GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

ESTRATEGIA DE EMPRENDIMIENTO Y EMPLEO JOVEN
garantía juvenil

DESARROLLADOR SOFTWARE – TALLER ANGULAR

FECHA- [18-19]- MAYO 2022 **Horario: 15:00 a 20:30 h**

Juan Antonio Herrerías Berbel

Índice

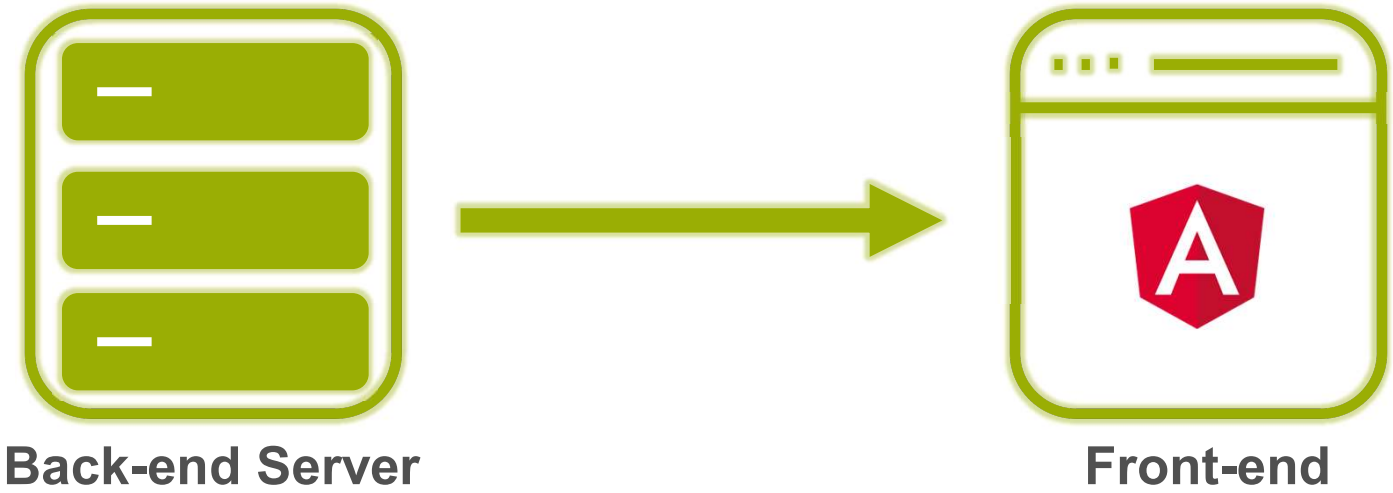
Herre - jaherrerias@gmail.com

01

Angular

What Is Angular?

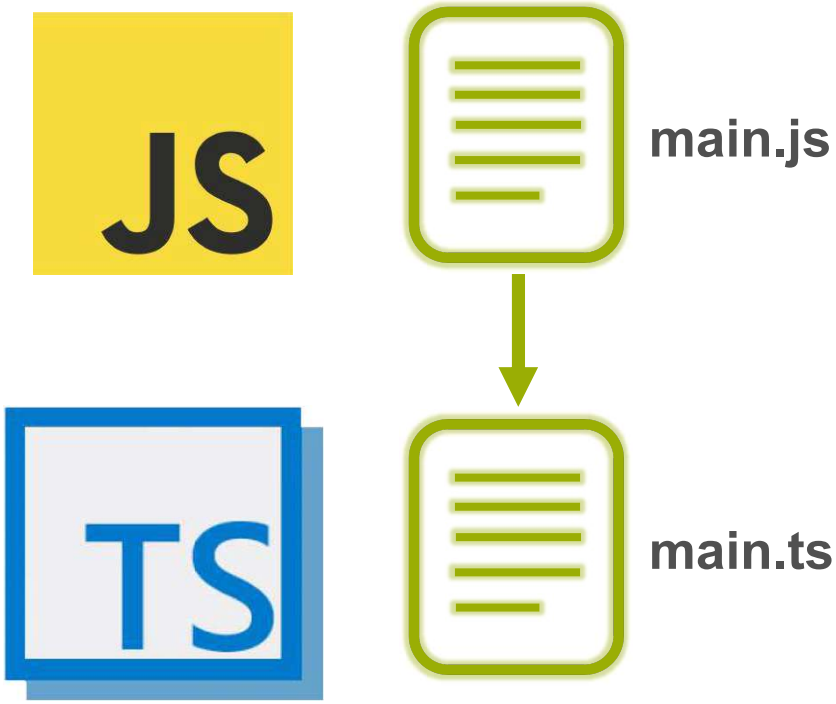
- Angular is a framework for dynamic web applications (SPA).
- Provides a way to organize your HTML, JavaScript, and CSS to keep your front-end code clean.
- Released in 2011.
- Mainly maintained by Google with the help of the open-source community.



Reminder, TypeScript: Our Language of Choice

- TypeScript is Microsoft's extension of JavaScript that allows the use of all ES2016 features and adds powerful type checking and object-oriented features.

- The Angular source is programmed with TypeScript.



