Understanding and Configuring Dependency Injection

Brice Wilson

@brice_wilson www.BriceWilson.net

Overview

What is dependency injection?

Providers

Injectors

Deciding where to provide services

Creating a Dependency

```
export class DashboardComponent {

  dataService: DataService;

  constructor() {

    this.dataService = new DataService();

  }

}
```

Creating a Dependency

```
export class DashboardComponent {

  dataService: DataService;

  constructor() {

    this.dataService = new DataService();

  }

}
```

Injecting a Dependency

```
export class DashboardComponent {

  constructor(private dataService: DataService) {

  }

}
```

Injecting a Dependency

```
export class DashboardComponent {

  constructor(private dataService: DataService) {

  }

}
```

Injecting a Dependency

```
export class DashboardComponent {

  constructor(private dataService: DataService) {

  }

}
```

Why Is Dependency Injection Important?

Loosely coupled code

More flexible code

Easier to test

Providers

A *provider* is a recipe for delivering a service associated with a *token*.

Angular Documentation

Define dependencies with providers

https://angular.io/guide/dependency-injection-in-action#define-dependencies-with-providers

Provider Tokens

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

Provider Tokens

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

Provider Tokens

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

Provider Tokens

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


```
export class DashboardComponent {
  constructor(private dataService: DataService) { }
}
```

Provider Tokens

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

Token

Token

```
export class DashboardComponent {
  constructor(private dataService: DataService) { }
}
```

Provider Recipes

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

Provider Recipes

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

Token

Provider Recipes

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

Token

Recipe

Provider Recipes

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

Provider Recipes

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

Demo

Multiple ways to provide services

Injectors

The Roles of Injectors

- Deliver provided services when they're requested via constructor injection
- Maintain a single instance of each service provided
- Determine what to inject based on emitted metadata
- Delegate injection to parent injectors if necessary

```
{
  "compilerOptions": {
    "outDir": "./dist/out-tsc",
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "target": "es5"
  }
}
```

The Importance of Metadata

Provides information about parameters to injectors

Enabled with the “emitDecoratorMetadata” compiler option

```
{
  "compilerOptions": {
    "outDir": "./dist/out-tsc",
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "target": "es5"
  }
}
```

The Importance of Metadata

Provides information about parameters to injectors

Enabled with the “emitDecoratorMetadata” compiler option

```
{
  "compilerOptions": {
    "outDir": "./dist/out-tsc",
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "target": "es5"
  }
}
```

