AngularJS

Services and dependency injection

Service

A class with a focused purpose

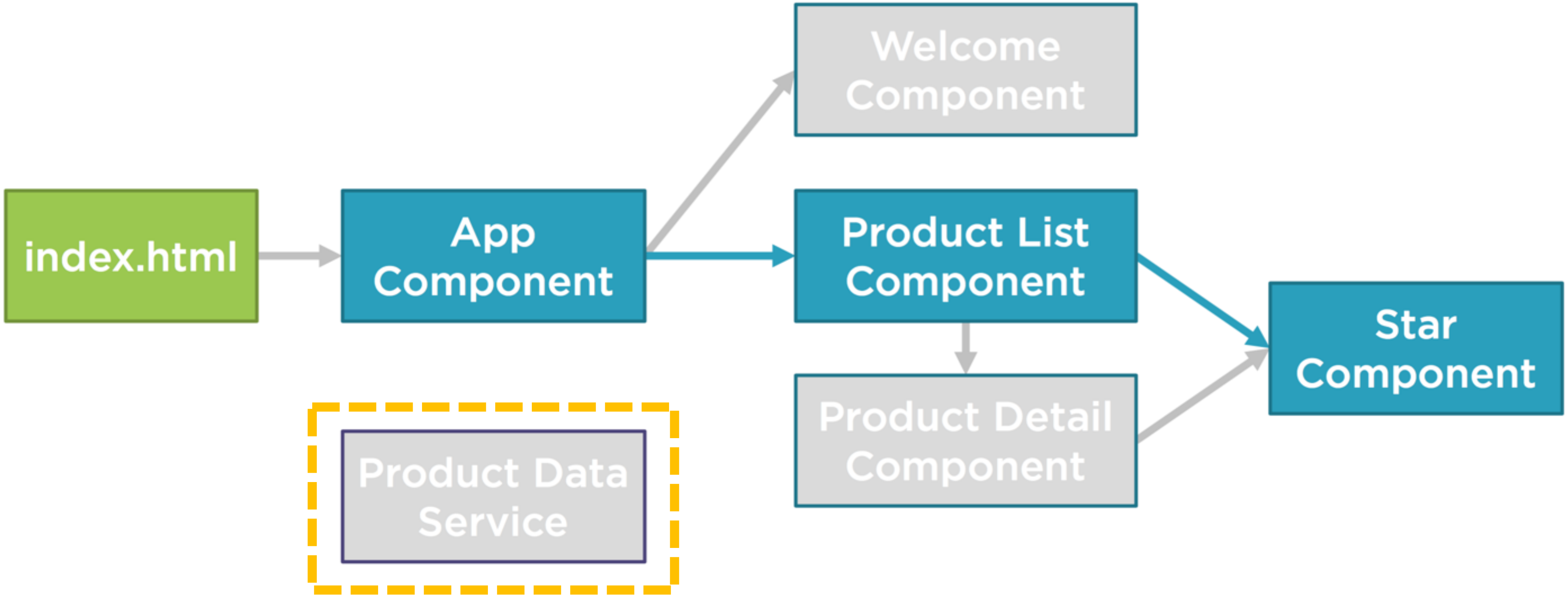
Used for features that:

- Are independent from ay 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?

