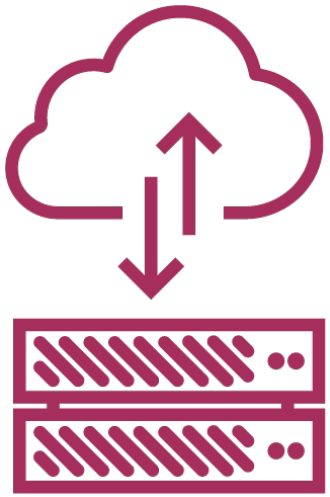
Overview

Http

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);
}
```

Index.html

```
<script src="../node_modules/angular2/bundles/http.dev.js"></script>
```

Http script

Http is in a separate module

Add the reference to http.dev.js

```
import { Component } from 'angular2/core';
import { HTTP_PROVIDERS } from 'angular2/http';

import { Vehicle, VehicleService } from './vehicle.service';
import { VehicleListComponent } from './vehicle-list.component';

@Component({
  selector: 'my-app',
  template: '<my-vehicle-list></my-vehicle-list>',
  directives: [VehicleListComponent],
  providers: [
    HTTP_PROVIDERS,
    VehicleService
  ]
})
export class AppComponent {}
```

Providers

Located in module angular2/http

Declaring the providers

Http Requirements

HTTP_PROVIDERS is an array of service providers for Http

vehicle.service.ts

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

  getVehicles() {
    return this._http.get('api/vehicles.json')
      .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) { }
ngOnInit() { this.getHeroes(); }
