Angular Is ...

A JavaScript framework

For building client-side applications

Using HTML, CSS and JavaScript

Why Angular?

Expressive HTML

Powerful Data Binding

Modular By Design

Built-in Back-End Integration

Why Angular 2?

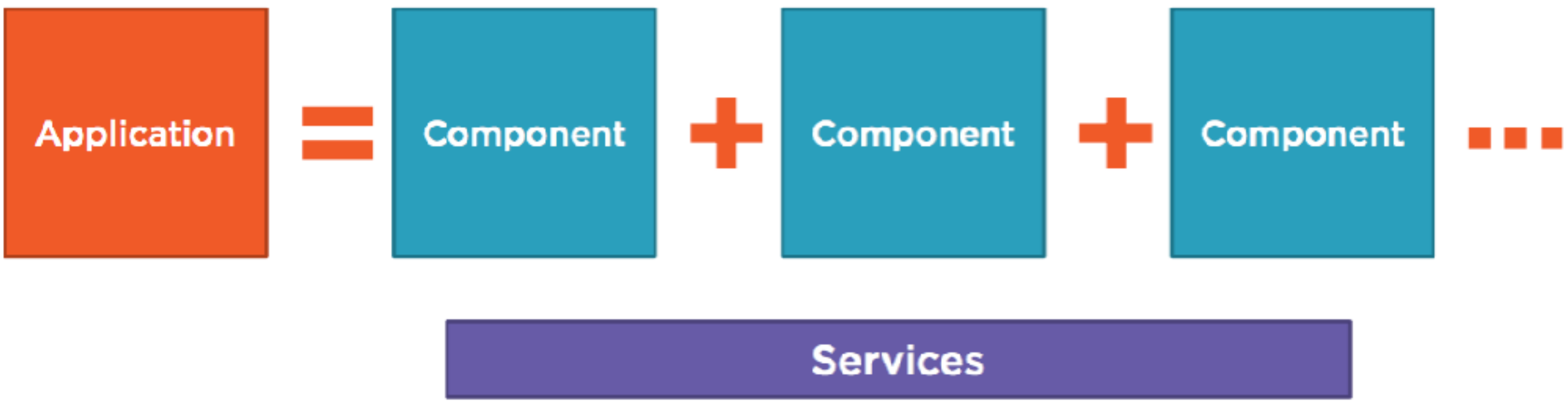
Built for Speed

Modern

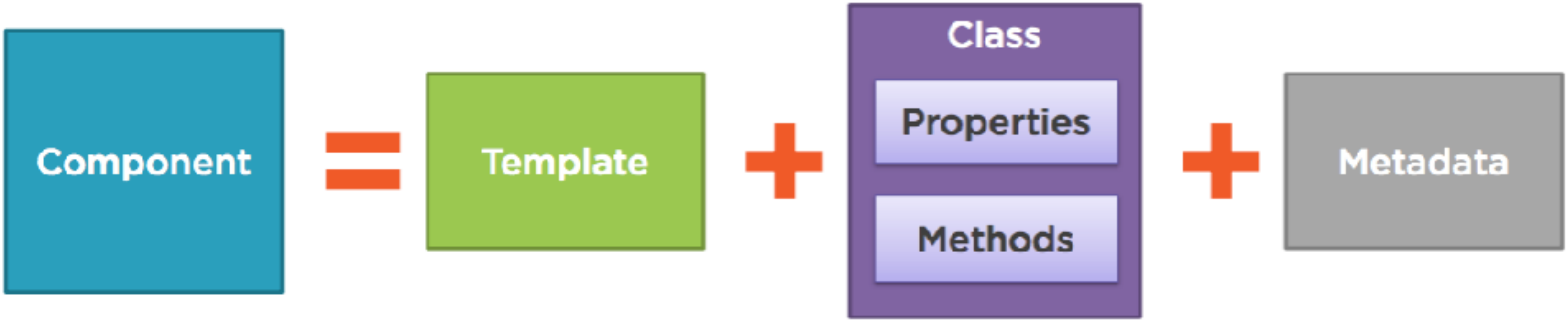
Simplified API

Enhances Productivity

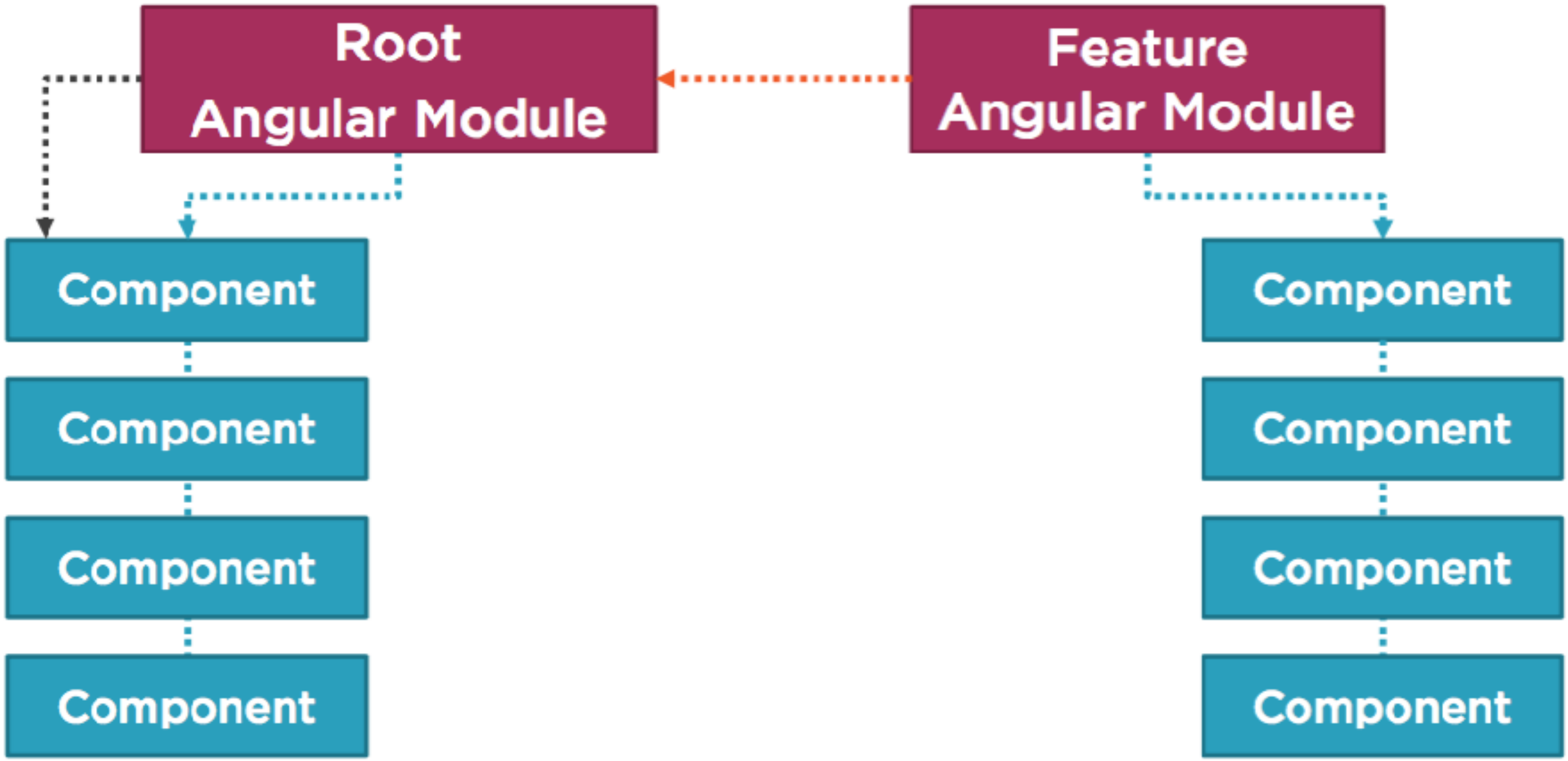
Anatomy of an Angular 2 Application



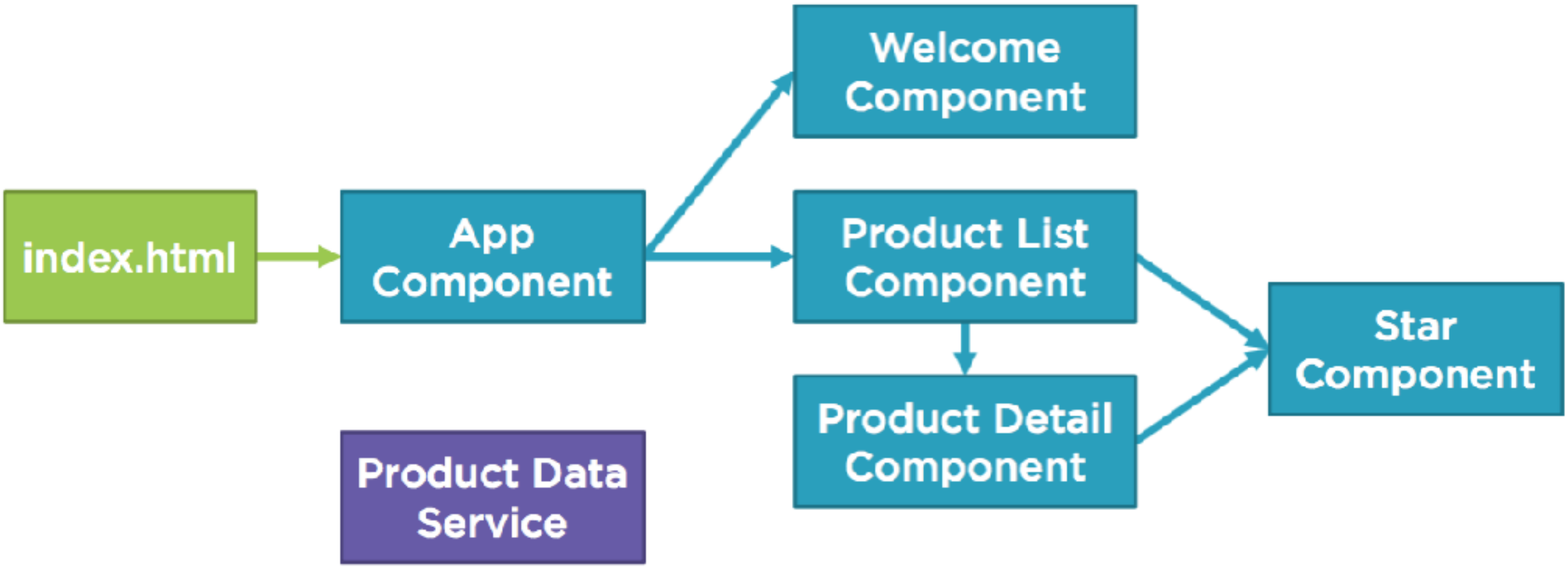
Component



Angular Modules



Sample Application Architecture



JavaScript Language Specification

ECMAScript (ES)

ES 3

ES 5

ES 2015 (formerly known as ES 6)

- Must be transpiled

Selecting a Language

ES 5

- Runs in the browser
- No compile required

ES 2015

- Lots of new features (classes, let, arrow, etc.)

TypeScript

- Superset of JavaScript
- Strong typing
- Great IDE tooling

Dart

- No JavaScript

What Is TypeScript?

TypeScript

Open source language

Superset of JavaScript

Transpiles to plain JavaScript

Strongly typed

- TypeScript type definition files (*.d.ts)