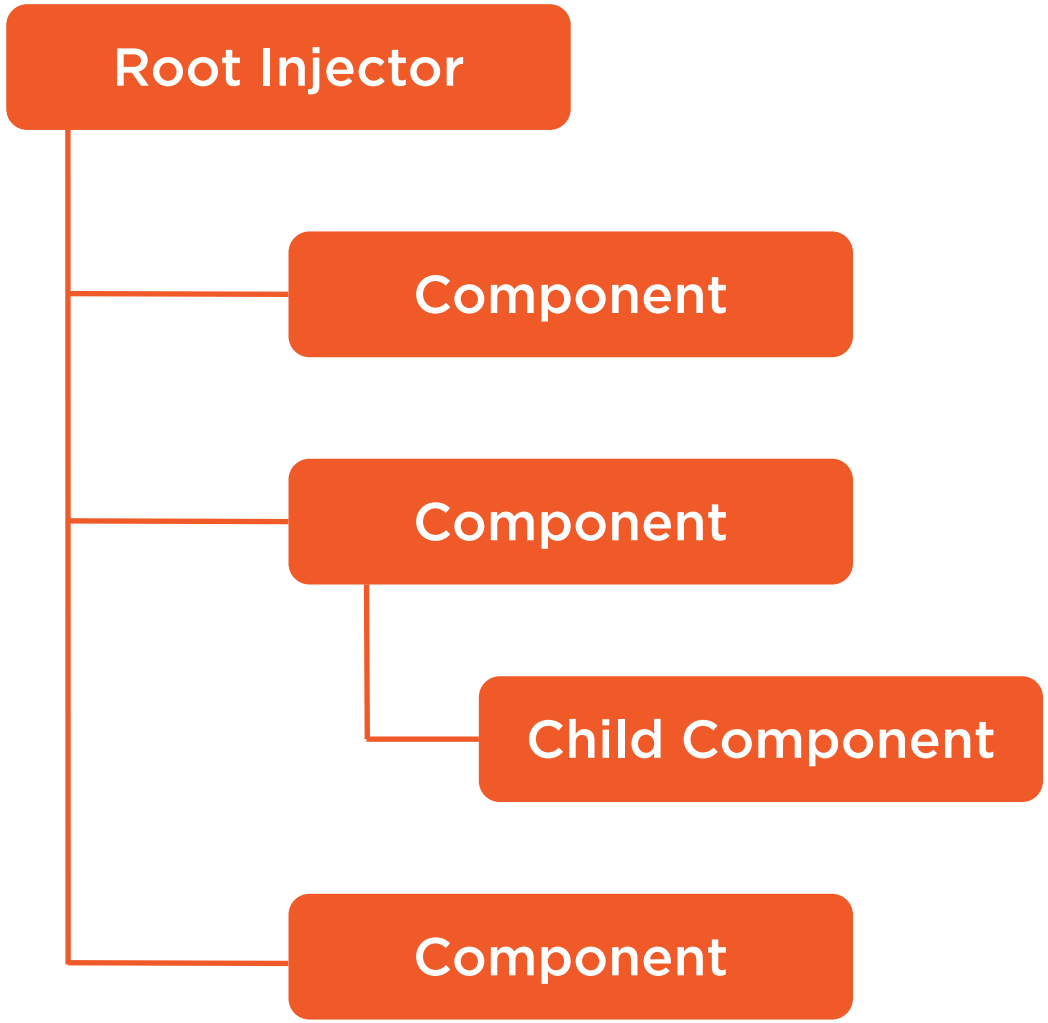
The Importance of Metadata

Provides information about parameters to injectors

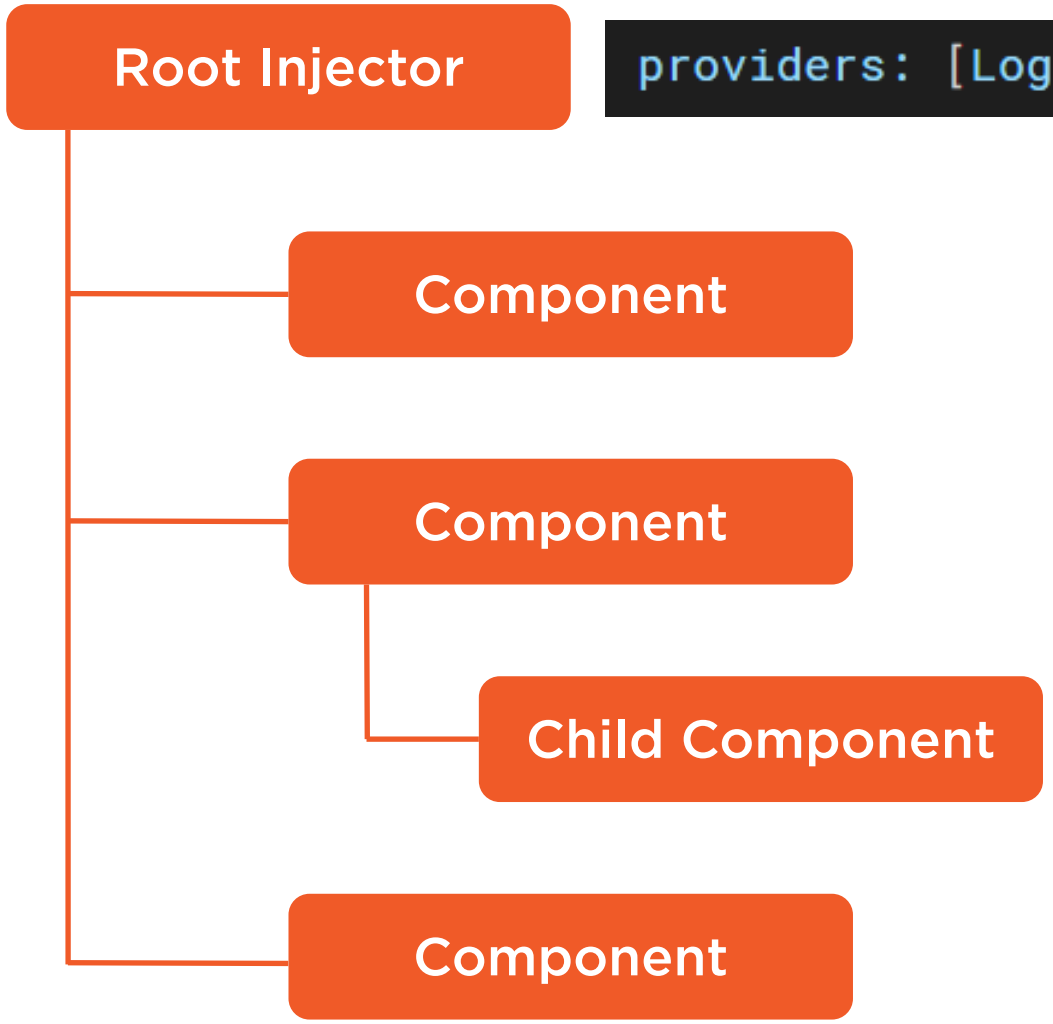
Enabled with the “emitDecoratorMetadata” compiler option

