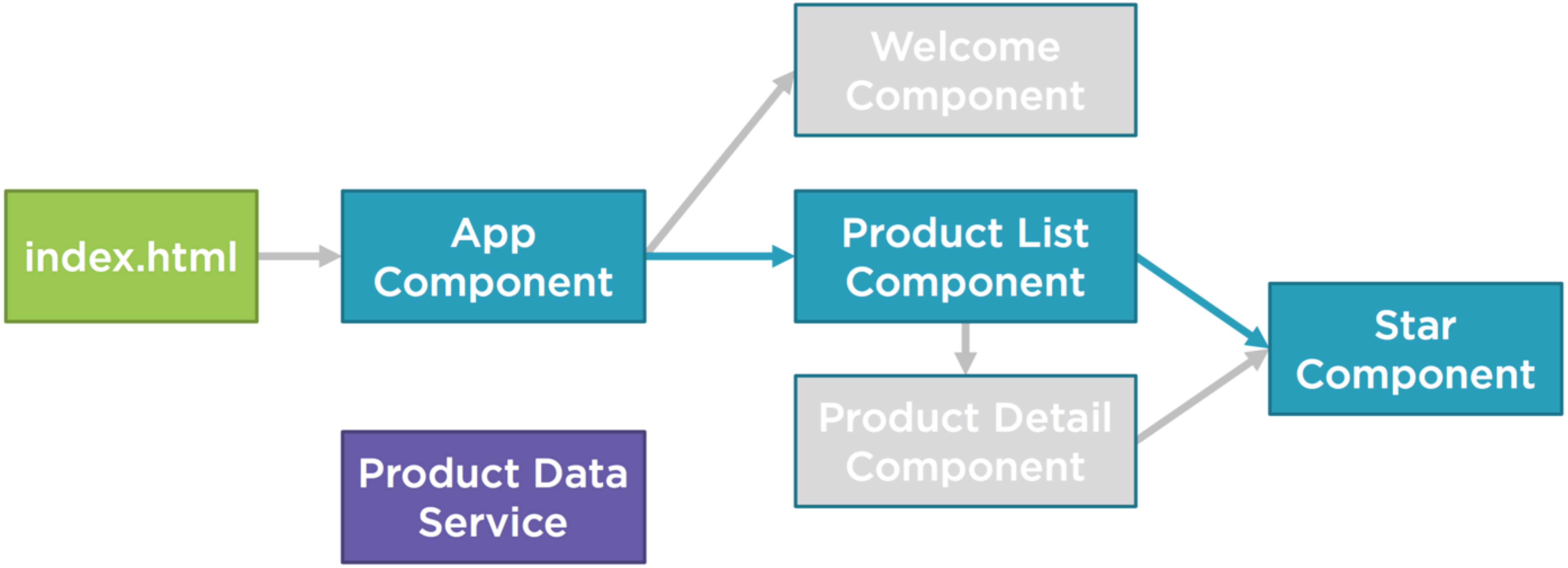
AngularJS

Retrieving data Using HTTP

Module Overview

- Observables and Reactive Extensions
- Sending an HTTP Request
- Exception Handling
- Subscribing to an Observable

