**1. Create an HTTP Interceptor**

TypeScript

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

import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable()

export class AuthInterceptor implements HttpInterceptor {

intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

const token = localStorage.getItem('token');

if (token) {

request = request.clone({

setHeaders: {

Authorization: `Bearer ${token}`

}

});

}

return next.handle(request);

}

}

* This interceptor adds an Authorization header to every outgoing request if a token is found in local storage.

**2. Register the Interceptor**

TypeScript

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

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

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

import { AuthInterceptor } from './app/interceptors/auth.interceptor';

bootstrapApplication(AppComponent, {

providers: [

provideHttpClient(withInterceptors([AuthInterceptor]))

]

});

* Register the AuthInterceptor in the bootstrapApplication function using withInterceptors.

**3. Make HTTP Requests**

* In your components or services, use the HttpClient to make HTTP requests as usual:

TypeScript

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

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

import { Observable } from 'rxjs';

@Injectable({ providedIn: 'root' })

export class DataService {

constructor(private http: HttpClient) {}

getData(): Observable<any[]> {

return this.http.get<any[]>('/api/data');

}

}

**Key Considerations:**

* **Order of Interceptors:** If you have multiple interceptors, they are executed in the order they are registered.1
* **Error Handling:** Handle errors within your interceptors (e.g., 401 Unauthorized) and implement appropriate error handling strategies.2
* **Testing:** Write unit tests for your interceptors to ensure they function correctly.
* **Security:** Store tokens securely (e.g., in a secure storage mechanism).

**Benefits of Using HTTP Interceptors:**

* **Centralized Logic:** Encapsulate common HTTP request logic (e.g., authentication, error handling, logging) in a single location.3
* **Improved Maintainability:** Easier to modify or update common HTTP request behavior across the application.
* **Increased Reusability:** Re-use the same interceptor logic in multiple parts of your application.
* **Enhanced Security:** Implement security measures like token-based authentication in a centralized and secure manner.

By effectively using HTTP interceptors, you can improve the efficiency, maintainability, and security of your Angular applications.4