

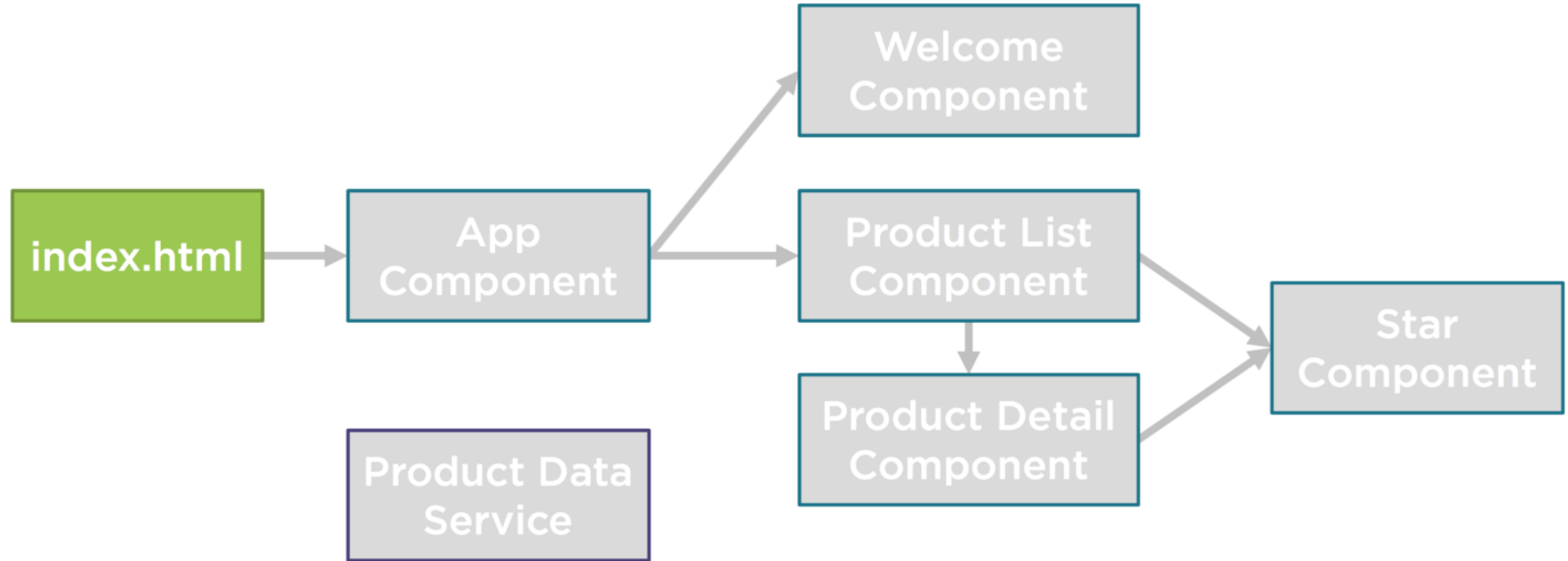
# AngularJS

Introduction to Components

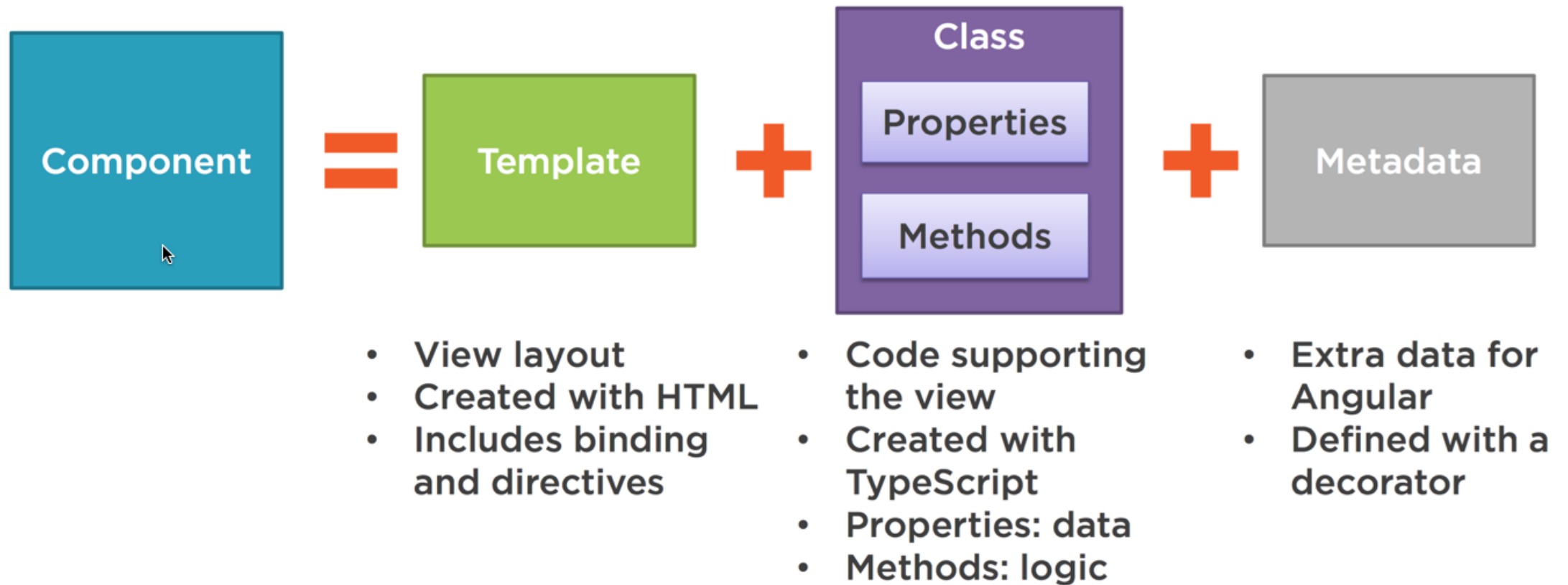
# Module Overview

- What Is a Component?
- Creating the Component Class
- Defining the Metadata with a Decorator
- Importing What We Need
- Bootstrapping Our App Component

# Application Architecture



# What is a Component



# Component

```
// app.component.ts
```

```
import { Component } from '@angular/core';
```

```
@Component({  
  selector: 'nat-app',  
  template: `  
    <h1>Angular2: Let's do it!</h1>  
  `,  
})
```

```
export class AppComponent {  
  pageTitle: string = 'Product Management';  
}
```