Class-based object-orientation

TypeScript Editors

- Visual Studio
- Visual Studio Code
- WebStorm
- Atom
- Eclipse
- Others

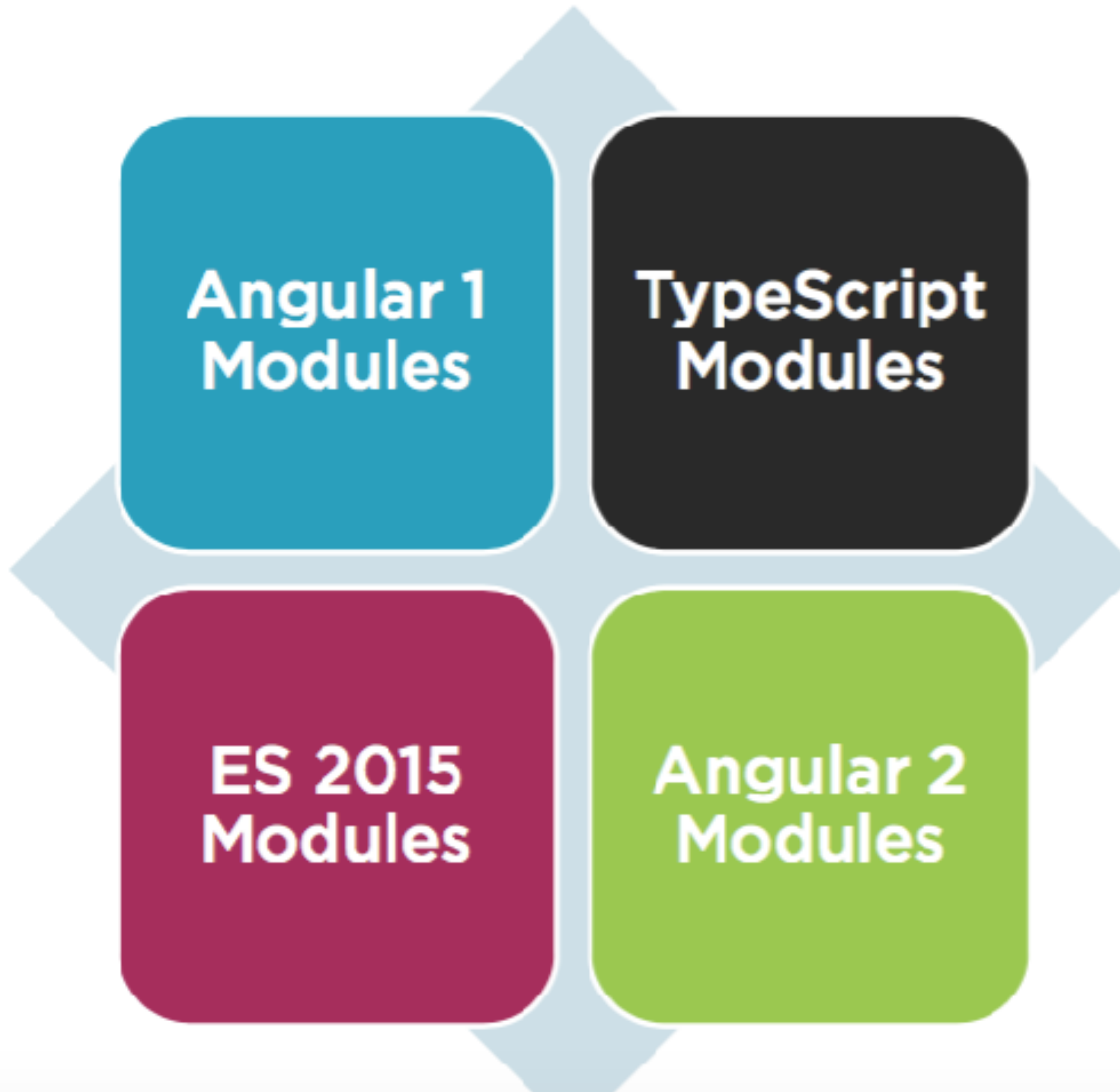
Setting up an Angular 2 Application

1. Create an application folder
2. Add package definition and configuration files
3. Install the packages
4. Create the app's Angular Module
5. Create the main.ts file
6. Create the host Web page (index.html)

