

# Services and Dependency Injection

---



**Deborah Kurata**

CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | [blogs.msmvps.com/deborahk/](https://blogs.msmvps.com/deborahk/)





Products

Logging



# Service

A class with a focused purpose.

Used for features that:

- Are independent from any particular component
- Provide shared data or logic across components
- Encapsulate external interactions



# Module Overview



**How Does It Work?**

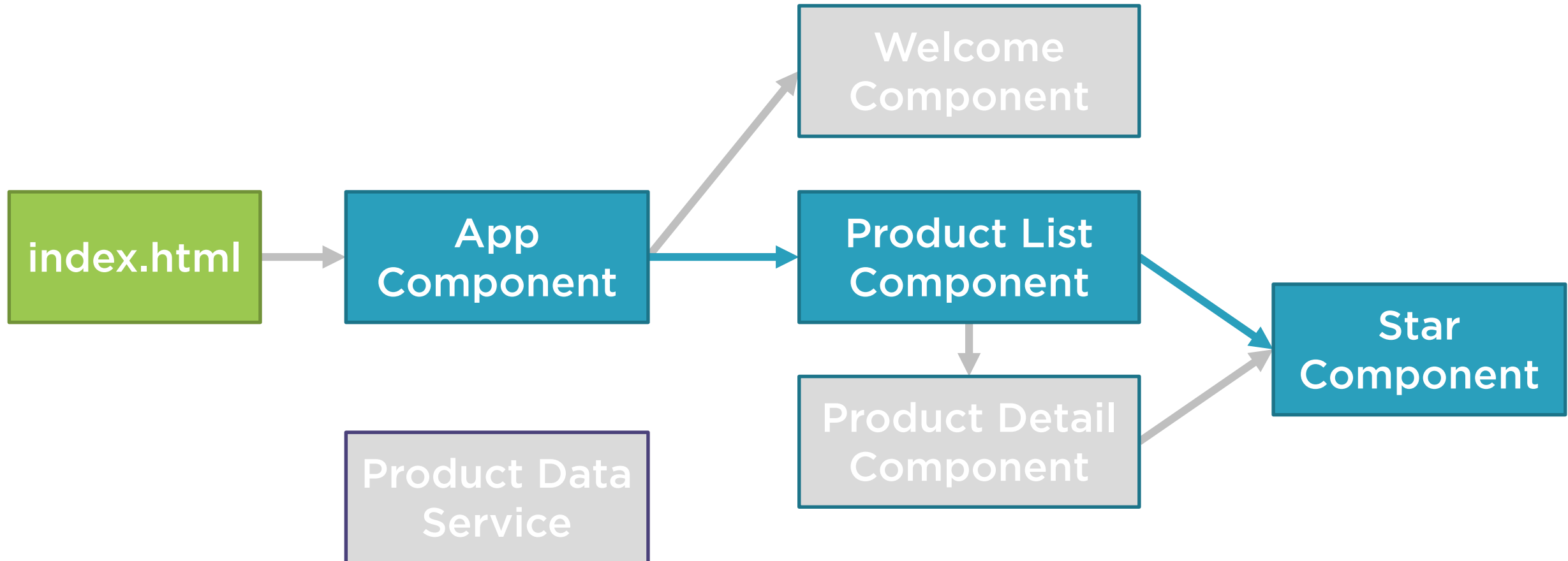
**Building a Service**

**Registering the Service**

**Injecting the Service**



# Application Architecture



# How Does It Work?

## Service

```
export class myService {}
```

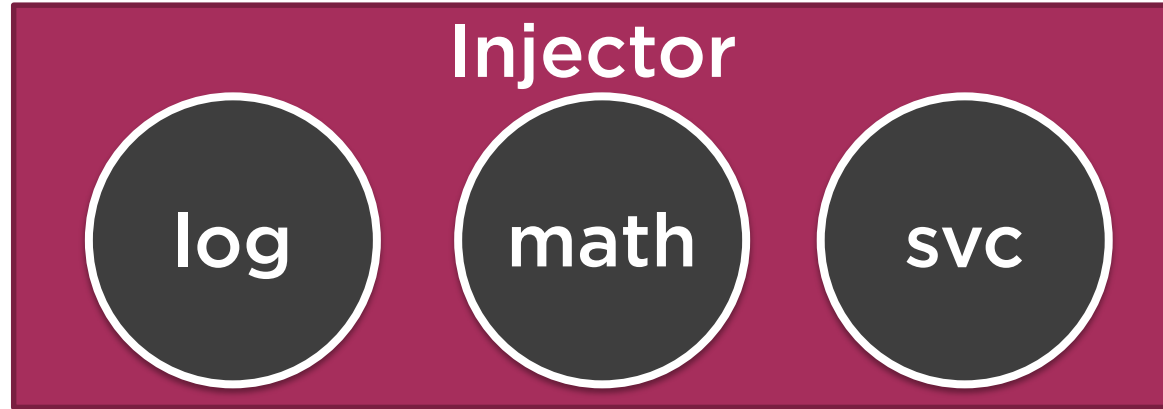
## Component

```
let svc = new myService();
```

