



# DESARROLLADOR SOFTWARE – TALLER ANGULAR

**FECHA- [18-19]- MAYO 2022**

**Horario: 15:00 a 20:30 h**

Juan Antonio Herrerías Berbel

# Índice

\_\_01 Angular

\_\_02 Structural Directives

\_\_03 Pipes & Methods

\_\_04 Splitting Components

\_\_05 Mocks & Models

\_\_06 Property & Class Binding

\_\_07 Event Binding

\_\_08 Two-way Binding

\_\_09 Service

\_\_10 HTTP

\_\_11 Component Interaction

\_\_12 Basic Routing

\_\_13 Share Data Between Components

\_\_14 Reactive Forms

Herre - jaherrerias@gmail.com



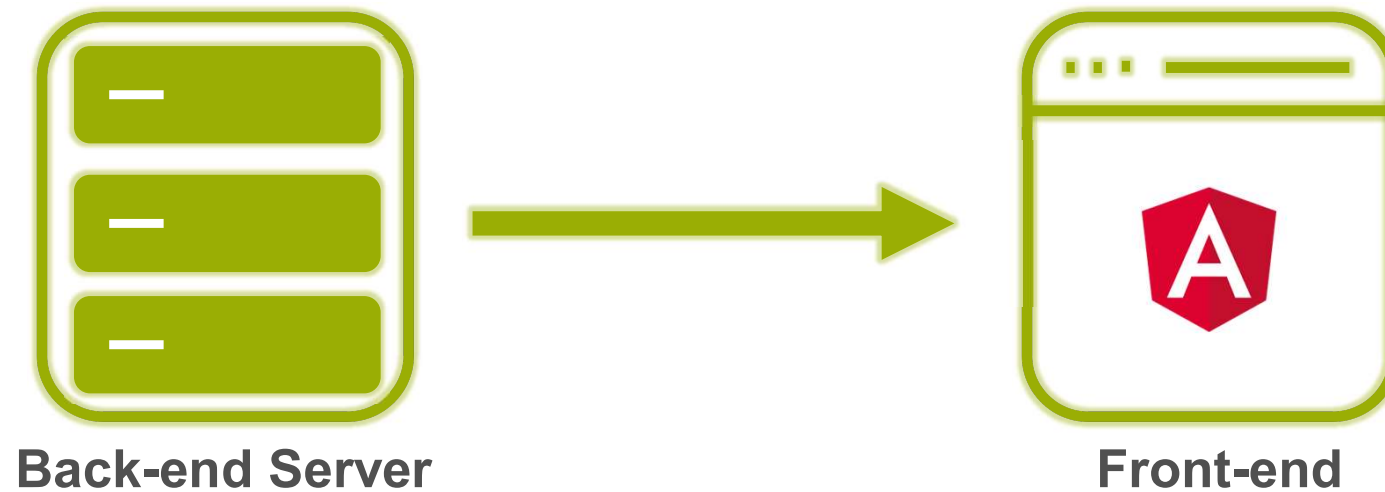


— **01**

# Angular

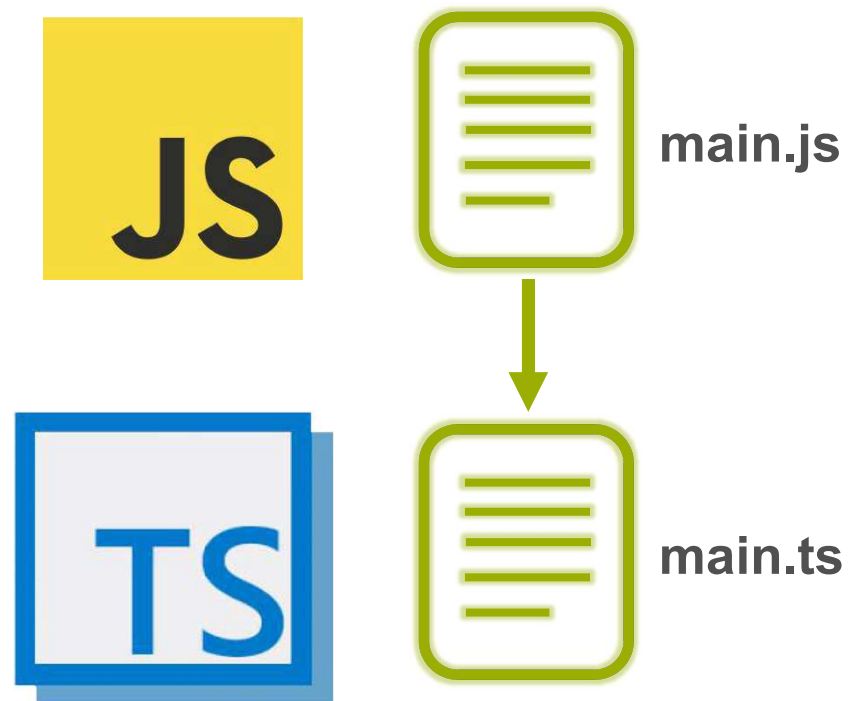


- Angular is a framework for dynamic web applications (SPA).
- Provides a way to organize your HTML, JavaScript, and CSS to keep your front-end code clean.
- Released in 2011.
- Mainly maintained by Google with the help of the open-source community.





