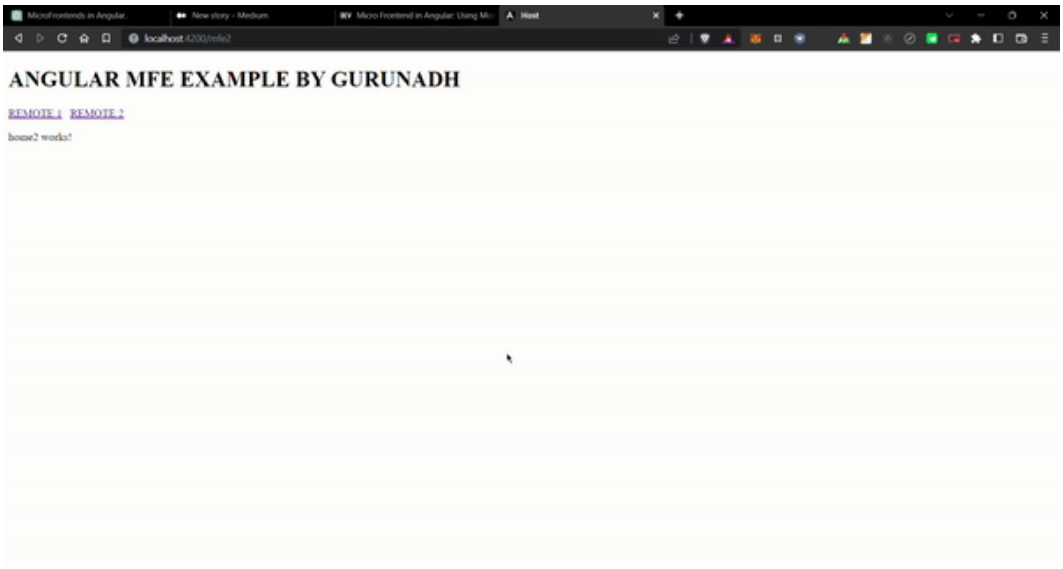
adding home.modult.ts in tsconfig.app.json in both remotes

11. Now reserve the remotes and modify the host app.component.ts like below



app.component.html of host

12. Now open the host application in the browser and test it.



Demo Working

Working.....

Repo Link : https://github.com/GuruNadh552/Micro-Frontends-in-Angular

Thank you if you have any doubts. drop the message in comments

Angular | Angularjs | Microfrontends | Angular Microfrontends | Angular 15

Follow

Written by Gurunadh Pukkalla

109 Followers · 1 Following

I am a Frontend Engineer and R&D Lead at Faclon Labs and Permanent Learner.

Responses (7)

What are your thoughts?

Respond

Gurunadh Pukkalla Author
Mar 7, 2023

Working on it

Reply

Nomis (sp90)
Apr 2, 2023 (edited)

Honestly whats the benefit over just structuring your code in one angular project ?

It does all the treeshaking and lazy loading with the built in router. If you use loadComponent/loadModule it also does not hit you on re compiles because it just... more

58 2 replies Reply

Kiran Pathe
Mar 5, 2023

Any articles on sharing state in micro frontend architecture?

