Data with Http

John Papa
PRINCIPAL ARCHITECT

@john_papa   www.johnpapa.net

Overview

Http

RxJS, Observables, and Subscriptions

Async Pipe

Promises

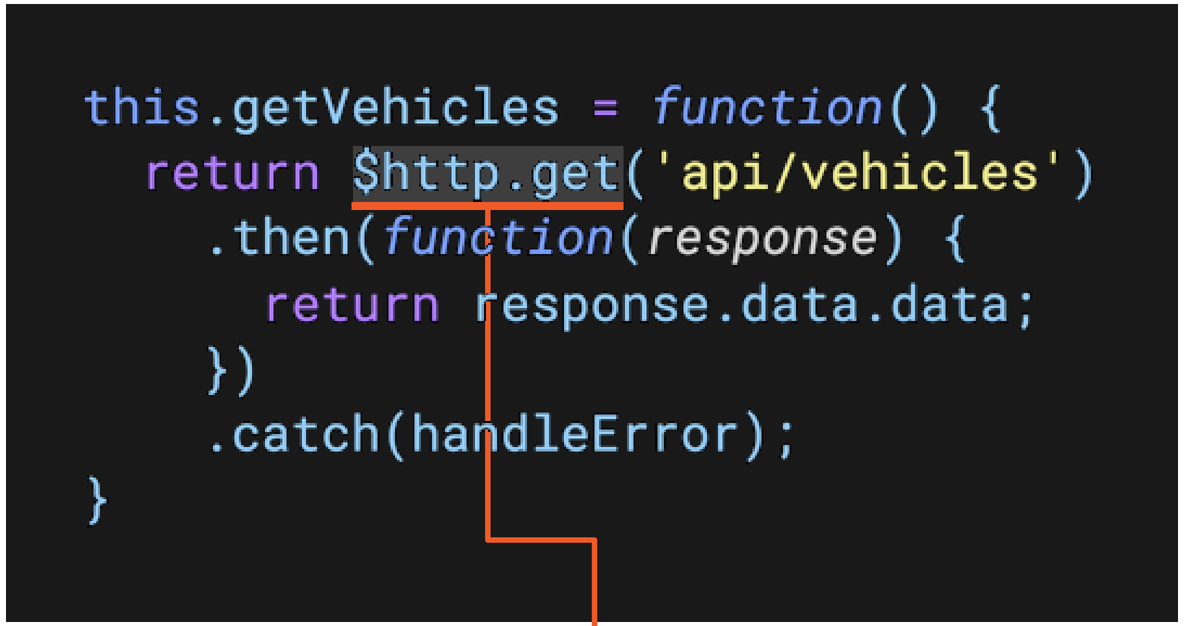
Http

Http

We use Http to get and save data with Promises or Observables. We isolate the http calls in a shared Service.