Modules

Namespaces

Code Organization



ES 2015 Modules

Export

product.ts

```
export class Product{
}
```

Transpile

product.js

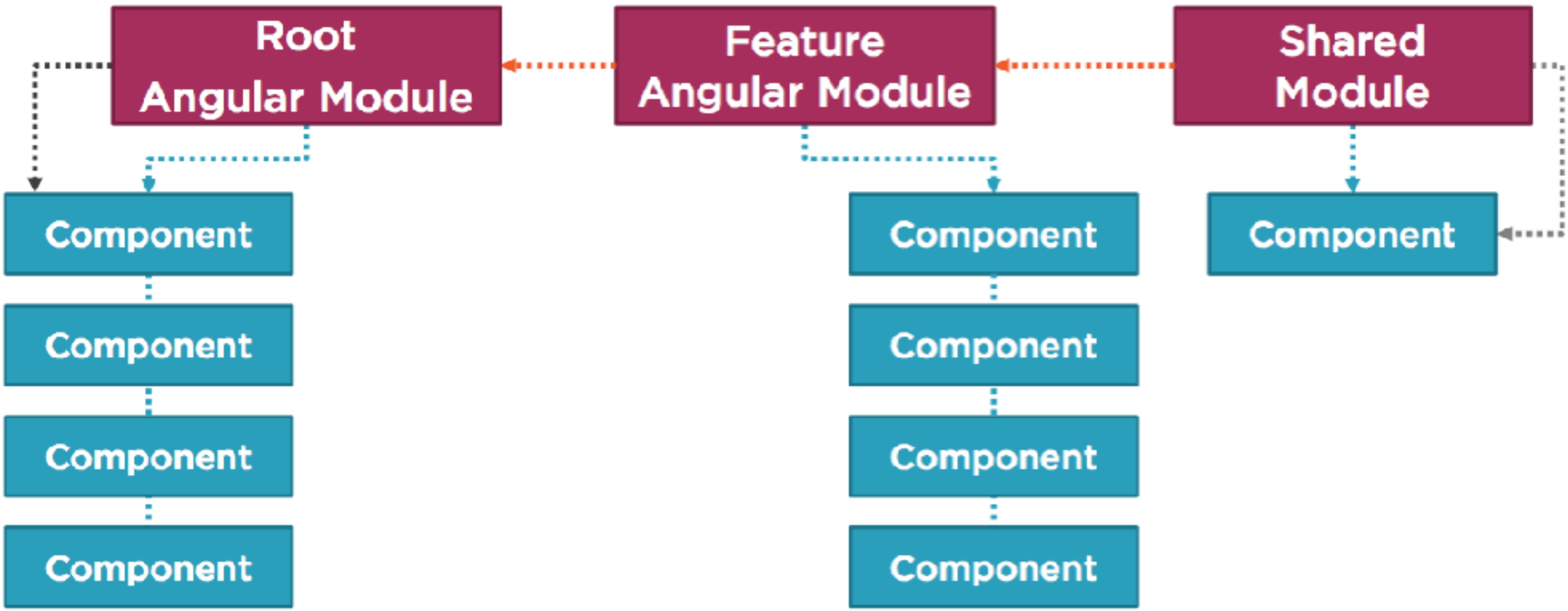
```
function Product() {
}
```

Import

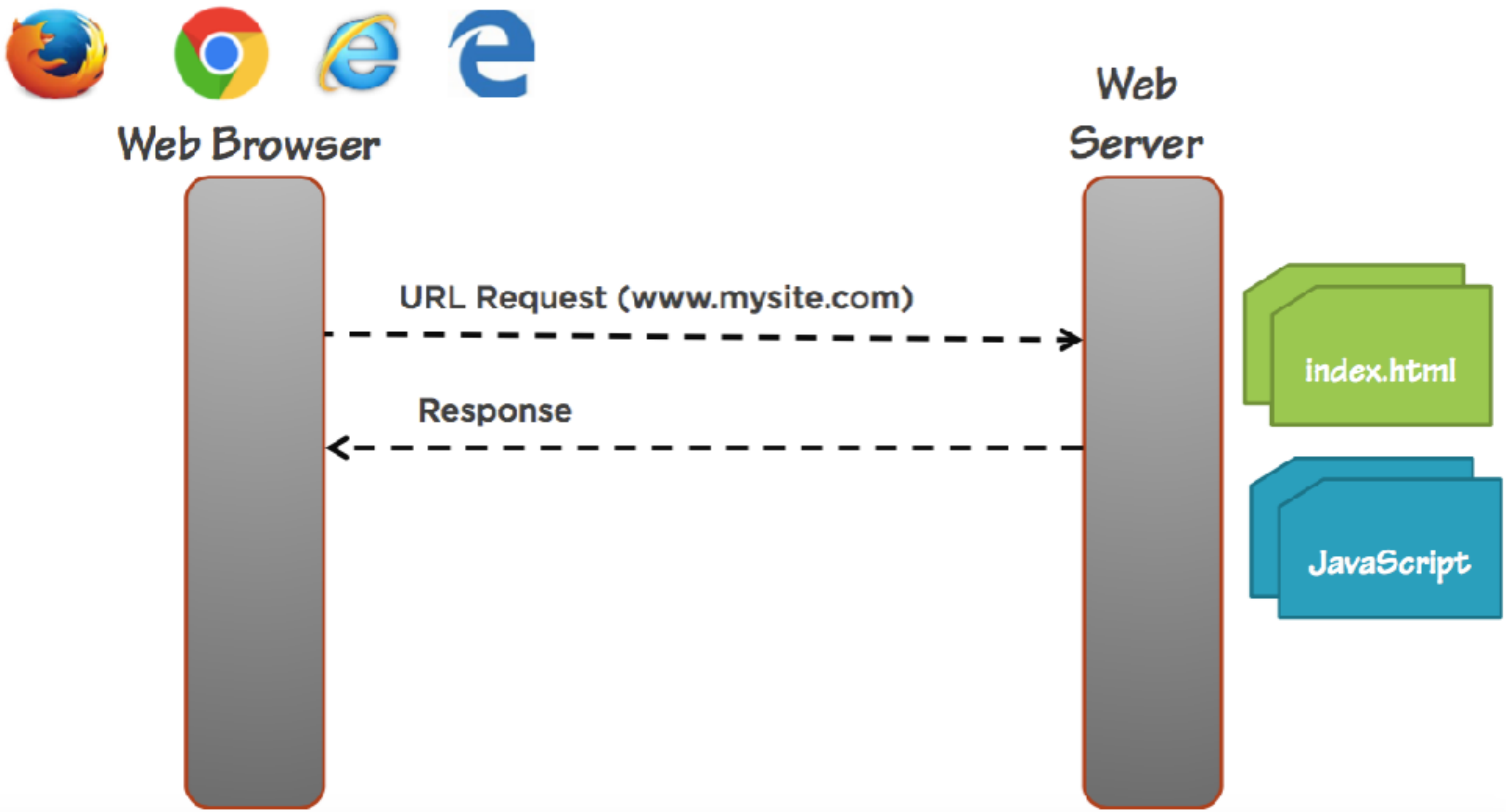
product-list.ts

```
import { Product } from
'./product'
```

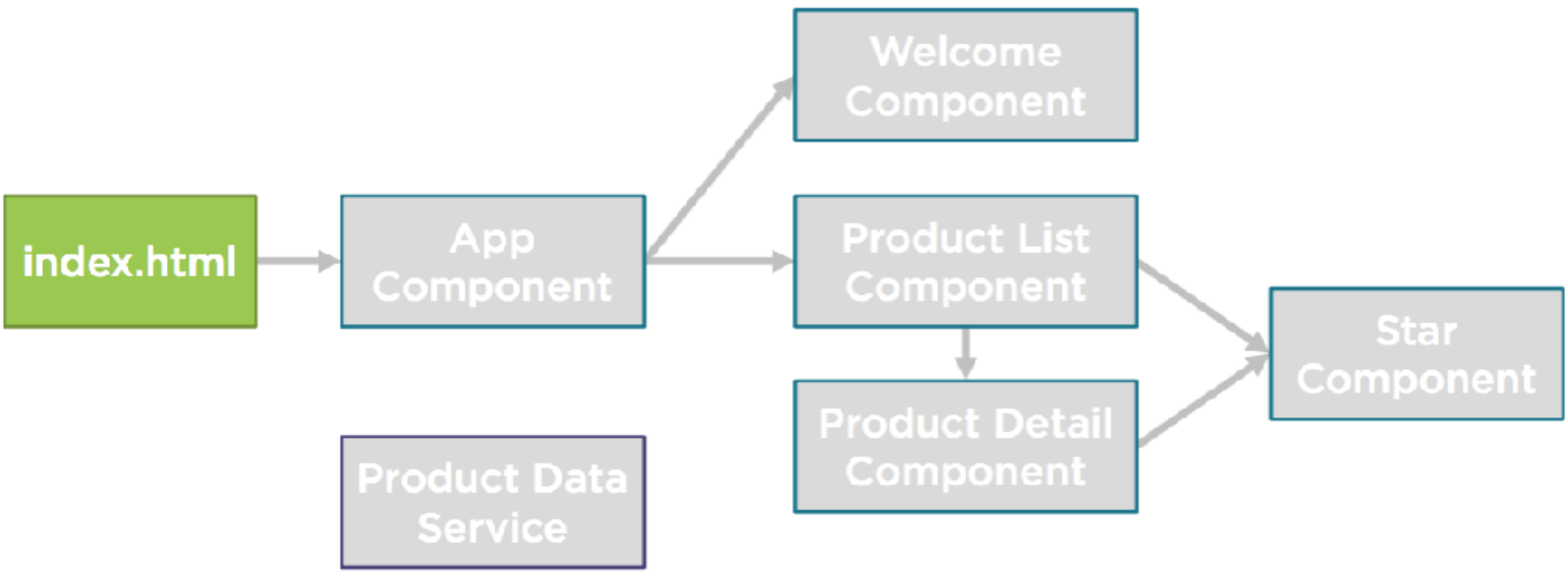
Angular Modules



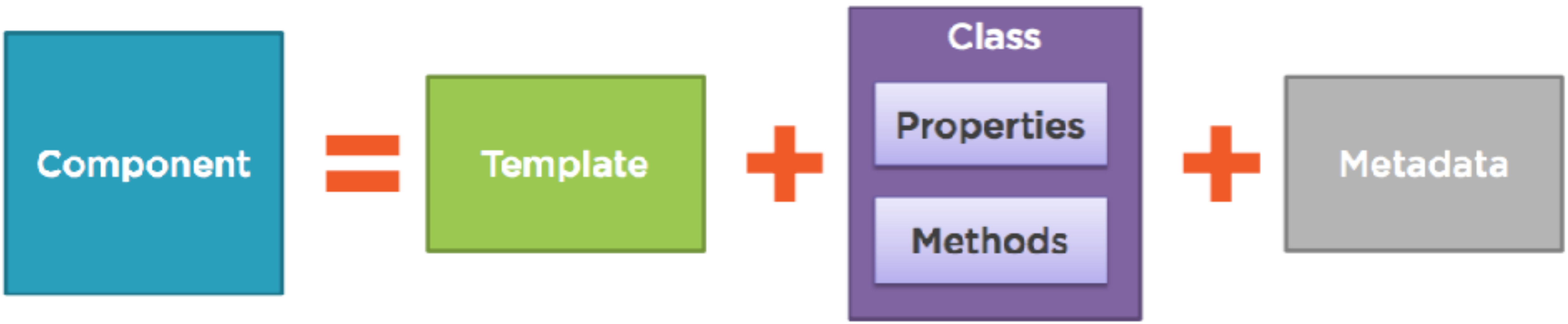
ES Modules	Angular Modules
Code files that import or export something	Code files that organize the application into cohesive blocks of functionality
Organize our code files	Organize our application
Modularize our code	Modularize our application
Promote code reuse	Promote application boundaries


Web Browser
Web Server
URL Request (www.mysite.com)
Response
index.html
JavaScript

Application Architecture



What Is a Component?



