Service and DI

Agenda

01 What is Dependency Injections

02 Scenarios without Dependency Injections

03 Scenarios with Dependency Injections

04 What are services

05 Steps to implement Service

06 Demo

Dependency Injection

Dependency Injection

- Code without DI
- DI as a design pattern
- DI as framework which angular provides

Code without DI

```
class Engine{
    constructor(){}
}
class Tires{
    constructor(){}
}
```

```
class Car{
    engine;
    tires;
    constructor()
    {
        this.engine = new Engine();
        this.tires = new Tires();
    }
}
```

Code without DI cont.…

```
class Engine{
    constructor(newparameter){}
}
class Tires{
    constructor(){}
}
```

```
class Car{
    engine;
    tires;
    constructor()
    {
        this.engine = new Engine();
        this.tires = new Tires();
    }
}
```

Code with and without DI

Without DI

```
class Car{
    engine;
    tires;
    constructor()
    {
        this.engine = new Engine();
        this.tires = new Tires();
    }
}
```

With DI

```
class Car{
    engine;
    tires;
    constructor(engine, tires)
    {
        this.engine = engine;
        this.tires = tires;
    }
}
```

Code with DI

```
var myEngine = new Engine();
var myTires = new Tires();
var myCar = new Car(myEngine, myTires);
```

```
var myEngine = new Engine(parameter);
var myTires = new Tires();
var myCar = new Car(myEngine, myTires);
```

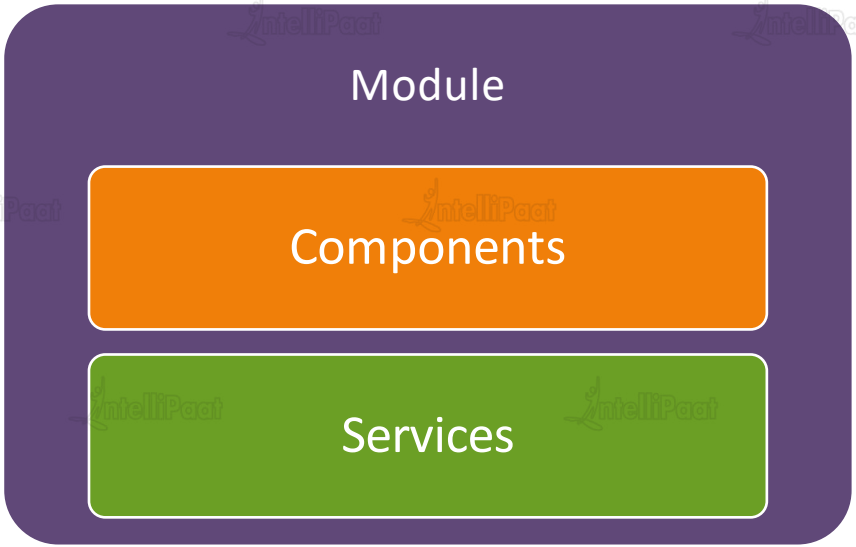
```
var myEngine = new Engine(parameter);
var myTires = new Tires(parameter);
var myCar = new Car(myEngine, myTires);
```

DI as a design pattern

- Dependency Injection is a coding pattern in which a class receives its dependencies from external resources rather that creating them itself.