- 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.





Command Line Interface



- 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 help us with:

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

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



## 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.
```

```
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

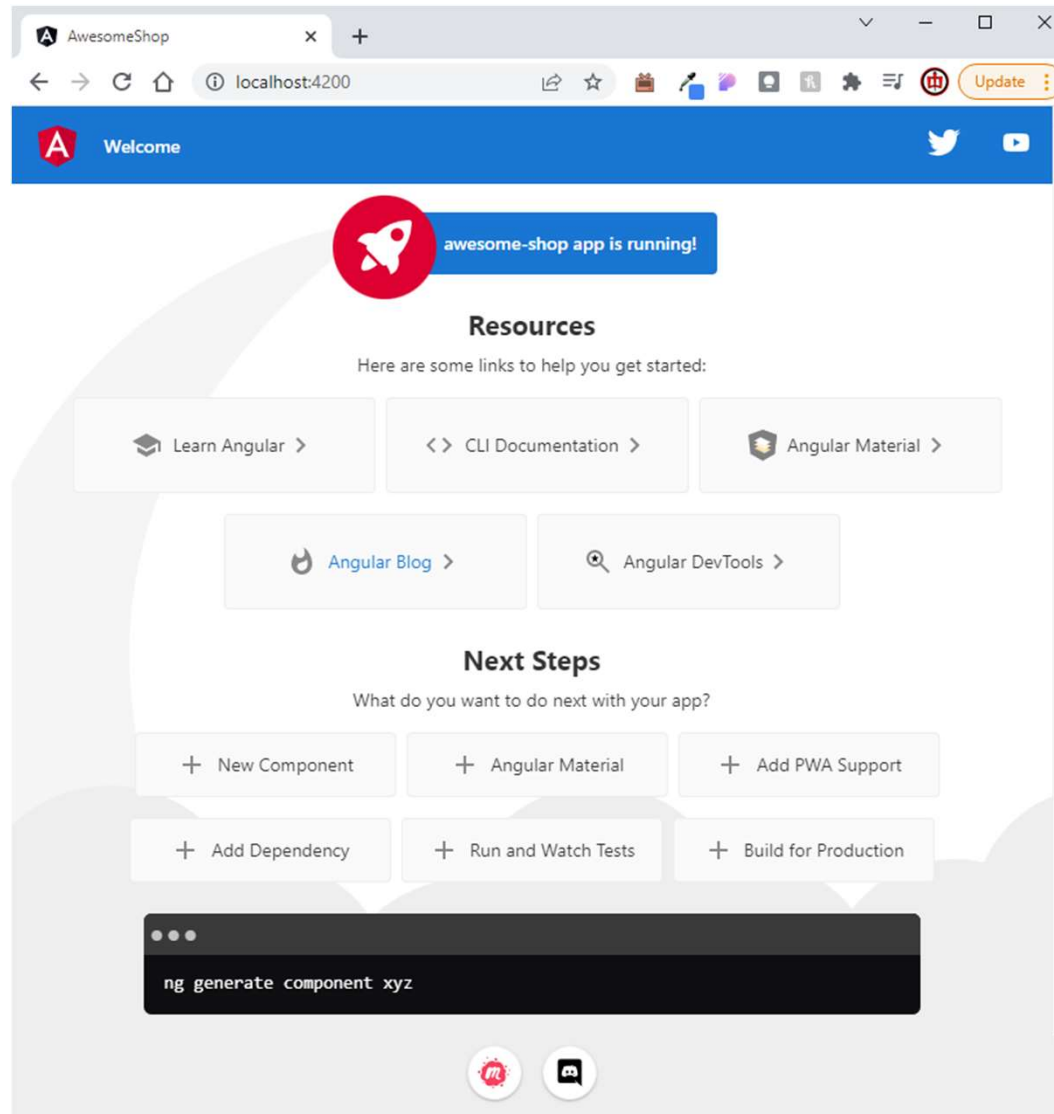
Build at: 2022-02-19T13:03:05.079Z - Hash: f6f5cd16073f553c - Time: 9835ms

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

✓ Compiled successfully.
```



It works!





This is where our Angular application will load.