Application Architecture



Observables and Reactive Extensions

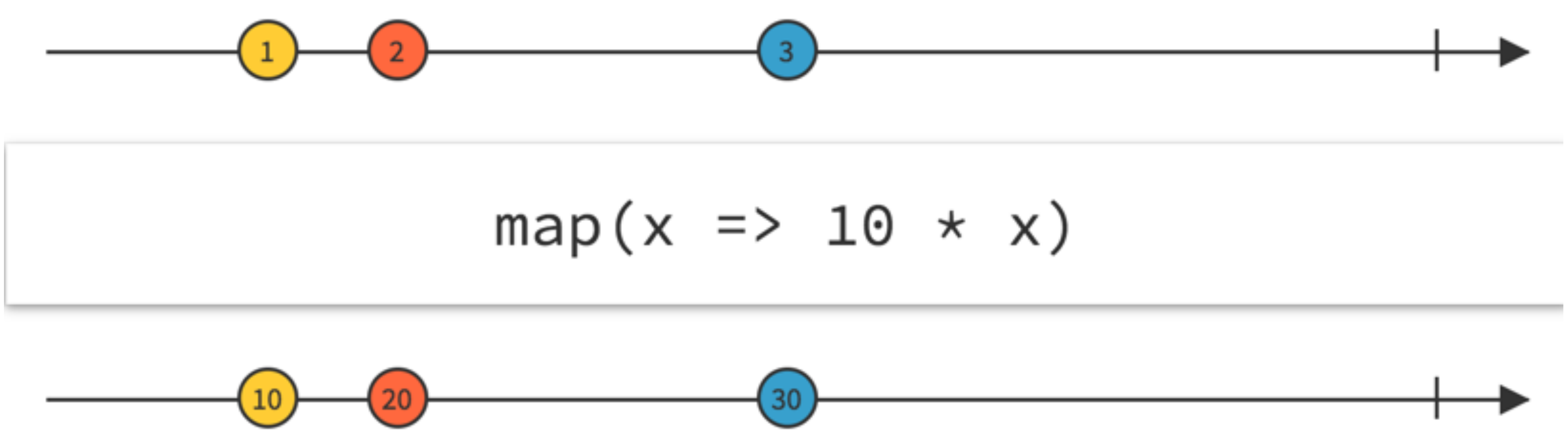
- Help manage asynchronous data
- Treat events as a collection
    - An array whose items arrive asynchronously over time
- Are a proposed feature for ES 2016
- Use Reactive Extensions (RxJS)
- Are used within built-in code in Angular

Observables Operators

- Methods on observables that compose new observables
- Transform the source observables in some way
- Process each value as it is emitted
- Examples: map, filter, take, merge, …