3 Reply

See all responses

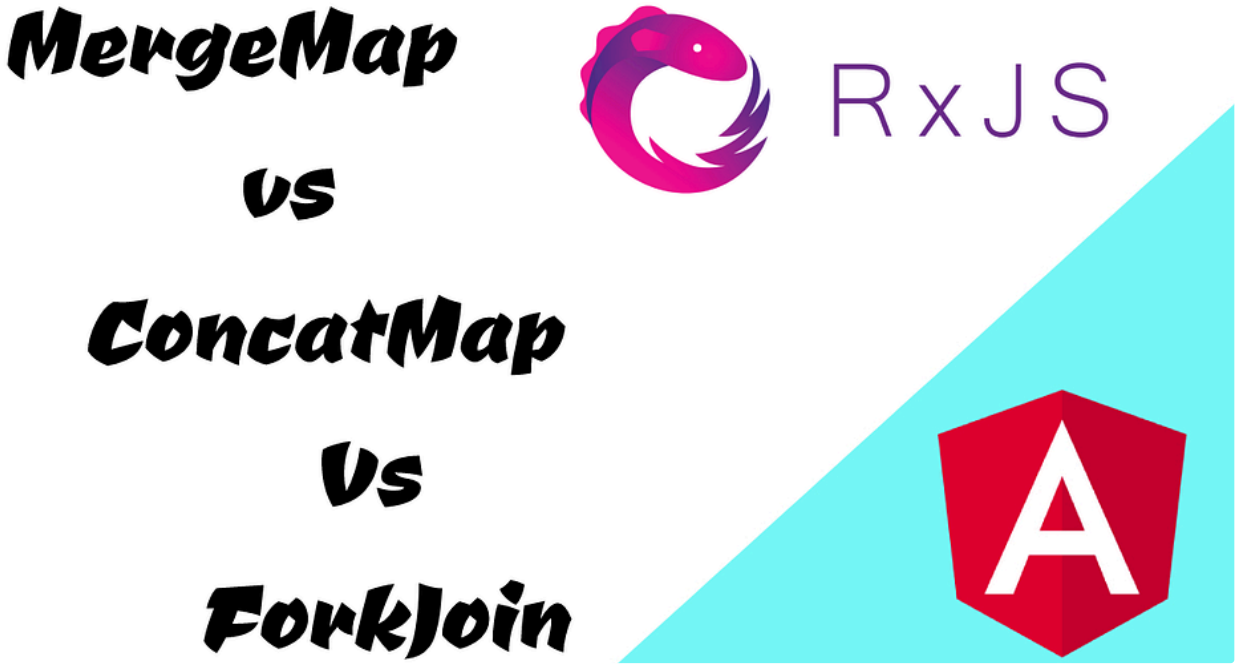
More from Gurunadh Pukkalla

Gurunadh Pukkalla

Router Events In Angular

Angular is a popular front-end framework for building single-page applications. One of the key features of Angular is its powerful routing...