- View layout
- Created with HTML
- Includes binding and directives

- Code supporting the view
- Created with TypeScript
- Properties: data
- Methods: logic

- Extra data for Angular
- Defined with a decorator

Component

app.component.ts

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

@Component({
    selector: 'pm-app',
    template: `
    <div><h1>{{pageTitle}}</h1>
        <div>My First Component</div>
    </div>
    `
})
export class AppComponent {
  pageTitle: string = 'Acme Product Management';
}
```

Import

Metadata & Template

Class

Creating the Component Class

app.component.ts

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

class keyword

Class Name

export keyword

Component Name when used in code

Creating the Component Class

app.component.ts

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

Property Name

Data Type

Default Value

Methods

Defining the Metadata

app.component.ts

```
@Component({
    selector: 'pm-app',
    template: `
    <div><h1>{{pageTitle}}</h1>
        <div>My First Component</div>
    </div>
    `
})
export class AppComponent {
 pageTitle: string = 'Acme 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()
```

Defining the Metadata

app.component.ts

```
@Component({
    selector: 'pm-app',
    template: `
    <div><h1>{{pageTitle}}</h1>
        <div>My First Component</div>
    </div>
    `
})
export class AppComponent {
  pageTitle: string = 'Acme Product Management';
}
```

Component decorator

Directive Name used in HTML

View Layout

Binding

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/animate

@angular/http

@angular/router

Importing What We Need

app.component.ts

```
@Component({
    selector: 'pm-app',
    template: `
    <div><h1>{{pageTitle}}</h1>
        <div>My First Component</div>
    </div>
    `
})
export class AppComponent {
 pageTitle: string = 'Acme Product Management';
}
```

Importing What We Need

app.component.ts

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

@Component({
    selector: 'pm-app',
    template: `
    <div><h1>{{pageTitle}}</h1>
        <div>My First Component</div>
    </div>
    `
})
export class AppComponent {
 pageTitle: string = 'Acme Product Management';
}
```

import keyword

Angular library module name

Member name

Completed Component

app.component.ts

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

@Component({
    selector: 'pm-app',
    template: `
    <div><h1>{{pageTitle}}</h1>
        <div>My First Component</div>
    </div>
    `
})
export class AppComponent {
 pageTitle: string = 'Acme Product Management';
}
```

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>
  <pm-app>Loading App ...</pm-app>
</body>
```