Observables

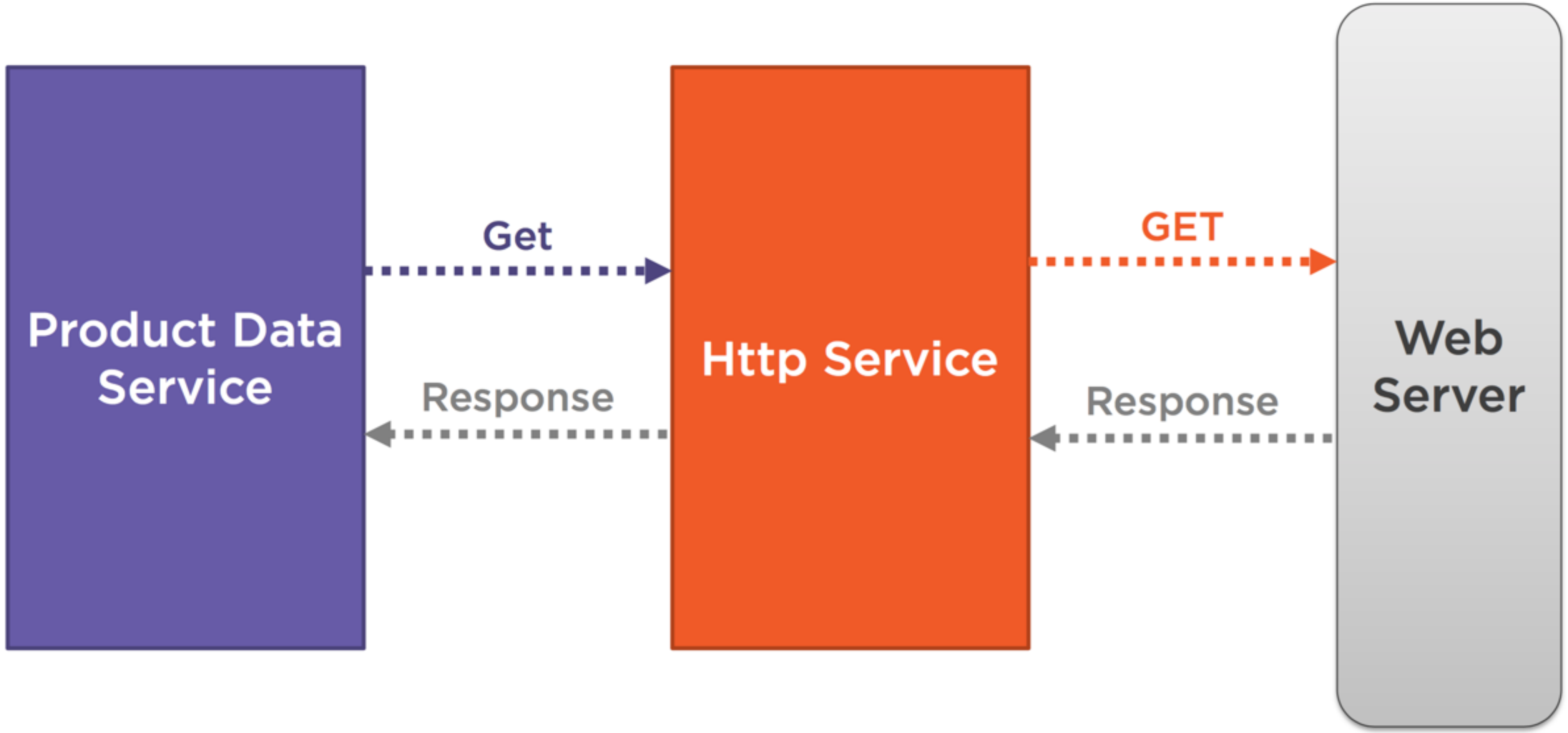
- http://rxmarbles.com



Promise vs Observables

Promise	**Observables**
Provides a single future value	Emits multiple value over time
Not lazy	Lazy
Not cancellable	Cancellable
	Suports map, filter and similar operators