Apr 22, 2023 21

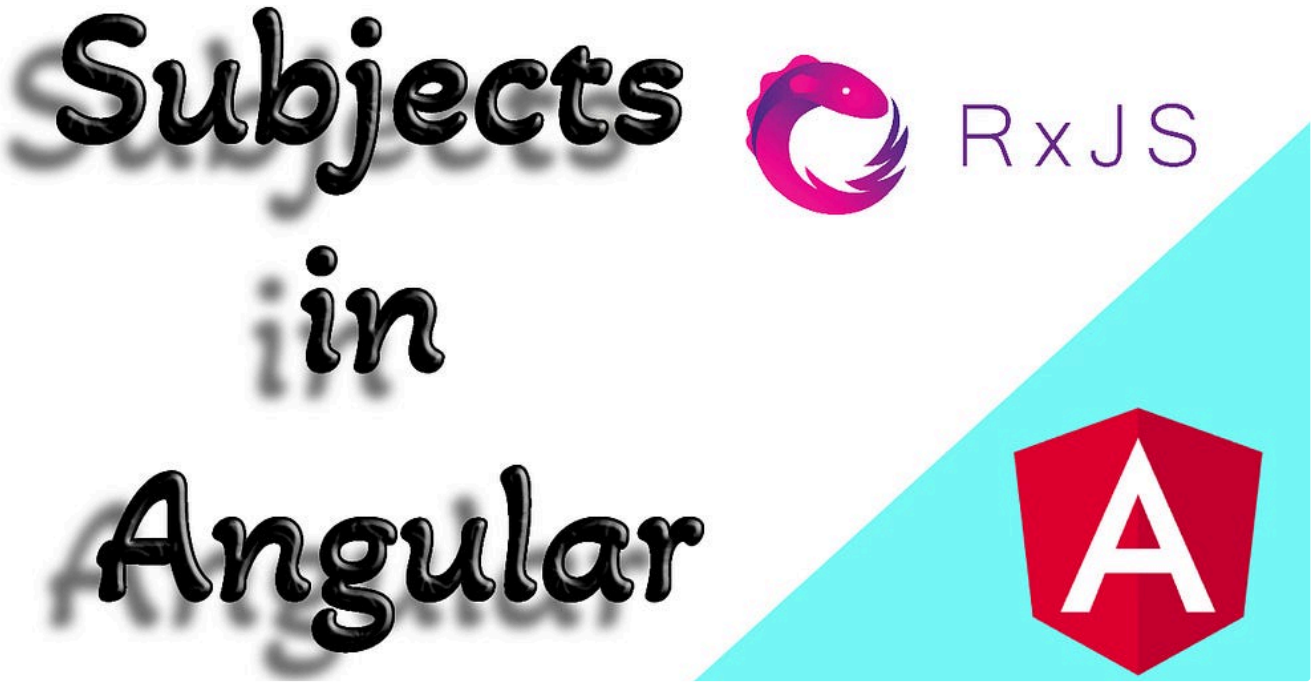


Gurunadh Pukkalla

MergeMap vs ConcatMap Vs ForkJoin

Angular Rxjs Operators

Mar 28, 2023 35

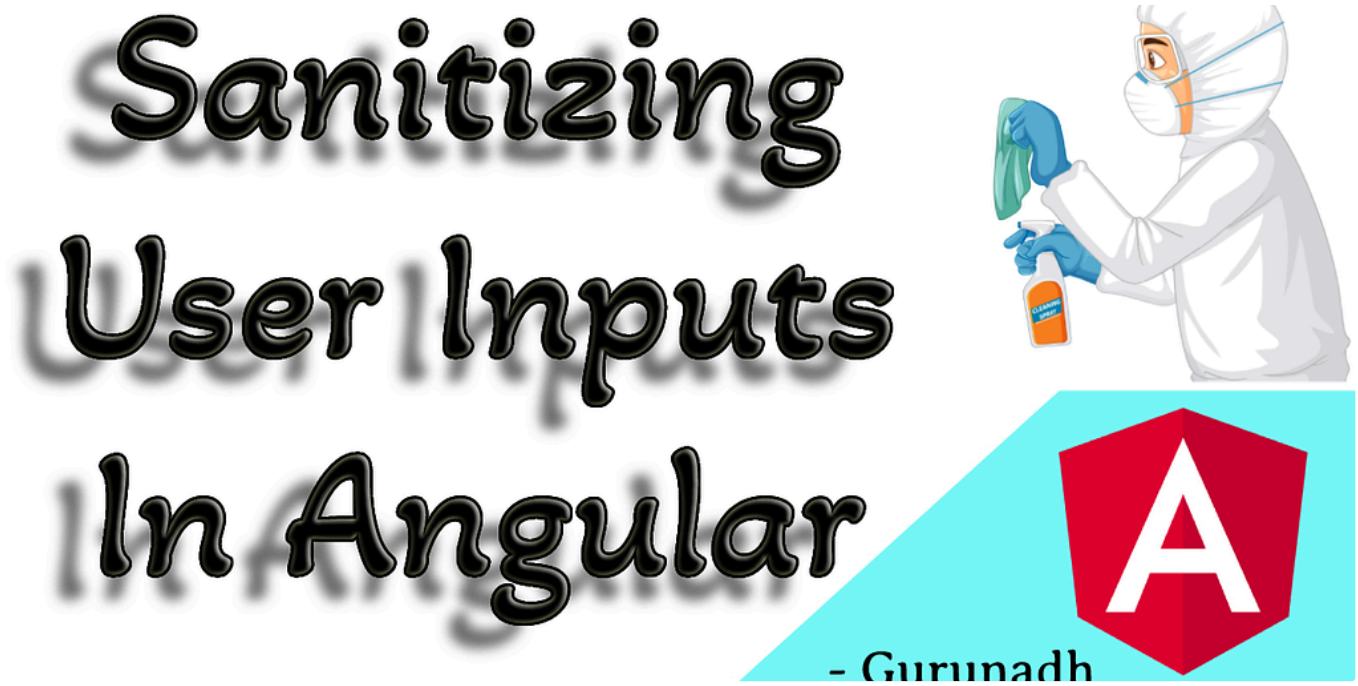


Gurunadh Pukkalla

Subjects in Angular

Subjects are an important part of the Angular framework, and they allow developers to create reactive and event-driven applications...