ANGULAR CLI

Command Line Interface

Angular CLI

- The Angular CLI is a tool to initialize, develop, scaffold and maintain Angular applications.

```
> npm install -g @angular/cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
```

Angular CLI

Angular CLI help us with:

- HTTP Server
- Live-reload system
- Testing tools
- Deploying tools
- To create new Angular components, services and pipes

Scaffold	Usage
Component	`ng g component my-new-component`
Directive	`ng g directive my-new-directive`
Pipe	`ng g pipe my-new-pipe`
Service	`ng g service my-new-service`
Class	`ng g class my-new-class`
Interface	`ng g interface my-new-interface`
Enum	`ng g enum my-new-enum`
Module	`ng g module my-module`

Generating and serving an Angular project via a development server

```
ng new awesome-shop

cd awesome-shop

ng serve
```

Cmder

```
CREATE awesome-shop/karma.conf.js (1429 bytes)
CREATE awesome-shop/tsconfig.app.json (287 bytes)
CREATE awesome-shop/tsconfig.spec.json (333 bytes)
CREATE awesome-shop/.vscode/extensions.json (130 bytes)
CREATE awesome-shop/.vscode/launch.json (474 bytes)
CREATE awesome-shop/.vscode/tasks.json (938 bytes)
CREATE awesome-shop/src/favicon.ico (948 bytes)
CREATE awesome-shop/src/index.html (297 bytes)
CREATE awesome-shop/src/main.ts (372 bytes)
CREATE awesome-shop/src/polyfills.ts (2338 bytes)
CREATE awesome-shop/src/styles.css (80 bytes)
CREATE awesome-shop/src/test.ts (745 bytes)
CREATE awesome-shop/src/assets/.gitkeep (0 bytes)
CREATE awesome-shop/src/environments/environment.prod.ts (51 bytes)
CREATE awesome-shop/src/environments/environment.ts (658 bytes)
CREATE awesome-shop/src/app/app-routing.module.ts (245 bytes)
CREATE awesome-shop/src/app/app.module.ts (393 bytes)
CREATE awesome-shop/src/app/app.component.html (23364 bytes)
CREATE awesome-shop/src/app/app.component.spec.ts (1091 bytes)
CREATE awesome-shop/src/app/app.component.ts (216 bytes)
CREATE awesome-shop/src/app/app.component.css (0 bytes)
√ Packages installed successfully.
```

cmd.exe

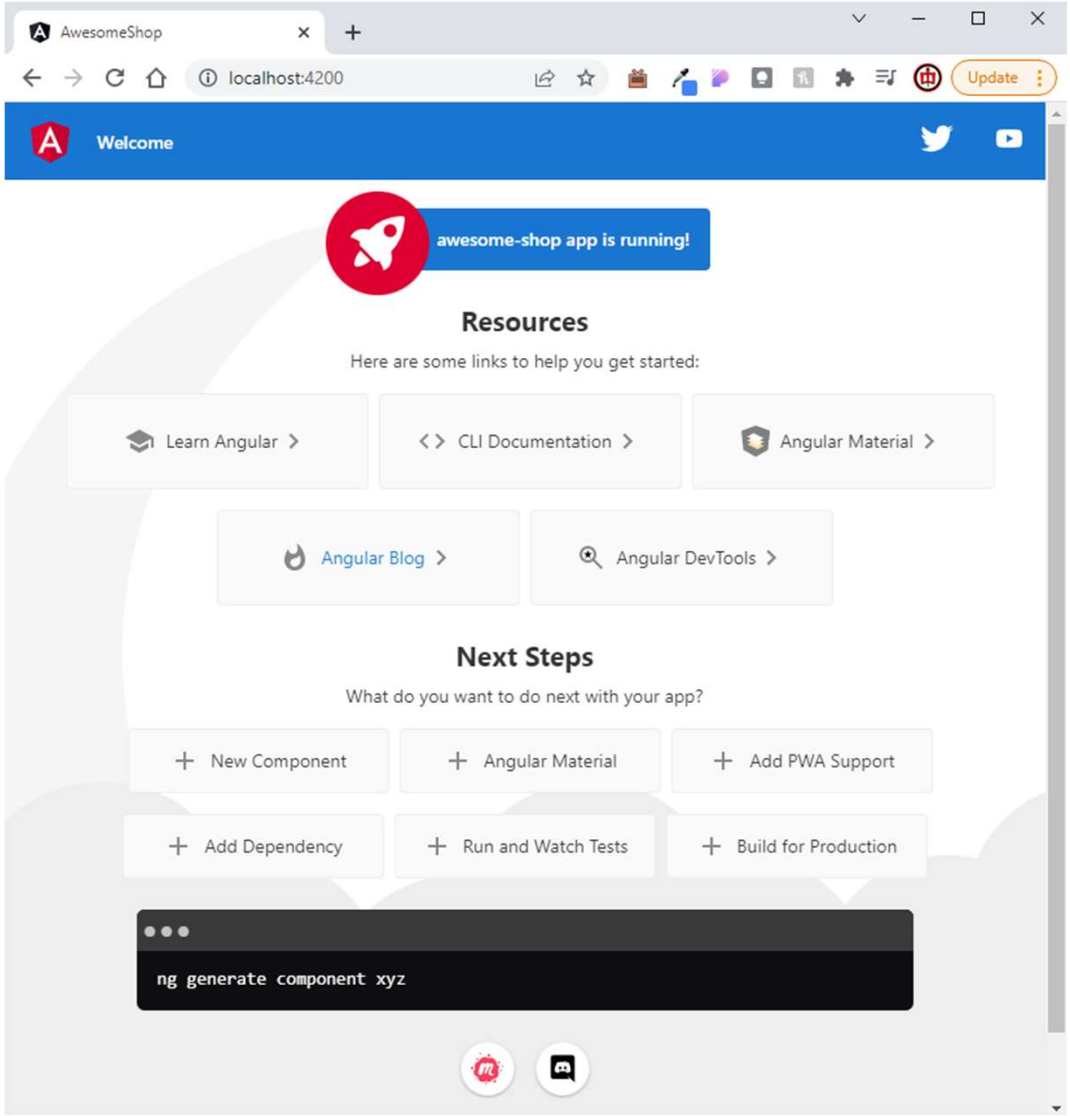
Cmder

```
C:\Dev\test\awesome-shop (master -> origin)
λ ng serve
√ Browser application bundle generation complete.

Initial Chunk Files   | Names         |  Raw Size
vendor.js             | vendor        |   1.97 MB |
polyfills.js          | polyfills     | 299.96 kB |
styles.css, styles.js | styles        | 173.23 kB |
main.js               | main          |  50.13 kB |
runtime.js            | runtime       |   6.52 kB |

                      | Initial Total |   2.49 MB

Build at: 2022-02-19T13:03:05.079Z - Hash: f6f5cd16073f553c - Time: 9835ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://l
ocalhost:4200/ **


√ Compiled successfully.
```