app.component.ts

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

@Component({
    selector: 'pm-app',
    template: `
    <div><h1>{{pageTitle}}</h1>
        <div>My First Component</div>
    </div>
    `
})
export class AppComponent {
 pageTitle: string = 'Acme Product Management';
}
```

Observables and Reactive Extensions

Help manage asynchronous data

Treat events as a collection

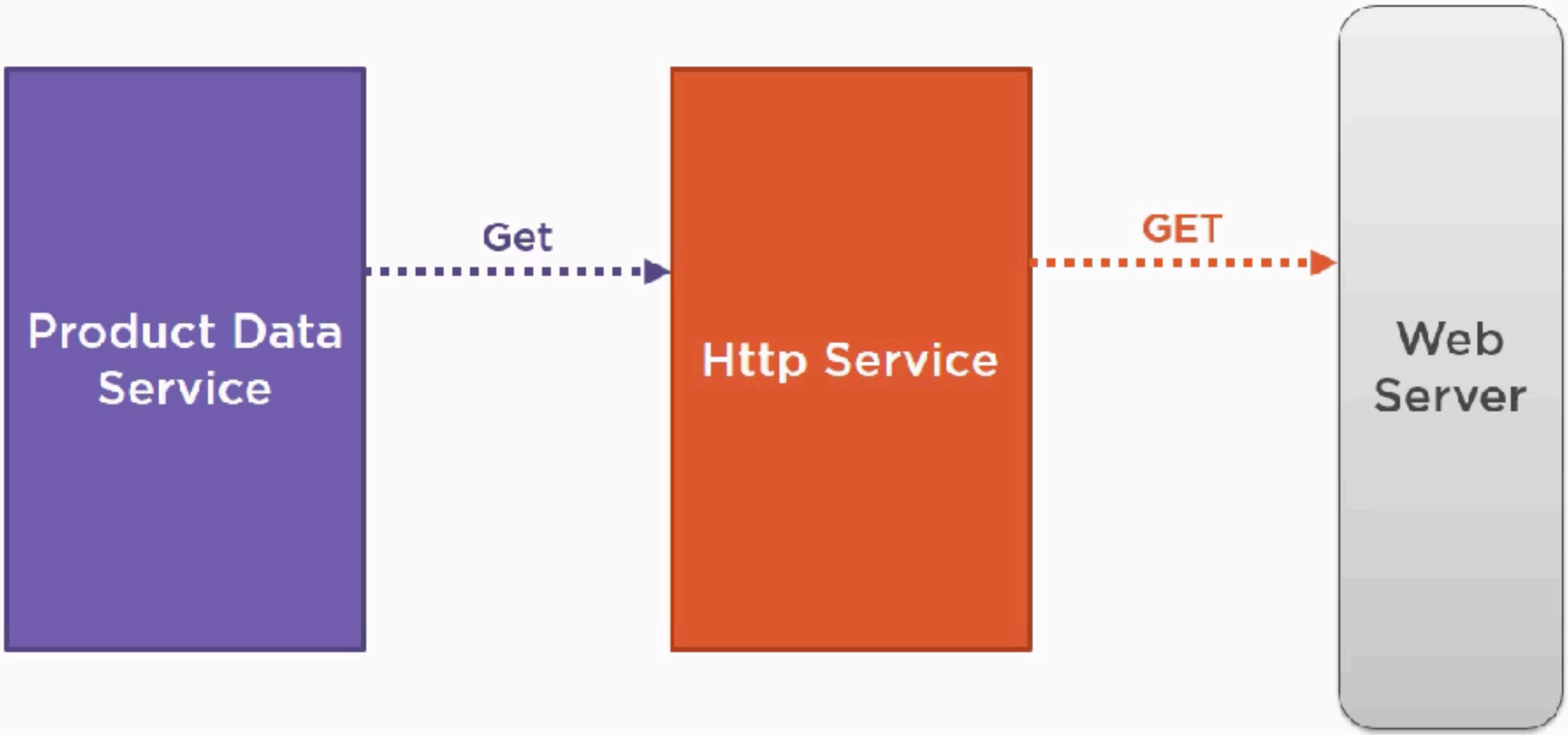