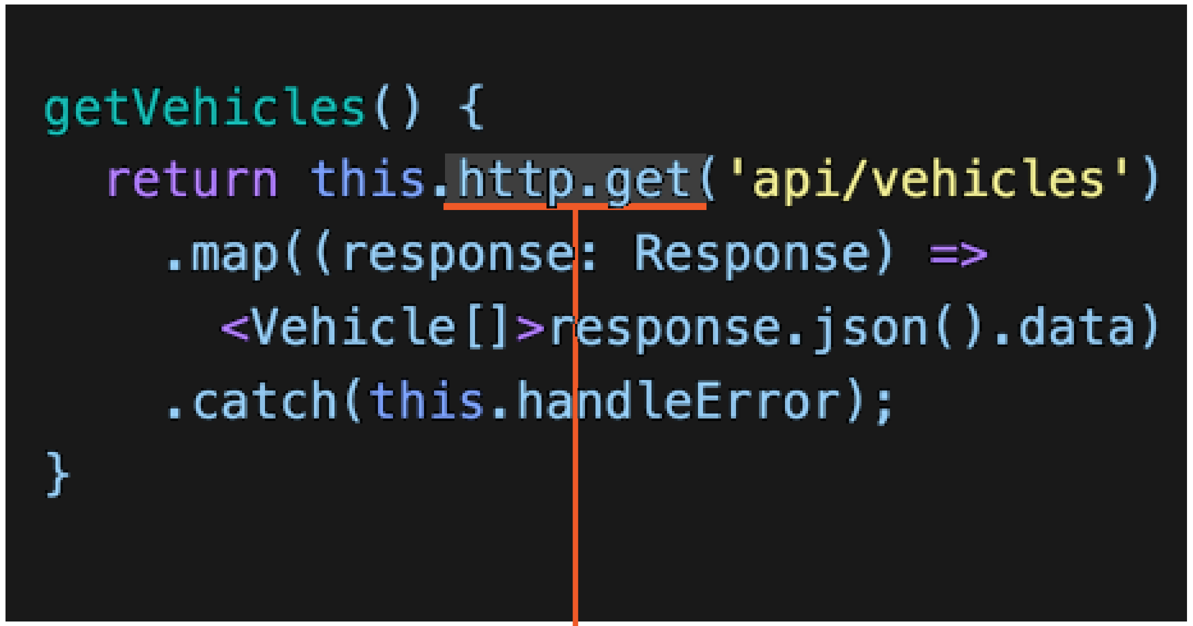
Http Then and Now

Angular 1

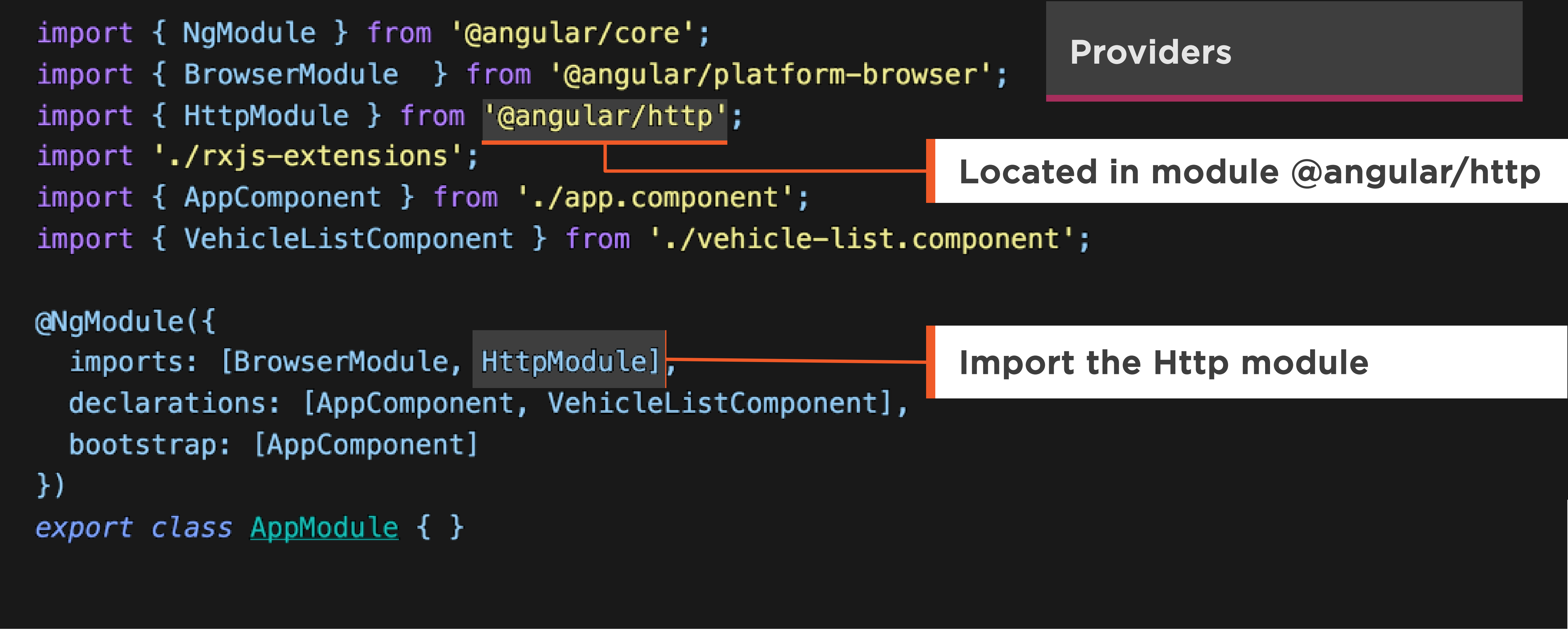


```
this.getVehicles = function() {
  return $http.get('api/vehicles')
    .then(function(response) {
      return response.data.data;
    })
    .catch(handleError);
}
```