node.exe

Our first Angular application

It works!



Our index

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>EverShop</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

This is where our Angular application will load.

This could be named anything, even <awesomeshop-app>

Our First Component

Angular library

Typescript Decorator

```
EXPLORADOR
EDITORES ABIERTOS
  app.component.ts src\app
EVER-SHOP
  e2e
  node_modules
  src
    app
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
```

app.component.ts

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'app';
10 }
11
```

index.html

```
<body>
  <app-root></app-root>
</body>
```

- Components are the basic building blocks of Angular applications.
- A component controls a portion of the screen.
- *@Component* is used to apply our component decorator to our class.
- *selector* is the HTML element where we want the component to load.
- *template* is the content we want to load inside our selector (could be HTML code)

Declaring the Root Angular Module

Modules are how we organize our application in Angular. Every Angular application must have a “root module,” which we’ll need to launch it.

TS app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

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

List of all components within the module

Loads required dependencies to launch our app in the browser

Indicates our root component

Declaring the Root Angular Module

Dependencies to render the application

```
TS main.ts ×
1  import { enableProdMode } from '@angular/core';
2  import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3
4  import { AppModule } from './app/app.module';
5  import { environment } from './environments/environment';
6
7  if (environment.production) {
8    enableProdMode();
9  }
10
11 platformBrowserDynamic().bootstrapModule(AppModule);
```

Angular library that will render the website.

This will allow us to bootstrap, or launch, the app.

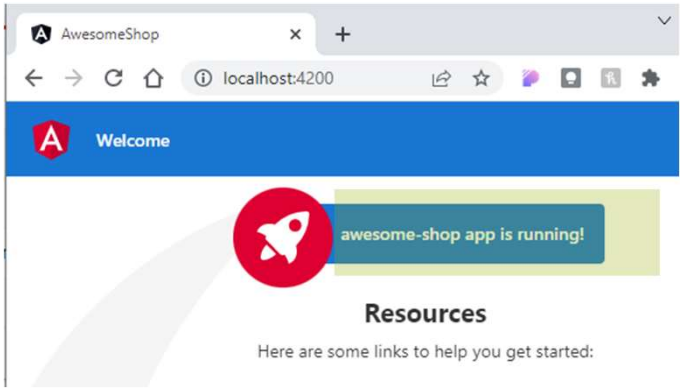
What if we have an object we want to print out onto the screen?

```
app.component.ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'My everShop';
}
```

Inside a TypeScript class, we don't use the *var* or *let* keywords to declare class properties.
Though we do in regular methods.

Curly braces allow us to load in component properties — this is called interpolation.

```
app.component.html
src > app > app.component.html > div.content > h2
340        </g>
341      </g>
342    </svg>
343
344    <span>{{ title }} app is running!</span>
```



Loading an Object

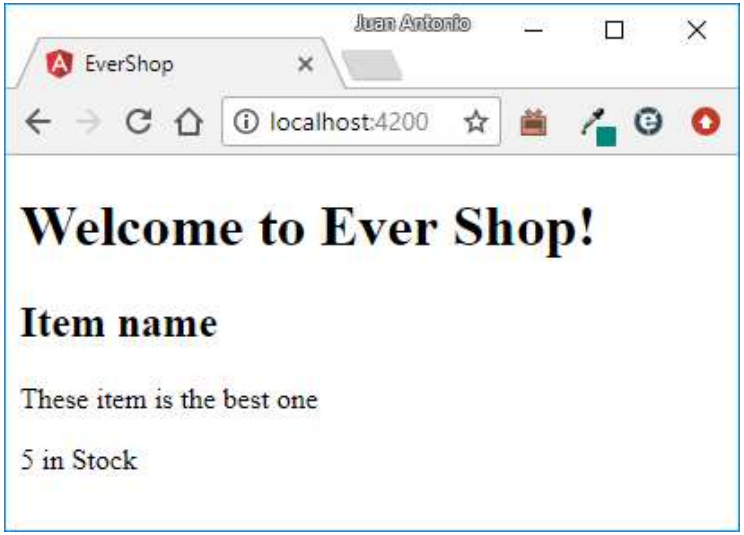
How do we send an object properties from our component class into our HTML?

app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Ever Shop';
  myItem = {
    'id': 1,
    'name': 'Item name',
    'description': 'These item is the best one',
    'stock': 5
  };
}
```

app.component.html

```
<div style="text-align:left">
  <h1>
    Welcome to {{title}}!
  </h1>

  <h2>{{myItem.name}}</h2>
  <p>{{myItem.description}}</p>
  <p>{{myItem.stock}} in Stock</p>
</div>
```



What'd We Learn?