- An array whose items arrive asynchronously over time

Are a proposed feature for ES 2016

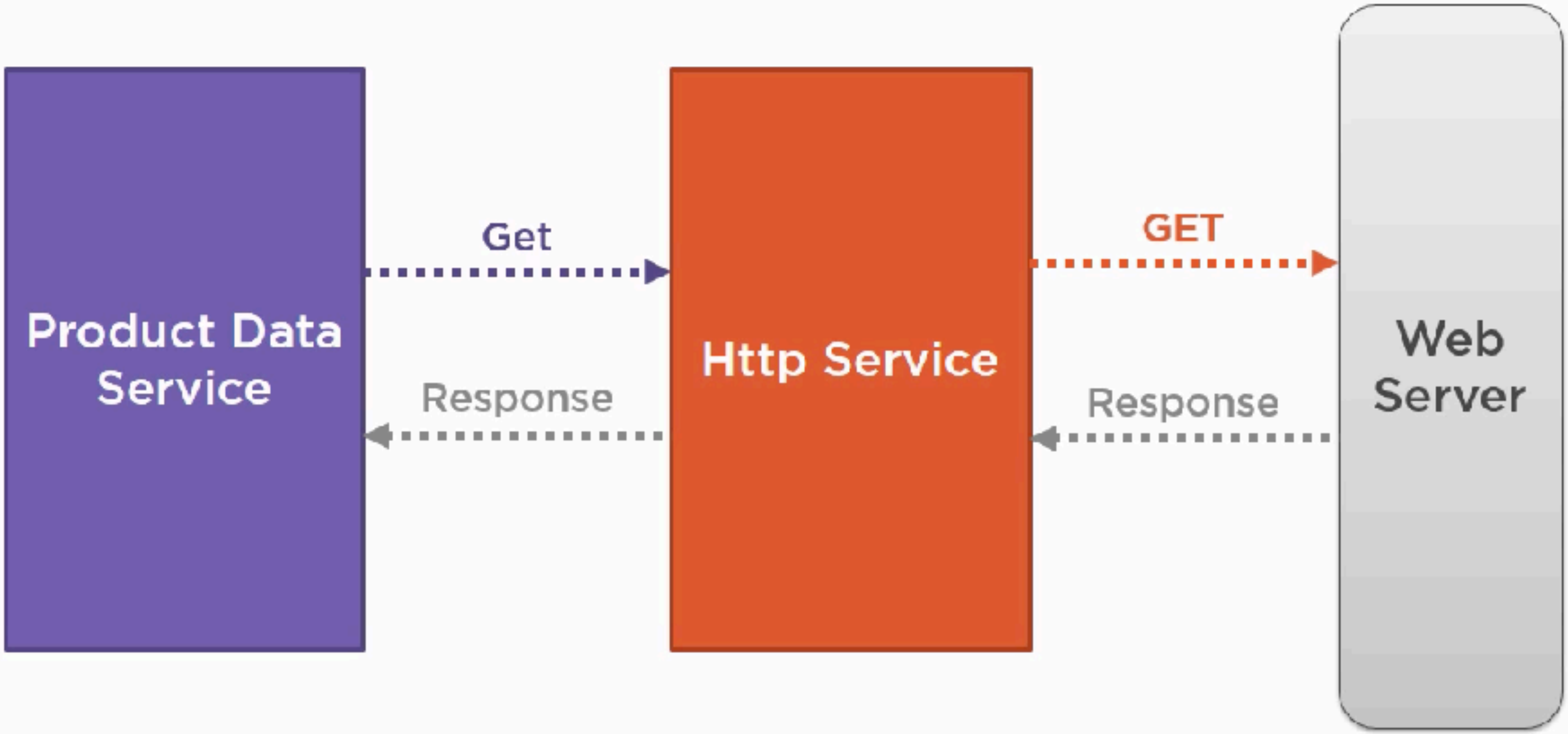
Use Reactive Extensions (RxJS)

Are used within Angular

Sending an Http Request



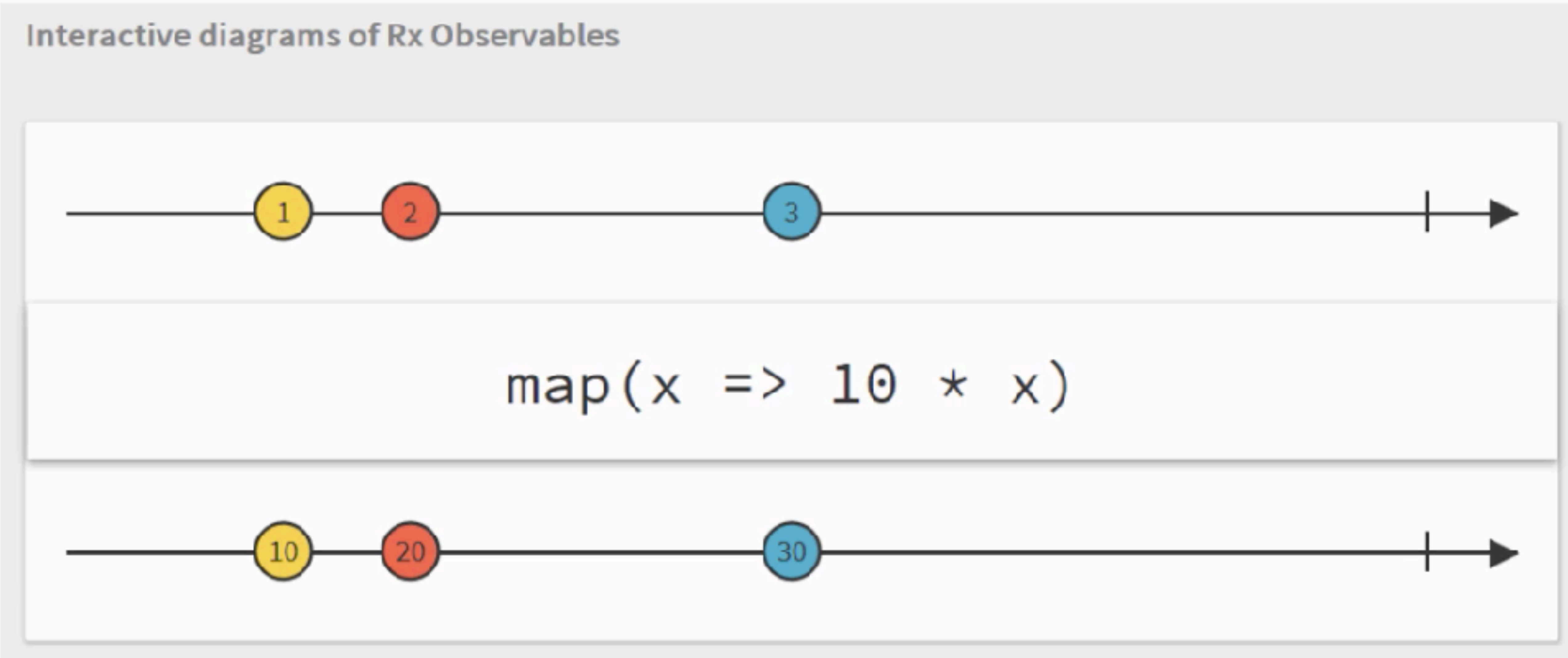
Sending an Http Request



Observables

Interactive diagrams of Rx Observables

Observables



Promise vs Observable

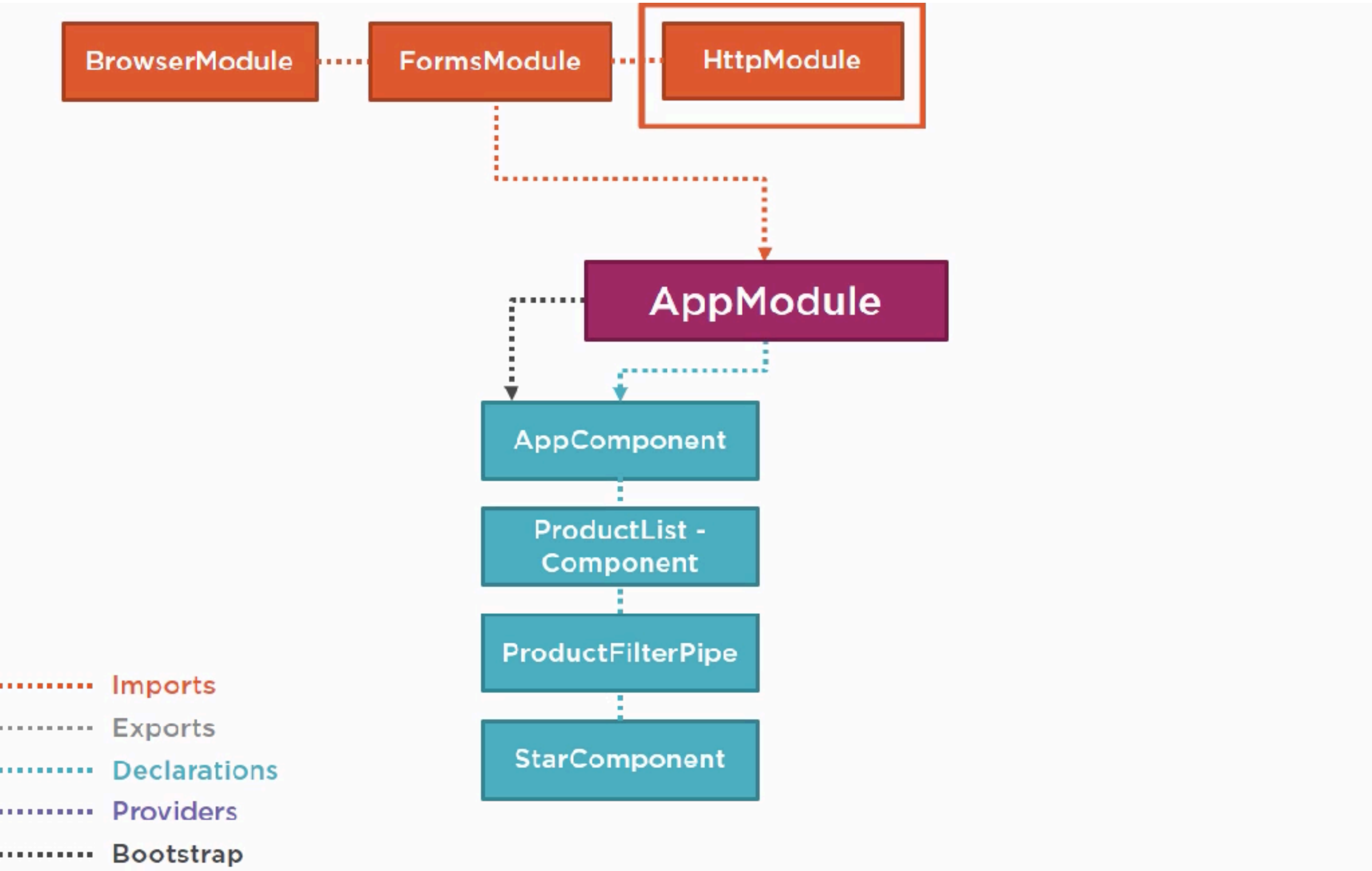
Promise	Observable
Provides a single future value	Emits multiple values over time