Angular 2



```
getVehicles() {
  return this.http.get('api/vehicles')
    .map((response: Response) =>
      <Vehicle[]>response.json().data)
    .catch(this.handleError);
}
```



```
import { NgModule } from '@angular/core';
import { BrowserModule  } from '@angular/platform-browser';
import { HttpModule } from '@angular/http';
import './rxjs-extensions';
import { AppComponent } from './app.component';
import { VehicleListComponent } from './vehicle-list.component';

@NgModule({
  imports: [BrowserModule, HttpModule],
  declarations: [AppComponent, VehicleListComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Http Requirements

We need the HttpModule to make Http calls

vehicle.service.ts

```
@Injectable()
export class VehicleService {
  constructor(private http: Http) { }

  getVehicles() {
    return this.http.get('api/vehicles')
      .map((response: Response) => <Vehicle[]>response.json().data)
      .catch(this.handleError);
  }

  private handleError(error: Response) {
    console.error(error);
    return Observable.throw(error.json().error || 'Server error');
  }
}
```

Make and return the async GET call

Map the response

Handle any exception

vehicle-list.component.ts

```
constructor(private vehicleService: VehicleService) { }
getHeroes() {
  this.vehicleService.getVehicles()
    .subscribe(
      vehicles => this.vehicles = vehicles,
      error =>  this.errorMessage = <any>error
  );
}
ngOnInit() { this.getHeroes(); }
```

Subscribe to the observable

Success and failure cases

Subscribing to the Observable

Component is handed an Observable

We Subscribe to it

1

2

3

Http Step by Step

Http Step by Step

1

2

3

1

Import HttpModule

Http Step by Step

1

2

3

1

Import HttpModule

2

Call Http.get in a Service and return the mapped result

Http Step by Step

1

2

3

1

Import HttpModule

2

Call Http.get in a Service and return the mapped result

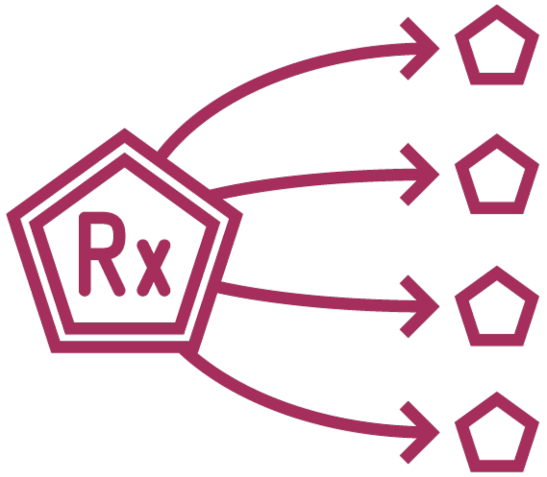
3

Subscribe to the Service's function in the Component

Demo

Http

RxJs



http://reactivex.io/

RxJs

RxJs (Reactive Js) implements the asynchronous observable pattern and is widely used in Angular 2 app.module.ts

