

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.....	6
4 Related articles.....	11

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

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:

```
ng add @angular/elements
```

- Install polyfills for browsers:

```
npm install @webcomponents/custom-elements
```

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

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

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

i **About NewsAPI.org:** 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 `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 }
```

2. Create a Service for Data Fetching

Generate a service to fetch data from an API:

ng generate service data

```

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
13   constructor(private httpClient: HttpClient) { }
14
15   get(): Observable<NewsApiResponse> {
16     return this.httpClient.get<NewsApiResponse>(`https://
newsapi.org/v2/top-headlines?sources=techcrunch&apiKey=${
this.apiKey}`);
17   }
18 }

```

3. Create a Component to Display Data**ng generate component news**

This command will generate three files and update `src/app/news/news.component.ts` 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
6  @Component({
7    selector: 'app-news',
8    templateUrl: './news.component.html',
9    styleUrls: ['./news.component.css']
10 })
11 export class NewsComponent implements OnInit {
12   articles!: Article[];
13
14   constructor(private dataService: DataService) { }
15
16   ngOnInit() {
17     this.dataService.get().subscribe((data: NewsApiResponse) => {
18       console.log(data);
19       this.articles = data.articles;
20     });
21   }
22 }

```

4. Update `src/app/news/news.component.html` :

```

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

```

5. 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 />

```

6. Run the application:

ng serve

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

Update `src/app.module.ts`:

```

1 import { Injector, NgModule, DoBootstrap } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { createCustomElement } from '@angular/elements';
4 import { AppComponent } from './app.component';
5 import { NewsComponent } from './news/news.component';
6 import { HttpClientModule } from '@angular/common/http';
7 import { AppRoutingModule } from './app-routing.module';
8
9 @NgModule({
10   declarations: [
11     AppComponent,
12     NewsComponent
13   ],
14   imports: [
15     BrowserModule,
16     HttpClientModule,
17     AppRoutingModule
18   ],
19   providers: []
20 })
21 export class AppModule implements DoBootstrap {
22   constructor(private injector: Injector) {
23     const el = createCustomElement(NewsComponent, { injector });

```



```

24     customElements.define('news-widget', el);
25   }
26
27   ngDoBootstrap() {}
28 }

```

8. 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

9. Use the web component

Create a new folder and copy files:

- Create a new folder for your web component, e.g., `news-widget`.
- Copy the following files from the `dist/angular-web-component` folder into this new folder:
 - `runtime.js`
 - `es2015-polyfills.js`
 - `polyfills.js`
 - `scripts.js`
 - `main.js`
 - `styles.css` (if present)
 - `favicon.ico` (if present)

Create an `index.html` file in the new folder:




```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-
6    scale=1.0">
7    <meta http-equiv="X-UA-Compatible" content="ie=edge">
8    <title>Testing the News Web Component</title>
9    <base href="/">
10 </head>
11 <body>
12   <news-widget></news-widget>
13
14   <script type="text/javascript" src="runtime.js"></script>
15   <script type="text/javascript" src="es2015-polyfills.js"
16   nomodule></script>
17   <script type="text/javascript" src="polyfills.js"></script>
18   <script type="text/javascript" src="scripts.js"></script>
19   <script type="text/javascript" src="main.js"></script>
20 </body>
21 </html>

```

10. Serve the folder using a simple HTTP server:
 - > npm install -g serve
 - > serve
- 11.

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](#)