Not lazy	Lazy
Not cancellable	Cancellable
	Supports map, filter, reduce and similar operators

Subscribing to an Observable

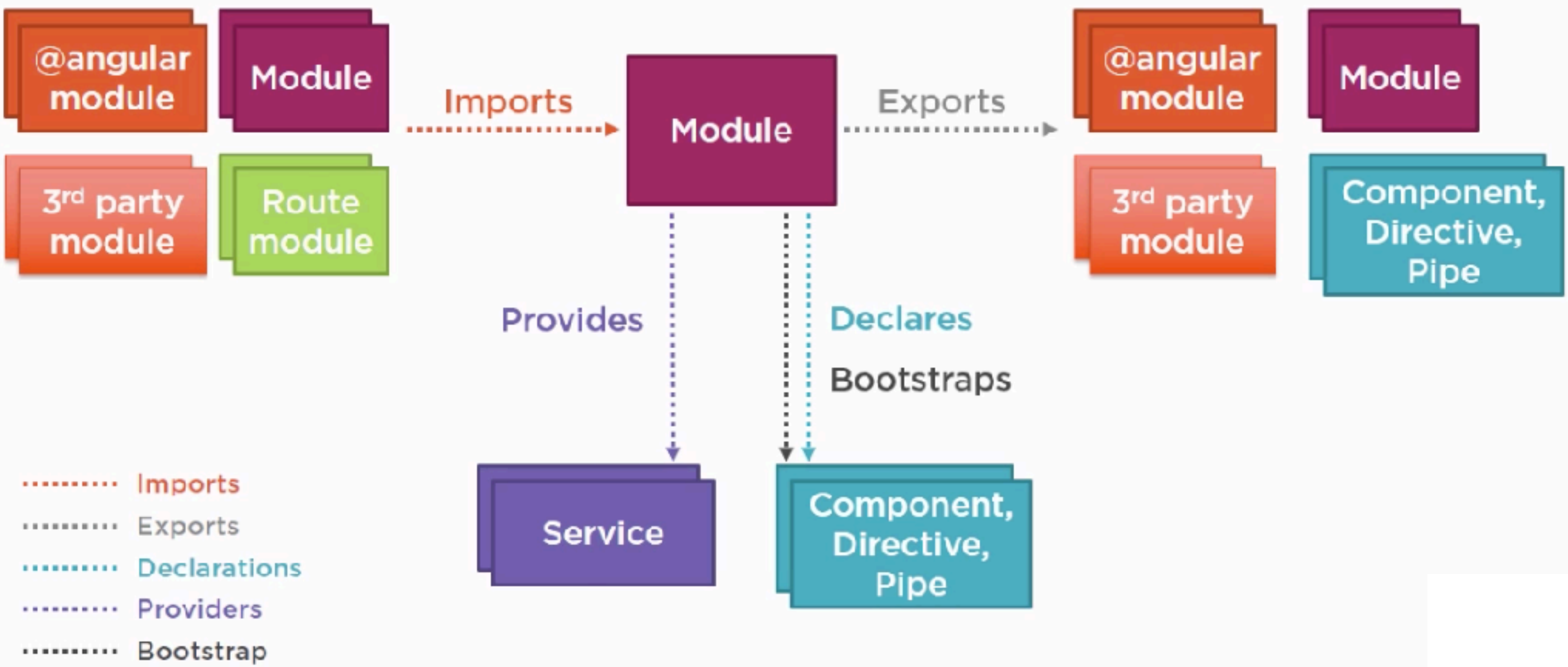
```
x.then(valueFn, errorFn)                         //Promise
x.subscribe(valueFn, errorFn)                    //Observable
x.subscribe(valueFn, errorFn, completeFn)        //Observable
let sub = x.subscribe(valueFn, errorFn, completeFn)
```

product-list.component.ts