- Angular is a framework for dynamic web applications.
- We are coding Angular using TypeScript, a language that transpiles into JavaScript.
- 'NgModules group Angular code into blocks of functionality.
- Components are the basic building blocks of any Angular application.
- We use a custom HTML tag (aka, selector) to show where we want our component to load inside our HTML.
- Decorators are what turn our plain TypeScript classes into Components.

02

Structural Directives

A directive (within Angular) is how we add dynamic behavior to HTML.

TS app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Ever Shop';
  myItem = {
    'id': 1,
    'name': 'Item name',
    'description': 'These item is the best one',
    'stock': 5
  };
}
```

What if we had more than one item?

TS app.component.ts

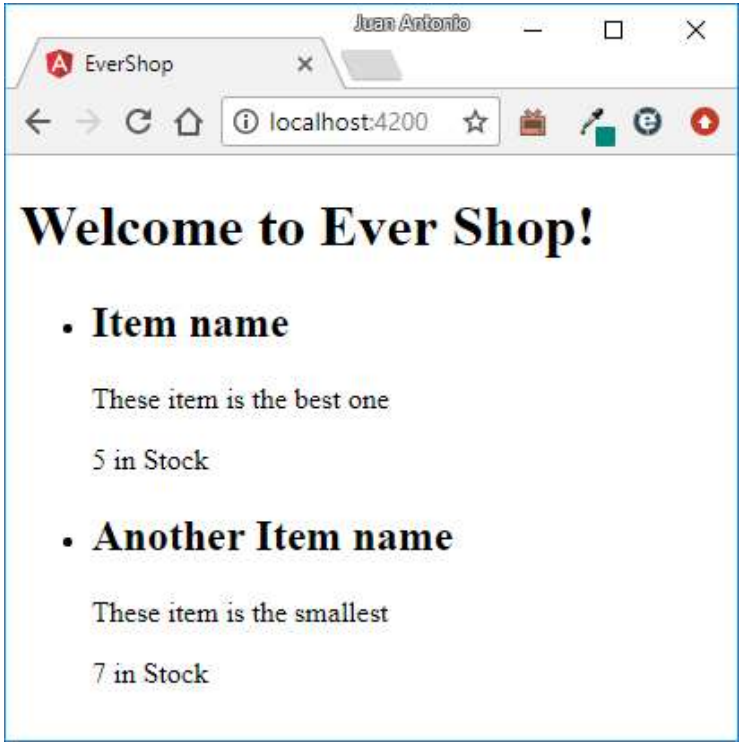
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Ever Shop';
  myItems = [{
    'id': 1,
    'name': 'Item name',
    'description': 'These item is the best one',
    'stock': 5
  },
  {
    'id': 2,
    'name': 'Another Item name',
    'description': 'These item is the smallest',
    'stock': 7
  }];
}
```

How do we loop through each of these?

app.component.html

```
<div style="text-align:left">
  <h1>
    Welcome to {{title}}!
  </h1>

  <ul>
    <li *ngFor="let item of myItems">
      <h2>{{item.name}}</h2>
      <p>{{item.description}}</p>
      <p>{{item.stock}} in Stock</p>
    </li>
  </ul>
</div>
```

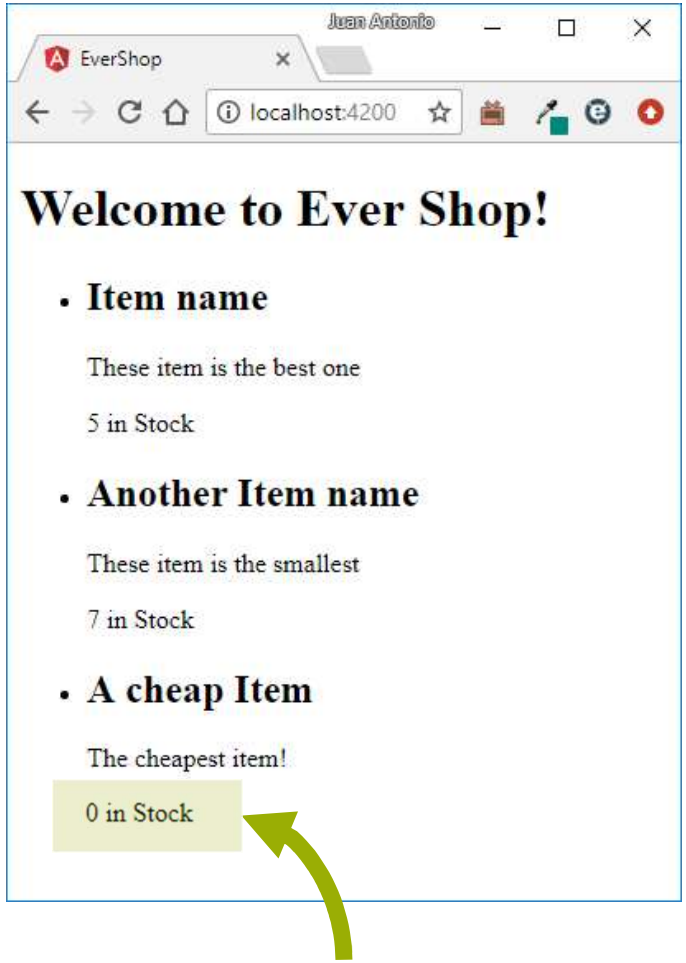
- **ngFor* is a structural directive.
- *item* is a local variable.
- *myItems* is the array to loop through
- The loop is run twice: once for each myItem