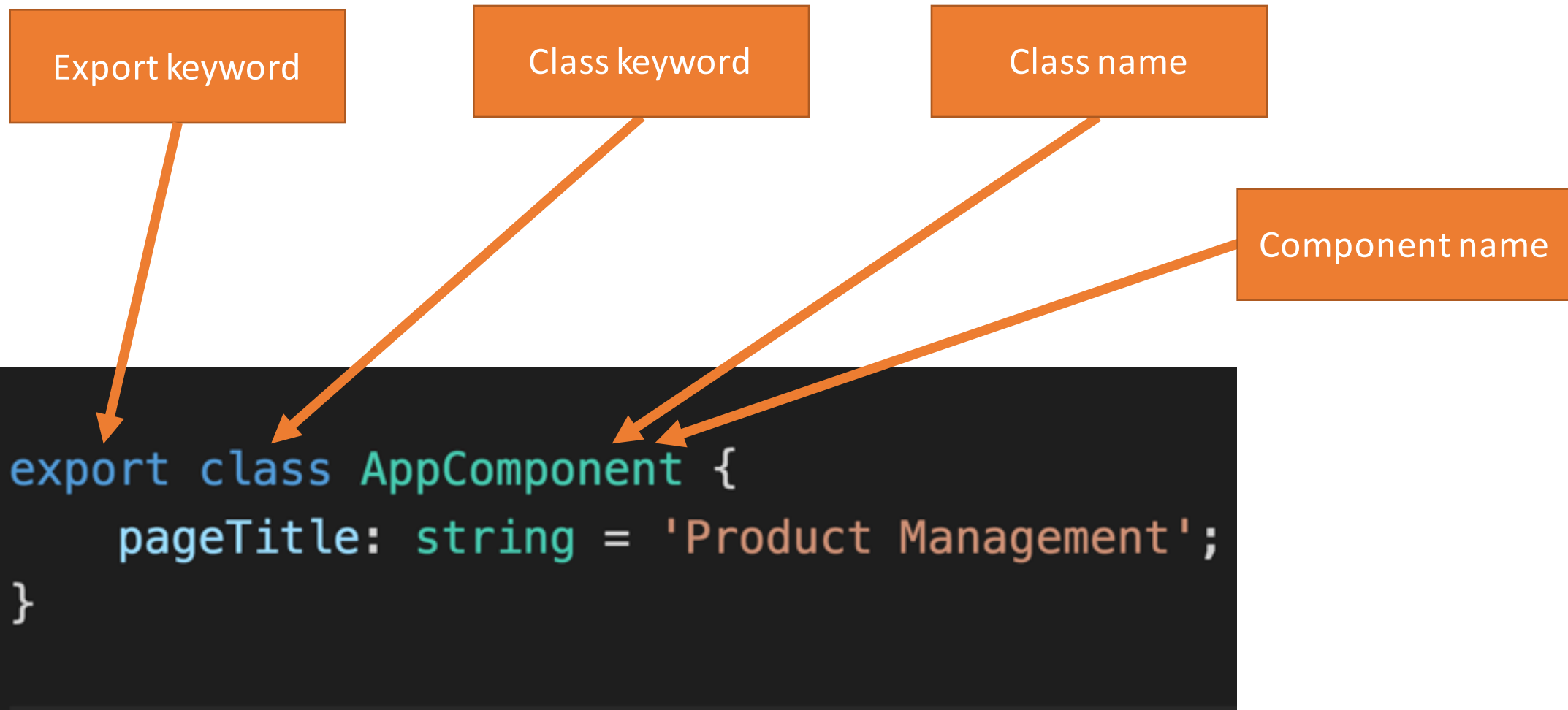
Import



Metadata &  
Template

Class

# 1. Creating the Component



# 1. Creating the Component

```
export class AppComponent {  
  pageTitle: string = 'Product Management';  
}
```

Property name

Data Type

Default value

Methods



## 2. Defining the Metadata

```
@Component({  
  selector: 'nat-app',  
  template: `  
    <h1>Angular2: Let's do it!</h1>  
  `,  
})
```

```
export class AppComponent {  
  pageTitle: string = 'Product Management';  
}
```



# Decorator

- A function that adds **metadata** to a class, its members, or its method arguments.
- Prefixed with an @.
- Angular provides built-in decorators.

## **@Component()**

Link to `@Component()`

## 2. Defining the Metadata

Decorator function

Decorator options  
{object}

HTML Element  
name

```
@Component({  
  selector: 'nat-app',  
  template: `  
    <h1>Angular2: {{pageTitle}}</h1>  
  `,  
})
```

Template (inline)

```
export class AppComponent {  
  pageTitle: string = 'Product Management';  
}
```

Bindings

### 3. Importing what we need

- Before we use an external function or class, we define where to find it
- import statement
- import allows us to use exported members from external ES modules
- Import from a third-party library, our own ES modules, or from Angular

# Angular is Modular

@angular/  
core

@angular/  
forms

@angular/  
http

@angular/  
router

<https://www.npmjs.com/~angular>

### 3. Importing what we need

Import keyword

```
import { Component } from '@angular/core';
```

Angular library  
module name

```
@Component({  
  selector: 'nat-app',  
  template: `  
    <h1>Angular2: {{pageTitle}}</h1>  
  `,  
})
```

Member name to  
import

```
export class AppComponent {  
  pageTitle: string = 'Product Management';  
}
```

# Completed component

```
// app.component.ts

import { Component } from '@angular/core';

@Component({
  selector: 'nat-app',