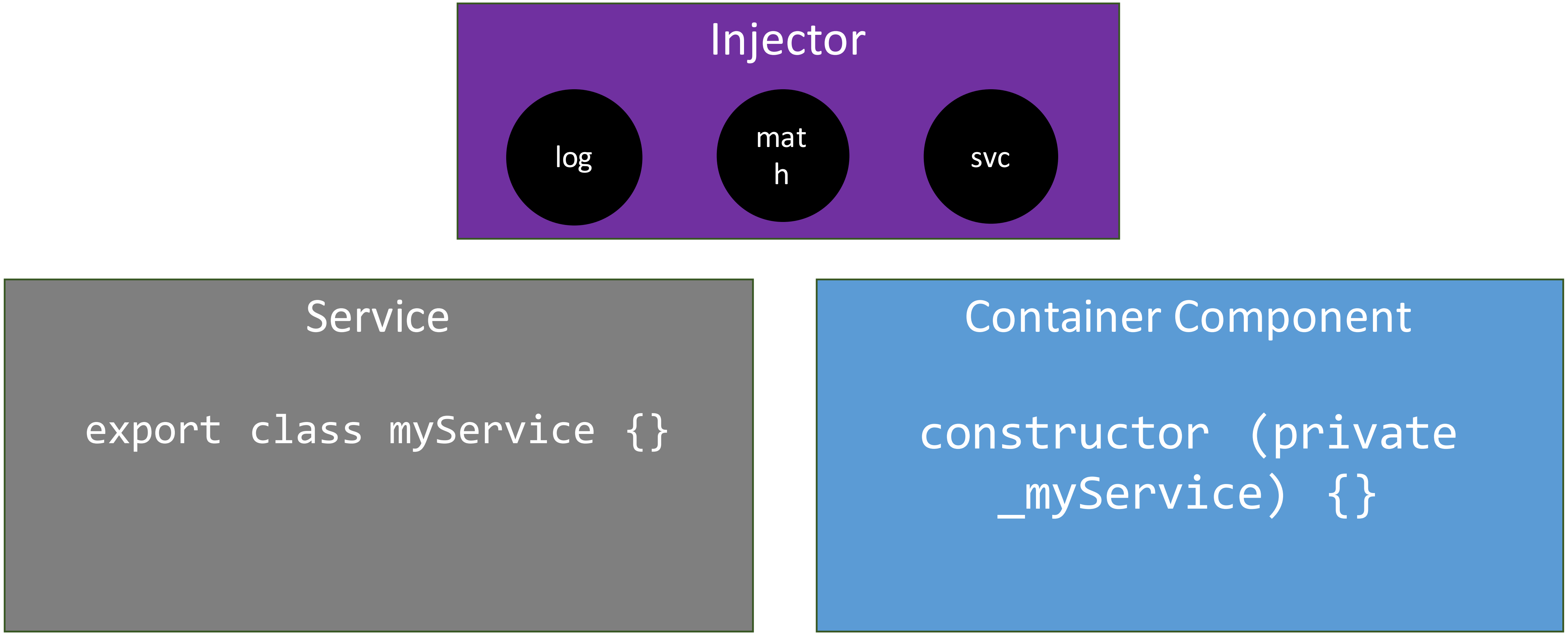
Service

```
export class myService {}
```

Container Component

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

How does it Work?



```
export class myService {}
```

```
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

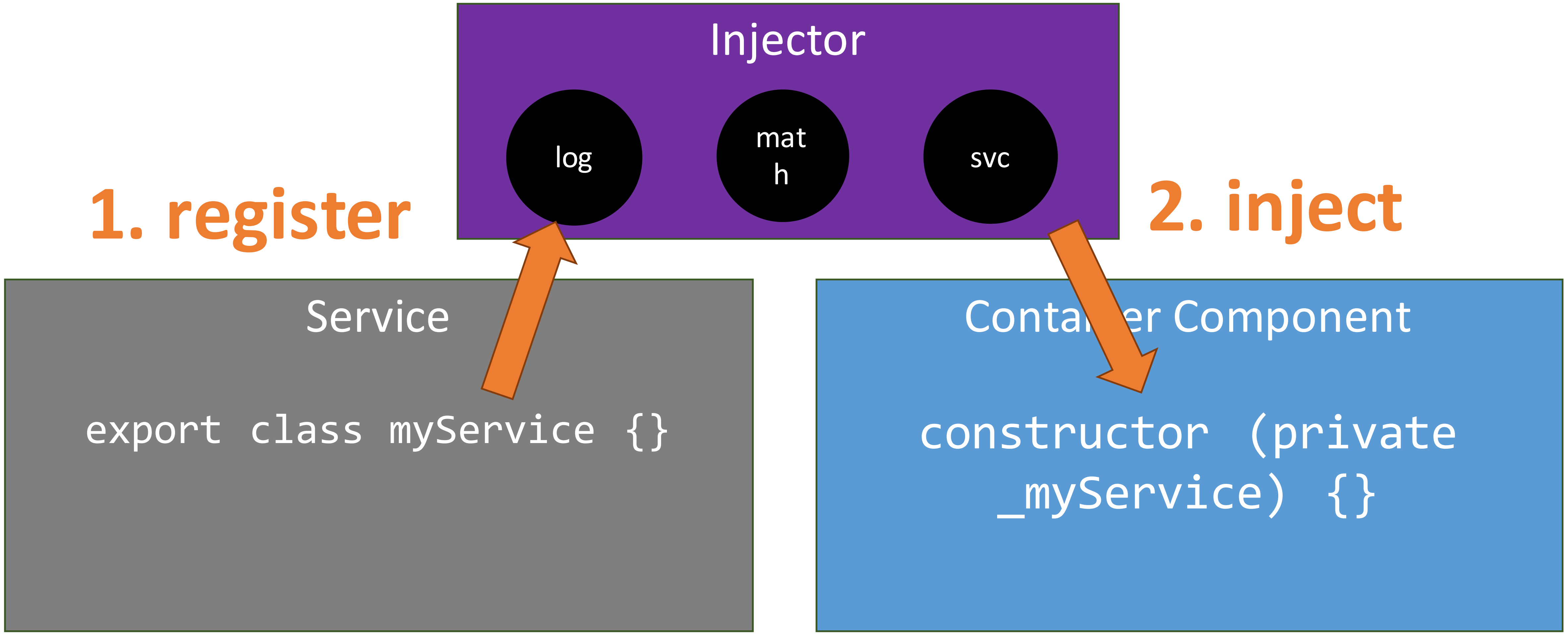
- Create the service class (with export keyword)
- Define the metadata with a decorator
- Import what we need
- We're done!

Building a service