When there are none in stock, how can we display “Out of Stock”?

app.component.html

```
<div style="text-align:left">
  <h1>
    Welcome to {{title}}!
  </h1>

  <ul>
    <li *ngFor="let item of myItems">
      <h2>{{item.name}}</h2>
      <p>{{item.description}}</p>
      <p>{{item.stock}} in Stock</p>
    </li>
  </ul>
</div>
```

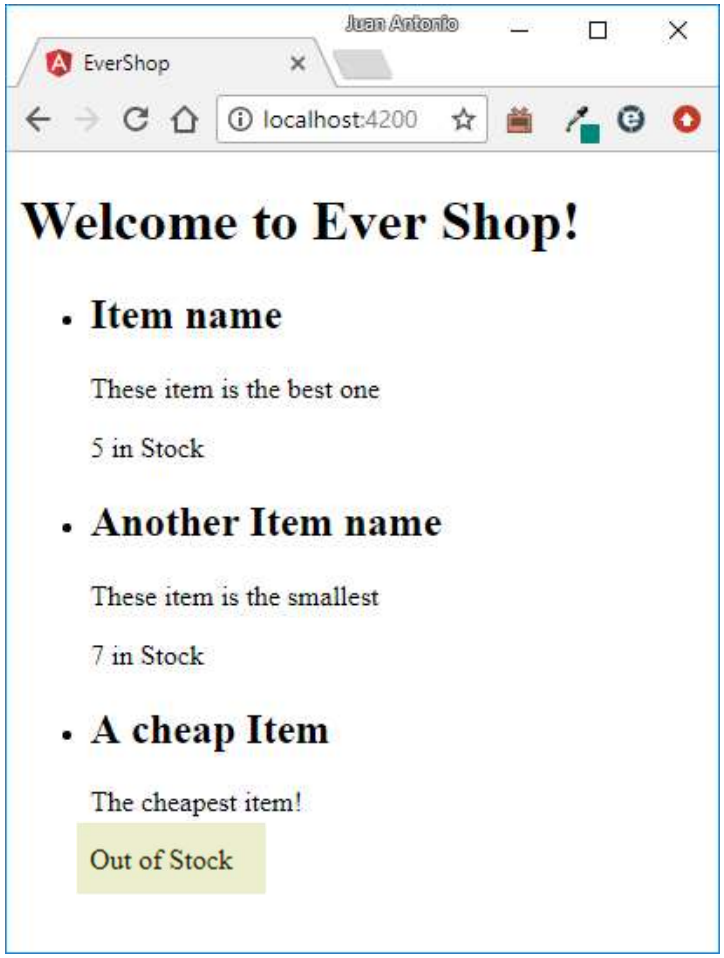


Should read “Out of Stock”

**ngIf* is another structural directive. It allows us to evaluate conditionals.

app.component.html

```
<div style="text-align:left">
  <h1>
    Welcome to {{title}}!
  </h1>

  <ul>
    <li *ngFor="let item of myItems">
      <h2>{{item.name}}</h2>
      <p>{{item.description}}</p>
      <p *ngIf="item.stock > 0">{{item.stock}} in Stock</p>
      <p *ngIf="item.stock === 0">Out of Stock</p>
    </li>
  </ul>
</div>
```



What'd We Learn?

- A directive (within Angular) is how we add dynamic behavior to HTML.
- A component directive has a template.
- A structural directive alters layout by adding, removing, or replacing HTML elements.
  - **ngFor* Loops through an array.
  - **ngIf* Shows content conditionally.
  - Angular documentation: https://angular.io/guide/structural-directives

