

Angular Hands-on Workshop: Beyond the Basics

Advanced Services and HTTP Communication

Providing Services

A provider tells an injector *how to create the service*.

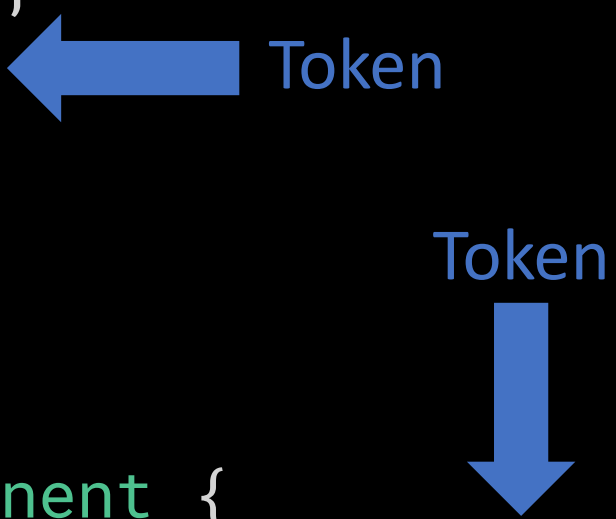
Angular Documentation

Configure an injector with a service provider

<https://angular.io/guide/dependency-injection#configure-an-injector-with-a-service-provider>

Provider Tokens

```
@NgModule({  
  declarations: [AppComponent, DashboardComponent],  
  bootstrap: [AppComponent],  
  providers: [DataService] ← Token  
})  
export class AppModule { }  
  
export class DashboardComponent {  
  constructor(private dataService: DataService) { }  
}
```



The diagram illustrates the concept of a 'Token' in Angular. A blue arrow points from the `providers: [DataService]` line in the `@NgModule` decorator to the word 'Token'. Another blue arrow points from the word 'Token' down to the `DataService` parameter in the `constructor` of the `DashboardComponent` class.

Provider Recipes

```
@NgModule({  
  declarations: [AppComponent, DashboardComponent],  
  bootstrap: [AppComponent],  
  providers: [  
    DataService,  
    { provide: LoggerService, useClass: LoggerService }  
  ]  
})  
export class AppModule { }
```

The diagram illustrates four provider recipes in the code:

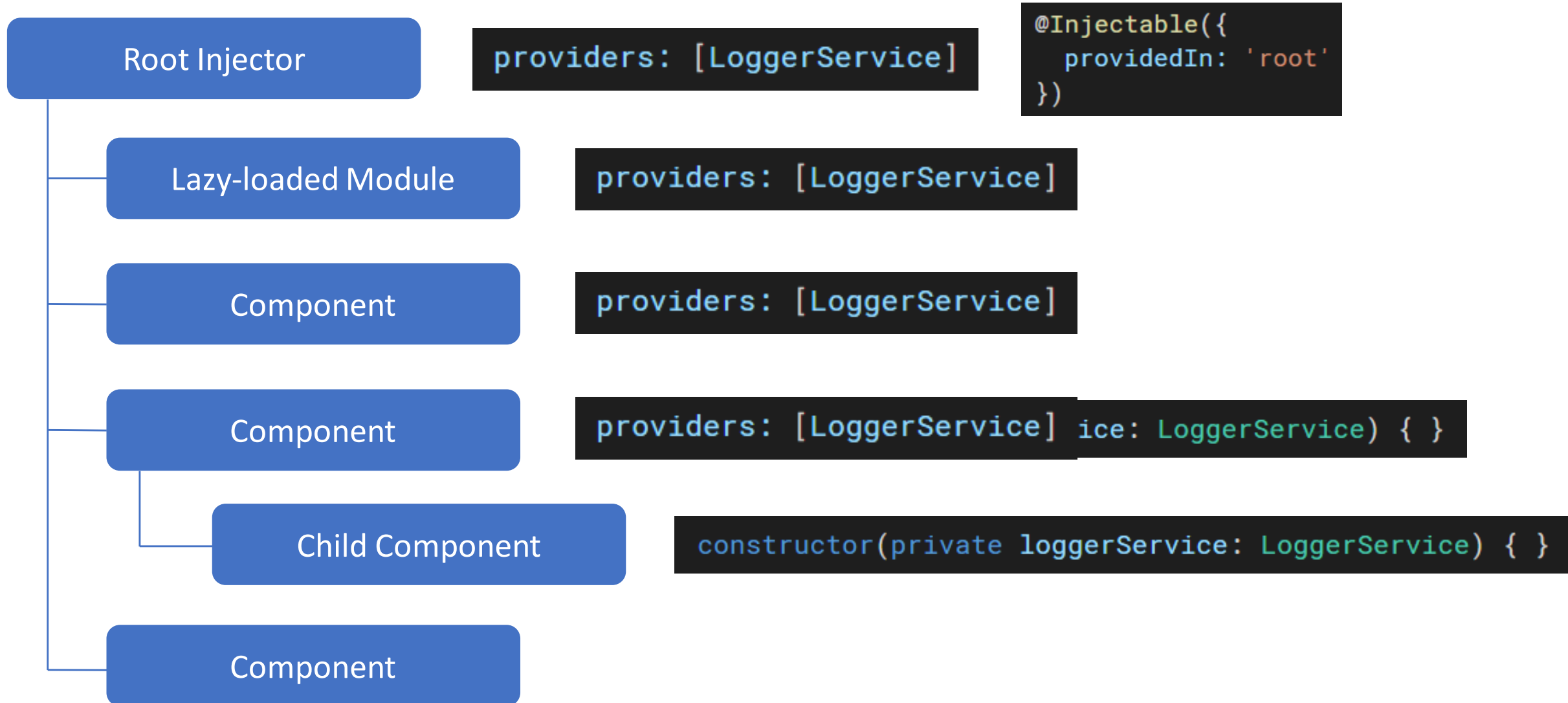
- Token**: A blue arrow points from the word "Token" to the `DataService` entry in the `providers` array.
- Class**: A blue arrow points from the word "Class" to the `LoggerService` in the `provide` property of the object provider.
- Recipe**: A blue arrow points from the word "Recipe" to the `useClass` property of the object provider.
- Class**: A blue arrow points from the word "Class" to the `LoggerService` in the `useClass` property of the object provider.