```
<> index.html x
1  <!doctype html>
2  <html lang="en">
3  <head>
4    <meta charset="utf-8">
5    <title>EverShop</title>
6    <base href="/">
7
8    <meta name="viewport" content="width=device-width, initial-scale=1">
9    <link rel="icon" type="image/x-icon" href="favicon.ico">
10 </head>
11 <body>
12   <app-root></app-root>
13 </body>
14 </html>
```

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



Angular library

Typescript Decorator

index.html

```
EXPLORADOR
├── EDITORES ABIERTOS
│   └── TS 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
└── TS app.component.ts x

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
```

```
<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)



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 x
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppComponent } from './app.component';
5
6  @NgModule({
7    declarations: [
8      AppComponent
9    ],
10   imports: [
11     BrowserModule
12   ],
13   providers: [],
14   bootstrap: [AppComponent]
15 })
16 export class AppModule { }
```

List of all components within the module

Loads required dependencies to launch our app in the browser

Indicates our root component



Dependencies to render the application

```
TS main.ts x
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.





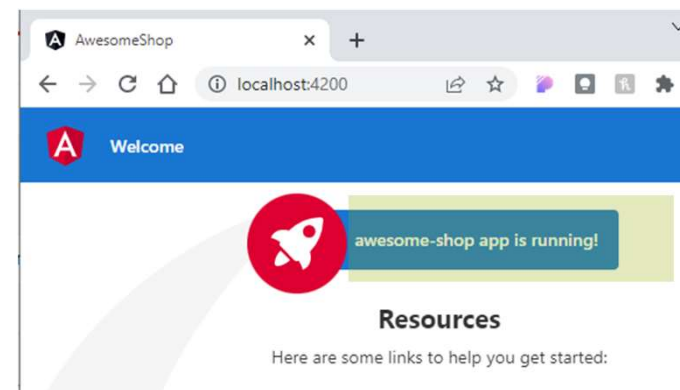
What if we have an object we want to print out onto the screen?

```
TS app.component.ts x
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 = 'My everShop';
10 }
```

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

```
5 app.component.html x
src > app > 5 app.component.html > div.content > h2
340      </g>
341    </g>
342  </svg>
343
344  <span>{{ title }} app is running!</span>
345
```

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

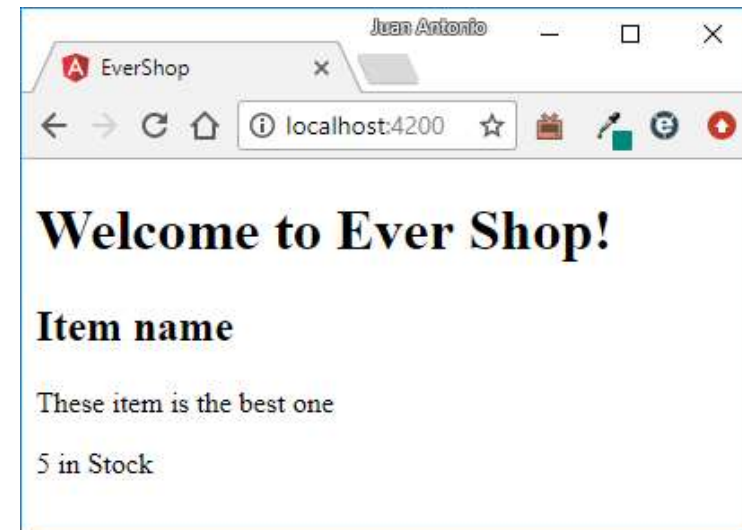




How do we send an object properties from our component class into our HTML?

```
TS app.component.ts x
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 = 'Ever Shop';
10
11   myItem = {
12     'id': 1,
13     'name': 'Item name',
14     'description': 'These item is the best one',
15     'stock': 5
16   };
17 }
```

```
app.component.html x
1 <div style="text-align:left">
2   <h1>
3     Welcome to {{title}}!
4   </h1>
5
6   <h2>{{myItem.name}}</h2>
7   <p>{{myItem.description}}</p>
8   <p>{{myItem.stock}} in Stock</p>
9 </div>
```