@Injectable() and @Component() decorator added to output metadata

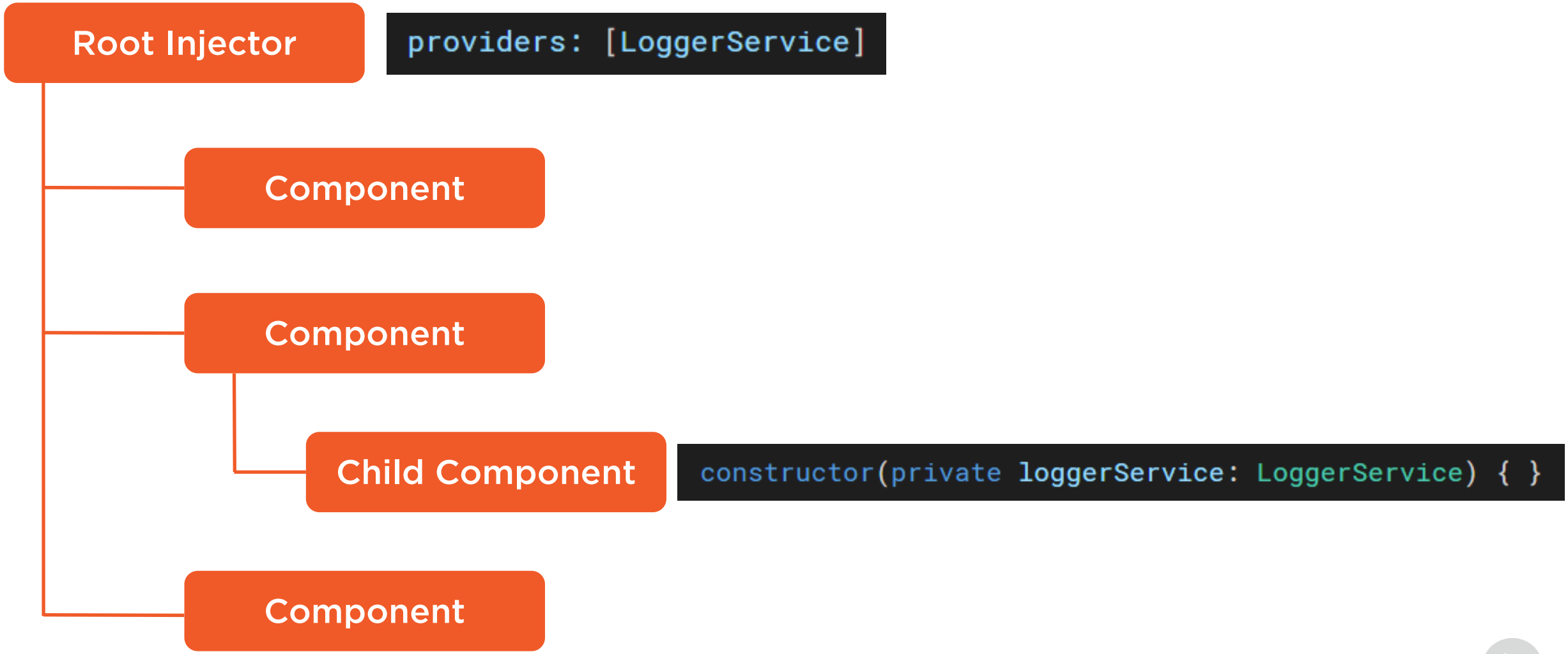
Hierarchical Injectors



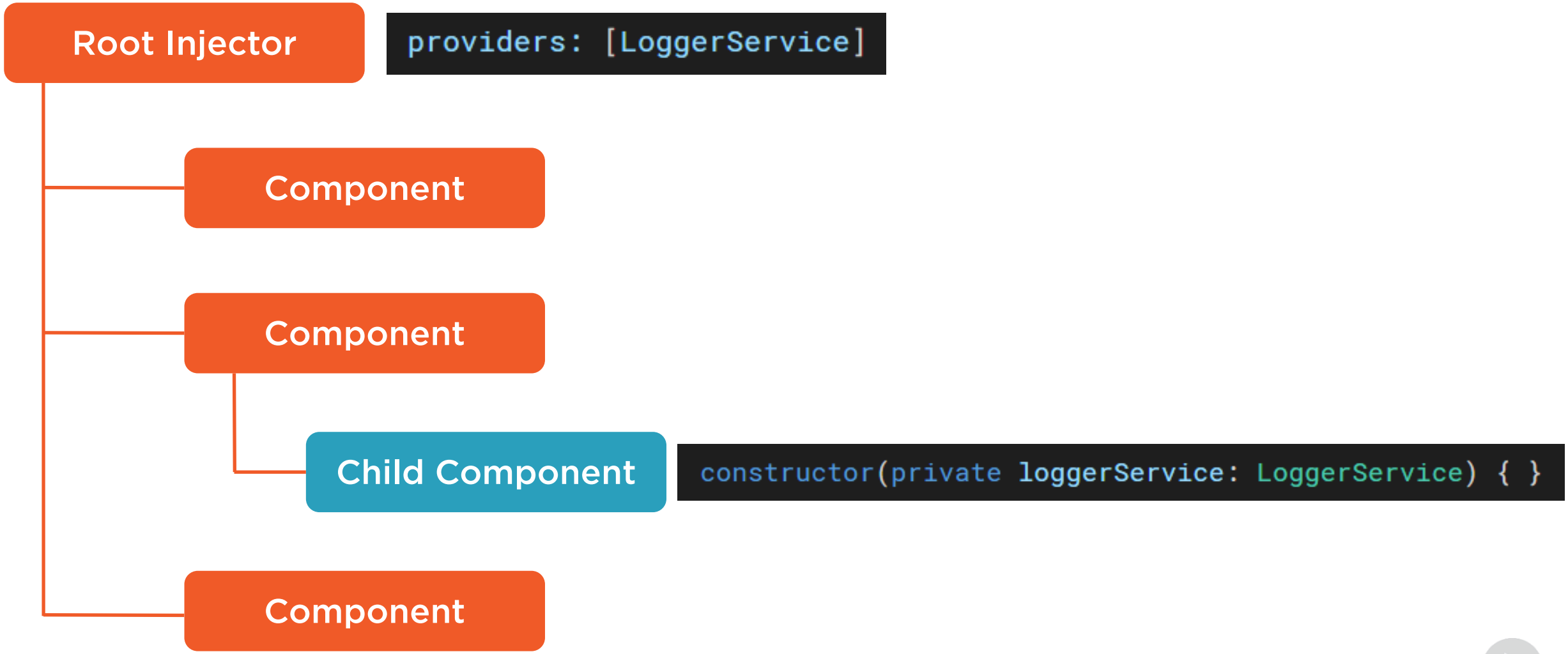
Hierarchical Injectors