  template: `
    <h1>Angular2: {{pageTitle}}</h1>
  `
})

export class AppComponent {
  pageTitle: string = 'Product Management';
}
```

# Demo time!



# Bootstrapping Our App Component

- Load the root component (bootstrapping)
- Host the application



# Single Page Application (SPA)

- index.html contains the main page for the application
- This is often the only Web page of the application
- Hence an Angular application is often called a Single Page Application (SPA)

# Hosting the application

```
<!-- index.html -->  
  
<body>  
  <nat-app></nat-app>  
  Product Manager Application starter files  
</body>
```

```
// app.component.ts  
  
import { Component } from '@angular/core';  
  
@Component({  
  selector: 'nat-app',  
  template: `  
    <h1>Angular2: Let's do it!</h1>  
  `,  
})  
  
export class AppComponent {  
  pageTitle: string = 'Product Management';  
}
```

# Angular Application Startup

```
<!-- index.html -->
<script>System.import('app')</script>
<body>
  <nat-app></nat-app>
  Product Manager Application starter files
</body>
```

```
// systemjs.config.js
packages: {
  app: {
    main: './main.js',
    defaultExtension: 'js'
  },
}
```

```
// main.ts
import { platformBrowserDynamic } from
'@angular/platform-browser-dynamic';
import { AppModule } from './app.module';

platformBrowserDynamic()
  .bootstrapModule(AppModule);
```

```
// app.component.ts

import { Component } from '@angular/core';

@Component({
  selector: 'nat-app',
  template: `
    <h1>Angular2: Let's do it!</h1>
  `
})

export class AppComponent {
  pageTitle: string = 'Product Management';
}
```

```
// app.module.ts
import { NgModule } from '@angular/core';
import { BrowserModule } from
'@angular/platform-browser';

import { AppComponent } from
  './app.component';

@NgModule({
  imports: [ BrowserModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})

export class AppModule { }
```