Apr 7, 2023 7



Gurunadh Pukkalla

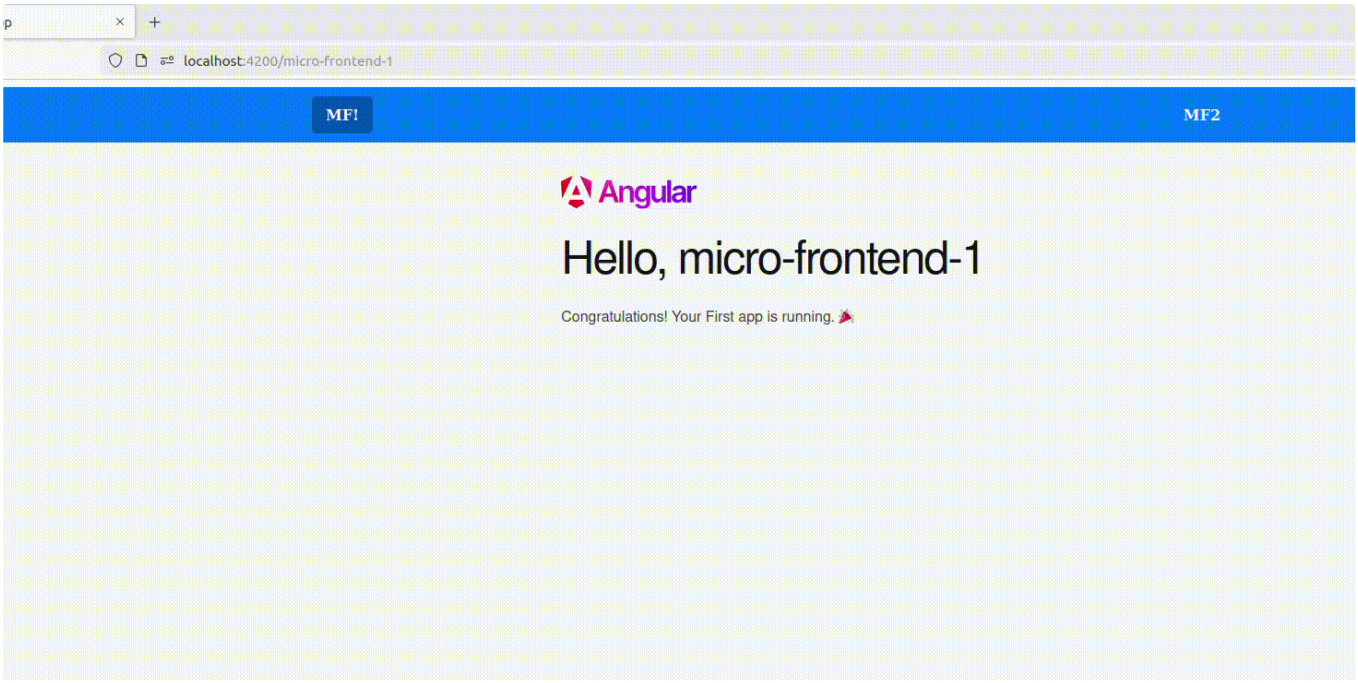
Sanitizing User Inputs in Angular 7

Sanitizing user inputs is a crucial aspect of web development when it comes to preventing malicious attacks such as cross-site scripting...