Sending an HTTP Request

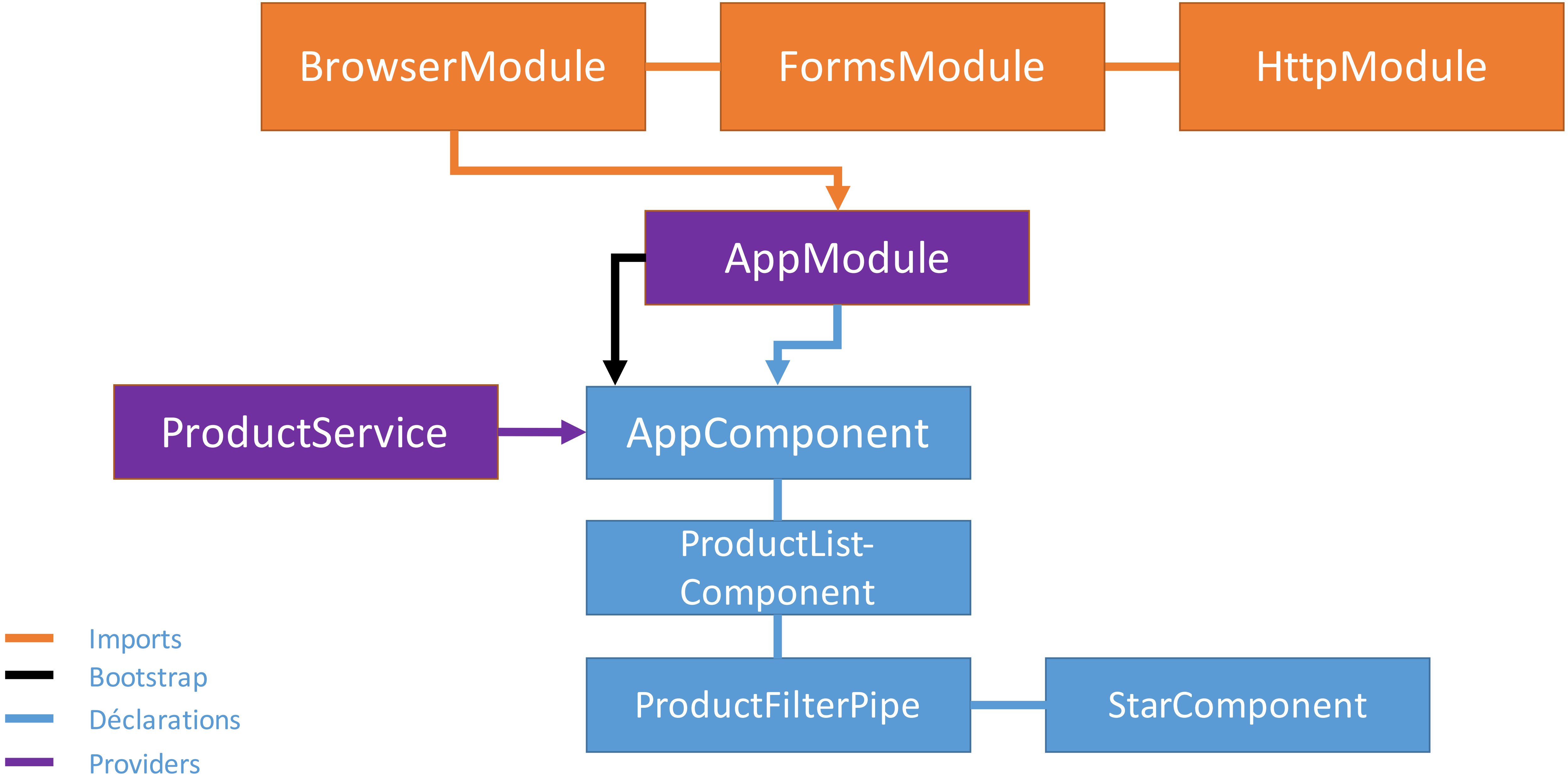


Sending an HTTP Request

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

import { Injectable } from '@angular/core';
import { Http } from '@angular/http';

import { IProduct } from 'product';
@Injectable()
export class ProductService {
  private _productUrl = 'localhost:3000/products';
  constructor(private _http: http) {}
  getProducts() {
    return this._http.get(this._productUrl);
  }
}
```


BrowserModule
FormsModule
HttpModule
AppModule
ProductService
AppComponent
ProductList-Component
ProductFilterPipe
StarComponent
Imports
Bootstrap
Déclarations
Providers

Declare http Module into AppModule

```
// app.module.ts
import { HttpModule } from '@angular/http';
@NgModule({
  imports: [
    BrowserModule, FormsModule, HttpModule
  ],
  declarations: [
    AppComponent, ProductListComponent,
    ProductFilterPipe, StarComponent
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule {}
```

Sending an HTTP Request

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

import { Injectable } from '@angular/core';
import { Http } from '@angular/http';

import { IProduct } from 'product';
@Injectable()
export class ProductService {
  private _productUrl = 'localhost:3000/products';
  constructor(private _http: http) {}
  getProducts() {
    return this._http.get(this._productUrl);
  }
}
```

Sending an HTTP Request

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

import { Injectable } from '@angular/core';
import { Http, Response } from '@angular/http';
import { Observable } from ’rxjs/Observable';

import { IProduct } from 'product';
@Injectable()
export class ProductService {
  private _productUrl = 'localhost:3000/products';
  constructor(private _http: http) {}
  getProducts(): Observable<Response> {
    return this._http.get(this._productUrl);
  }
}
```

Sending an HTTP Request

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

import { Injectable } from '@angular/core';
import { Http, Response } from '@angular/http';
import { Observable } from ’rxjs/Observable';
import 'rxjs/add/operator/map';

import { IProduct } from 'product';
@Injectable()
export class ProductService {
  private _productUrl = 'localhost:3000/products';
  constructor(private _http: http) {}
  getProducts(): Observable<Response> {
    return this._http.get(this._productUrl)
        .map((response: Response) => {
          return <IProduct[]>response.json();
        });
  }
}
```

Exception handling

```
// product/product.service.ts
…
import 'rxjs/add/operator/do';
import 'rxjs/add/operator/catch';
…

  getProducts(): Observable<Response> {
    return this._http.get(this._productUrl)
        .map((response: Response) => <IProduct[]>response.json())
        .do(data => console.log('All: ' + JSON.stringify(data))
        .catch(this.handleError);
       });
  }

  handleError(error: Response) { ... }
}
```