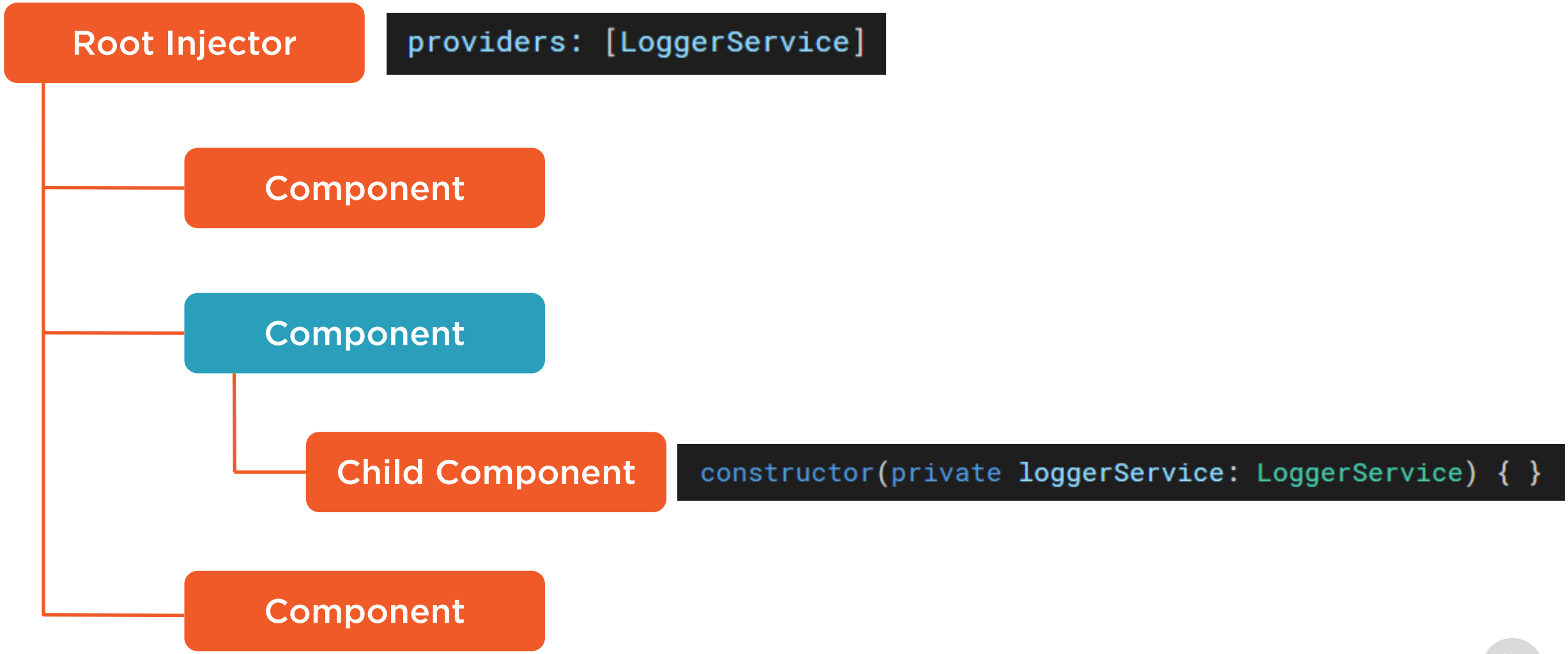
Hierarchical Injectors



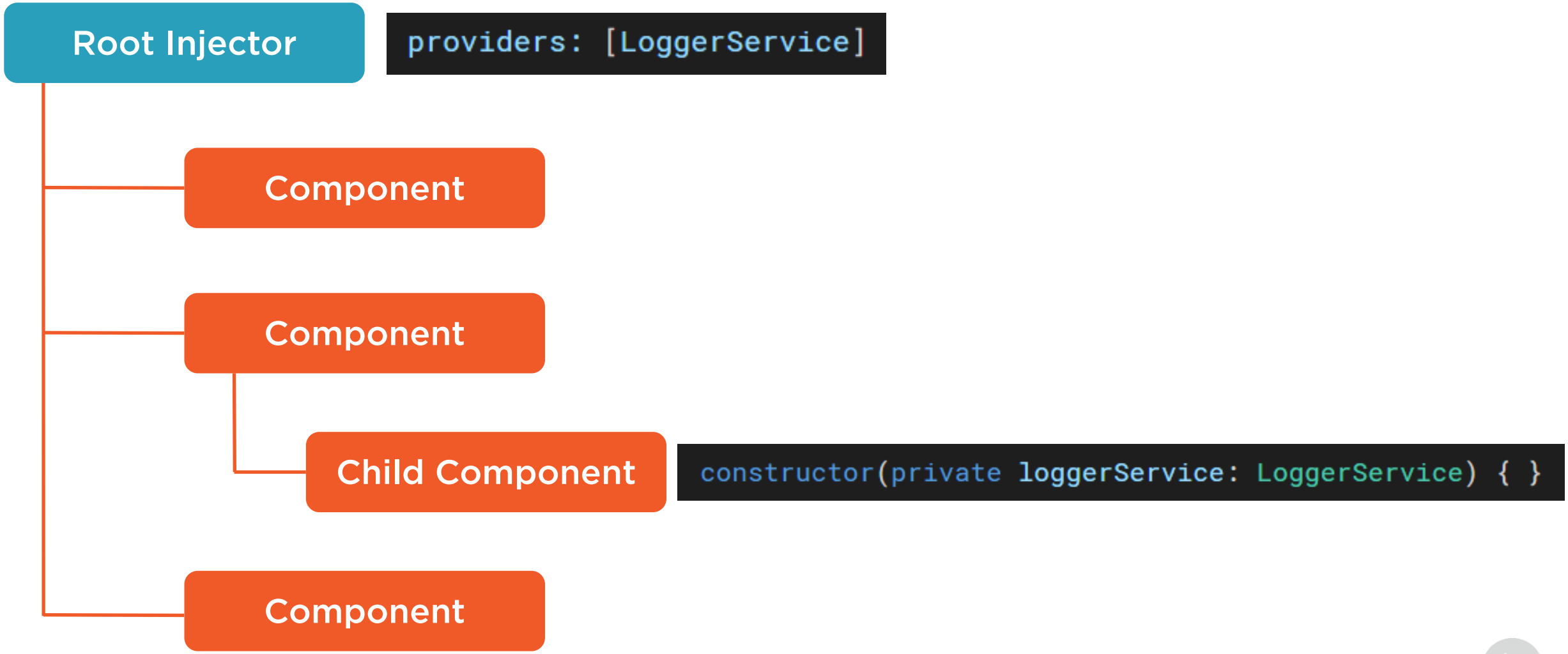
Hierarchical Injectors



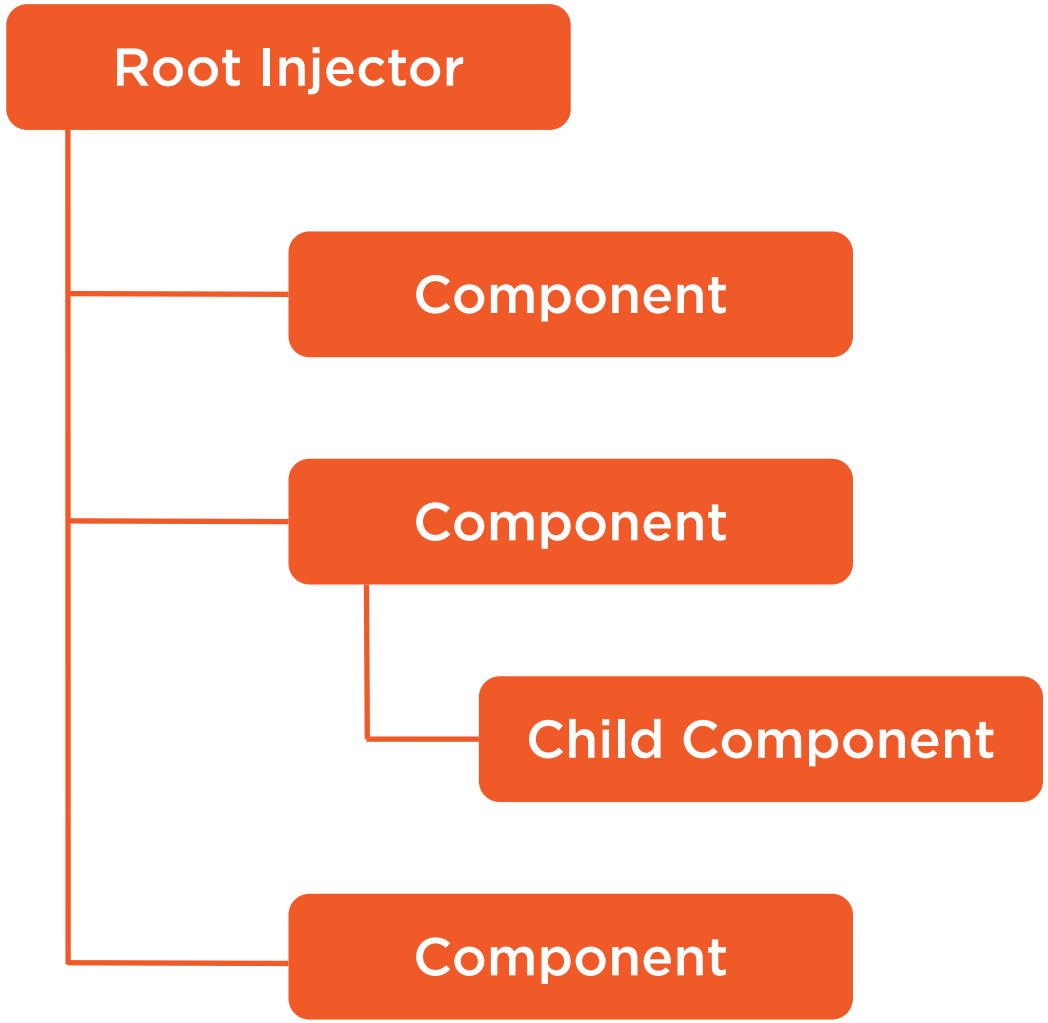
Hierarchical Injectors



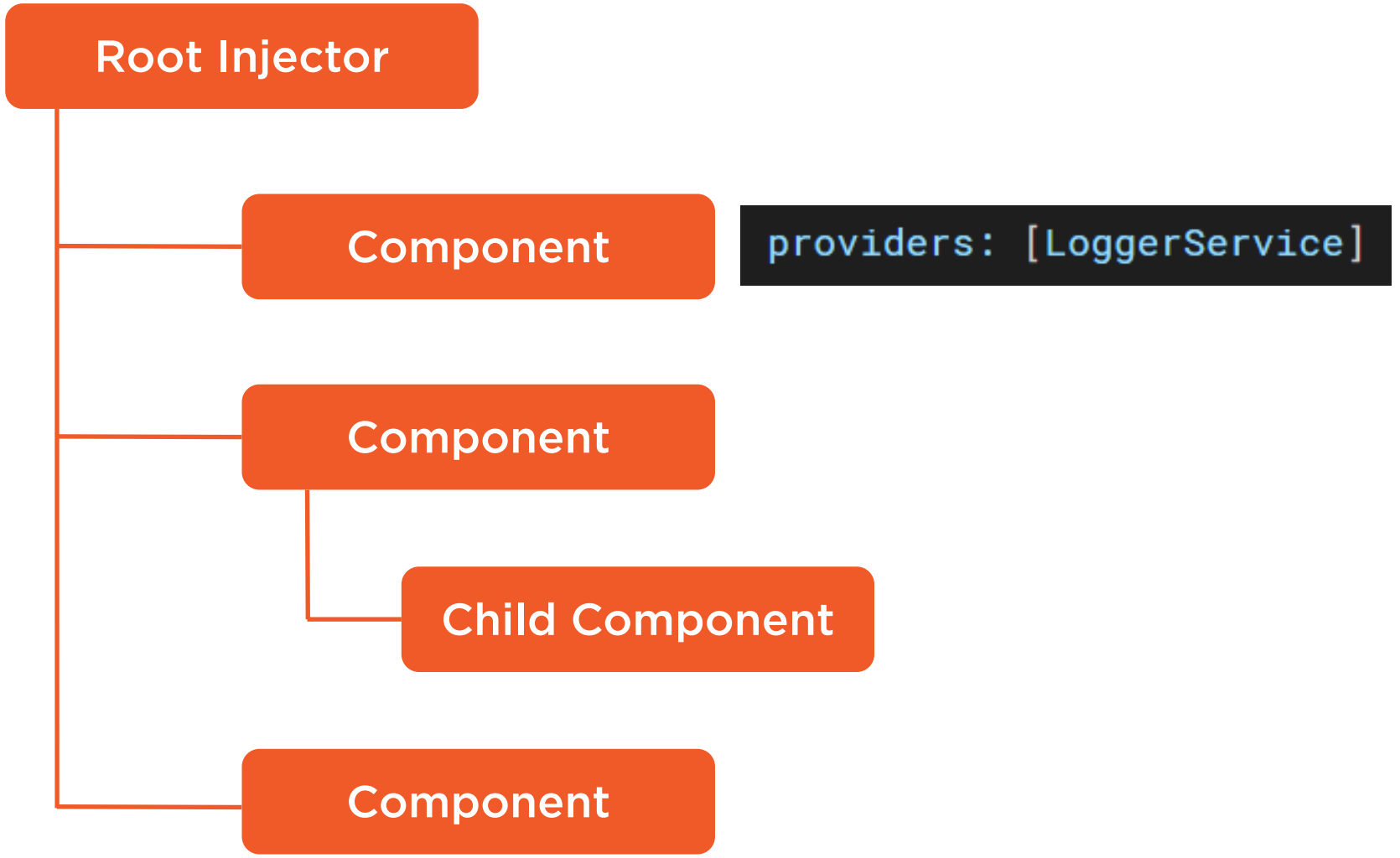