- 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 x
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 = 'Ever Shop';
10   myItem = {
11     'id': 1,
12     'name': 'Item name',
13     'description': 'These item is the best one',
14     'stock': 5
15   };
16 }
```

What if we had more than one item?

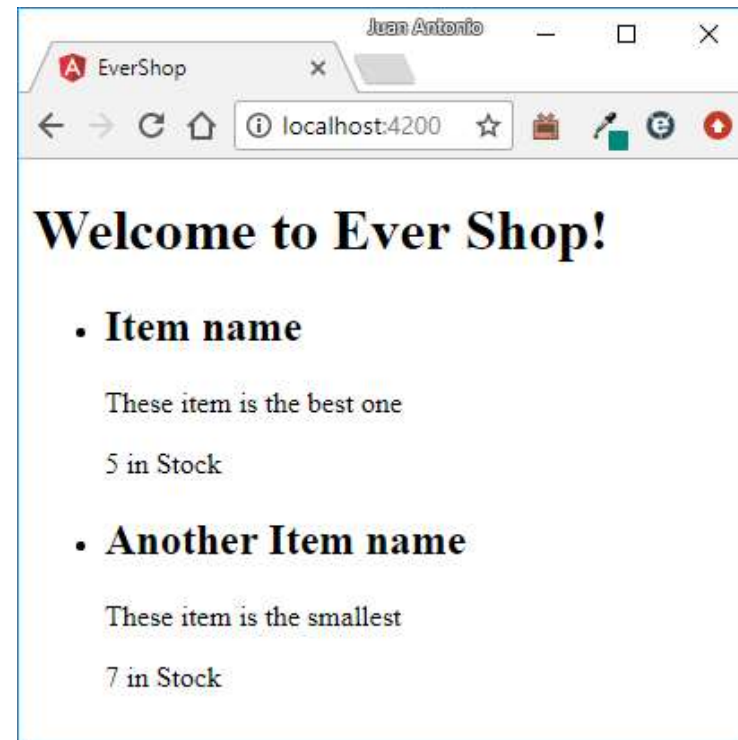


```
TS app.component.ts x
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'Ever Shop';
10   myItems = [{
11     'id': 1,
12     'name': 'Item name',
13     'description': 'These item is the best one',
14     'stock': 5
15   },
16   {
17     'id': 2,
18     'name': 'Another Item name',
19     'description': 'These item is the smallest',
20     'stock': 7
21   }
22 ];
```

How do we loop through each of these?

```
app.component.html x
1 | <div style="text-align:left">
2 |   <h1>
3 |     Welcome to {{title}}!
4 |   </h1>
5 |
6 |   <ul>
7 |     <li *ngFor="let item of myItems">
8 |       <h2>{{item.name}}</h2>
9 |       <p>{{item.description}}</p>
10 |      <p>{{item.stock}} in Stock</p>
11 |     </li>
12 |   </ul>
13 | </div>
```

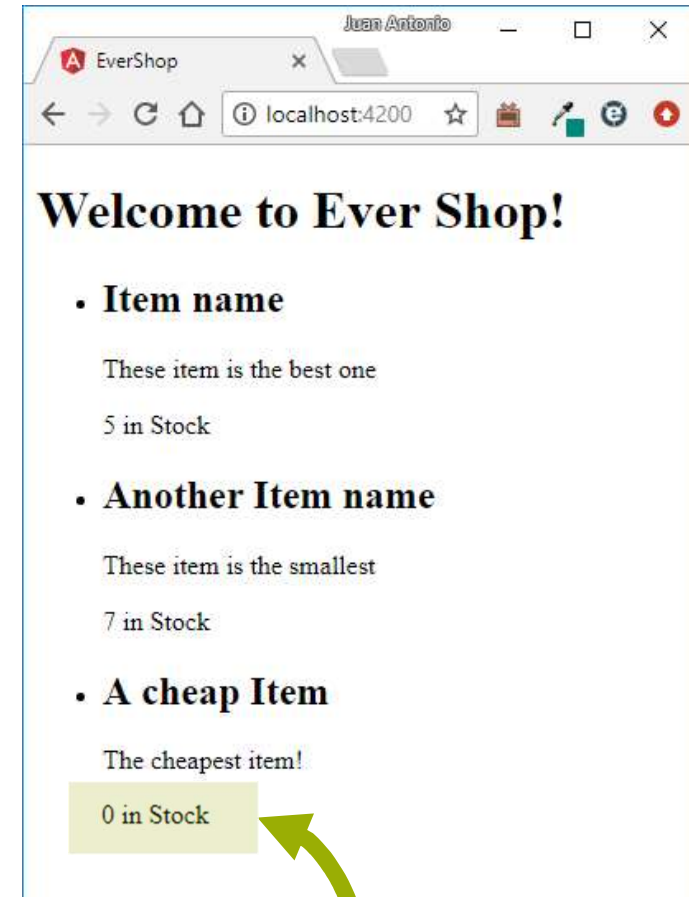
- *\*ngFor* is a structural directive.
- *item* is a local variable.
- *myItems* is the array to loop through
- The loop is run twice: once for each myItem





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