03

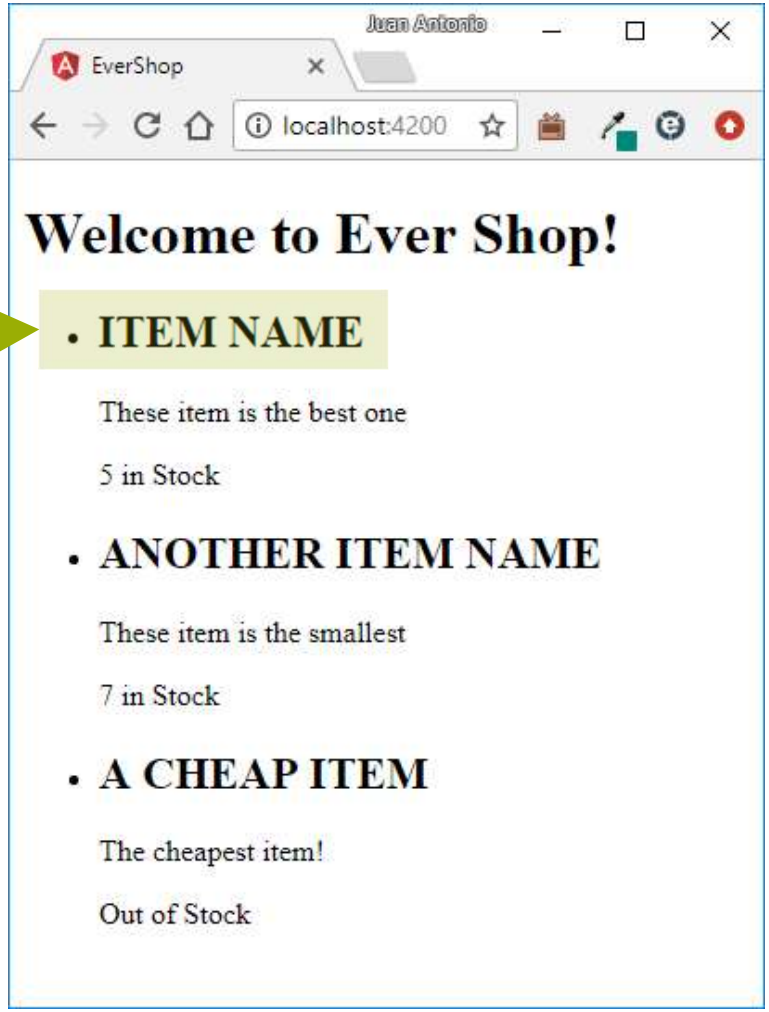
Pipes & Methods

A pipe takes in data as input and transforms it to a desired output.

How can we write out item name in capital letters?

app.component.html

```
<div style="text-align:left">
  <h1>
    Welcome to {{title}}!
  </h1>

  <ul>
    <li *ngFor="let item of myItems">
      <h2>{{item.name | uppercase}}</h2>
      <p>{{item.description}}</p>
      <p *ngIf="item.stock > 0">{{item.stock}} in Stock</p>
      <p *ngIf="item.stock === 0">Out of Stock</p>
    </li>
  </ul>
</div>
```



Adding a Price to Our Data

```
TS app.component.ts ×
8   export class AppComponent {
9     title = 'Ever Shop';
10    myItems = [{
11      'id': 1,
12      'name': 'Item name',
13      'description': 'These item is the best one',
14      'stock': 5,
15      'price': 14.99
16    },
17    {
18      'id': 2,
19      'name': 'Another Item name',
20      'description': 'These item is the smallest',
21      'stock': 7,
22      'price': 5
23    },
```

How do we format the price attribute properly?

Looking for the right pipe to use

Angular documentation: https://angular.io/guide/pipes

Es seguro | https://angular.io/api/common/CurrencyPipe

ANGULAR FEATURES DOCS RESOURCES EVENTS BLOG

GETTING STARTED
TUTORIAL
FUNDAMENTALS
Architecture
Template & Data Binding
Displaying Data
Template Syntax
Lifecycle Hooks
Component Interaction
Component Styles
Dynamic Components
Attribute Directives
Structural Directives
Pipes
Animations
Forms
Bootstrapping

CurrencyPipe PIPE

npm Package	@angular/common
Module	`import { CurrencyPipe } from '@angular/common';`
Source	common/src/pipes/number_pipe.ts
NgModule	`CommonModule`

Formats a number as currency using locale rules.

How To Use

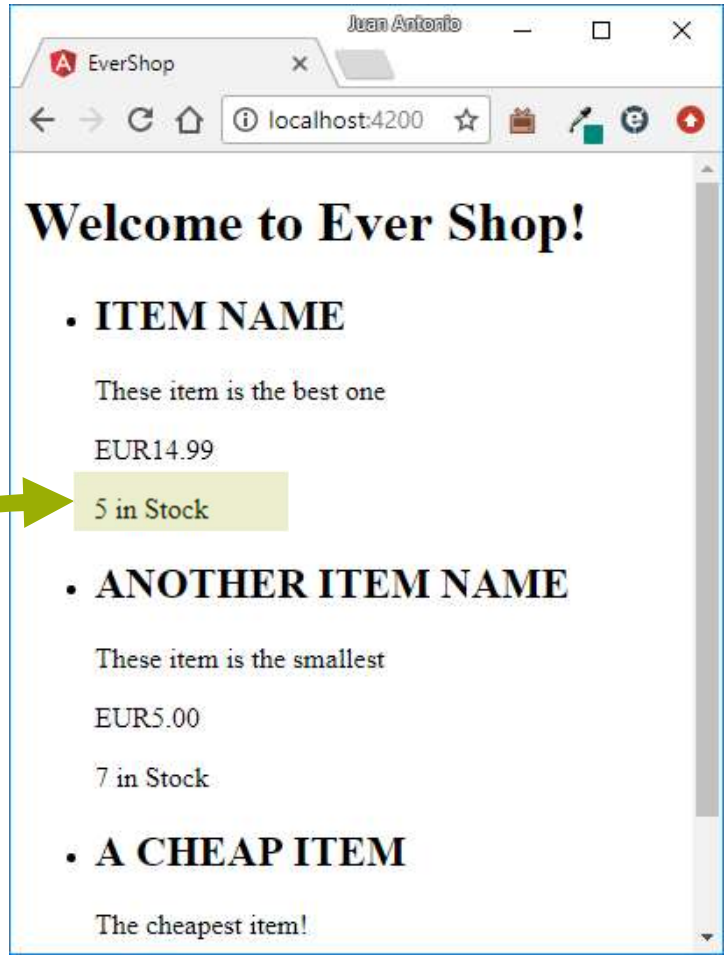
```
number_expression | currency[:currencyCode[:symbolDisplay[:digitInfo]]]
```

