**Micro Front-Ends - Angular**

The term "Micro Front-ends" has been a buzzword for breaking up growing front-end code into easy-to-maintain parts. The front-end is divided into its multiple functions or parts. These parts are implemented and deployed by independent teams. This increases the testability, reusability, and offers the possibility to select different technologies for each micro front-end.

We will need a few dependencies to build and run Angular custom elements.

Generate two projects.

Ng new app1

Ng new app2

Ng new mainapp

ng add @angular/elements

ng add ngx-build-plus

npm i -g http-server --save

generate components in every app.

demo.component.html

<!--The content below is only a placeholder and can be replaced.-->

<div style="text-align:center">

<a href="javascript:alert('Welcome to App1!!');" style="font-size:25px;">{{ title }}</a>

</div>

demo.component.ts

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-demo',

templateUrl: './demo.component.html',

styleUrls: ['./demo.component.scss']

})

export class DemoComponent implements OnInit {

title = ‘Harsha App';

constructor() { }

ngOnInit() {

}

Update app.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { NgModule, Injector } from '@angular/core';

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

import { AppComponent } from './app.component';

import { createCustomElement } from '@angular/elements';

import { DemoComponent } from './demo/demo.component';

@NgModule({

declarations: [

AppComponent,

DemoComponent

],

imports: [

BrowserModule,

AppRoutingModule

],

providers: [],

bootstrap: [],

entryComponents: [

DemoComponent

]

})

export class AppModule {

constructor(private injector: Injector) {

}

ngDoBootstrap() {

const myCustomElement = createCustomElement(DemoComponent, { injector: this.injector });

customElements.define('app-demo’, myCustomElement);

}

}

Update angular.json:

"architect": { "build": { "builder": "ngx-build-plus:build", ....

"serve": { "builder": "ngx-build-plus:dev-server", ...

"test": { "builder": "ngx-build-plus:karma",

Running App:

ng build --prod --output-hashing none --single-bundle true

if u got any error like:

Schema validation failed with the following errors: Data path ".budgets[1].type" should be equal to one of the allowed values.

In angular.json remove:

{

"type": "anyComponentStyle",

"maximumWarning": "6kb",

"maximumError": "10kb"

}

Finnaly wrapping custom element:

In Mainapp/index.html

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>MainApp</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

</head>

<body>

<app-root></app-root>

<app-demo></app-demo>

<app-demo1></app-demo1>

<script src="https://cdnjs.cloudflare.com/ajax/libs/zone.js/0.9.1/zone.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/webcomponentsjs/2.2.10/custom-elements-es5-adapter.js"></script>

<script type="text/javascript" src="http://localhost:8081/main.js"></script>

<script type="text/javascript" src="http://localhost:8082/main.js"></script>

</body>

</html>

**Here,**Angular requires **zones.** and **custom-elements-es5-adapter.js**provides custom element support within the browser. We also included main.js from our custom elements.

Modify mainapp/angular.json to override the default server port.

"serve": {

"builder": "@angular-devkit/build-angular:dev-server",

"options": {

"browserTarget": "mainapp:build",

"port": 8080

}

…