```
// product/product.service.ts

import { Injectable } from '@angular/core';
import { IProduct } from 'product';
@Injectable()
export class ProductService {
  getProducts(): IProduct[] { … }
}
```

Registering the Service



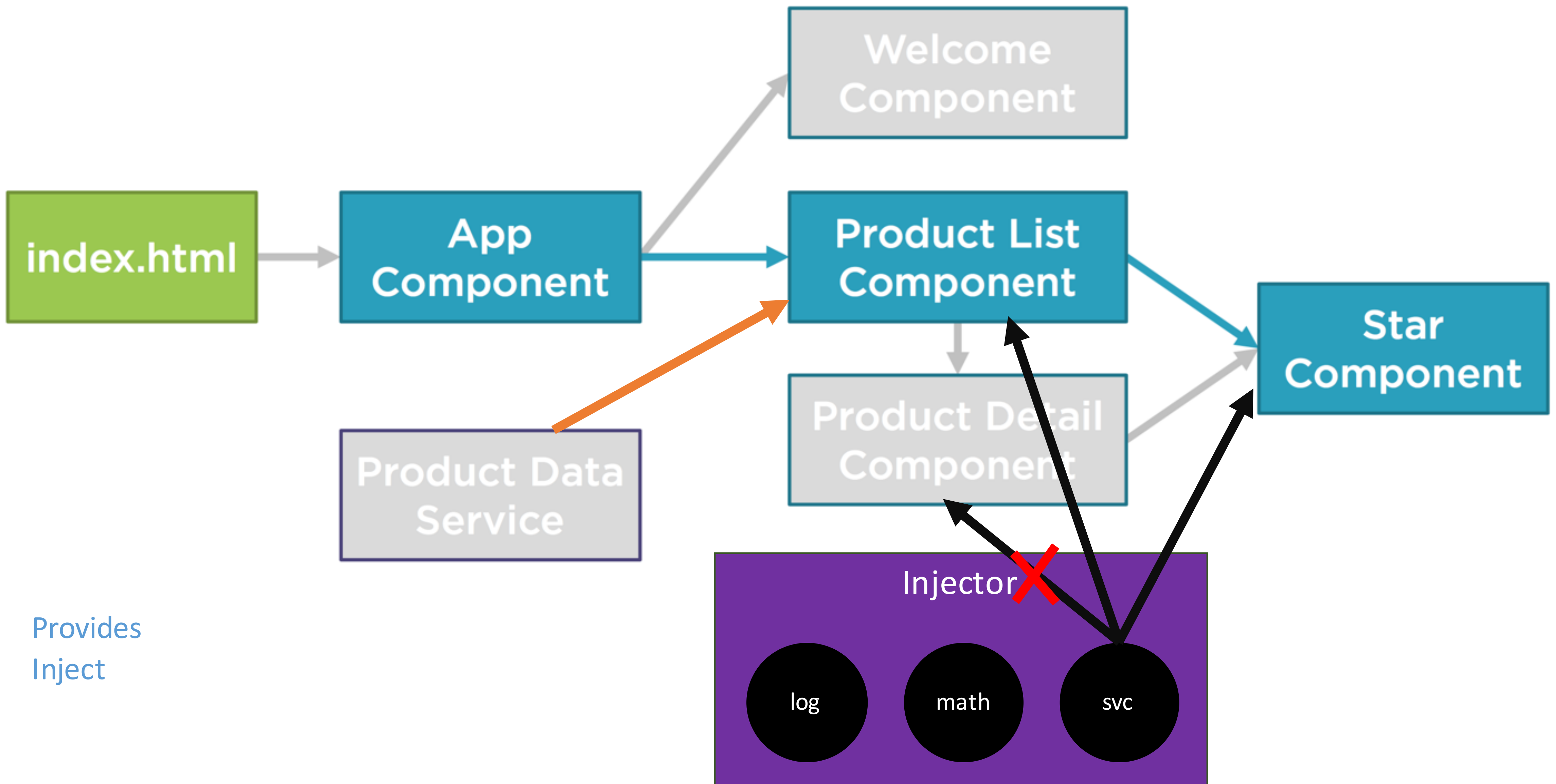
```
export class myService {}
```

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

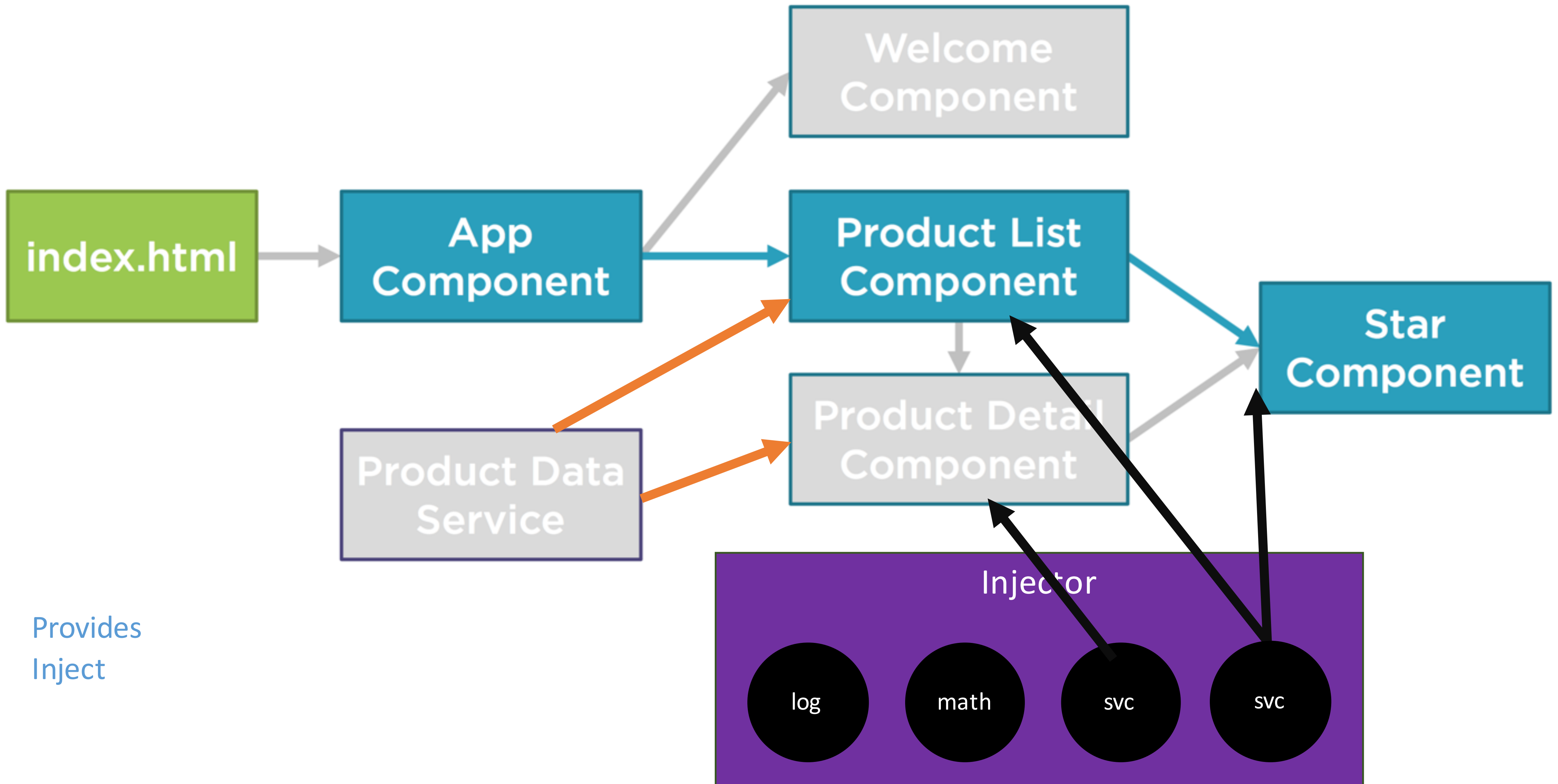
