Angular 18 Microfrontend Build Guide for Single SPA

This guide covers building an Angular 18 application as a microfrontend for Single SPA deployment via CDN, with minimal file output and style isolation.

Prerequisites

- Angular CLI 18+
- Node.js 18+
- single-spa-angular package

Initial Setup

1. Create New Angular Project

```bash
ng new my-microfrontend --routing=true --style=scss
cd my-microfrontend
```

2. Install Required Dependencies

```bash
# Single SPA dependencies
npm install single-spa-angular single-spa

# For style isolation
npm install -D postcss postcss-prefixwrap autoprefixer
```

Project Configuration

3. Configure angular.json (No Webpack Needed)

Replace the build configuration in `angular.json`:

```json
```

```
{
  "projects": {
    "my-microfrontend": {
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist",
            "index": "src/index.html",
            "main": "src/main.single-spa.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "assets": [],
            "styles": ["src/styles.scss"],
            "scripts": [],
            "optimization": true,
            "outputHashing": "none",
            "extractCss": true,
            "namedChunks": false,
            "vendorChunk": false,
            "commonChunk": false,
            "buildOptimizer": true
          }
        }
      }
    }
  }
}
```

5. Create Single SPA Entry Point

Create `src/main.single-spa.ts`:

```typescript
import { enableProdMode, NgZone } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { Router } from '@angular/router';
import { LifeCycles, singleSpaAngular } from 'single-spa-angular';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

const lifecycles: LifeCycles = singleSpaAngular({
  bootstrapFunction: singleSpaProps => {
    return platformBrowserDynamic().bootstrapModule(AppModule);
  },
  template: '<app-root />',
```

```
  Router,
  NgZone,
  domElementGetter: () => {
    // Create a unique container for this microfrontend
    let container = document.getElementById('my-microfrontend-container');
    if (!container) {
      container = document.createElement('div');
      container.id = 'my-microfrontend-container';
      document.body.appendChild(container);
    }
    return container;
  }
});

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

Style Isolation

6. Configure Style Scoping

Update `src/styles.scss` with a unique prefix:

```scss
// Wrap all global styles in a unique container
.my-microfrontend-container {
  // Import your component libraries here with scoping
  @import '~bootstrap/dist/css/bootstrap.css';
  @import '~@angular/material/prebuilt-themes/indigo-pink.css';

  // Your global styles
  * {
    box-sizing: border-box;
  }

  // Override any global styles that might leak
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;

  // Ensure component libraries are scoped
  .mat-button, .btn {
    // Component-specific overrides
  }
}
```

4. Configure PostCSS for Automatic Style Prefixing

Create `postcss.config.js` in the root:

```javascript
```

```
module.exports = {
  plugins: [
    require('autoprefixer'),
    require('postcss-prefixwrap')('#my-microfrontend-root')
  ]
};
```

8. Update App Component

Modify `src/app/app.component.ts`:

typescript

```typescript
import { Component, ViewEncapsulation } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `
    <div class="my-microfrontend-container">
      <router-outlet></router-outlet>
    </div>
  `,
  styleUrls: ['./app.component.scss'],
  encapsulation: ViewEncapsulation.Emulated // Ensures style isolation
})
export class AppComponent {
  title = 'my-microfrontend';
}
```

Third-Party Library Integration

8. Handle Third-Party Components

For libraries like Angular Material, PrimeNG, etc.:

typescript

```typescript
// In your module
import { MatButtonModule } from '@angular/material/button';
import { MatCardModule } from '@angular/material/card';

@NgModule({
  imports: [
    // Only import what you need to minimize bundle size
    MatButtonModule,
    MatCardModule
  ],
  // ...
})
export class AppModule { }
```

Create component-specific style overrides:

```scss
// component.scss
:host {
  // All styles are automatically scoped to this component
  .mat-button {
    // Override material styles safely
    border-radius: 8px;
  }
}
```

Build Configuration

9. Build Scripts

Update `package.json`:

```json
{
  "scripts": {
    "build:microfrontend": "ng build --configuration=production",
    "build:serve": "npx http-server dist -p 4200 --cors",
    "analyze": "npx webpack-bundle-analyzer dist/main.js"
  }
}
```

10. Build Command

```bash
npm run build:microfrontend
```

This will generate:

- `dist/main.js` (your Angular app)
- `dist/styles.css` (all styles)

CDN Deployment Setup

11. Final Output

Create a simple `dist/index.html` for testing:

```html
```

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Microfrontend Test</title>
  <link rel="stylesheet" href="./styles.css">
  <script type="systemjs-importmap">
  {
    "imports": {
      "@angular/core": "https://unpkg.com/@angular/core@18/bundles/core.umd.js",
      "@angular/common": "https://unpkg.com/@angular/common@18/bundles/common.umd.js",
      "@angular/platform-browser": "https://unpkg.com/@angular/platform-browser@18/bundles/platform-browser.u
      "@angular/platform-browser-dynamic": "https://unpkg.com/@angular/platform-browser-dynamic@18/bundles/p
      "@angular/router": "https://unpkg.com/@angular/router@18/bundles/router.umd.js",
      "single-spa": "https://unpkg.com/single-spa@6/lib/system/single-spa.min.js",
      "single-spa-angular": "https://unpkg.com/single-spa-angular@9/lib/system/single-spa-angular.min.js"
    }
  }
  </script>
  <script src="https://unpkg.com/systemjs@6/dist/system.min.js"></script>
</head>
<body>
  <script>
    System.import('./main.js');
  </script>
</body>
</html>
```

Single SPA Integration

13. Single SPA Integration

In your Single SPA root configuration:

javascript

```javascript
import { registerApplication, start } from 'single-spa';

registerApplication({
  name: 'my-microfrontend',
  app: () => System.import('https://your-cdn.com/my-microfrontend/main.js'),
  activeWhen: ['/my-microfrontend'],
  customProps: {
    // Pass any props needed
  }
});

start();
```

Style Isolation Best Practices

14. Critical Rules for Zero Style Leakage

Always follow these rules:

1. **Import 3rd party CSS inside the scoped container** in `styles.scss`
2. **Use the same ID** (`#my-microfrontend-root`) in PostCSS config and app component
3. **Never import global CSS** outside the scoped container
4. **Test with multiple microfrontends** loaded simultaneously

15. Component-Level Isolation

For individual components with 3rd party libraries:

```scss
// component.scss
:host {
  // Component styles are automatically scoped by Angular
  .custom-override {
    // Override 3rd party component styles safely
    .mat-button {
      border-radius: 4px;
      font-weight: 500;
    }

    .p-button {
      border-radius: 4px;
      padding: 0.5rem 1rem;
    }
  }
}
```

16. Build Verification

After building, verify:

```bash
# Check file output
ls -la dist/
# Should see: main.js, styles.css (and index.html if created)

# Check style scoping - all CSS should be prefixed
grep "#my-microfrontend-root" dist/styles.css
```

Final File Structure

After build, you'll have exactly:

```
dist/
├── main.js        # Angular microfrontend with Single SPA lifecycles
├── styles.css     # All styles scoped to #my-microfrontend-root
└── index.html     # Optional test file
```

Troubleshooting

Style Leakage: Ensure all 3rd party CSS imports are inside `#my-microfrontend-root` in styles.scss and PostCSS is configured correctly.

Multiple Files: Check that `vendorChunk: false`, `commonChunk: false`, and `namedChunks: false` are set in angular.json.

3rd Party Conflicts: Always import component library CSS inside the scoped container, never as separate entries in the styles array.

This configuration will give you a production-ready Angular 18 microfrontend that can be deployed via CDN and integrated with Single SPA while maintaining style isolation.