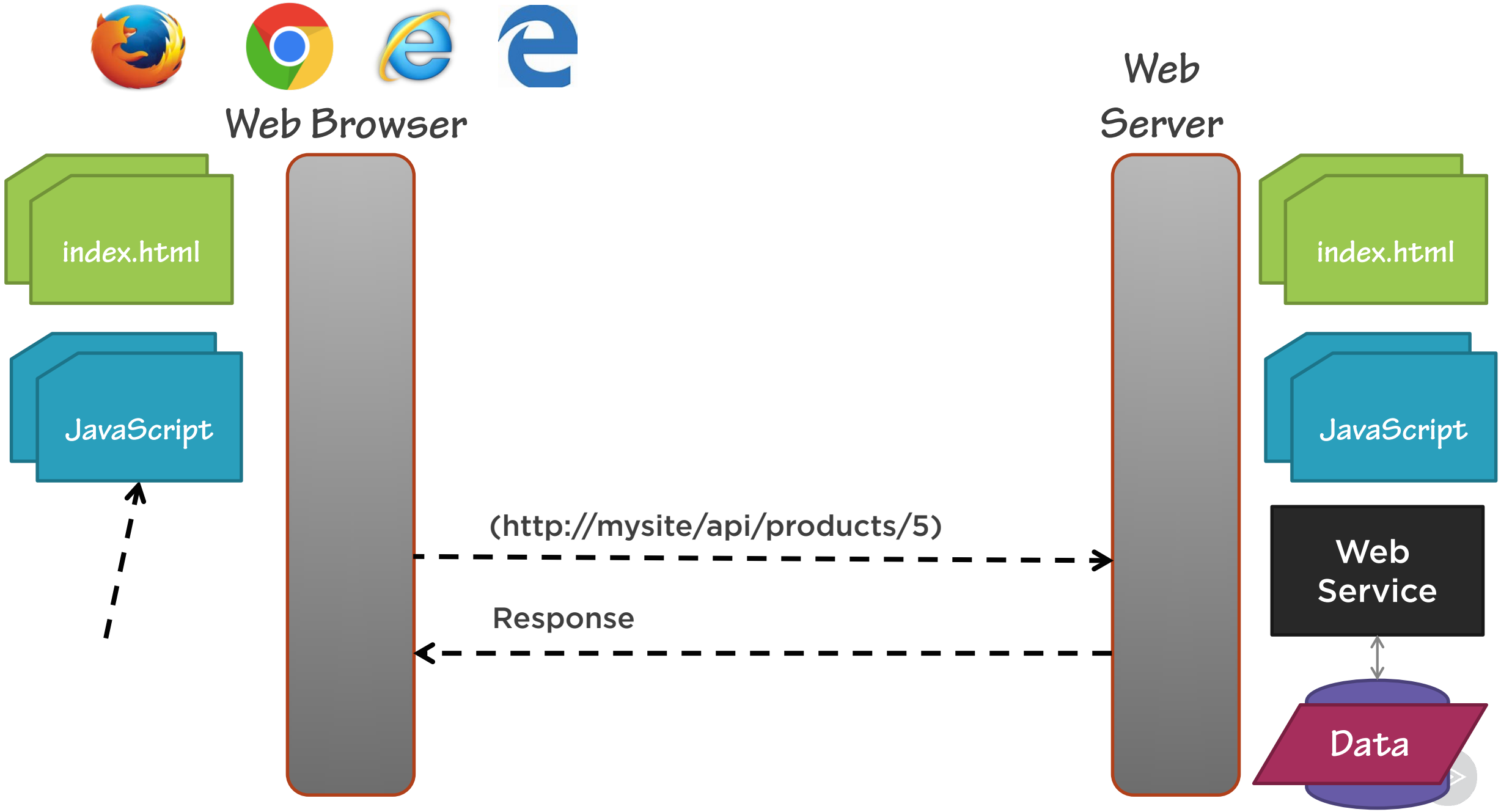

Web Browser
Web Server
index.html
JavaScript
(http://mysite/api/products/5)
Response
index.html
JavaScript
Web Service
Data