Hierarchical Injectors



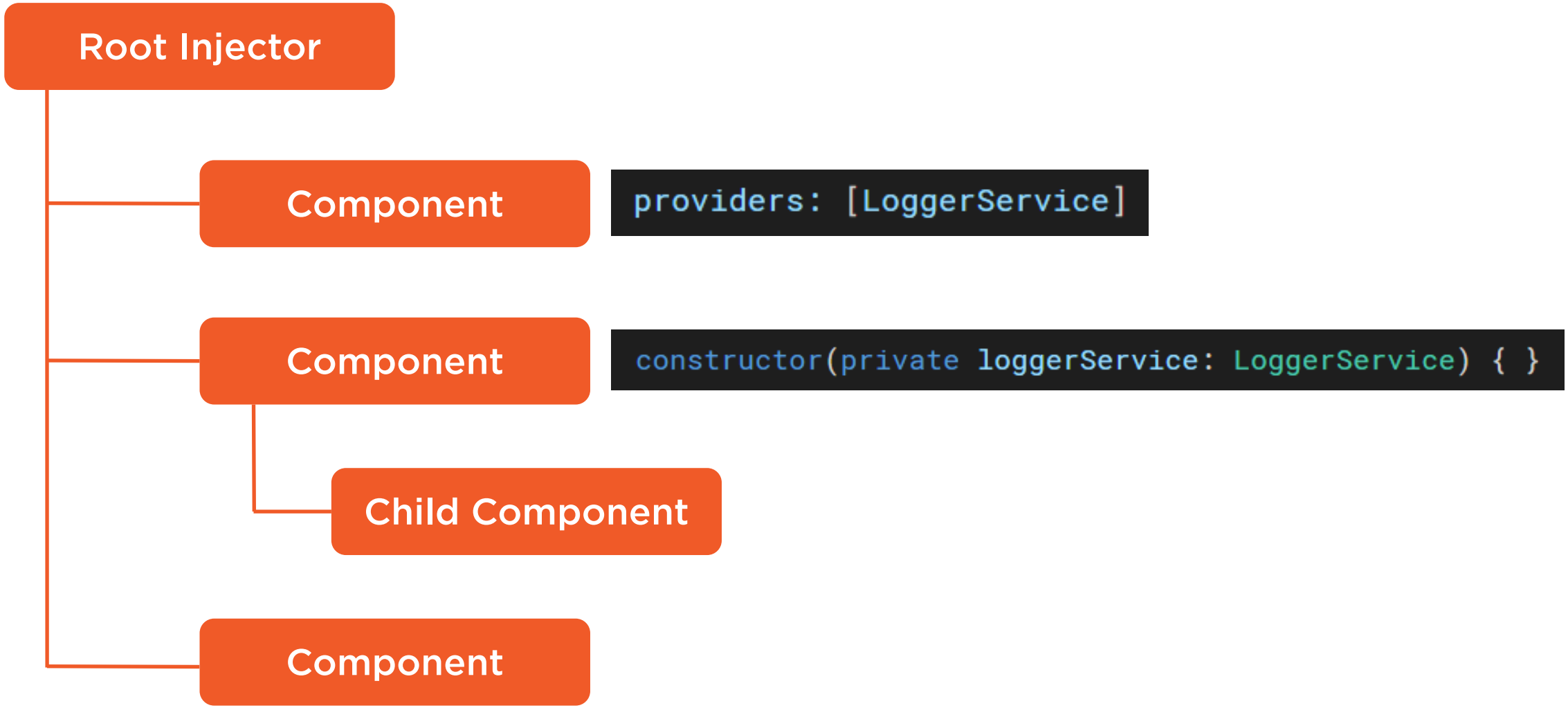
Hierarchical Injectors



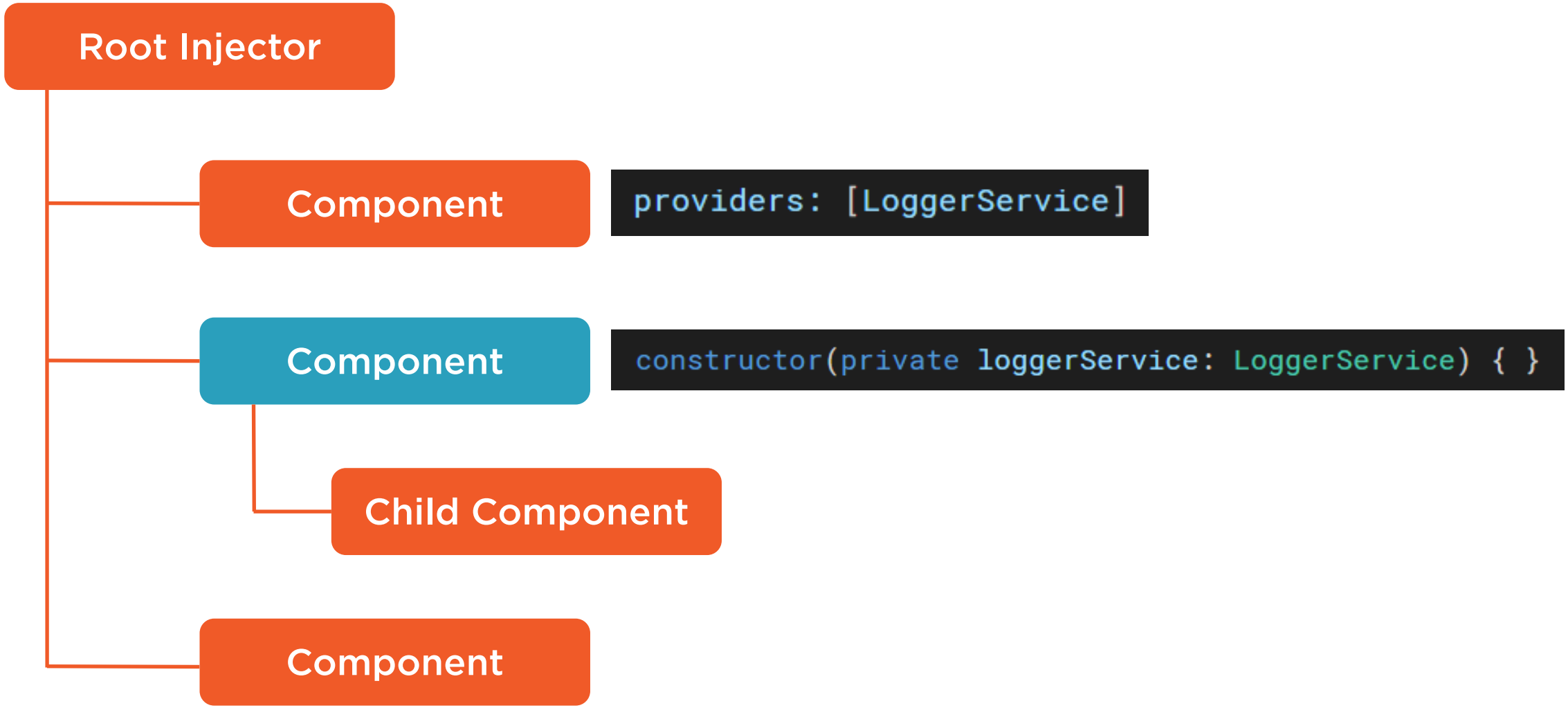
Hierarchical Injectors



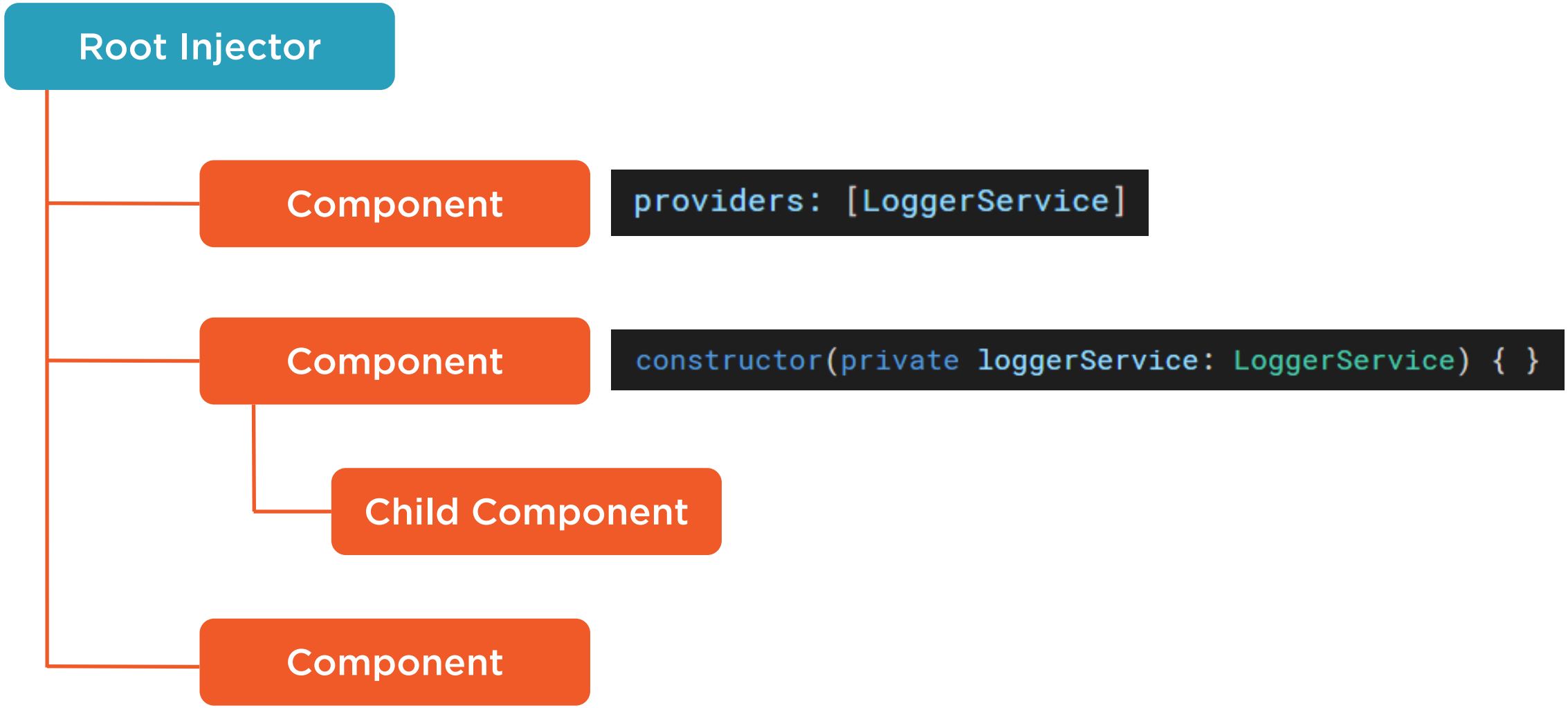