```
app.component.html x
1 | <div style="text-align:left">
2 |   <h1>
3 |     Welcome to {{title}}!
4 |   </h1>
5 |
6 |   <ul>
7 |     <li *ngFor="let item of myItems">
8 |       <h2>{{item.name}}</h2>
9 |       <p>{{item.description}}</p>
10 |      <p>{{item.stock}} in Stock</p>
11 |     </li>
12 |   </ul>
13 | </div>
```

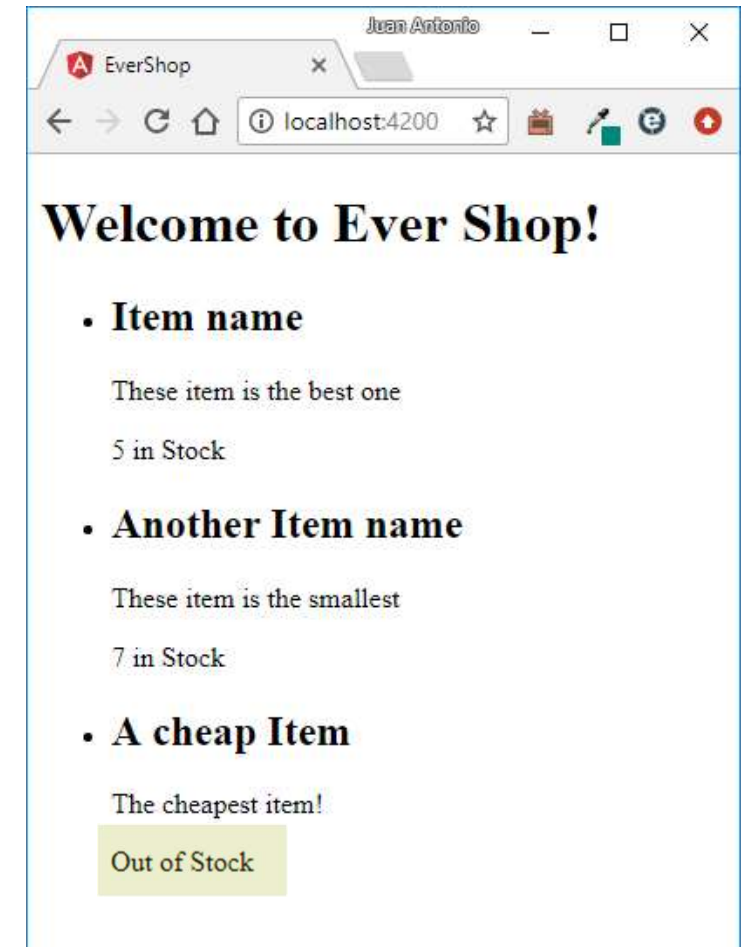


Should read “Out of Stock”



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