```
import 'rxjs/Rx';
```

Imports all of RxJs

Importing RxJs

RxJs is a large library

For production, only import the modules you require

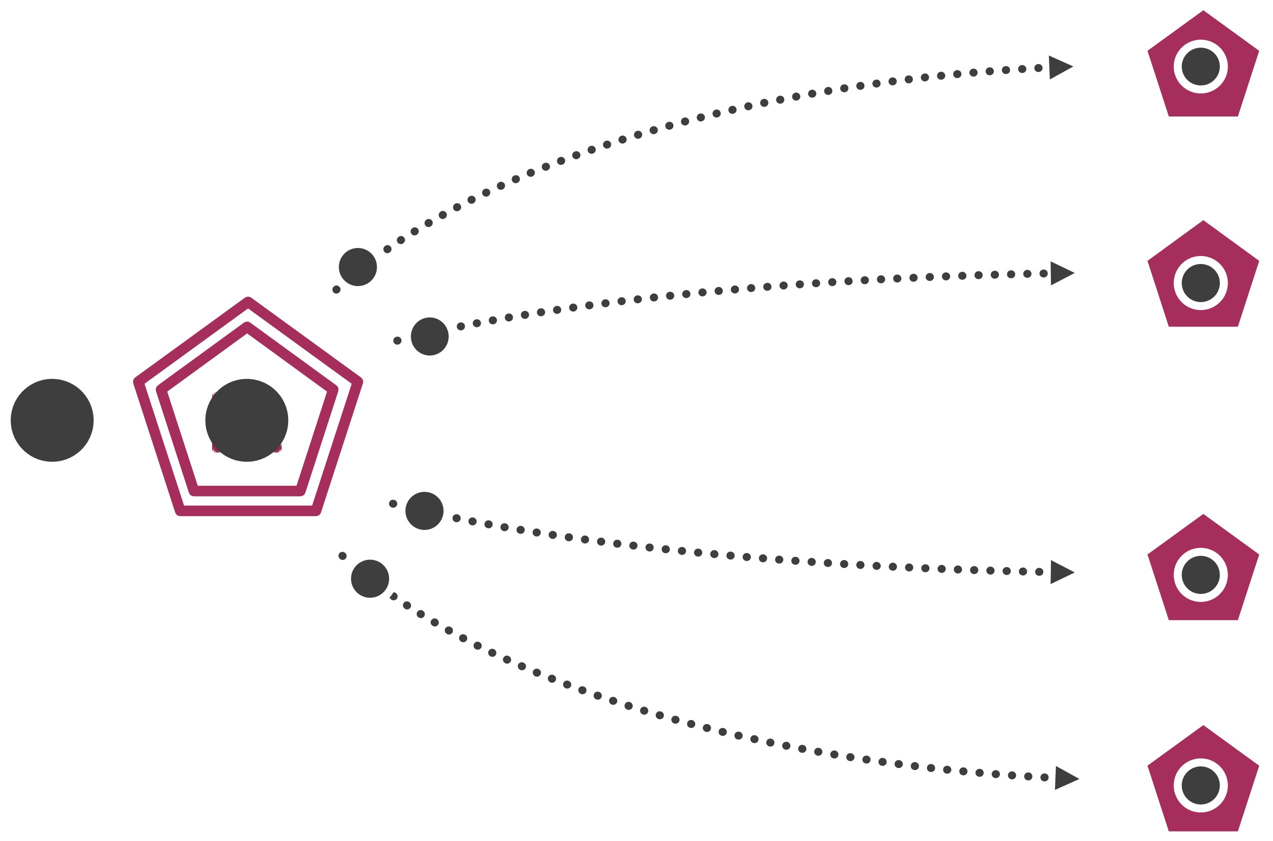
app.module.ts

```
import './rxjs-extensions';
```

rxjs-extensions.ts

```
import 'rxjs/add/operator/catch';
import 'rxjs/add/operator/do';
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/toPromise';
```

Prefer importing only what we use from RxJs



vehicle.service.ts

```
return this.http.get('api/vehicles')
  .map((response: Response) =>
    <Vehicle[]>response.json().data
  )
  .catch(this.handleError);
```

Retrieve the JSON

data is what we defined on the server

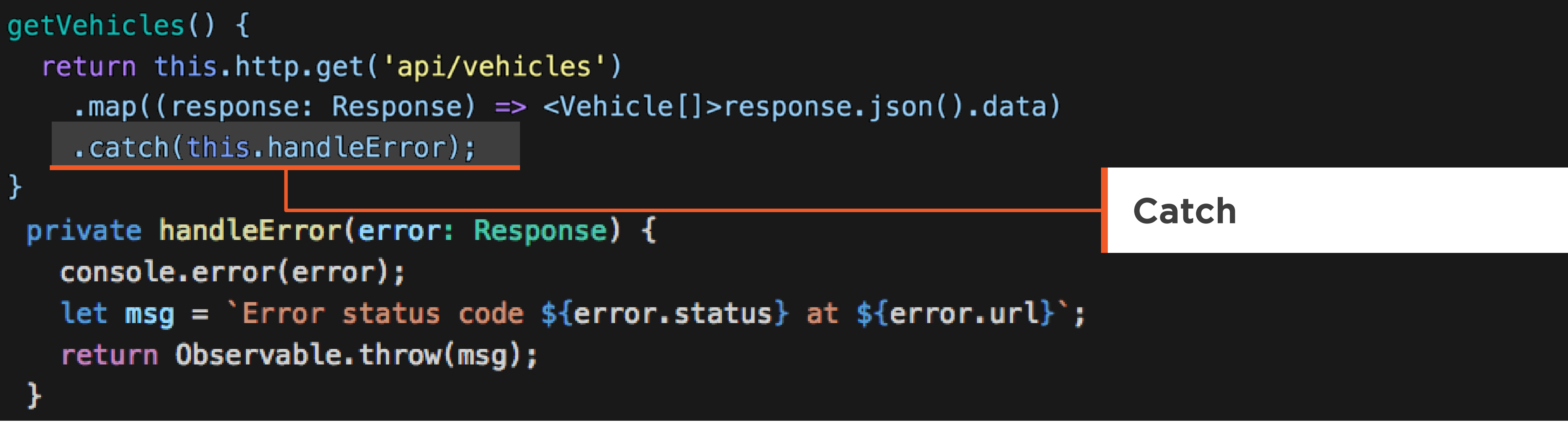
Returning from Http

We don’t return the response

Service does the work

The consumers simply get the data

Catching Errors