Hierarchical Injectors



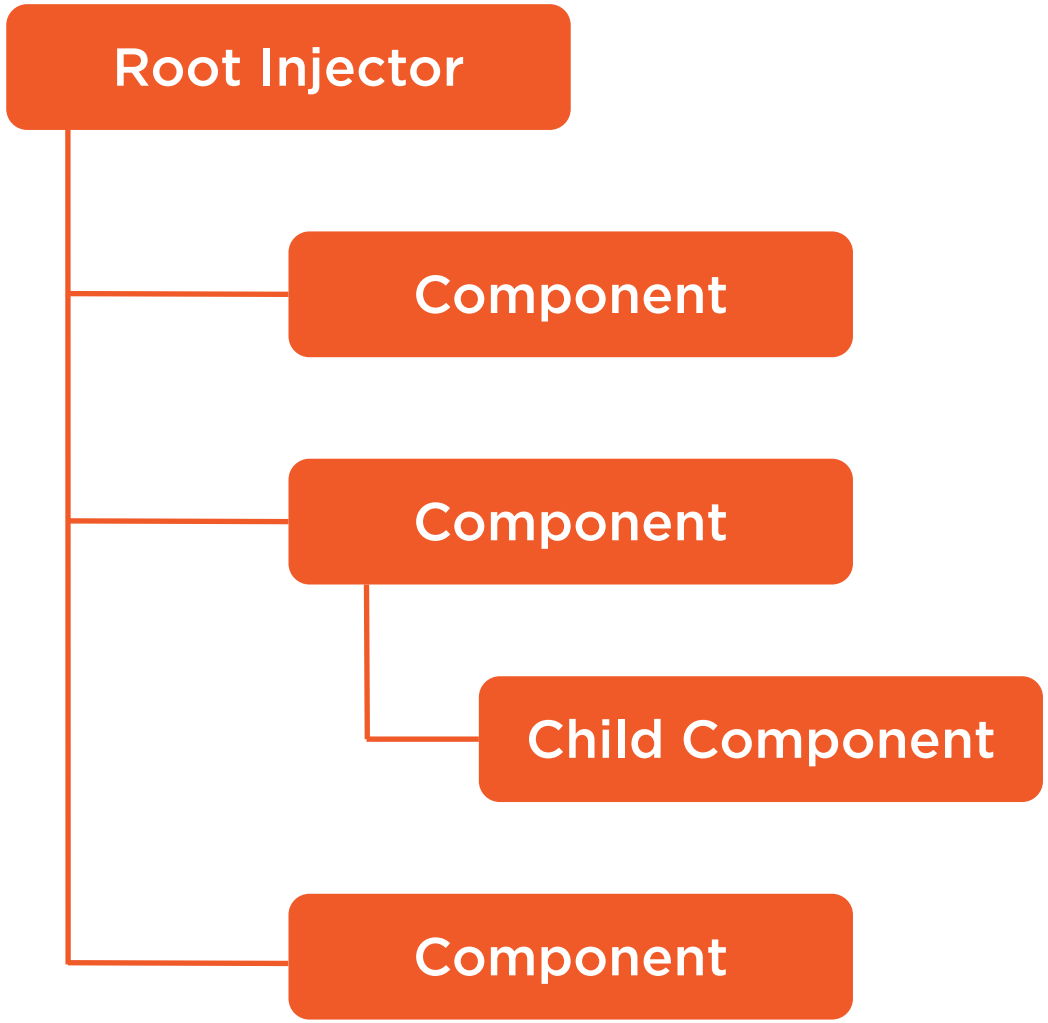
Hierarchical Injectors



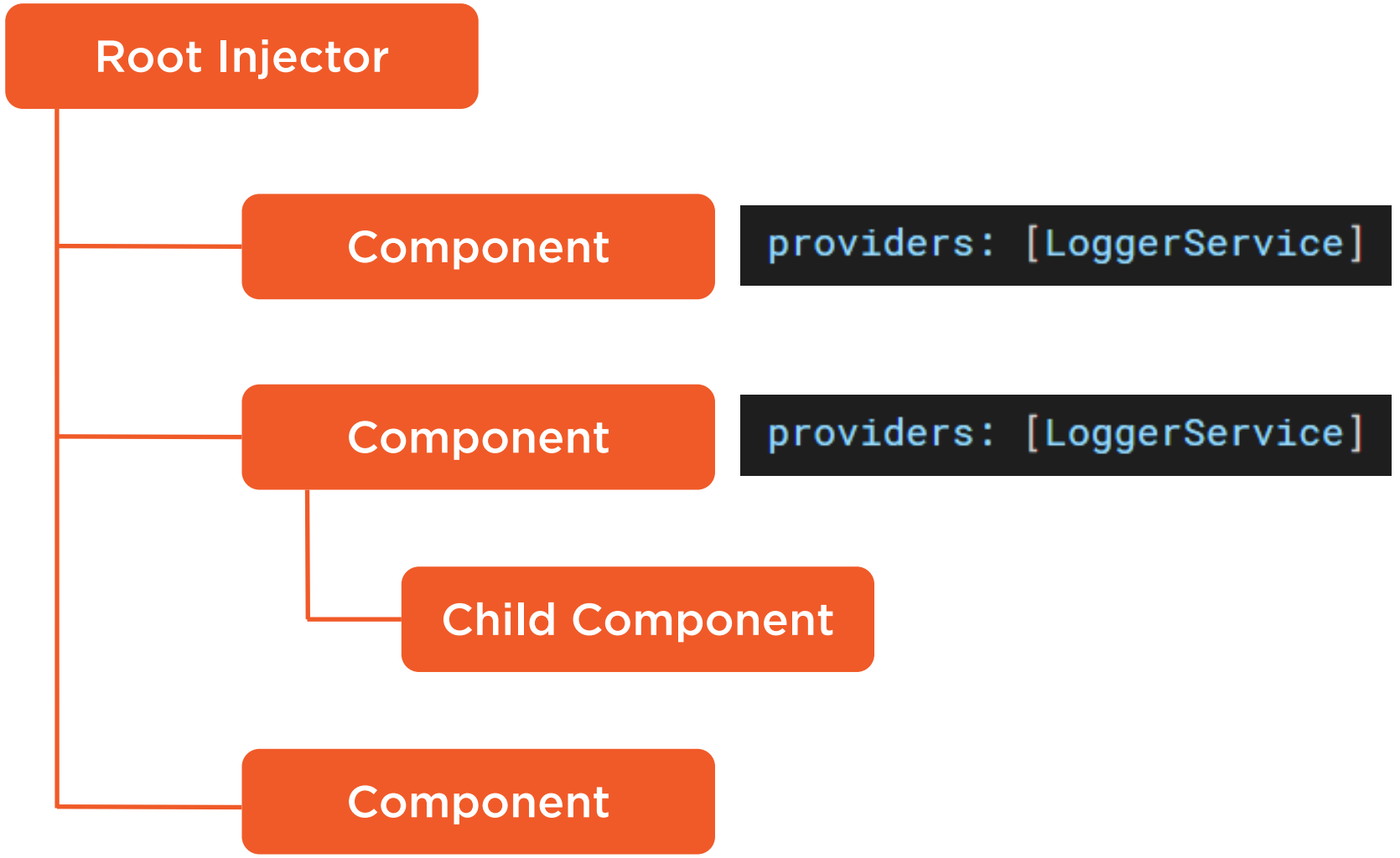
Hierarchical Injectors



Hierarchical Injectors



Hierarchical Injectors



Demo

Hierarchical injectors

Deciding Where to Provide Services

- Provide in an NgModule if needed everywhere
- Provide at the root AppModule rather the root AppComponent
- Provide component-specific services directly to component
- Consider creating a core module

Demo

Providing feature services

Demo

Creating a core module

Summary

- Providers
- Injectors
- Recipes for providers
- Injector hierarchy
- Deliver the right service at the right time