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

Http Step by Step

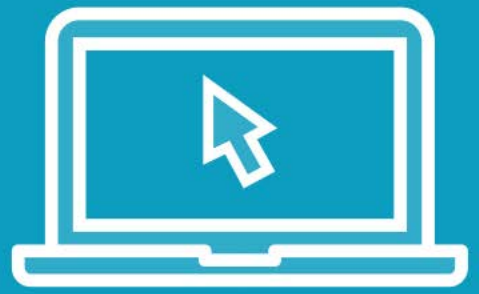
Add script reference to http in index.html

Register the Http providers

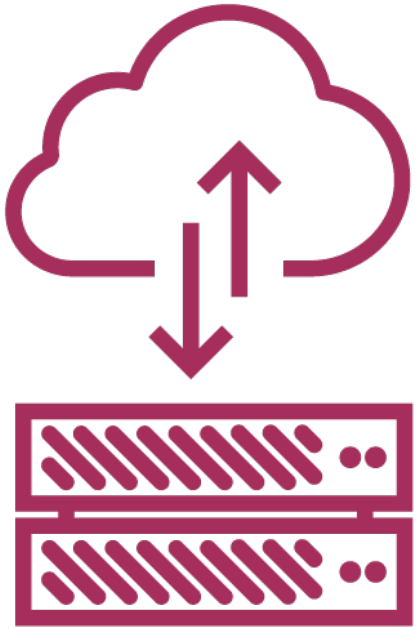
Call Http.get in a Service and return the mapped result

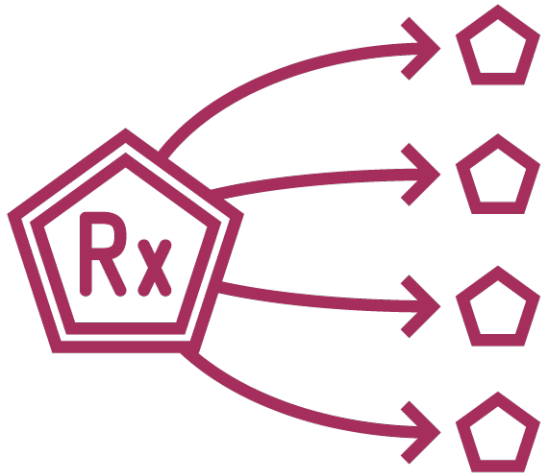
Subscribe to the Service's function in the Component

Demo



Http





RxJs

http://reactivex.io/

RxJs

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

main.ts

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

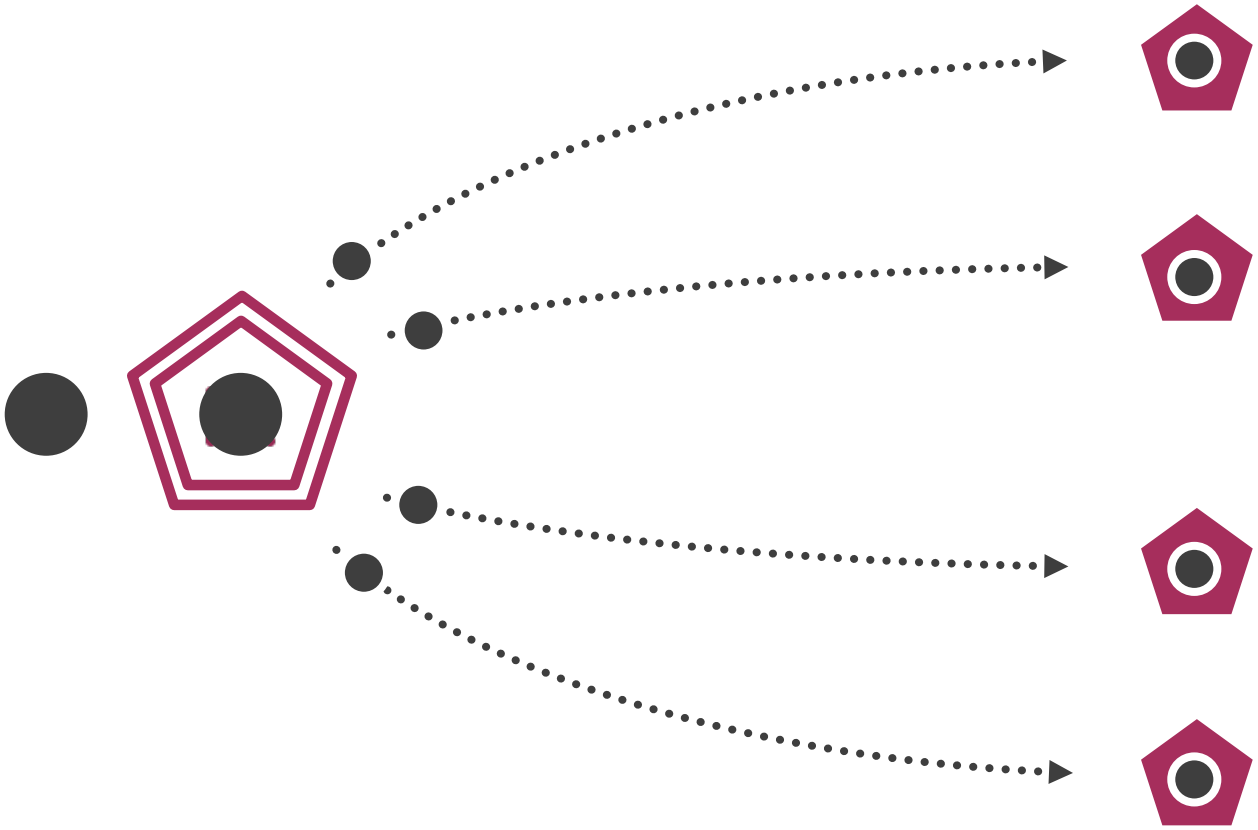
Import all of RxJs ... for now

Importing RxJs

RxJs is a large library

For learning, import it all

For production, only import the modules you require



vehicle.service.ts

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

json() is defined by the http spec

data is what we defined on the server

Returning from Http

We do not return the response

Service does the dirty 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);
  return Observable.throw(error.json().error || 'Server error');
}
```