```
getVehicles() {
  return this.http.get('api/vehicles')
    .map((response: Response) => <Vehicle[]>response.json().data)
    .catch(this.handleError);
}
 private handleError(error: Response) {
   console.error(error);
   let msg = `Error status code ${error.status} at ${error.url}`;
   return Observable.throw(msg);
 }
```

Exception Handling

We catch errors in the Service

We pass some error messages to the consumer for presentation

vehicle-list.component.ts

```
getHeroes() {
  this.vehicleService.getVehicles()
   .subscribe(
     vehicles => this.vehicles = vehicles,
     error =>  this.errorMessage = <any>error
  );
}
```

Subscribe to the observable

Success and failure cases

Subscribing to the Observable

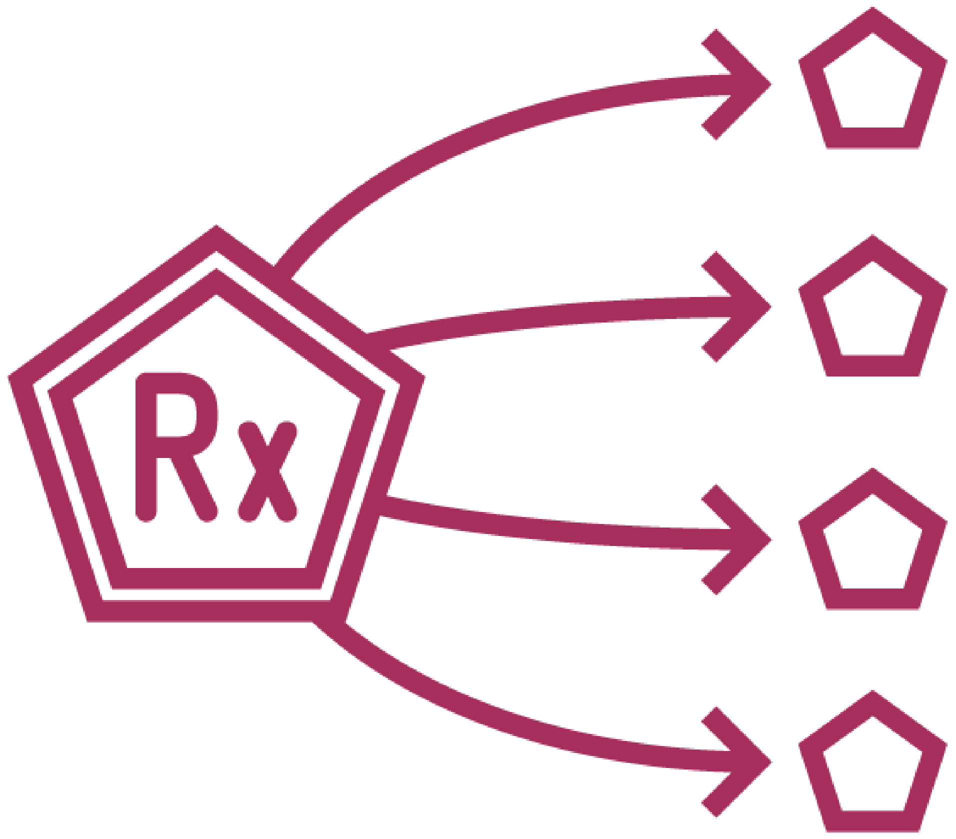
Component is handed an Observable