Module Overview

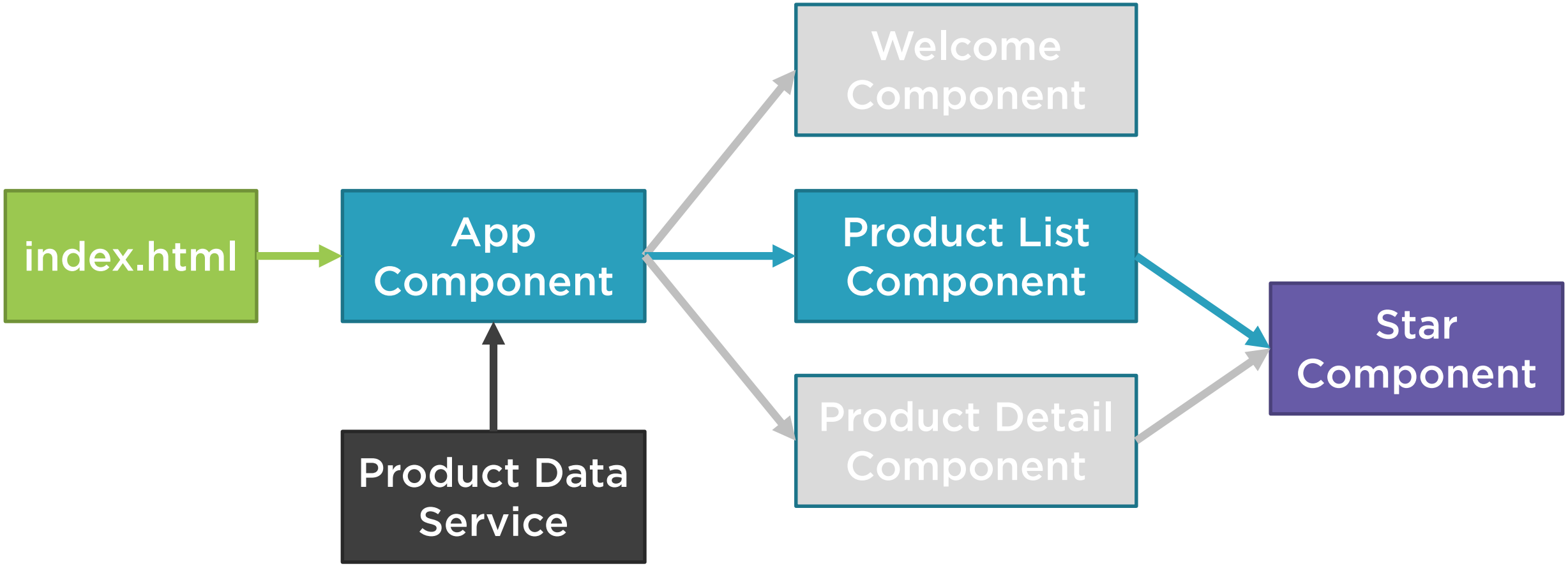
Observables and Reactive Extensions

Setting Up

Sending an Http Request

Subscribing to an Observable

Application Architecture



Observables and Reactive Extensions

An array whose items arrive asynchronously over time

Helps manage asynchronous data

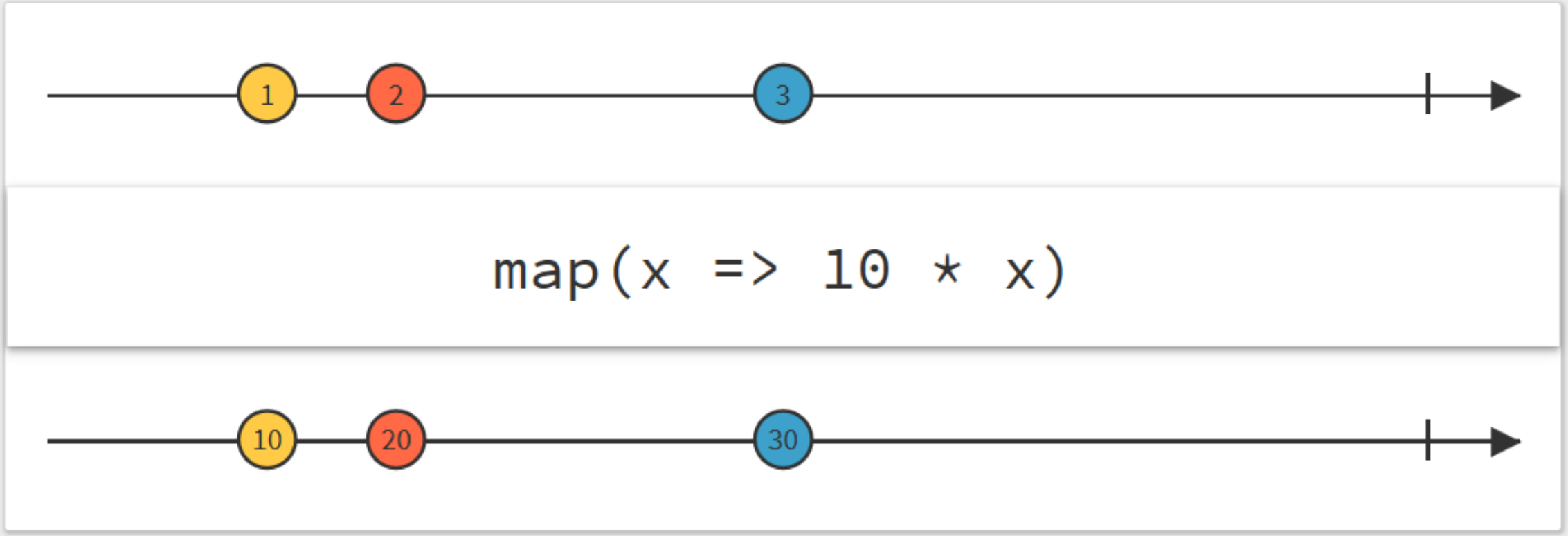
Proposed feature for ES 2016

Use Reactive Extensions (RxJS)

Used within Angular

Observables

Interactive diagrams of Rx Observables



Promise vs Observable

Promise	Observable
Returns a single value	Works with multiple values over time
Not cancellable	Cancellable
	Supports map, filter, reduce and similar operators

Setting Up

Include the Angular 2 Http script

Register HTTP_PROVIDERS

Import RxJS

Sending an Http Request

product.service.ts

```
import { Http, Response} from 'angular2/http'
import { Observable } from 'rxjs/Observable'

@Injectable()
export class ProductService {
  private _productUrl = 'www.myWebService.com/api/products';
  constructor(private _http: Http) { }

  getProducts(): Observable<IProduct[]> {
   return this._http.get(this._productUrl)
                .map((response: Response) => <IProduct[]>response.json());
  }
}
```

Handling Errors

product.service.ts

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

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

Subscribing to an Observable

product-list.component.ts

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

Http Checklist: Setup

- Include the Angular 2 Http script
- Register HTTP_PROVIDERS
- Import RxJS

Http Checklist: Service

Import what we need

Define a dependency for the http client service

- Use a constructor parameter

Create a method for each http request

Call the desired http method, such as get

- Pass in the Url

Map the Http response to a JSON object

Add error handling

Http 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

Summary

- Observables and Reactive Extensions
- Setting Up
- Sending an Http Request
- Subscribing to an Observable

Application Architecture