**svc**



# How Does It Work?



## Service

```
export class myService {}
```

## Component

```
constructor(private _myService) {}
```



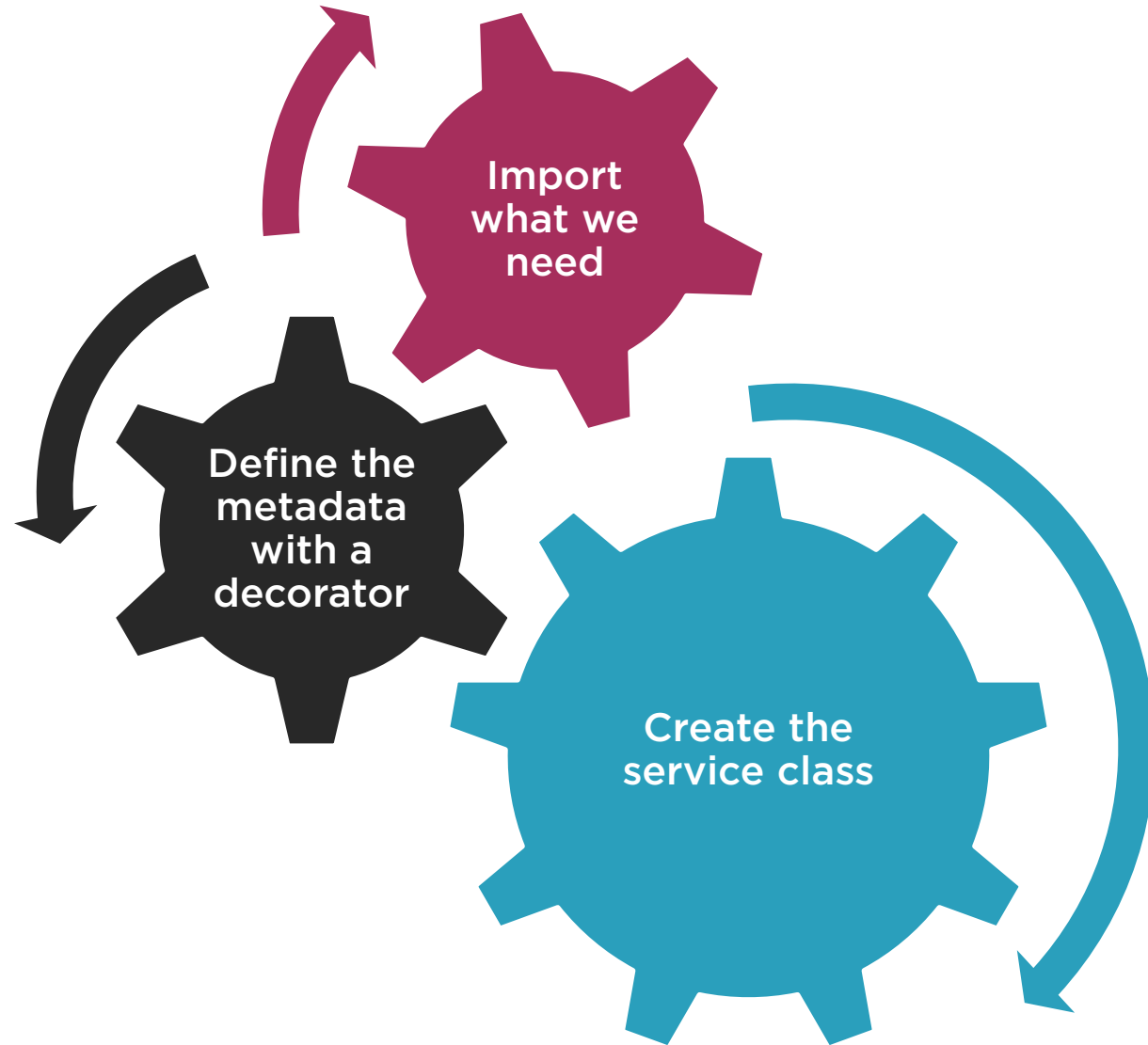
# Dependency Injection

A coding pattern in which a class receives the instances of objects it needs (called **dependencies**) from an external source rather than creating them itself.