Provider Recipes

```
@Injectable({  
    providedIn: 'root'  
})  
export class DataService {  
    constructor(private http: HttpClient) { }  
}
```

Hierarchical Injectors

Hierarchical Injectors



Deciding Where to Provide Services

- Provide in the root injector if needed everywhere
- Provide component-specific services directly to component

Demo

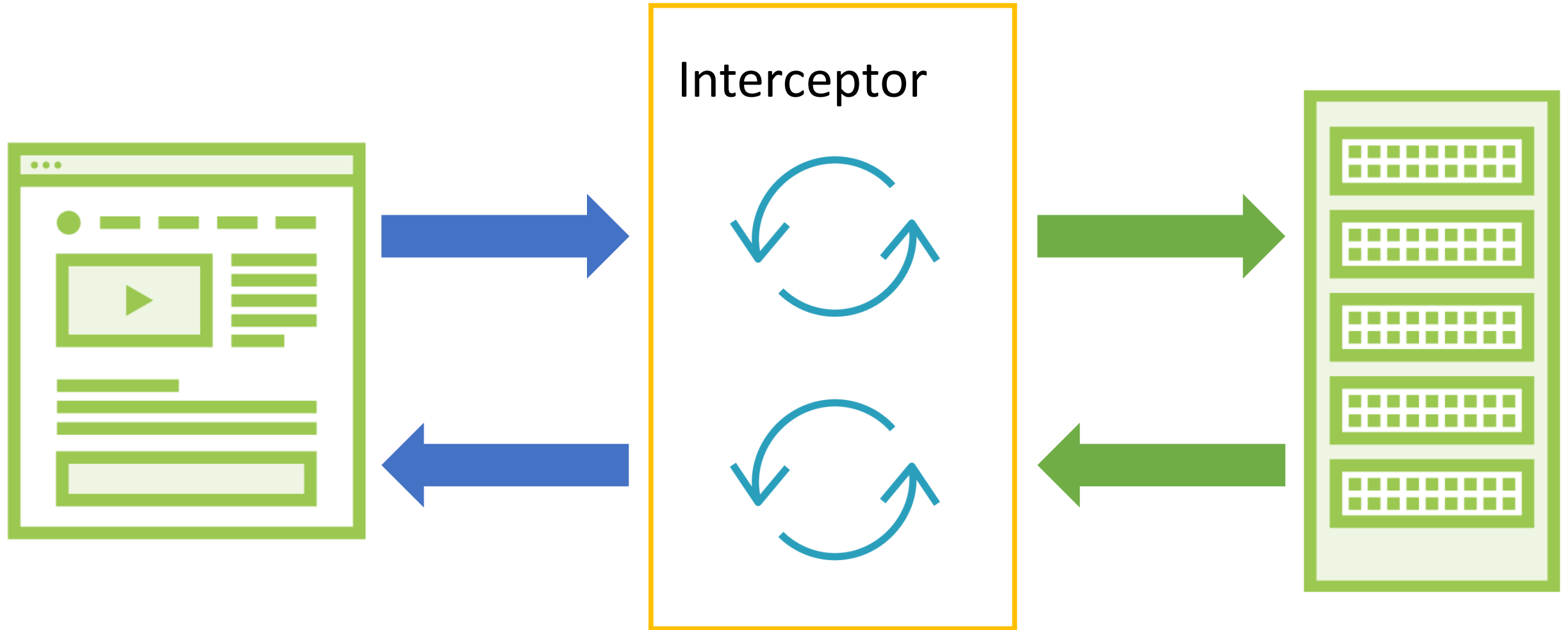
Providing services

Creating Interceptors

What Are Interceptors?

- Services
- Implement the `HttpInterceptor` interface
- Manipulate HTTP requests before they're sent to the server
- Manipulate HTTP responses before they're returned to your app

Intercepting Requests and Responses



Demo

Creating interceptors

Lab

Creating interceptors

Advanced Services and HTTP Communication

Links to Additional
Information

- Dependency Injection in Angular
 - <https://angular.io/guide/dependency-injection>
- HttpClient
 - <https://angular.io/guide/http>