Services

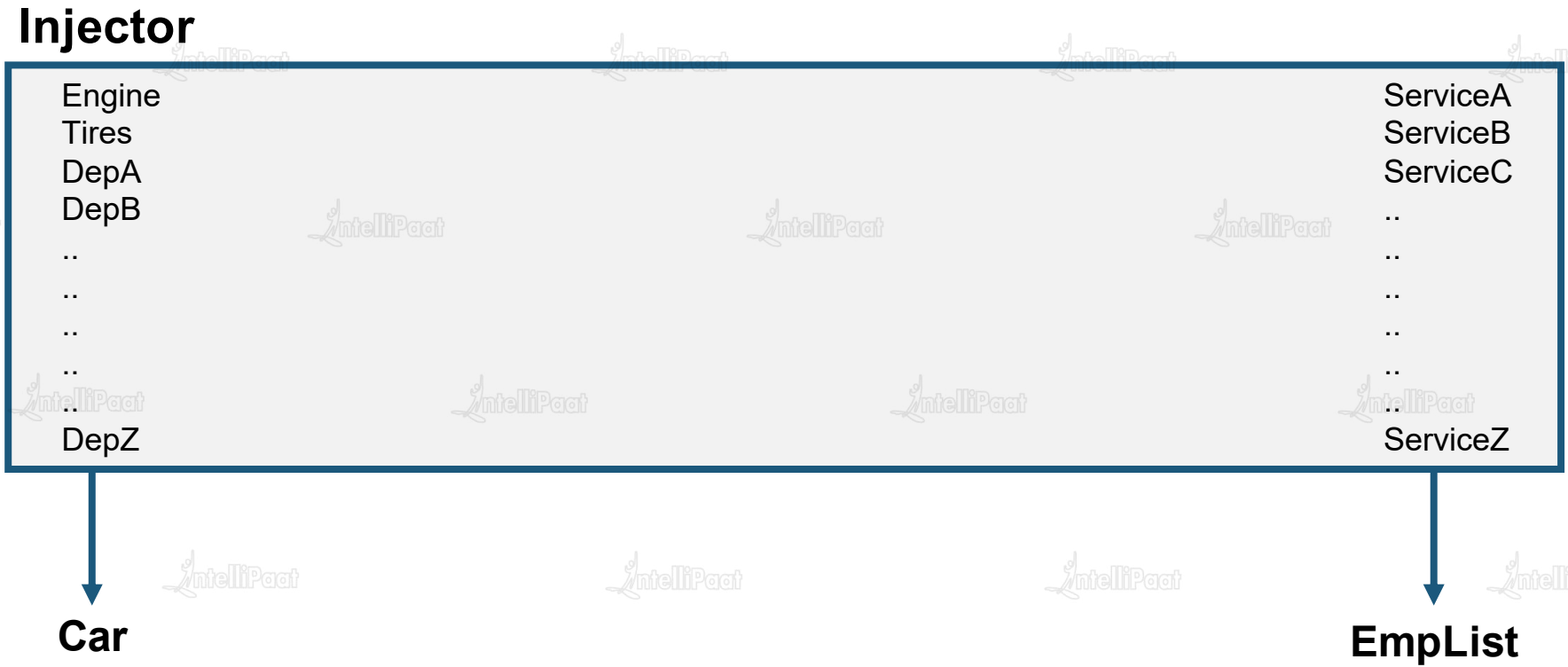
Services



Services

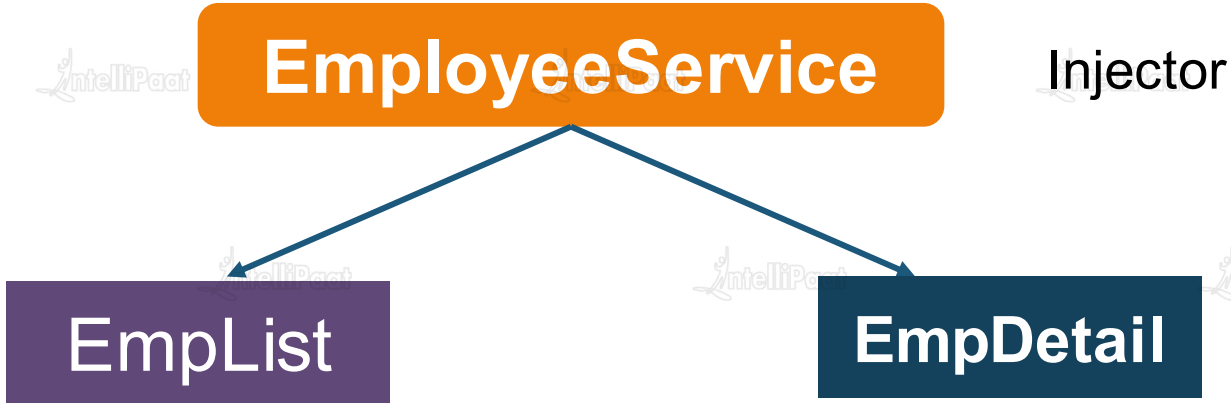
- Service is a class
- Share the data
- Implement application logic
- Services are a great way to share information among classes that *don't know each other*
- Components shouldn't fetch or save data directly and they certainly shouldn't knowingly present fake data. They should focus on presenting data and delegate data access to a service.