# Building a Service



# Building a Service

product.service.ts

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

@Injectable()
export class ProductService {

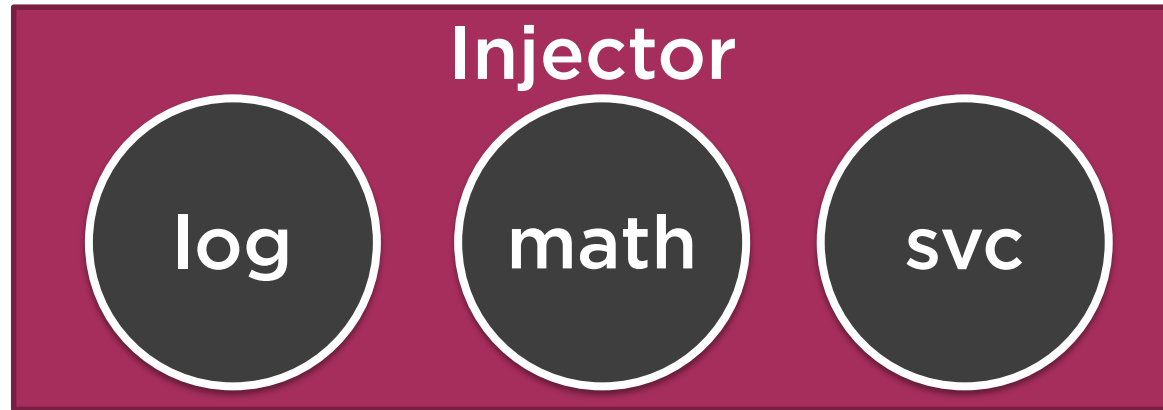
  getProducts(): IProduct[] {

  }

}
```



# Registering the Service



## Service

```
export class myService {}
```

## Component

```
constructor(private _myService) {}
```



# Registering a Service

Injector

log

math

svc

## Register a provider

- Code that can create or return a service
- Typically the service class itself

## Define in component OR Angular module metadata

## Registered in component:

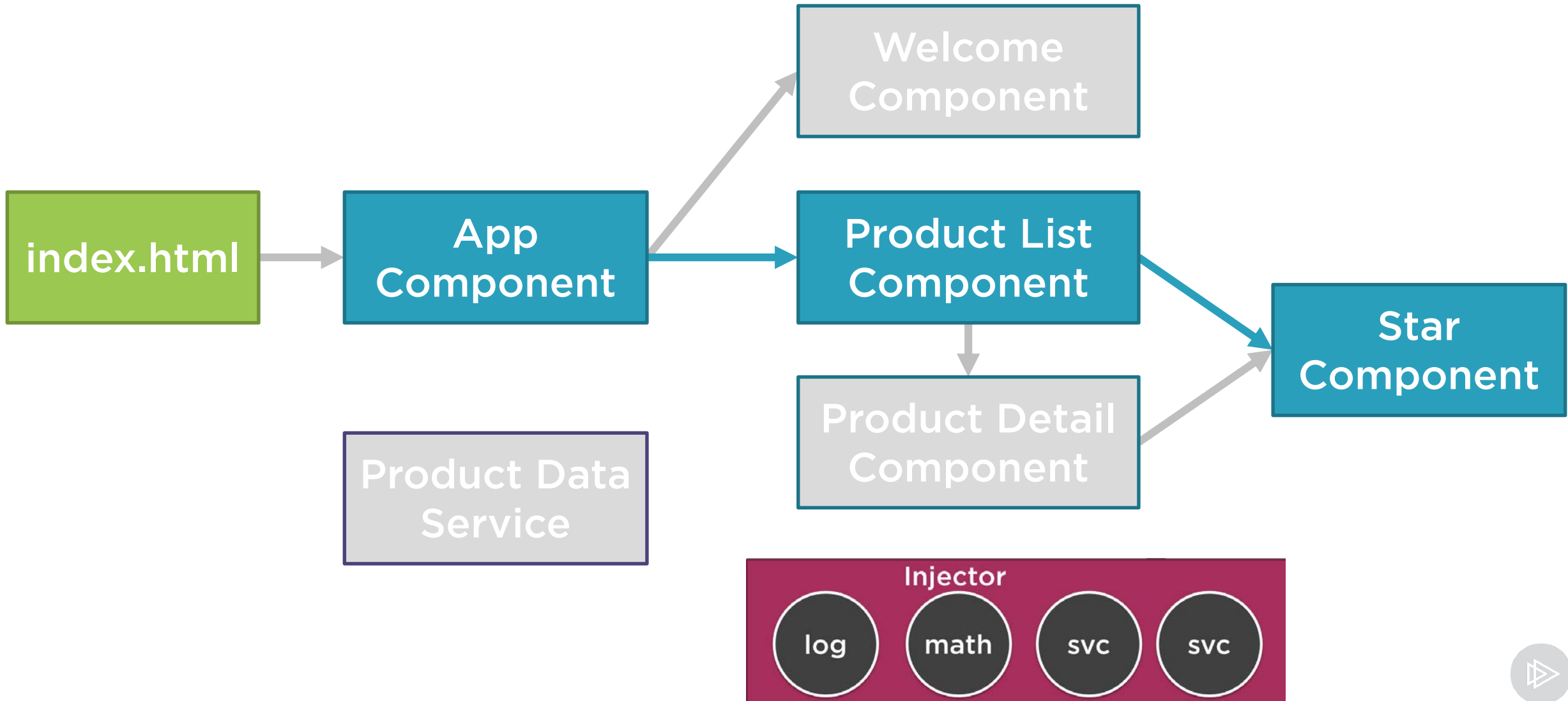
- Injectable to component AND its children

