

How to use Web Components in Angular Applications

BPR
VUE

Vermögen und Eigenkapital

Exportiert am

27/07/2024

durch

Nab Raj Roshyara

GFS
SEU

Eigentum der Atruvia AG. Exportiert aus Confluence, bereitgestellt
durch GFS SEU.

Exportiert mit "SEU Export für Archivierung" (V1, 2022-06-23)

Table of Contents

1 What are Web Components?.....	4
2 Advantages of Using Web Components.....	5
3 Step-by-Step Guide to Implementing Web Components in Angular Standalone project.....	6
4 Related articles	13

Dr. Nab Raj Roshyara

Angular is a popular framework for building modern web applications. It provides a number of features and tools that make it easy to create complex applications. In this post, we will explore how to use web components in Angular with examples.

Web components are a set of web platform APIs that allow developers to create custom HTML elements with their own functionality and styles. They are a powerful tool for building reusable and modular components that can be used across different projects.

1 What are Web Components?

Web Components are custom, reusable HTML elements that encapsulate functionality and styles, making them modular and reusable across different projects and frameworks. They consist of three main technologies:

1. **Custom Elements:** APIs to define new HTML elements.
2. **Shadow DOM:** Encapsulation of internal DOM and styles, preventing them from being affected by the main document's CSS.
3. **HTML Templates:** `<template>` and `<slot>` elements for defining reusable HTML structures.

2 Advantages of Using Web Components

1. **Reusability:** Web components are highly reusable across different projects and frameworks.
2. **Encapsulation:** They provide encapsulation of styles and behavior, reducing the risk of conflicts.
3. **Interoperability:** Web components work seamlessly with various JavaScript frameworks and libraries.
4. **Future-Proof:** As a web standard, they are expected to be supported long-term across all browsers.
5. **Modularity:** They promote a modular approach to building applications, making code more maintainable and scalable.

3 Step-by-Step Guide to Implementing Web Components in Angular Standalone project

Prerequisites

- Angular CLI version 17.1.0
- Node.js and npm installed

To check your Angular CLI version, use the following command:

ng version

If you need to install Angular CLI version 17.1.0, you can do so with:

npm install -g @angular/cli@17.1.0

1. Set Up a New Angular Project
Create a new Angular project using Angular CLI:

ng new angular-web-component
cd angular-web-component

2. Install Angular Elements and Dependencies
 - Install Angular Elements and other necessary packages:

```
1 npm install @angular/elements@17.1.0 --legacy-peer-deps
2 npm install @webcomponents/custom-elements --legacy-peer-deps
```

- Install polyfills for browsers:

```
1 npm install @webcomponents/custom-elements --legacy-peer-deps
```

- Add polyfills to `src/polyfills.ts`:

```
1 touch src/polyfills.ts
```

Add the following lines to `src/polyfills.ts`. If this file does not exist, then create it.

```
1 import '@webcomponents/custom-elements/src/native-shim';
2 import '@webcomponents/custom-elements/custom-elements.min';
3 import 'zone.js'; // Included with Angular CLI.
```

- Edit `tsconfig.app.json` file as following

```

1  /* To learn more about this file see: https://angular.io/
2  config/tsconfig. */
3  {
4    "extends": "./tsconfig.json",
5    "compilerOptions": {
6      "outDir": "./out-tsc/app",
7      "types": []
8    },
9    "files": [
10     "src/main.ts",
11     //this line has been added
12     "src/polyfills.ts"
13   ],
14   "include": [
15     "src/**/*.d.ts"
16   ]
17 }

```

- ***If necessary edit Angular.json file to add polyfills.ts .
Search for the lines "polyfills" in Angular.json and change like following :***

```

1    "build": {
2      "builder": "@angular-devkit/build-
3      angular:application",
4      "options": {
5        "outputPath": "dist/angular-web-component",
6        "index": "src/index.html",
7        "browser": "src/main.ts",
8        "polyfills": [
9          "src/polyfills.ts",
10        ],
11        "tsConfig": "tsconfig.app.json",
12        "assets": [
13          "src/favicon.ico",
14          "src/assets"
15        ],
16        "styles": [
17          "src/styles.css"
18        ],
19        "scripts": []
20      },
21      ...
22    },
23    ...
24    ...
25    "test": {
26      "builder": "@angular-devkit/build-angular:karma",
27      "options": {
28        "polyfills": [

```

```

29         "src/polyfills.ts",
30     ],
31     "tsConfig": "tsconfig.spec.json",
32     "assets": [
33         "src/favicon.ico",
34         "src/assets"
35     ],
36     "styles": [
37         "src/styles.css"
38     ],
39     "scripts": []
40 }

```

i About NewsAPI.org: [NewsAPI.org](https://newsapi.org/) is a simple HTTP REST API for searching and retrieving live articles from all over the web. It includes breaking news headlines, news articles, and blogs from reliable news sources and blogs. It offers a free plan for open source and development projects. To use this service, you need to register on their website to get a free API key. With the API key registered, you can define a variable called `apiKey` in your service as shown above.

Let's create component to fetch news from the NewsApi and display as Angular Component . For this, we follow the following steps.