DI as framework

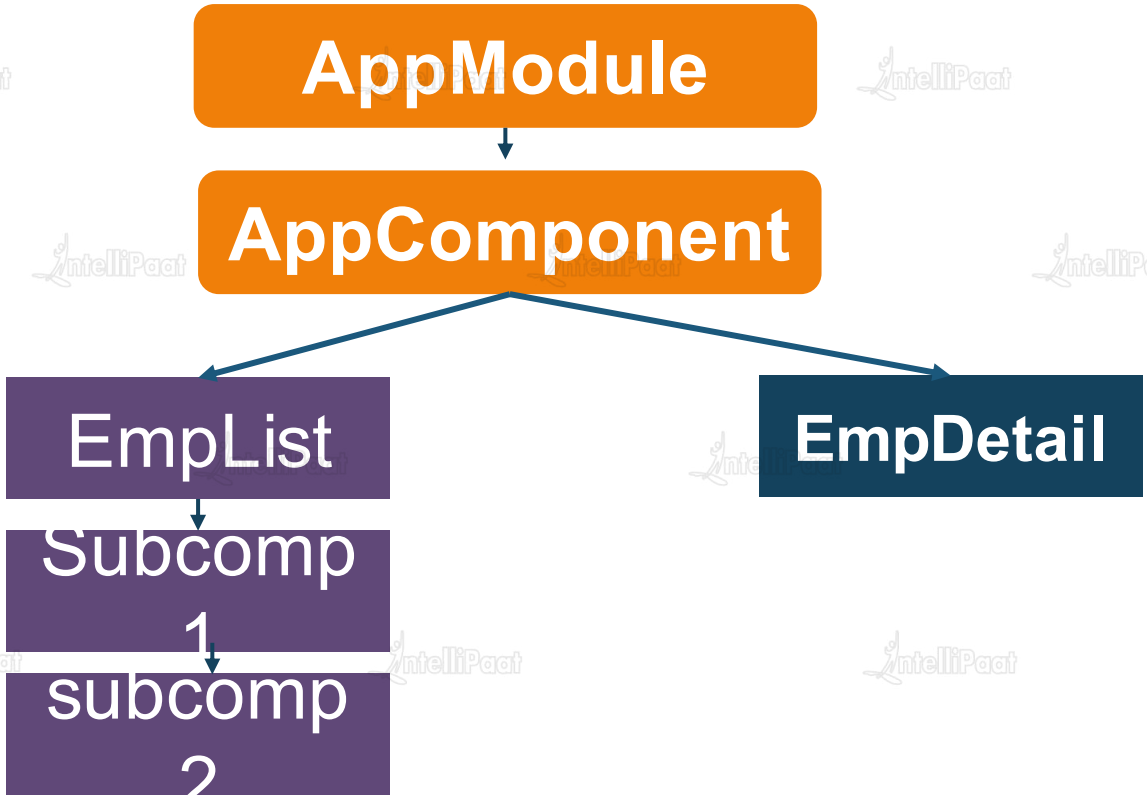


Steps to implement DI in Angular

- Define the EmployeeService class
- Register with Injector
- Declare as dependency in EmpList and EmpDetails



Steps to implement DI in Angular



Demo

India : +91-7847955955

US : 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24X7 Chat with our Course Advisor