```
<> app.component.html x
1 | <div style="text-align:left">
2 |   <h1>
3 |     Welcome to {{title}}!
4 |   </h1>
5 |
6 |   <ul>
7 |     <li *ngFor="let item of myItems">
8 |       <h2>{{item.name}}</h2>
9 |       <p>{{item.description}}</p>
10 |      <p *ngIf="item.stock > 0">{{item.stock}} in Stock</p>
11 |      <p *ngIf="item.stock === 0">Out of Stock</p>
12 |     </li>
13 |   </ul>
14 | </div>
```







- 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

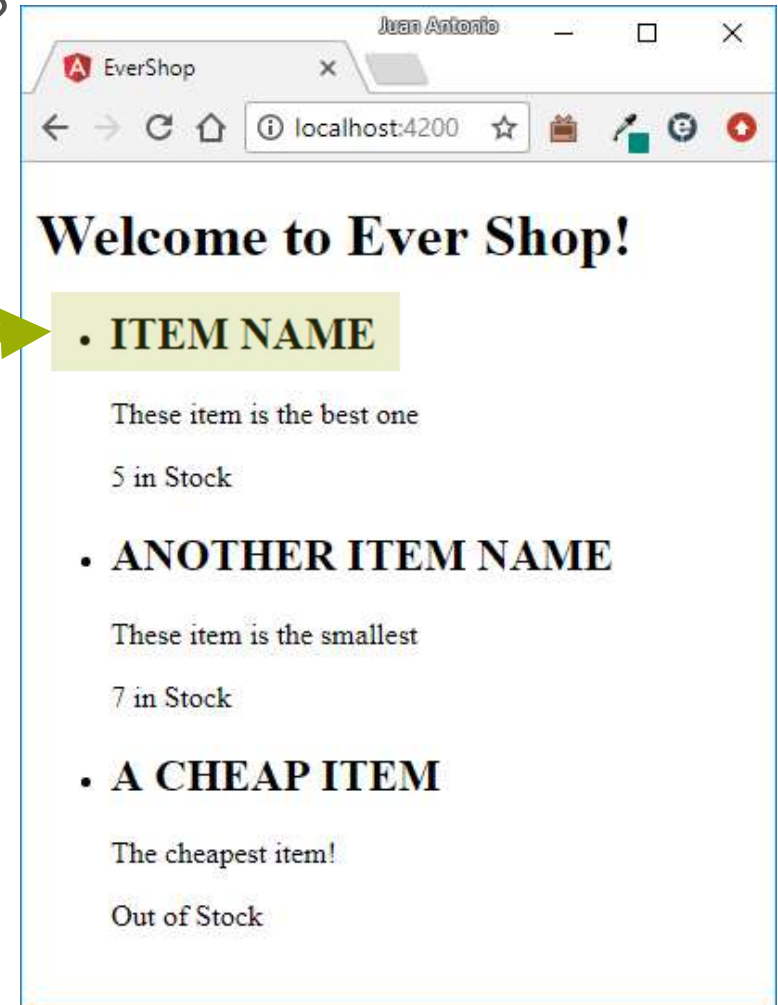
# Pipes & Methods



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 x
1 | <div style="text-align:left">
2 |   <h1>
3 |     Welcome to {{title}}!
4 |   </h1>
5 |
6 |   <ul>
7 |     <li *ngFor="let item of myItems">
8 |       <h2>{{item.name | uppercase}}</h2>
9 |       <p>{{item.description}}</p>
10 |       <p *ngIf="item.stock > 0">{{item.stock}} in Stock</p>
11 |       <p *ngIf="item.stock === 0">Out of Stock</p>
12 |     </li>
13 |   </ul>
14 | </div>
```





```
TS app.component.ts x
 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?



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

The screenshot shows the Angular documentation page for the `CurrencyPipe`. The page has a blue header with the Angular logo and navigation links: FEATURES, DOCS, RESOURCES, EVENTS, and BLOG. A left sidebar contains