Apr 20, 2023 9

See all from Gurunadh Pukkalla

Recommended from Medium

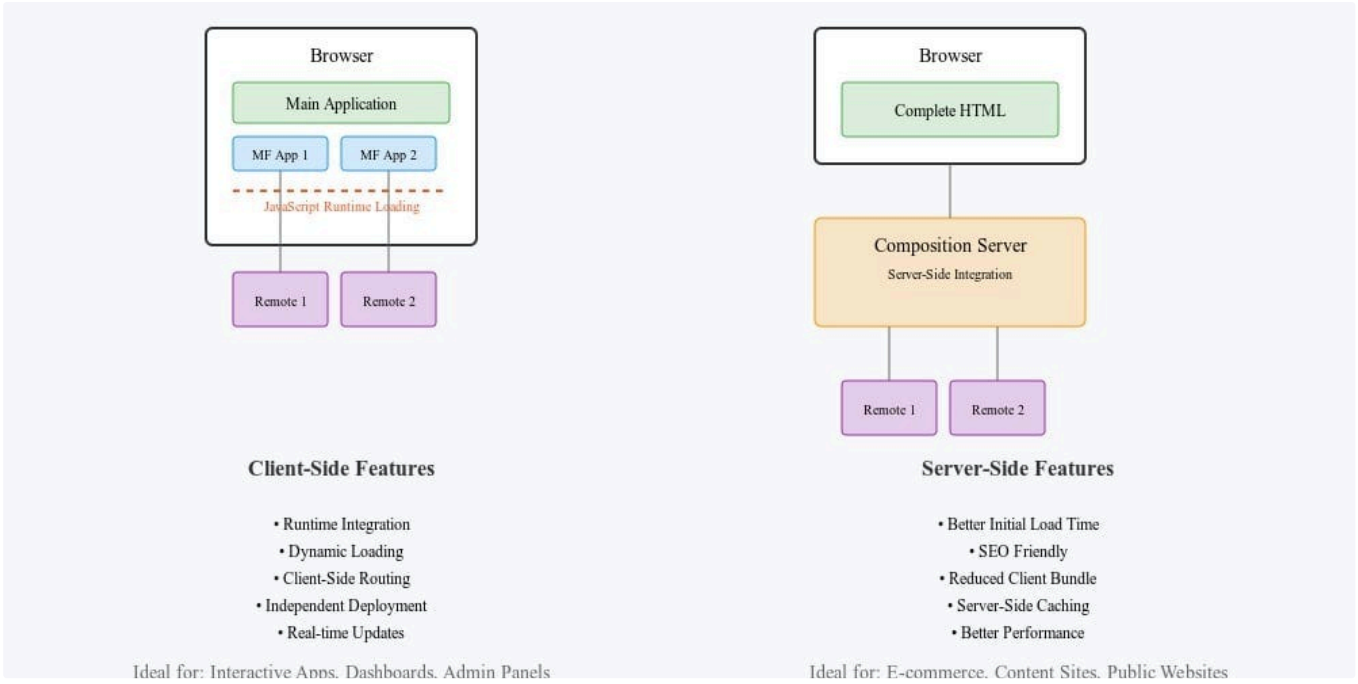


In Developwithmi by Midhul

Micro Frontends with Angular : Practical Multi-Repo Guide

Set up a micro frontend architecture using Angular with multiple repositories.