Registering the Service

- ✓ 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 it’s children
- ✓ Registered in Angular Module
    - ✓ Injectable everywhere in the application

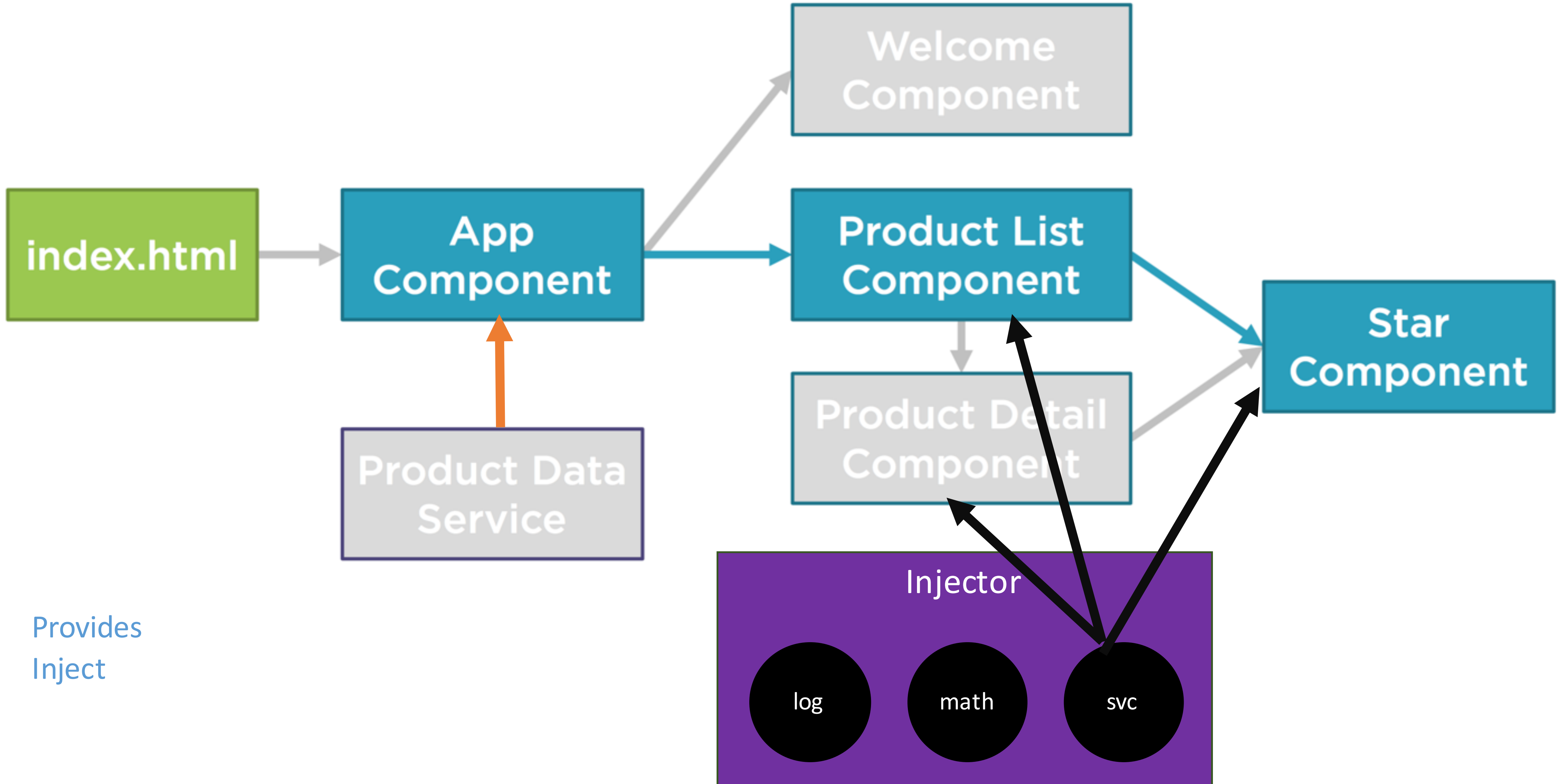
Application Architecture



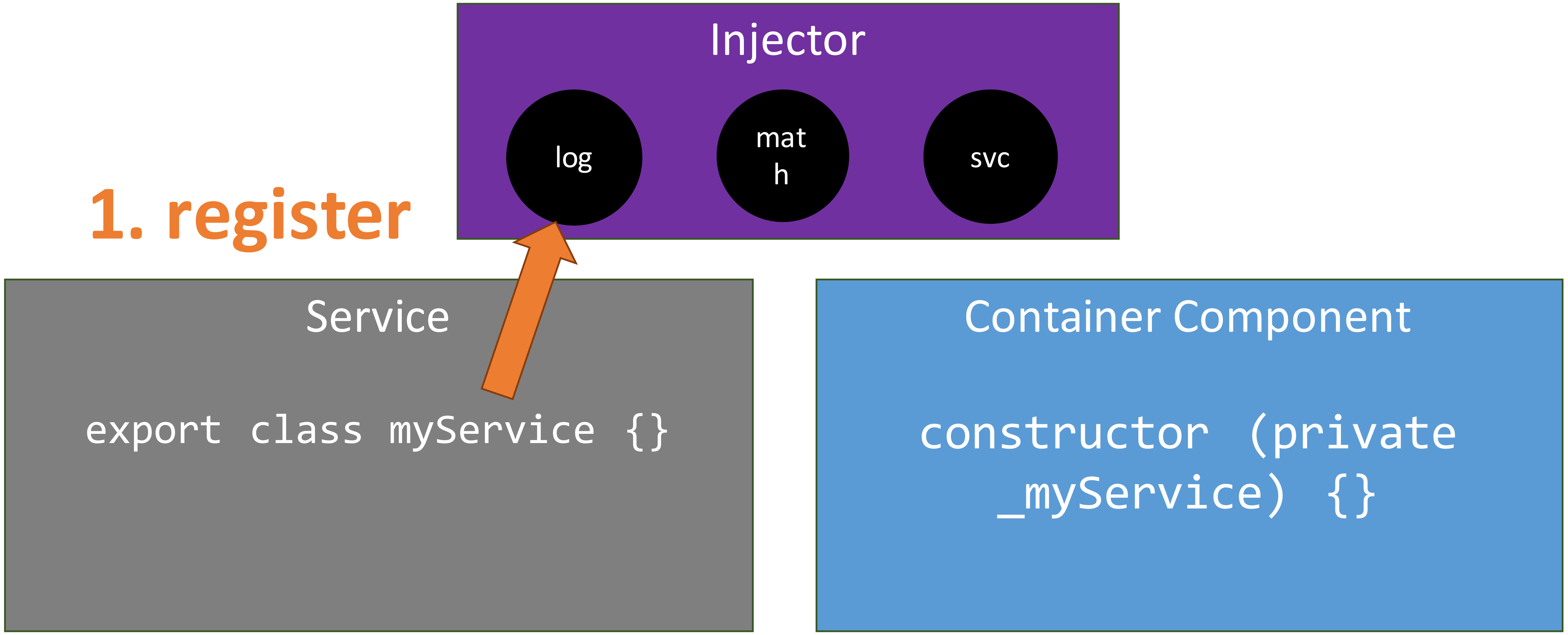
Application Architecture



Application Architecture



Registering the Service