a table of contents with categories like GETTING STARTED, TUTORIAL, FUNDAMENTALS, and Forms. The main content area is titled "CurrencyPipe" with a "PIPE" tag. It contains a table with the following information:

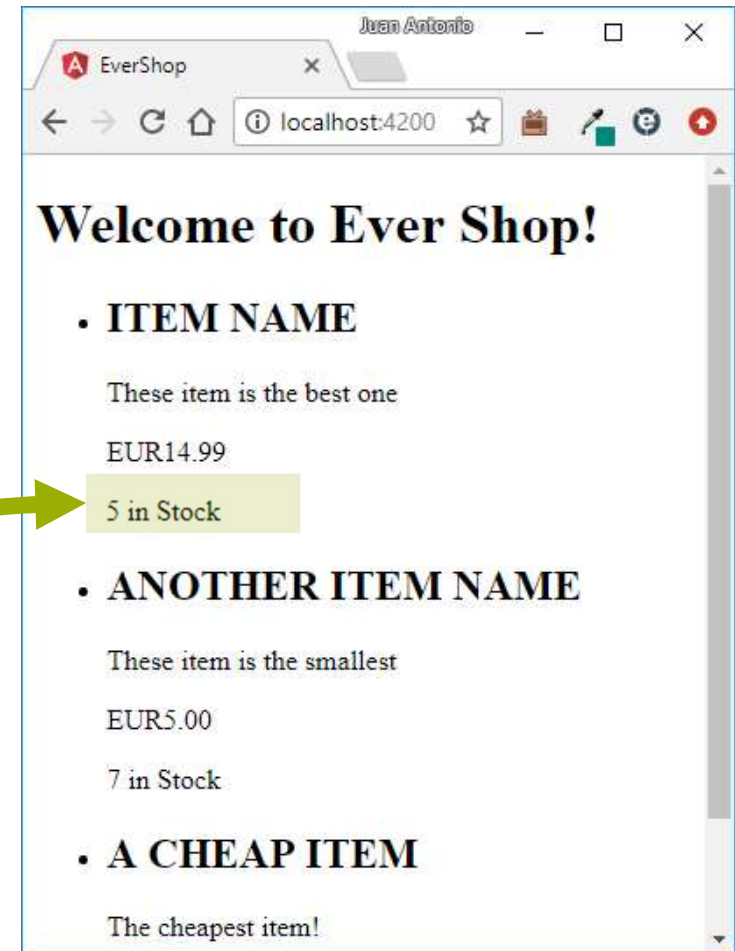
npm Package	<a href="#">@angular/common</a>
Module	<code>import { CurrencyPipe } from '@angular/common';</code>
Source	<a href="#">common/src/pipes/number_pipe.ts</a>
NgModule	<code>CommonModule</code>

Below the table, there is a description: "Formats a number as currency using locale rules." and a "How To Use" section with the following code snippet:

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

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

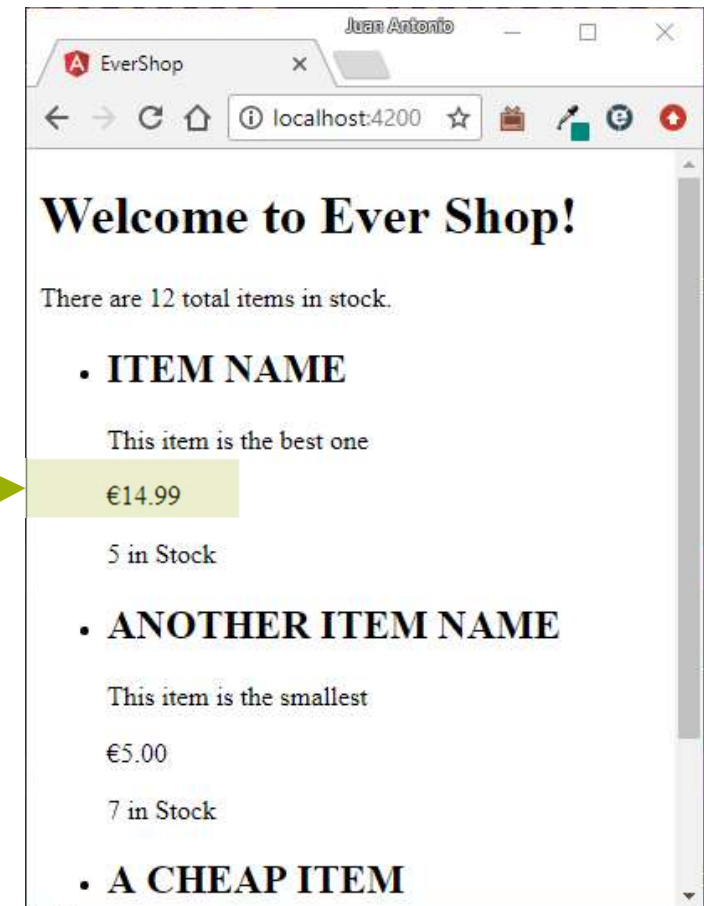
```
app.component.html x
1 | <div style="text-align:left">
2 |   <h1>
3 |     Welcome to {{title}}!
4 |   </h1>
5 |
6 |   <ul>
7 |     <li *ngFor="let item of myItems">
8 |       <h2>{{item.name | uppercase}}</h2>
9 |       <p>{{item.description}}</p>
10 |      <p>{{item.price | currency:'EUR'}}</p>
11 |      <p *ngIf="item.stock > 0">{{item.stock}} in Stock</p>
12 |      <p *ngIf="item.stock === 0">Out of Stock</p>
13 |    </li>
14 |  </ul>
15 | </div>
```



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

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