## Registered in Angular module:

- Injectable everywhere in the application



# Application Architecture



# Registering a Provider

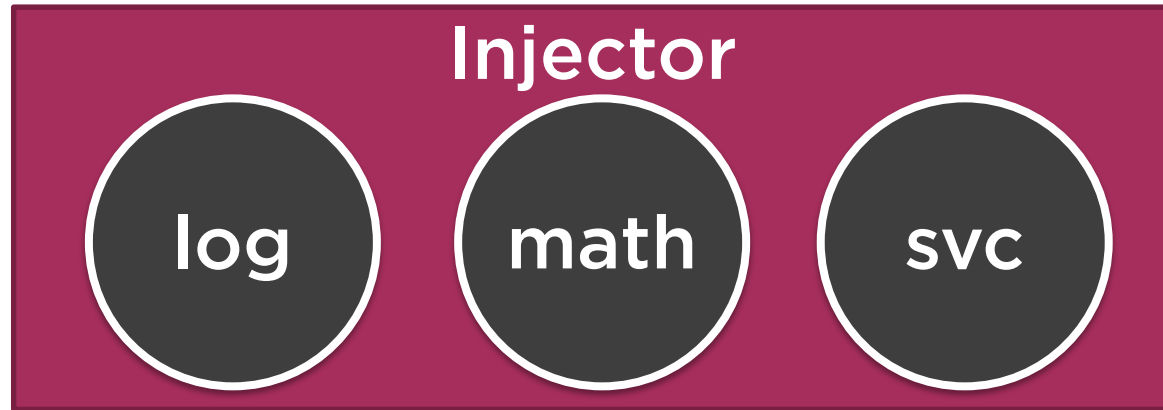
app.component.ts

```
...
import { ProductService } from '../products/product.service';

@Component({
  selector: 'pm-root',
  template: `
    <div><h1>{{pageTitle}}</h1>
      <pm-products></pm-products>
    </div>
  `,
  providers: [ProductService]
})
export class AppComponent { }
```



# Injecting the Service



## Service

```
export class myService {}
```

## Component

```
constructor(private _myService) {}
```



# Injecting the Service

## product-list.component.ts

```
...