1. Create a Component to Display Data

```
1 ng generate component news
```

This command will create four files as following

```

IDCP+xcx7004@CP28271 MINGW64 /c/roshyara/testing/frontend/examples/web_components/web_component (main)
$ ng generate component news
CREATE src/app/news/news.component.html (20 bytes)
CREATE src/app/news/news.component.spec.ts (605 bytes)
CREATE src/app/news/news.component.ts (238 bytes)
CREATE src/app/news/news.component.css (0 bytes)

```

2. Create `src/app/news/news.model.ts`:

```

1 //src/app/news/news.model.ts
2 export interface Article {
3     source: {
4         id: string | null;
5         name: string;
6     };
7     author: string;
8     title: string;
9     description: string;
10    url: string;
11    urlToImage: string;
12    publishedAt: string;
13    content: string;

```



```

14 }
15 export interface NewsApiResponse {
16   status: string;
17   totalResults: number;
18   articles: Article[];
19 }

```

3. Create a Service for Data Fetching

Generate a service to fetch data from an API:

```

1 ng generate service data
2 //This command will generate two files as following

```

```

IDCP+xcx7004@CP28271 MINGW64 /c/roshyara/testing/frontend/examples/web_components/web_component (main)
$ ng generate service data
CREATE src/app/data.service.spec.ts (363 bytes)
CREATE src/app/data.service.ts (142 bytes)

```

4. Edit the file `src/app/data.service.ts` as following

```

1 //src/app/data.service.ts:
2 import { Injectable } from '@angular/core';
3 import { HttpClient } from '@angular/common/http';
4 import { Observable } from 'rxjs';
5 import { NewsApiResponse } from '../news/news.model';
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class DataService {
11   apiKey = 'YOUR_API_KEY';
12   // my api key
13   //apiKey =dcceb22b096c4c4ba9dc1e616055ec50
14   constructor(private httpClient: HttpClient) { }
15
16   get(): Observable<NewsApiResponse> {
17     return this.httpClient.get<NewsApiResponse>(`https://
newsapi.org/v2/top-headlines?sources=techcrunch&apiKey=${
this.apiKey}`);
18   }
19 }

```

5. Edit the news components as following

```

1 //src/app/news/news.component.ts
2 import { Component, OnInit } from '@angular/core';
3 import { DataService } from '../data.service';
4 import { Article, NewsApiResponse } from '../news.model';
5 import { CommonModule } from '@angular/common';
6
7 @Component({

```

```

8      selector: 'app-news',
9      templateUrl: './news.component.html',
10     styleUrls: ['./news.component.css'],
11     standalone: true,
12     imports: [CommonModule]
13   })
14   export class NewsComponent implements OnInit {
15     articles!: Article[];
16
17     constructor(private dataService: DataService) { }
18
19     ngOnInit() {
20       this.dataService.get().subscribe((data: NewsApiResponse) => {
21         console.log(data);
22         this.articles = data.articles;
23       });
24     }
25   }

```

6. **Update** `src/app/news/news.component.html`:

```

1  // src/app/news/news.component.html: <!-- src/app/news/
2  news.component.html -->
3  <div *ngFor="let article of articles">
4  <h2>{{ article.title }}</h2>
5  <p>{{ article.description }}</p>
6  <a href="{{ article.url }}">Read full article</a>
7  </div>

```

7. Update `src/app/app.component.html`:

```

1  <!-- src/app/app.component.html -->
2  <main class="main">
3  <app-news></app-news>
4  </main>
5
6  <router-outlet />

```

8. Run the application:

ng serve

9. **Transform the Component into a Web Component**

Update `app.component.ts`

```

1  import { Component, Injector } from '@angular/core';
2  import { createCustomElement } from '@angular/elements';

```

```

3 import { RouterOutlet } from '@angular/router';
4 import { NewsComponent } from '../news/news.component';
5
6 @Component({
7   selector: 'app-root',
8   standalone: true,
9   imports: [RouterOutlet, NewsComponent],
10  templateUrl: './app.component.html',
11  styleUrls: ['./app.component.css']
12 })
13 export class AppComponent {
14   title = 'angular-web-component';
15
16   constructor(private injector: Injector) {
17     const el = createCustomElement(NewsComponent, { injector });
18     customElements.define('news-widget', el);
19   }
20 }

```

10. Update `app.config.ts`

```

1 import { ApplicationConfig } from '@angular/core';
2 import { provideHttpClient } from '@angular/common/http';
3 import { provideRouter } from '@angular/router';
4
5 import { routes } from './app.routes';
6
7 export const appConfig: ApplicationConfig = {
8   providers: [
9     provideHttpClient(),
10    provideRouter(routes)
11  ]
12 };

```

11. Update `main.ts` if necessary

Ensure your `main.ts` file bootstraps the application correctly:

```

1 import { bootstrapApplication } from '@angular/platform-browser';
2 import { appConfig } from './app/app.config';
3 import { AppComponent } from './app/app.component';
4
5 bootstrapApplication(AppComponent, appConfig)
6   .catch((err) => console.error(err));

```

12. Build the Web Component

Build the project for production:

```
ng build --configuration production --output-hashing none
```




This will generate the necessary files in the `dist` folder

13. Serve the folder using a simple HTTP server:

```
1  #install serve
2  npm install -g serve
3  #serve the web component
4  serve -s dist/angular-web-component/browser
5  #or
6  serve -s dist/angular-web-component/
```

Using web components in Angular allows us to create reusable and modular components that can be easily shared across different projects. With the help of the `@angular/elements` package, we can create and register custom elements with the browser and use them in our application just like any other HTML element.

4 Related articles

-  [How to use Web Components in Angular Applications](#)
-  [DZPB-STP - Generate frontend models and services with the YAML file](#)
-  [DZPB-VV - Generate frontend models and services with the YAML file](#)