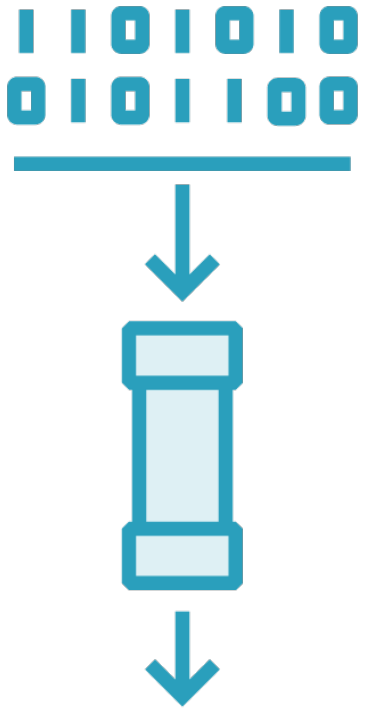
We Subscribe to it

We can handle errors here, for presenting to the user if we wish

http://reactivex.io/

RxJs

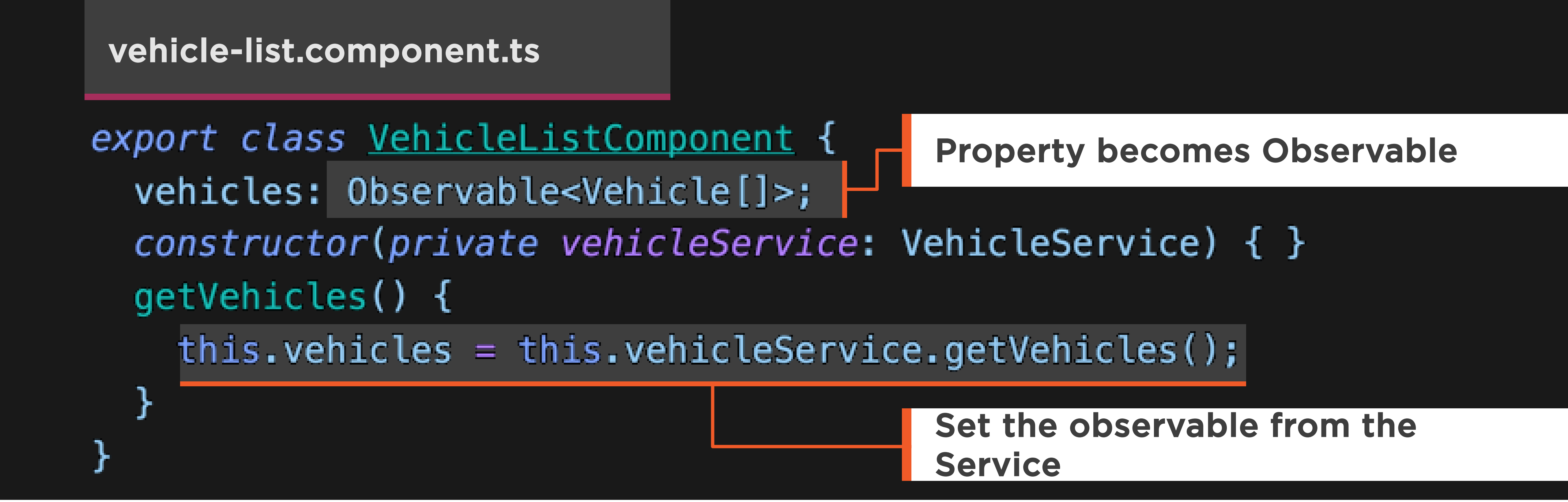




Async Pipe

Async Pipe

The Async Pipe receives a Promise or Observable as input and subscribes to the input, eventually emitting the value(s) as changes arrive.


vehicle-list.component.ts

```
export class VehicleListComponent {
  vehicles: Observable<Vehicle[]>;
  constructor(private vehicleService: VehicleService) { }
  getVehicles() {
    this.vehicles = this.vehicleService.getVehicles();
  }
}
```

Property becomes Observable
Set the observable from the Service

Observable Properties

Component is simplified

Grab the Observable and set it to the property

vehicle-list.component.html