@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {

  constructor() {
  }

}
```





# Injecting the Service

## product-list.component.ts

```
...
import { ProductService } from '../product.service';

@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {
  private _productService;
  constructor(productService: ProductService) {
    _productService = productService;
  }
}
```



# Injecting the Service

## product-list.component.ts

```
...
import { ProductService } from '../product.service';

@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {

  constructor(private _productService: ProductService) {
  }

}
```



# Checklist: Creating a Service



## Service class

- Clear name
- Use PascalCasing
- Append "Service" to the name
- export keyword

## Service decorator

- Use Injectable
- Prefix with @; Suffix with ()

## Import what we need

# Checklist: Registering a Service in a Component



## Select the appropriate level in the hierarchy

- Root component if service is used throughout the application
- Specific component if only that component uses the service
- Otherwise, common ancestor

## Component metadata

- Set the providers property
- Pass in an array

## Import what we need

# Checklist: Dependency Injection



\_\_\_\_\_



\_\_\_\_\_



\_\_\_\_\_

Specify the service as a dependency

Use a constructor parameter

Service is injected when component is  
instantiated



# Summary



**How Does It Work?**

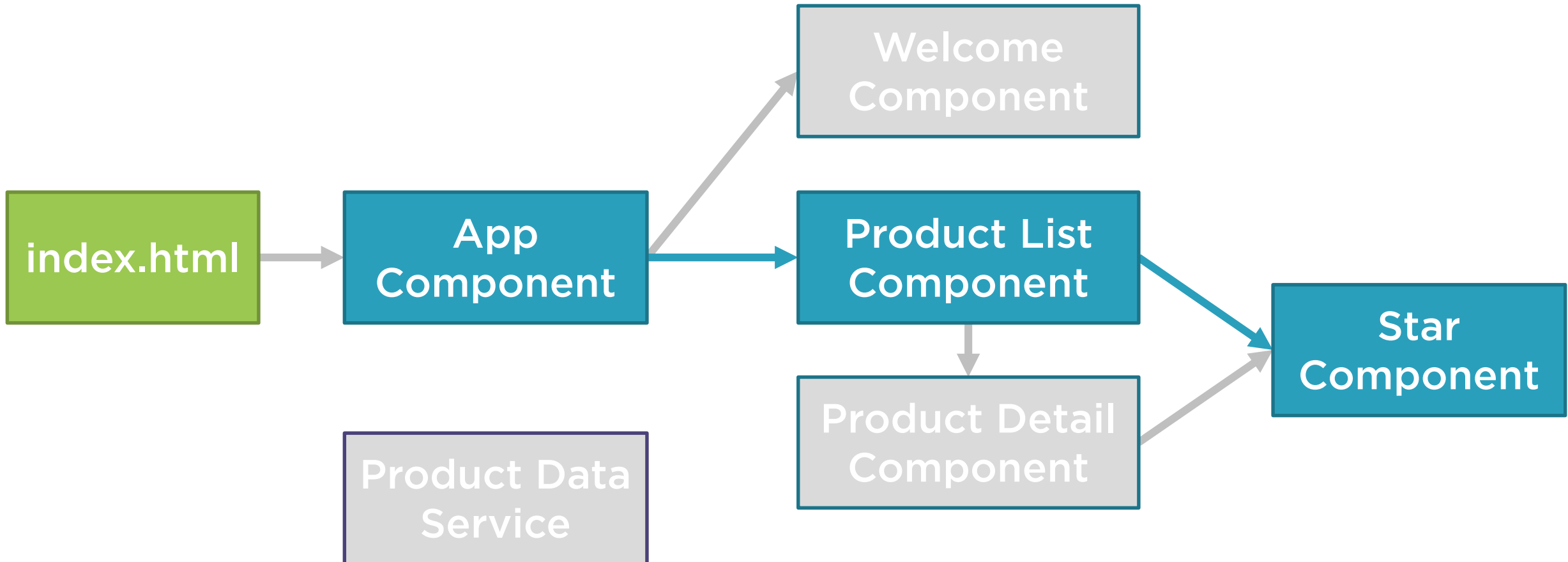
**Building a Service**

**Registering the Service**

**Injecting the Service**



# Application Architecture



# Application Architecture

