**Interceptors**

In Angular, interceptors are a powerful feature that allows you to intercept and modify HTTP requests and responses globally across your application. They provide a way to pre-process or post-process HTTP requests or responses before they are sent or received by the actual HTTP client.

Interceptors in Angular are implemented as classes that implement the `HttpInterceptor` interface. This interface requires the implementation of the `intercept` method, which receives an `HttpRequest` object and a `HttpHandler` object as parameters.

The `intercept` method allows you to inspect, modify, or even short-circuit the request or response flow. You can modify request headers, add authentication tokens, handle errors, perform logging, and more. The `intercept` method returns an observable of `HttpEvent`, which allows you to chain multiple interceptors or handle the response.

Interceptors can be used for various purposes, such as:

1. Authentication: You can add authentication headers or tokens to outgoing requests or handle authentication-related errors.

2. Logging: Interceptors can log request and response information for debugging or analytics purposes.

3. Error handling: You can intercept and handle HTTP errors globally, providing a centralized error handling mechanism.

4. Caching: Interceptors can implement caching strategies, allowing you to cache responses and serve them without making additional requests.

5. Response transformation: You can modify the response data, such as parsing, transforming, or filtering the data before it reaches the client code.

To use an interceptor, you need to provide it as a provider in your Angular module using the `HTTP\_INTERCEPTORS` token. By specifying `multi: true`, you allow multiple interceptors to be registered.

import { HTTP\_INTERCEPTORS } from '@angular/common/http';

@NgModule({

// ...

providers: [

// ...

provide: HTTP\_INTERCEPTORS,

useClass: MyInterceptor,

multi: true,

},

})

export class AppModule {}

The order of the interceptors in the provider array determines the order of execution. The interceptors are executed in the order they are registered.

For example if you provide three interceptors called A, B, C in that order, as the Angular Docs say: "... Requests flow in A->B->C and responses flow C->B->A

By using interceptors, you can centralize common functionalities related to HTTP requests and responses and make your code more modular, reusable, and maintainable.