```
<ul>
  <li *ngFor="let vehicle of vehicles | async">
    {{ vehicle.name }}
  </li>
</ul>
```

Subscribes to the Vehicles

Async Pipe in the Template

Apply the async Pipe

Demo

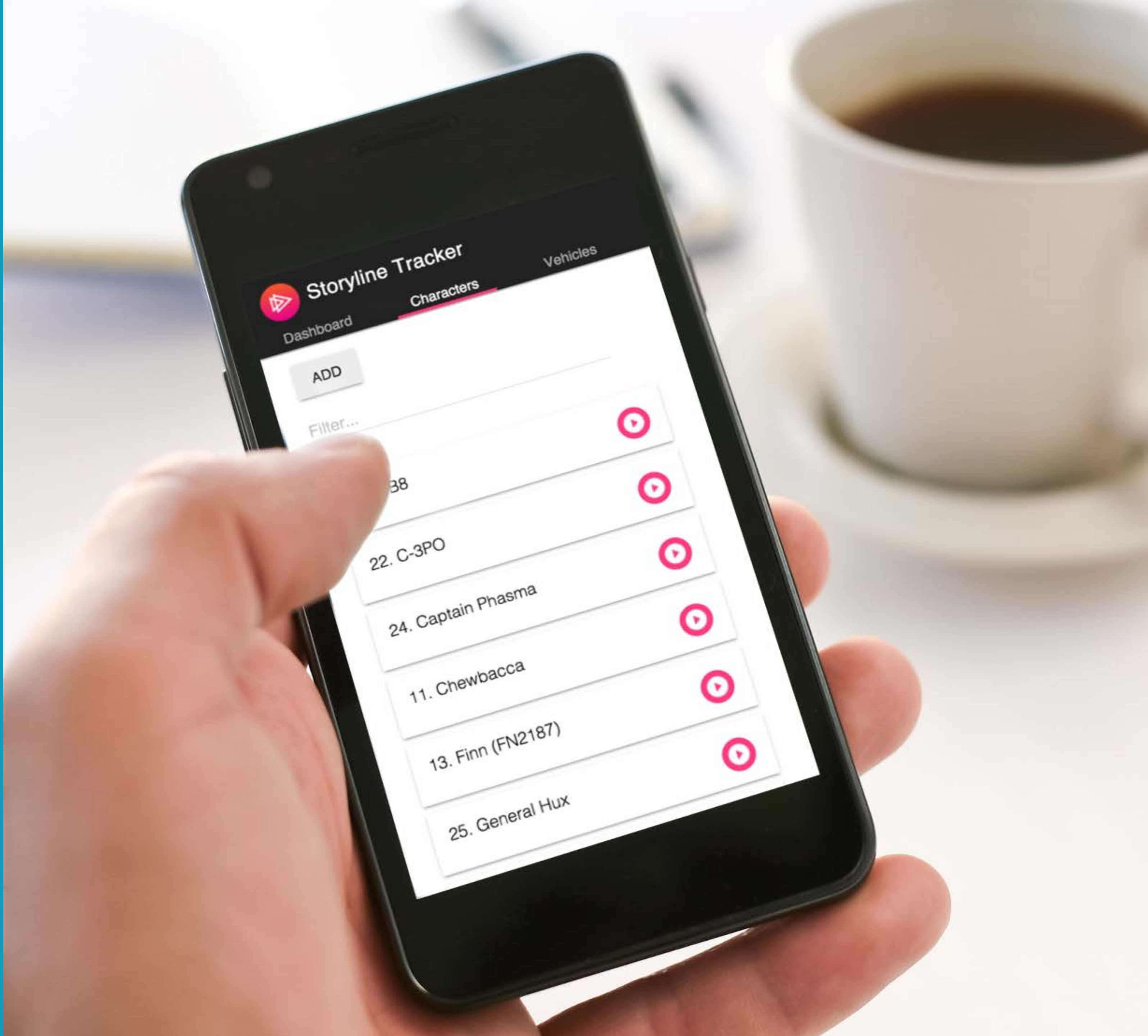
Async

Demo

Promises

Demo

Putting It All Together



Http

Http

Observables and Subscriptions

Async Pipe

Promises