Catch

Exception Handling

We catch errors in the Service

We sometimes pass 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

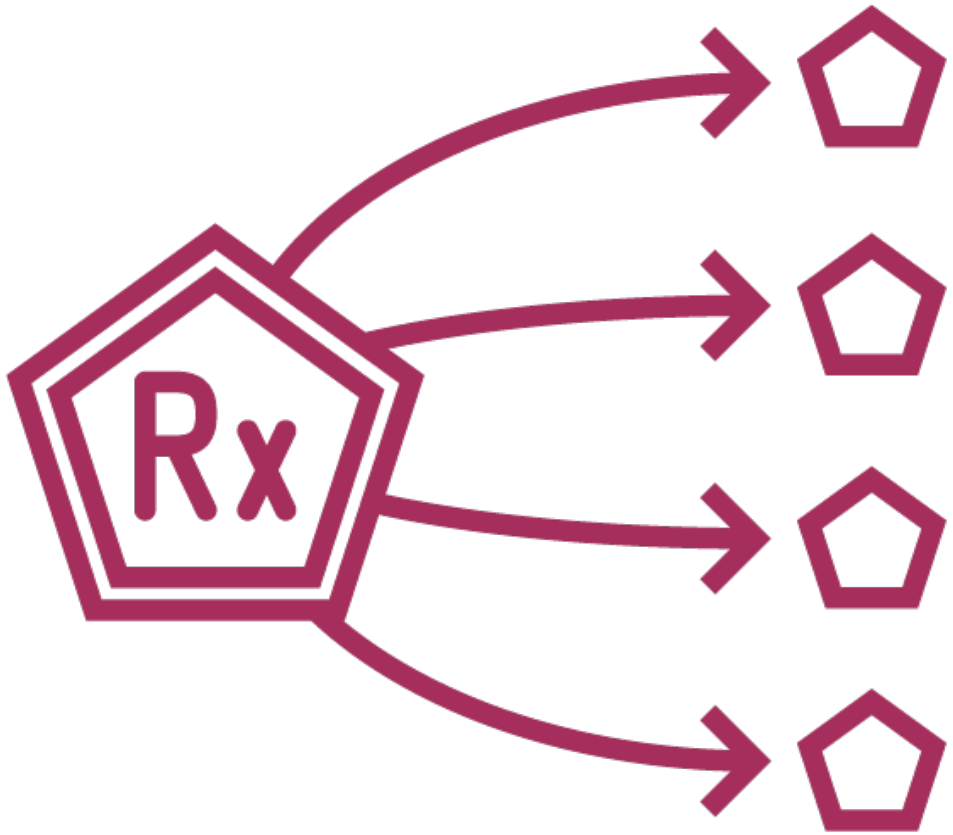
Subscribing to the Observable

Component is handed an Observable

We Subscribe to it

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

```
vehicles: Observable<Vehicle[]>;

getHeroes() {
  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

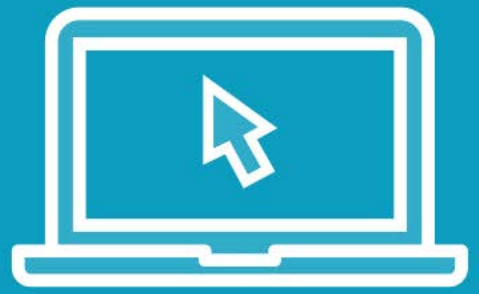
```