```
app.component.html x
1 <div style="text-align:left">
2   <h1>
3     Welcome to {{title}}!
4   </h1>
5
6   <ul>
7     <li *ngFor="let item of myItems">
8       <h2>{{item.name | uppercase}}</h2>
9       <p>{{item.description}}</p>
10      <p>{{item.price | currency:'EUR':'symbol'}}</p>
11      <p *ngIf="item.stock > 0">{{item.stock}} in Stock</p>
12      <p *ngIf="item.stock === 0">Out of Stock</p>
13    </li>
14  </ul>
15 </div>
```



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





## Introducing Locale Concept

```
app.module.ts M x
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>
```





`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.

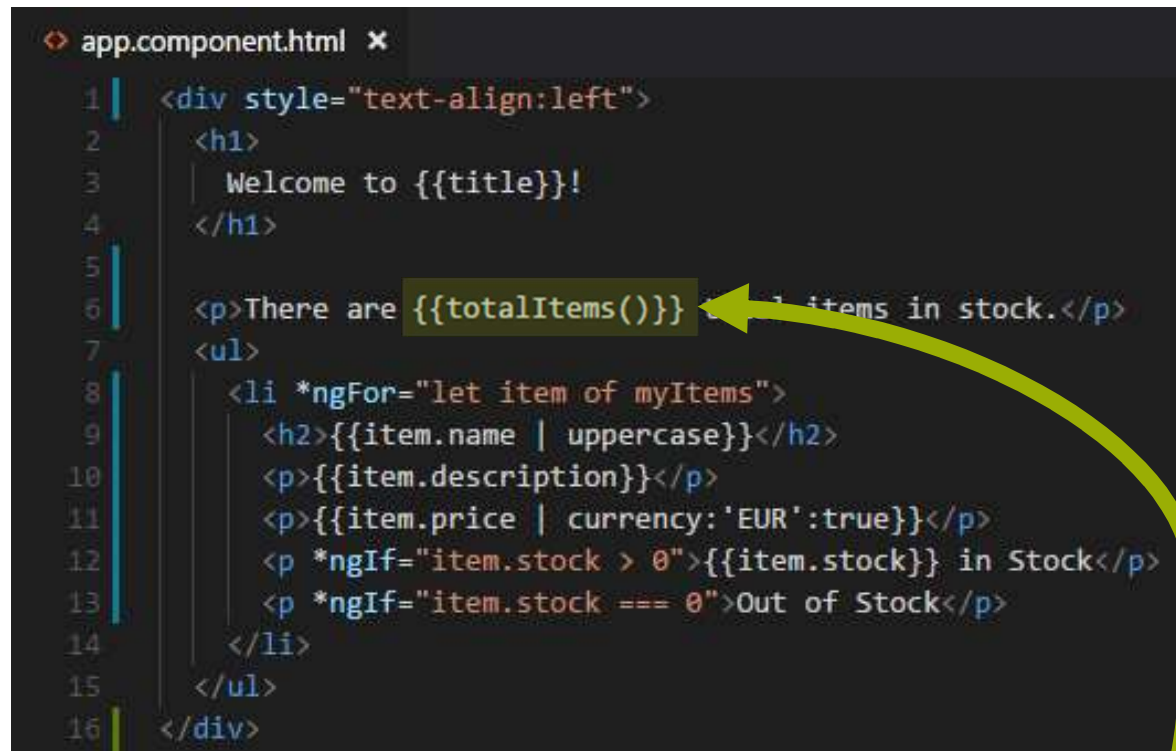
`custom pipe`

You can write your own custom pipes.

<https://angular.io/guide/pipes#custom-pipes>

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.

A screenshot of a code editor showing an Angular HTML template file named 'app.component.html'. The code is as follows:

```
1 | <div style="text-align:left">
2 |   <h1>
3 |     Welcome to {{title}}!
4 |   </h1>
5 |
6 |   <p>There are {{totalItems()}} items in stock.</p>
7 |   <ul>
8 |     <li *ngFor="let item of myItems">
9 |       <h2>{{item.name | uppercase}}</h2>
10 |      <p>{{item.description}}</p>
11 |      <p>{{item.price | currency:'EUR':true}}</p>
12 |      <p *ngIf="item.stock > 0">{{item.stock}} in Stock</p>
13 |      <p *ngIf="item.stock === 0">Out of Stock</p>
14 |    </li>
15 |  </ul>
16 | </div>
```

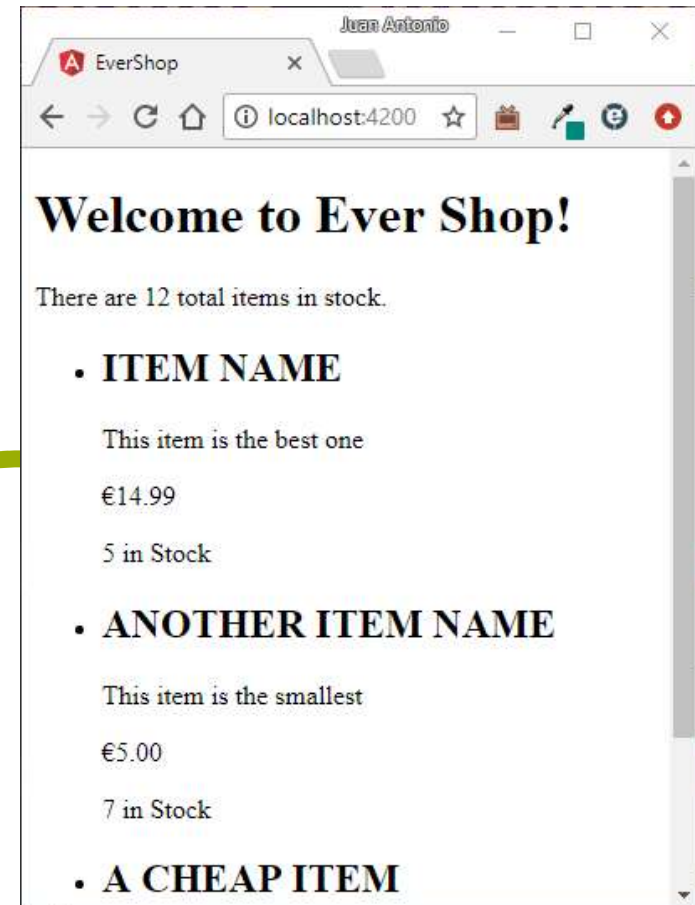
A yellow arrow points from the text 'We define this method inside of our component class.' to the `{{totalItems()}}` expression in the template.

We define this method inside of our component class.



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


```
TS app.component.ts x
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 = '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   ...
32   totalItems() {
33     let sum = 0;
34     for (let myItem of this.myItems) {
35       sum += myItem.stock;
36     }
37     return sum;
38   }
}
```






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 { }
```



```
@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?





```
@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?