**Fetching data in Angular: HttpClient or Fetch API?**

Many angular developers use Angular’s HttpClient class to fetch data from a rest API in a data service. Not only this requires you to import Observable class from RxJS library, you also need to set up a provider for HttpClientModule in your app.config.ts. Your data service often looks like this:

import { Injectable } from '@angular/core';  
import { HttpClient } from '@angular/common/http';  
import { Observable } from 'rxjs';  
  
interface User {  
 firstname: string;  
 lastname: string;  
 birthday: string;  
 id: string;  
}  
  
@Injectable({  
 providedIn: 'root'  
})  
export class DataService {  
 private apiUrl = 'https://6561036783aba11d99d1cff0.mockapi.io/api/test';  
  
 constructor(private http: HttpClient) {}  
  
 getData(): Observable<User> {  
 const users = this.http.get<User>(this.apiUrl);  
 return users;  
 }  
}

Then you use it in your component like this:

export class MyComponent {  
 constructor(private dataService: DataService) {}  
  
 logData() {  
 this.dataService.getData().subscribe(data => console.log(data));  
 }  
}

Get Saeid’s stories in your inbox

Join Medium for free to get updates from this writer.

Subscribe

However since Rest APIs rely on HTTP it means you always get one response per request, so there is really no need to use an Observable and you can achieve the same result using JavaScript’s Promises by using Fetch. Here is DataService written using promises rather than observables:

import { Injectable } from '@angular/core';  
  
interface User {  
 firstname: string;  
 lastname: string;  
 birthday: string;  
 id: string;  
}  
  
@Injectable({  
 providedIn: 'root'  
})  
export class DataService {  
 private apiUrl = 'https://6561036783aba11d99d1cff0.mockapi.io/api/test';  
  
 getData(): Promise<User> {  
 const users = fetch(this.apiUrl).then(res => res.json());  
 return users;  
 }  
}

And you use it in your component like this:

export class MyComponent {  
 constructor(private dataService: DataService) {}  
  
 logData() {  
 this.dataService.getData().then(data => console.log(data));  
 }  
}

As you see we no longer need to import HttpClient and Observable classes and the code is more reusable since it can be used in any other framework or even without a framework.

**Do this …this works :**

To fetch data using HTTP in Angular, you typically use the HttpClient service provided by @angular/common/http. Here's a complete guide with practical examples and video tutorials to help you master this process across Angular versions.

Step-by-Step Guide to Fetch Data Using HTTP in Angular

1. ✅ Import HttpClientModule

In your AppModule, import HttpClientModule:

import { HttpClientModule } from '@angular/common/http'; @NgModule({ imports: [HttpClientModule], }) export class AppModule {}

import { HttpClientModule } from '@angular/common/http';

@NgModule({

imports: [HttpClientModule],

})

export class AppModule {}

2. 🛠️ Create a Service

Generate a service to handle API calls:

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({ providedIn: 'root' })

export class DataService {

private apiUrl = 'https://jsonplaceholder.typicode.com/posts';

constructor(private http: HttpClient) {}

getPosts(): Observable<any> {

return this.http.get(this.apiUrl);

}

}

3. 📦 Inject and Use in Component

Use the service in your component:

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

import { DataService } from './data.service';

@Component({

selector: 'app-posts',

template: `<ul><li \*ngFor="let post of posts">{{ post.title }}</li></ul>`,

})

export class PostsComponent implements OnInit {

posts: any[] = [];

constructor(private dataService: DataService) {}

ngOnInit() {

this.dataService.getPosts().subscribe((data) => {

this.posts = data;

});

}

}

To **execute your Angular 20 component** that uses ngOnInit() and a DataService, follow these steps to set up and run the project successfully. I’ll also point you to video tutorials that walk through the exact process.

🛠️ Step-by-Step Execution Guide

1. **Install Angular CLI (if not already installed)**

npm install -g @angular/cli

ng new angular-http-demo

cd angular-http-demo

✅ Choose **standalone components** if prompted during setup (Angular 20 supports them natively).

3. **Generate Your Component and Service**

ng generate component posts

ng generate service data

4. **Add HttpClientModule to app.config.ts or AppModule**

If using standalone setup:

import { provideHttpClient } from '@angular/common/http';

export const appConfig: ApplicationConfig = {

providers: [provideHttpClient()]

};

If using traditional AppModule:

import { HttpClientModule } from '@angular/common/http';

@NgModule({

imports: [HttpClientModule],

})

export class AppModule {}

5. **Implement Your Service (data.service.ts)**

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({ providedIn: 'root' })

export class DataService {

private apiUrl = 'https://jsonplaceholder.typicode.com/posts';

constructor(private http: HttpClient) {}

getPosts(): Observable<any> {

return this.http.get(this.apiUrl);

}

}

6. **Update Your Component (posts.component.ts)**

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

import { DataService } from '../data.service';

@Component({

selector: 'app-posts',

template: `<ul><li \*ngFor="let post of posts">{{ post.title }}</li></ul>`,

})

export class PostsComponent implements OnInit {

posts: any[] = [];

constructor(private dataService: DataService) {}

ngOnInit() {

this.dataService.getPosts().subscribe((data) => {

this.posts = data;

});

}

}

7. **Add to app.component.html**

<app-posts></app-posts>

8. **Run the Application**

ng serve

Visit http://localhost:4200 in your browser to see the fetched data displayed.