```
export class myService {}
```

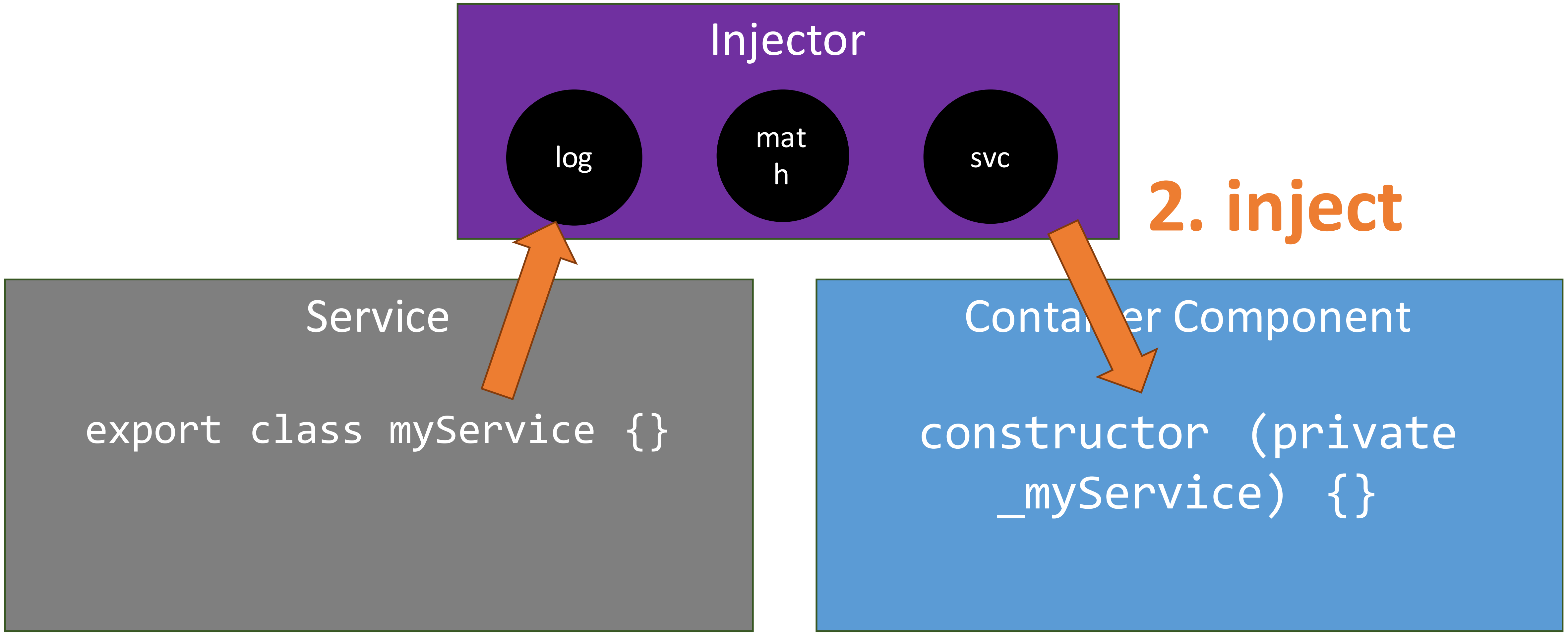
```
constructor (private
_myService) {}
```

Registering a Provider