Oct 16, 2024 94



Charu Goyal

Implementing Micro-frontends with Vite Federation : A Practical Guide

Introduction

Nov 1, 2024 39 1

Lists

Staff picks

817 stories · 1632 saves

Stories to Help You Level-Up at Work

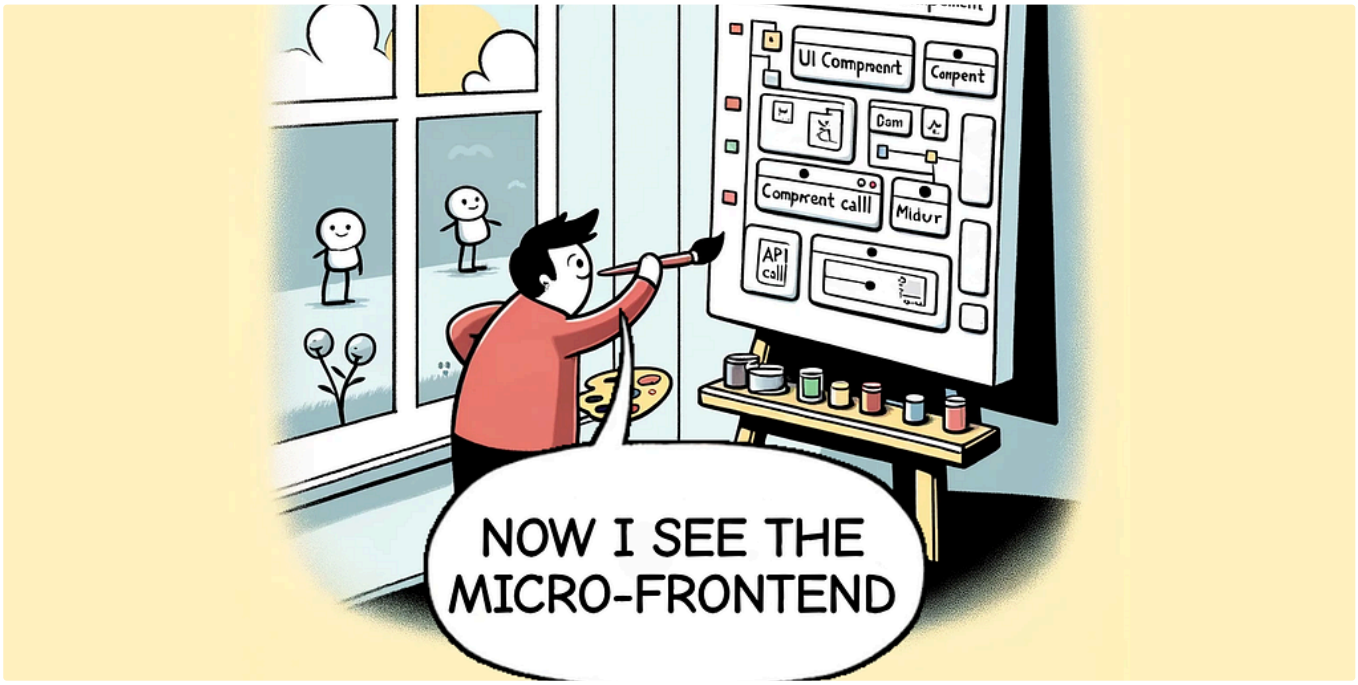
19 stories · 942 saves

Self-Improvement 101

20 stories · 3313 saves

Productivity 101

20 stories · 2787 saves



David Rodenas PhD

Overcoming the Implementation Challenges of Micro-Frontends

A comprehensive guide of the micro-frontend architecture and implementation challenges.

Mar 30, 2024 142 3

Priyabrata Saha

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

As Angular continues evolving, coding practices must adapt to align with its latest features and optimizations. Angular in 2025 is a...

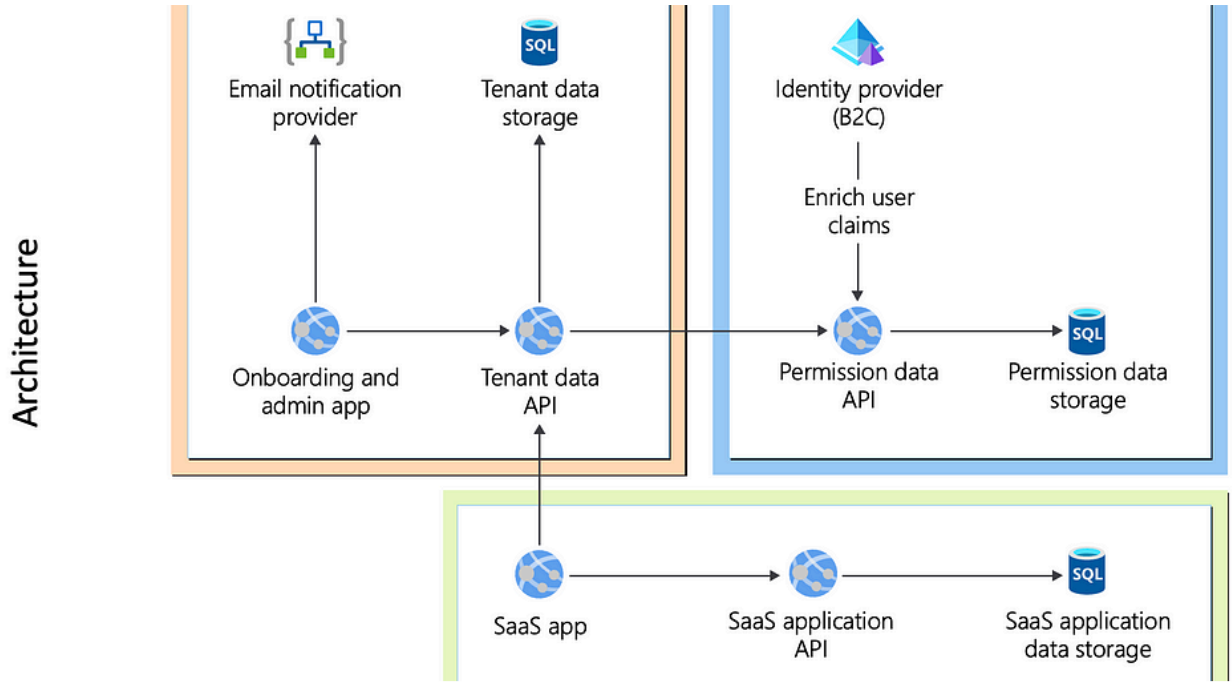
Dec 10, 2024 116 4

In Stackademic by Yue Wang

Deploying Angular SSR （v17, 18 & 19）Websites on AWS Lambda

Since version 17, Angular has made significant improvements to SSR (Server-Side Rendering). However, if you attempt to deploy your SSR app...

Nov 16, 2024 62 1



Balram Chavan

How to Build a Scalable Multi-Tenant SaaS Platform Using Angular, C#.NET,

Checkout my YouTube channel and if you would like to me create a YouTube videos on this topic, drop me a message in the comment section.

Dec 15, 2024 58

See more recommendations