# Demo time: bootstrapping the App component!



# Component Checklist

- ✓ Class → Code
- ✓ Decorator → Metadata
- ✓ Import what we need

# Component Checklist: Class

- ✓ Clear name
  - ✓ Use PascalCasing
  - ✓ Append "Component" to the name
- ✓ Export keyword
- ✓ Data in properties
  - ✓ Appropriate data type
  - ✓ Appropriate default value
  - ✓ camelCase with first letter lowercase
- ✓ Logic in methods
  - ✓ camelCase with first letter lowercase

# Component Checklist: Class

- ✓ Clear name
  - ✓ Use PascalCasing
  - ✓ Append "Component" to the name
- ✓ Export keyword
- ✓ Data in properties
  - ✓ Appropriate data type
  - ✓ Appropriate default value
  - ✓ camelCase with first letter lowercase
- ✓ Logic in methods
  - ✓ camelCase with first letter lowercase

# Component Checklist: Metadata

- ✓ Component decorator
  - ✓ Prefix with @; Suffix with ();
- ✓ selector: Component name in HTML
  - ✓ Prefix for clarity
- ✓ template: View's HTML
  - ✓ Correct HTML syntax



# Component Checklist: Import

- ✓ Defines where to find the members that this component needs
- ✓ import keyword
- ✓ Member name
  - ✓ Correct spelling/casing
- ✓ Module path
  - ✓ Enclose in quotes
  - ✓ correct spelling/casing

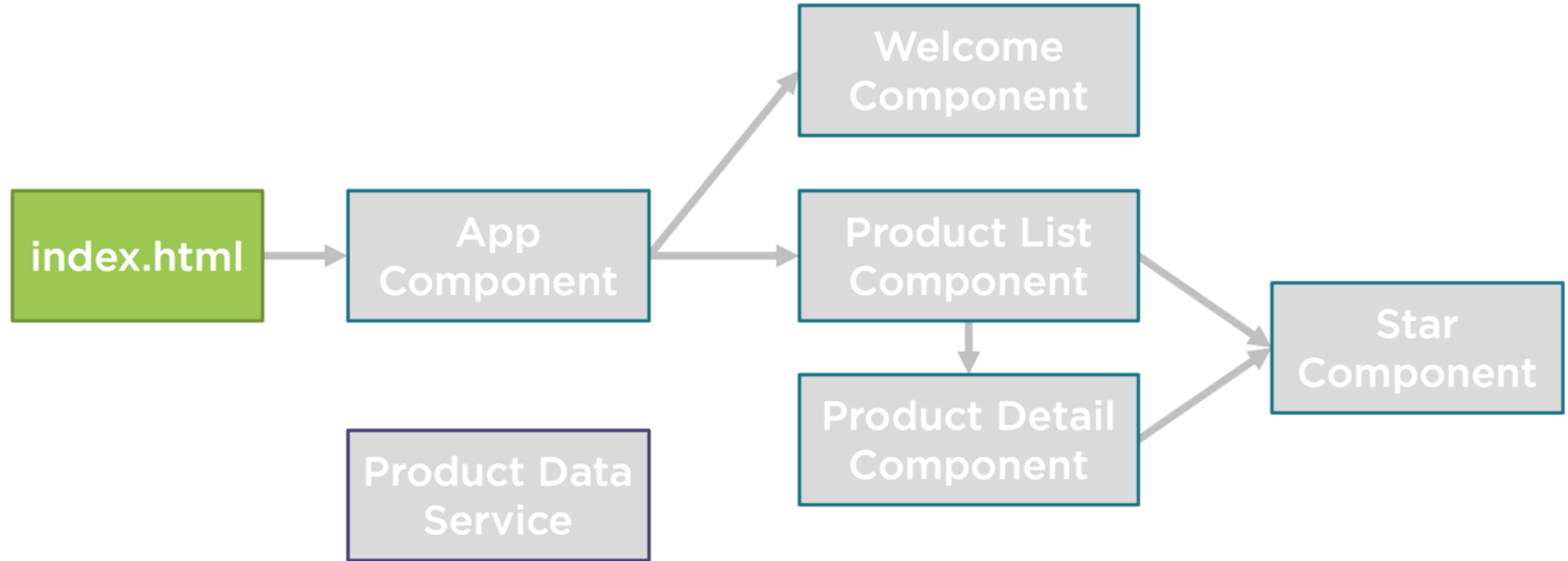
# Something Wrong! Checklist

- ✓ F12 is your friend (devTool)
- ✓ Recheck your code
  - ✓ HTML
    - ✓ Close tags
    - ✓ Angular directives are case sensitive
  - ✓ TypeScript
    - ✓ Close braces
    - ✓ TypeScript is case sensitive

# Summary

- What Is a Component?
- Creating the Component Class
- Defining the Metadata with a Decorator
- Importing What We Need
- Bootstrapping Our App Component

# Application Architecture



# Application Architecture