<ul>
  <li *ngFor="#vehicle of vehicles | async">
    {{ vehicle.name }}
  </li>
</ul>
```

Subscribes to the Vehicles

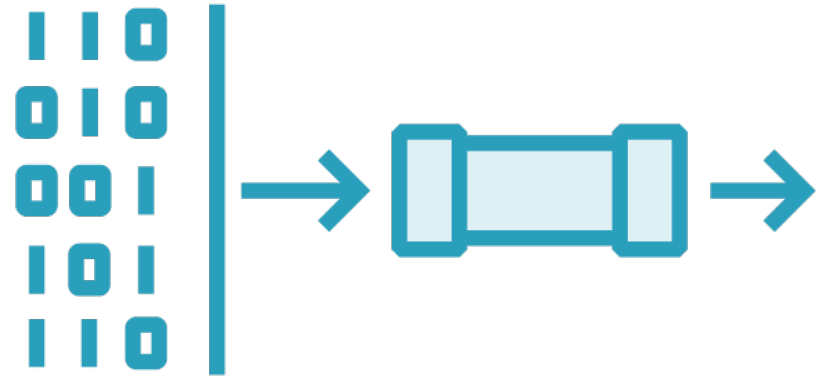
Async Pipe in the Template

Apply the async Pipe

Demo

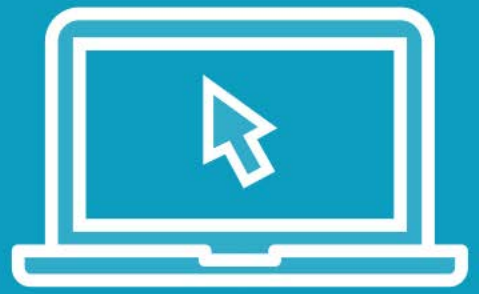


Async

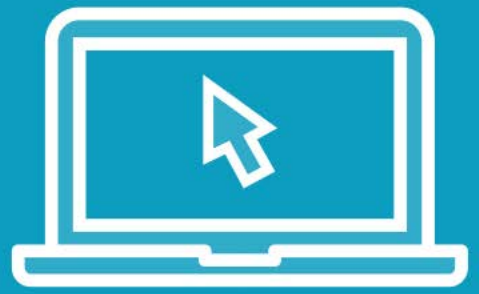


Promises

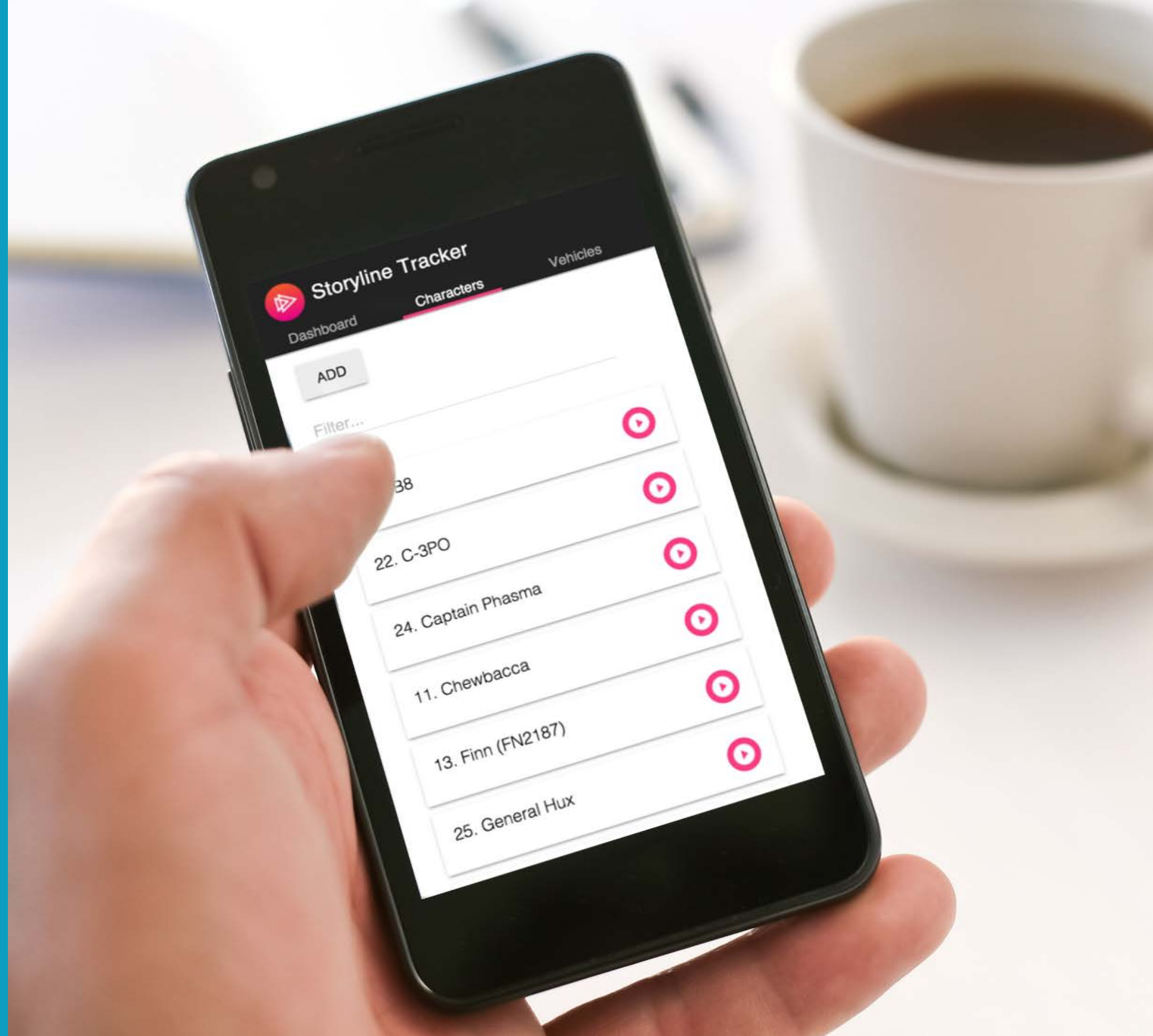
Demo



Demo



Putting It All Together



Http

Http

Observables and Subscriptions

Async Pipe

Promises