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)

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

Building a service

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

Checklist: Setup HttpModule

- ✓ Install @angular/http NPM package
- ✓ Add HttpModule to the imports array of one of the application's Angular Modules

Checklist: Service

- ✓ Import what we need
- ✓ Define a dependecy for th ehttp client service
  - ✓ Use the product service construtor
- ✓ Create a method for each HTTP Request
- ✓ Call the desired http method, such as get
  - ✓ Pass in the API URL
- ✓ Map the http response to a JSON object
- ✓ Add error handling

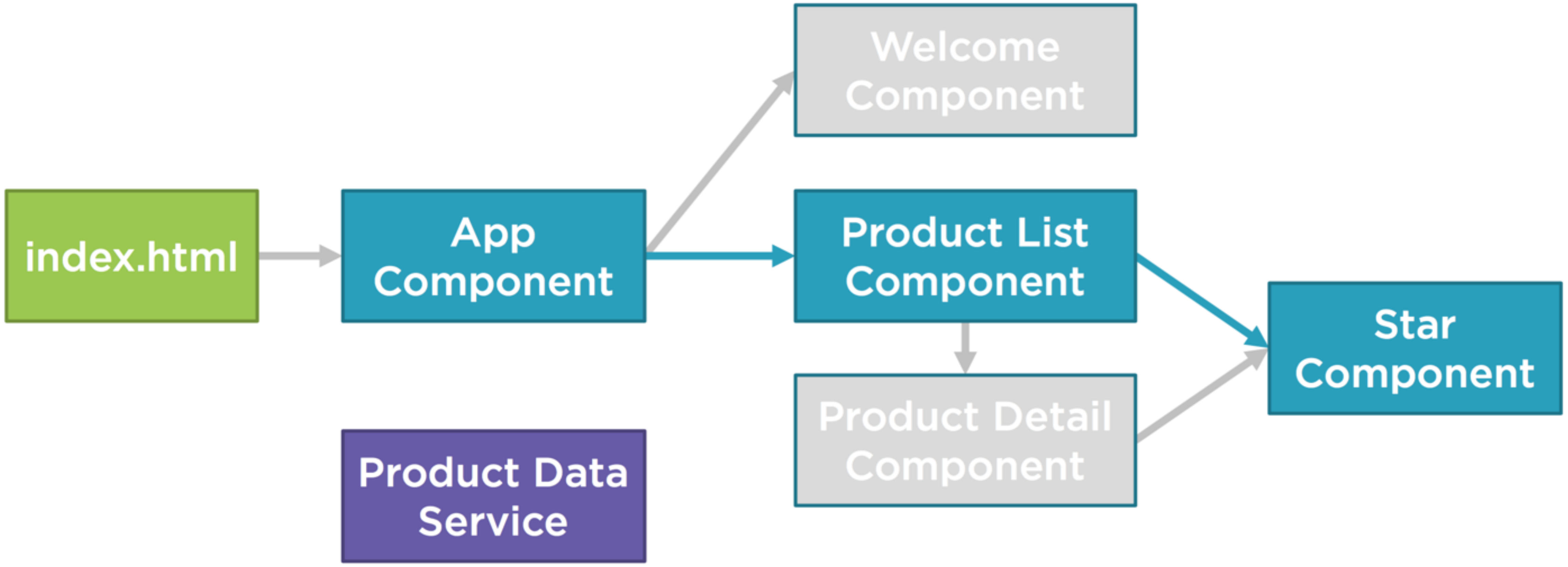
Checklist: Subscribing

- ✓ Call the subscribe method of the returned observable
- ✓ Provide a function to handle an emitted item
  - ✓ Normally assigns a property to the returned JSON Object
- ✓ Provide an error function to handle any returned errors

Module Overview

- Observables and Reactive Extensions
- Sending an HTTP Request
- Exception Handling
- Subscribing to an Observable

Application Architecture


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