```
ngOnInit(): void {
    this._productService.getProducts()
            .subscribe(
              products => this.products = products,
              error => this.errorMessage = <any>error);
}
```

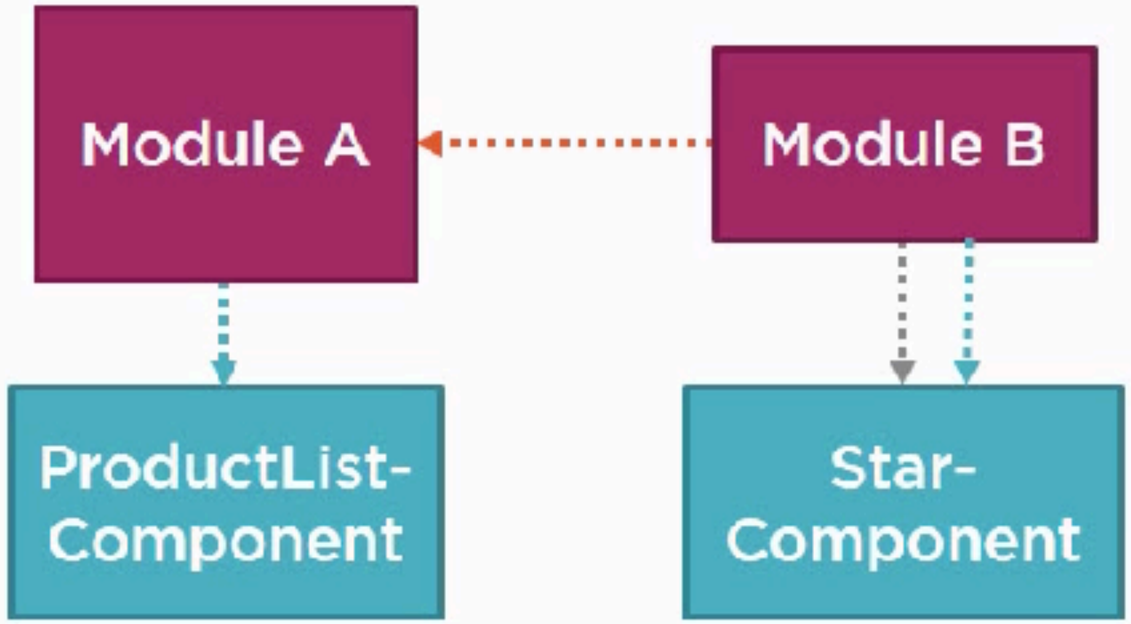

BrowserModule
FormsModule
HttpModule
AppModule
AppComponent
ProductList -
Component
ProductFilterPipe
StarComponent
Imports
Exports
Declarations
Providers
Bootstrap

Angular Module



Declarations Array Truth #3

Never re-declare components, directives, or pipes that belong to another module



Declarations Array Truth #4

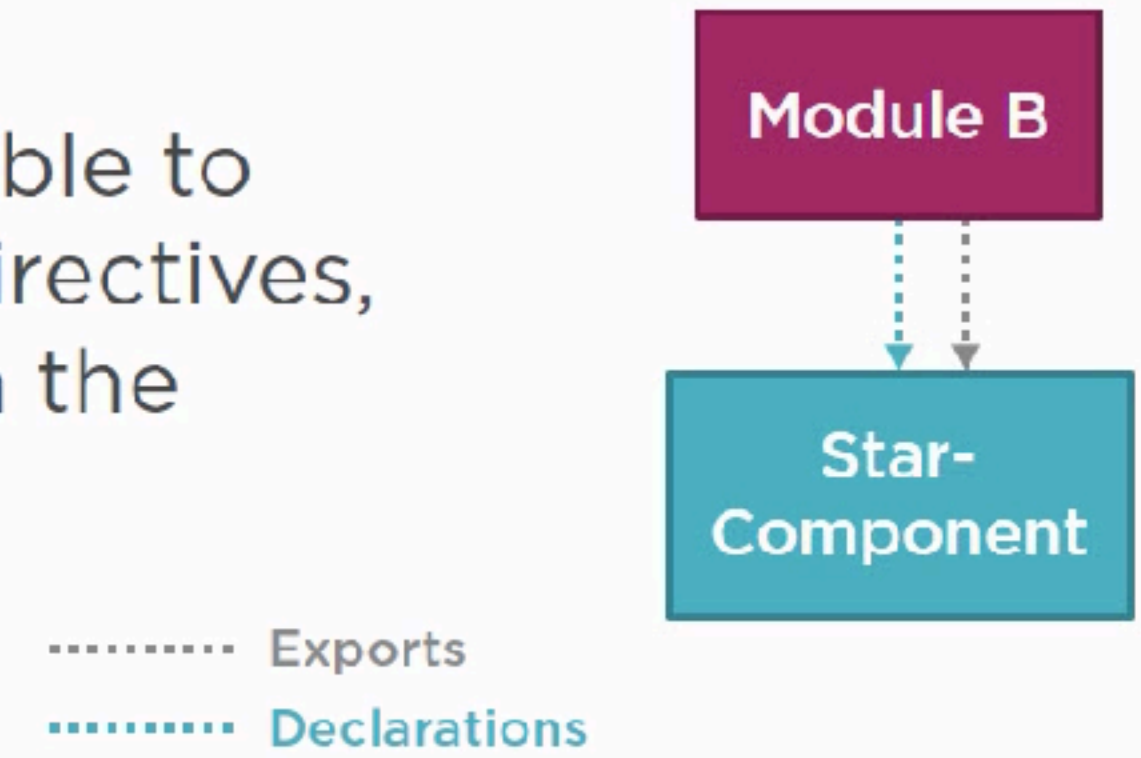
All declared components, directives, and pipes are private by default.

They are only accessible to other components, directives, and pipes declared in the same module.

Declarations Array Truth #4

All declared components, directives, and pipes are private by default.

They are only accessible to other components, directives, and pipes declared in the same module.



Declarations Array Truth #5

The Angular module provides the template resolution environment for its component templates.