Using the Currency Pipe With One Parameter

By default, currency pipe uses ISO 4217 currency code, such as USD for dollar or EUR for the euro.

app.component.html

```
<div style="text-align:left">
  <h1>
    Welcome to {{title}}!
  </h1>

  <ul>
    <li *ngFor="let item of myItems">
      <h2>{{item.name | uppercase}}</h2>
      <p>{{item.description}}</p>
      <p>{{item.price | currency:'EUR'}}</p>
      <p *ngIf="item.stock > 0">{{item.stock}} in Stock</p>
      <p *ngIf="item.stock === 0">Out of Stock</p>
    </li>
  </ul>
</div>
```



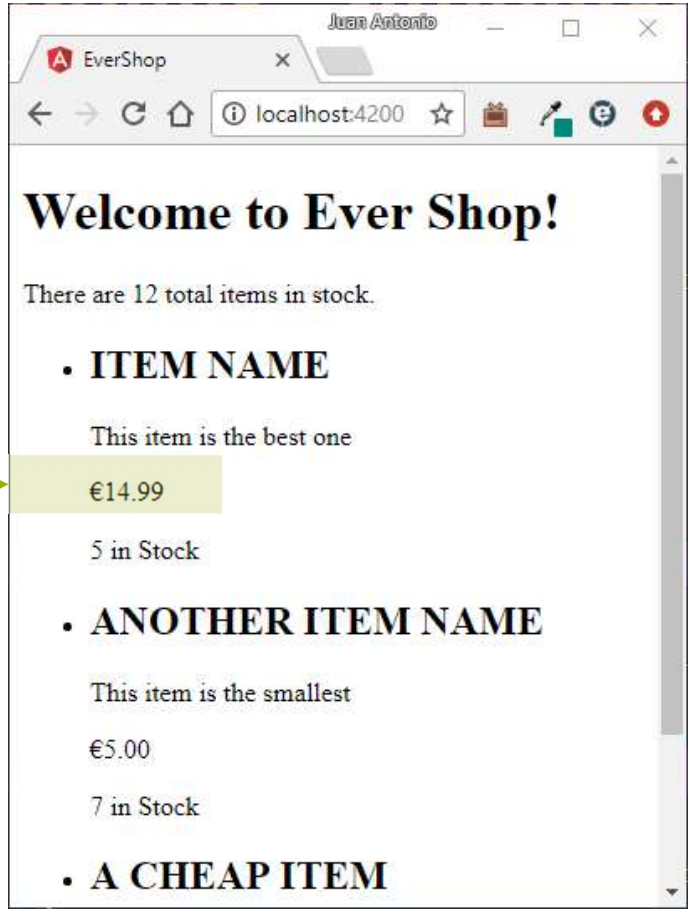
But we want the EUR symbol — how do we do that?

Using the Currency Pipe With Two Parameters

The second parameter is a boolean indicating if we should use the currency symbol.

app.component.html

```
<div style="text-align:left">
  <h1>
    Welcome to {{title}}!
  </h1>

  <ul>
    <li *ngFor="let item of myItems">
      <h2>{{item.name | uppercase}}</h2>
      <p>{{item.description}}</p>
      <p>{{item.price | currency:'EUR':'symbol'}}
      <p *ngIf="item.stock > 0">{{item.stock}} in Stock</p>
      <p *ngIf="item.stock === 0">Out of Stock</p>
    </li>
  </ul>
</div>
```



What we have to do if we want Spanish currency? Something like 14,2 €

Using the Currency Pipe With Two Parameters

Introducing Locale Concept

app.module.ts M

src > app > app.module.ts > ...

```
1  import { registerLocaleData } from '@angular/common';  64.7K (gzipped: 17.5K)
2  import es from '@angular/common/locales/es';  1.5K (gzipped: 885)
3  import { LOCALE_ID, NgModule } from '@angular/core';  218.5K (gzipped: 70.2K)
4  import { BrowserModule } from '@angular/platform-browser';  27K (gzipped: 7.4K)
5
6  import { AppRoutingModule } from './app-routing.module';
7  import { AppComponent } from './app.component';
8
9  @NgModule({
10   declarations: [
11     AppComponent
12   ],
13   imports: [
14     BrowserModule,
15     AppRoutingModule
16   ],
17   providers: [{
18     provide: LOCALE_ID,
19     useValue: 'es-ES'
20   }],
21   bootstrap: [AppComponent]
22 })
23 export class AppModule { }
24 registerLocaleData(es);
```

```
<h2>{{myItem.price | currency:'EUR':'symbol':'1.1-1'}}</h2>
```

Additional Pipes to Use

`lowercase`	Well, lowercase...
`date`	Formats dates how youlike them.
`number`	Formats numbers.
`decimal`	Formats decimals.
`replace`	Creates a new string, replacing specified characters.
`slice`	Creates a new list or string containing a subset of the elements.
`json`	Transforms any input to a JSON-formatted string.
`custom pipe`	You can write your own custom pipes. https://angular.io/guide/pipes#custom-pipes