```
// app.component.ts

import { Component } from '@angular/core';
import { ProductService } from 'products/product.service';
@Component({
  selector: 'nat-app',
  template: `
    <h1>Angular2: Let's do it!</h1>
    <nat-products></nat-products>
  `,
  providers: [ ProductService ]
})
export class ProductListComponent {}
```

Injecting the Service



```
export class myService {}
```

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

Injecting the Service

```
// product-list.components.ts

import { Component } from '@angular/core';
import { ProductService } from ’./product.service';
@Component({
  selector: 'nat-products',
  templateUrl: 'app/products/product-list.component.html'
})
export class ProductListComponent {
  private _productService: ProductService;
  constructor (productService: ProductService) {
    this._productService = productService;
  }
}
```

Injecting the Service

```
// product-list.components.ts

import { Component } from '@angular/core';
import { ProductService } from ’./product.service';
@Component({
  selector: 'nat-products',
  templateUrl: 'app/products/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 elvel 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

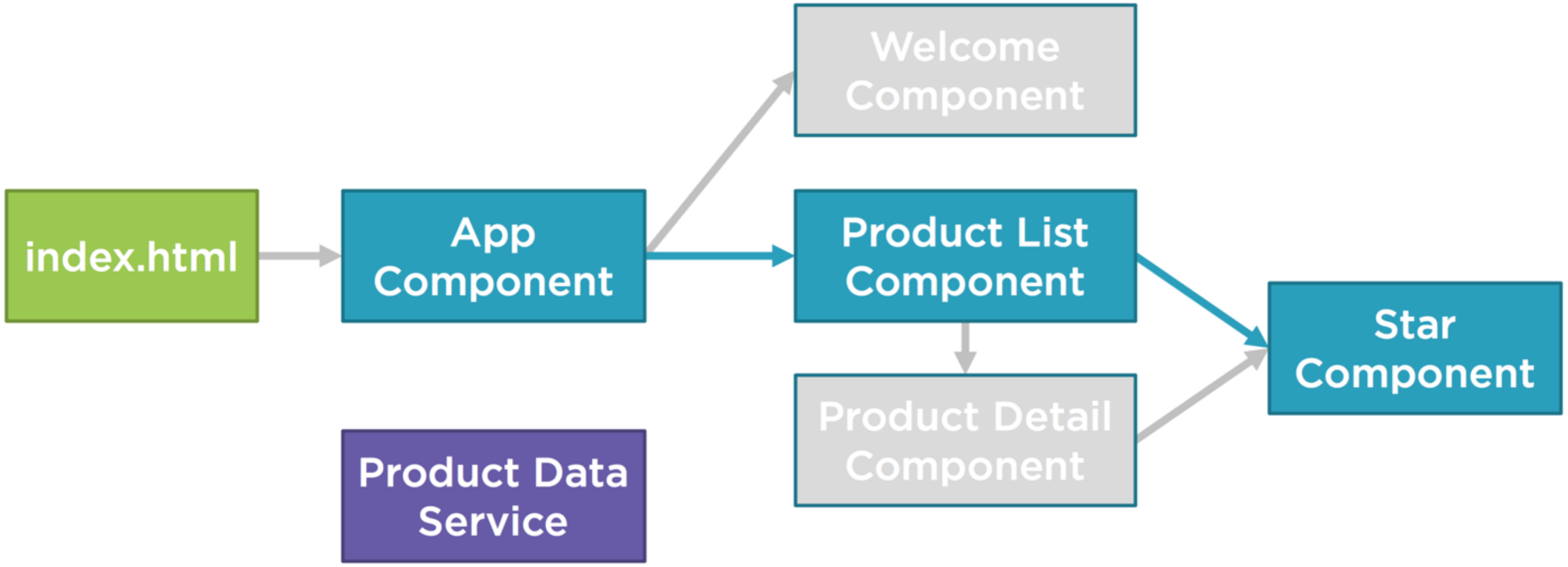
Checklist: Dependency Injection

- ✓ Specify the service as a dependency
- ✓ Use a constructor parameter
- ✓ Service is injected when component is instantiated

Module Overview

- How does it Work?
- Building a Service
- Registering the Service
- Injecting the Service

Application Architecture


index.html
App Component
Welcome Component
Product List Component
Product Detail Component
Star Component
Product Data Service