Listing the Number of Items in Stock

How could we display the total number of items in stock?
We'll add new code to our HTML template and print the result of a method we're about to define.

app.component.html

```
<div style="text-align:left">
  <h1>
    Welcome to {{title}}!
  </h1>

  <p>There are {{totalItems()}} total items in stock.</p>
  <ul>
    <li *ngFor="let item of myItems">
      <h2>{{item.name | uppercase}}</h2>
      <p>{{item.description}}</p>
      <p>{{item.price | currency:'EUR':true}}</p>
      <p *ngIf="item.stock > 0">{{item.stock}} in Stock</p>
      <p *ngIf="item.stock === 0">Out of Stock</p>
    </li>
  </ul>
</div>
```

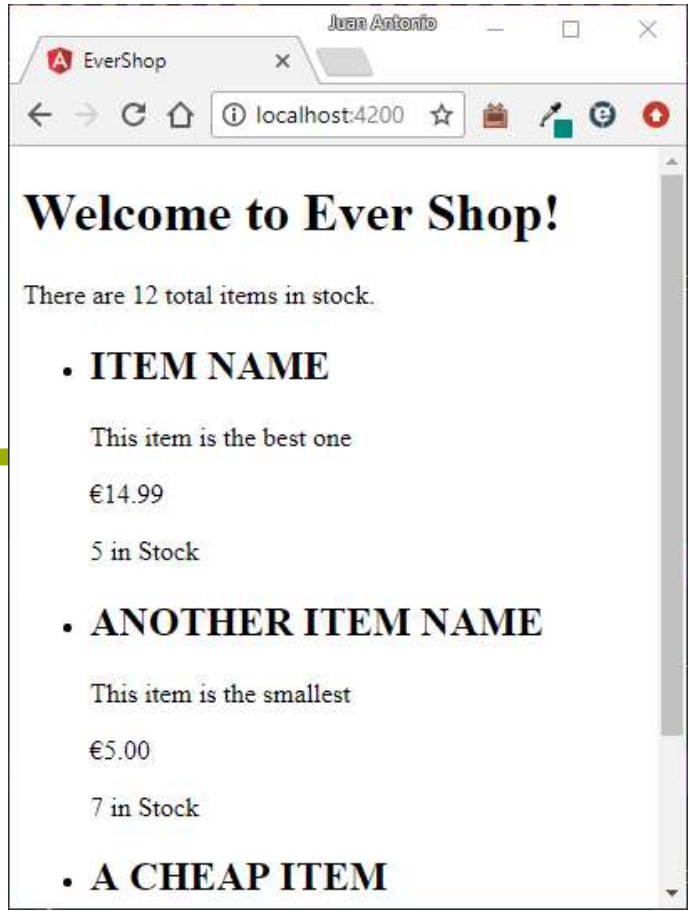
We define this method inside of our component class.

Implementing the Sum of my Items

Let's use an ES2015 *for of* loop, like in our template.

TS app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Ever Shop';
  myItems = [{
    'id': 1,
    'name': 'Item name',
    'description': 'These item is the best one',
    'stock': 5,
    'price': 14.99
  },
  ...
  totalItems() {
    let sum = 0;
    for (let myItem of this.myItems) {
      sum += myItem.stock;
    }
    return sum;
  }
```



Implementing the Sum of my Items

It looks really awful… so….. How could we simplify this code using fat arrow (arrow function) ?

```
totalItems() {
  let sum = 0;
  for (let myItem of this.myItems) {
    sum += myItem.stock;
  }
  return sum;
}
```

```
totalItems() {
  return this.myItems.reduce(function(prev, current) { return prev + current.stock; }, 0);
}
```

```
totalItems() {
  return this.myItems.reduce( (prev, current) => prev + current.stock, 0);
}
```

Is there another way to develop this?

ng g pipe Total

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'total'
})
export class TotalPipe implements PipeTransform {

  transform(value: any, args?: any): any {
    return null;
  }

}
```

```
@NgModule({
  declarations: [
    AppComponent,
    PricePipe,
    TotalPipe
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Custom Pipes

```
@Pipe({
  name: 'total'
})
export class TotalPipe implements PipeTransform {

  transform(value: any, args?: any): any {
    return value.reduce( (prev, current) => prev + current.stock, 0);
  }

}
```

```
<div style="text-align:center">
  <h1 *ngFor="let item of myItems">
    Welcome to {{ title }} {{item.amount | price: item.stock | currency:'EUR'}}!
  </h1>

  <h2>{{myItems | total}}</h2>
```

```
@Pipe({
  name: 'total'
})
export class TotalPipe implements PipeTransform {

  transform(value: any, args?: any): any {
    return value.reduce( (prev, current) => prev + current.stock, 0);
  }

}
```

What if I want the Pipe to receive the attribute to be added?

Custom Pipes

```
@Pipe({
  name: 'total'
})
export class TotalPipe implements PipeTransform {

  transform(value: any, arg: string): any {
    return value.reduce( (prev, current) => prev + current[arg], 0);
  }

}
```

```
<div style="text-align:center">
  <h1 *ngFor="let item of myItems">
    Welcome to {{ title }} {{item.amount | price: item.stock | currency:'EUR'}}!
  </h1>

  <h2>{{myItems | total:'stock'}}</h2>
```

Could you develop a Pipe to transform from EUR to USD?

And to any Currency?