



AN INTRODUCTION TO ANGULAR

May 2019

OVERVIEW, OBJECTIVES

- Learning basics:
 - How Angular works
 - Modules, components, directives, pipes
 - Services, Observables and reactive extensions
 - Routing
- Create Angular applications, reusable Component Libraries

Welcome to ngFirstApp!

Hello, world!

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

 [Slides >](#)

Articles - grid: col-xs-4

Grid: col-xs-4

Article: article title 1	Article: article title 2	Article: article title 3
 <small>tech</small>	 <small>tech Liked!</small>	 <small>tech</small>
<p>Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.</p> <p>Like it! Dislike</p>	<p>Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.</p> <p>Like it! Dislike</p>	<p>Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.</p> <p>Like it! Dislike</p>



ANGULAR COURSE REPOSITORY

- **REPOSITORY:**

<https://github.com/fdelfelice-sopra/introduction-to-angular>

- **COMMITS LIST:**

<https://github.com/fdelfelice-sopra/introduction-to-angular/commits/firstApp>

Welcome to ngFirstApp!

Hello, world!

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

[Slides >](#)

Articles - grid: col-xs-4

Grid: col-xs-4

Article: article title 1	Article: article title 2	Article: article title 3
 <small>tech</small>	 <small>tech Liked!</small>	 <small>tech</small>
<p>Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.</p> <p>Like it! Dislike</p>	<p>Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.</p> <p>Like it! Dislike</p>	<p>Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.</p> <p>Like it! Dislike</p>



ANGULARJS IS NOT ANGULAR:

Angular 7 vs Angular 6 vs Angular 2 vs Angular 1



[<https://angular.io/>]

Angular 1

Complete Re-Write

“AngularJS”

2010 ANGULARJS

Not really related to Angular 2+

[<https://angularjs.org/>]



ANGULAR IS:



- A javascript framework
- Using HTML, CSS, JS and TYPESCRIPT
- Based on ModuleS/Components logic
- Can be used for:
 - Web Client side application
 - Native mobile application (with Cordova, Ionic and NativeScript)
 - Native desktop application (with Electron or UWP)
 - Server Side Rendering (with Angular Universal)
- It means:
 - Easy learning and use for FE people, but also BE
 - Cross Platform target
 - Speed and performance (hierarchical dependency injection)
 - Cleaner Code, Productivity, code re-use
 - Full development story (prototyping, presentation/logic/data splitting , development, unit & integration testing, documentation)



TYPESCRIPT



Open source language

Superset of JavaScript

Transpiles to plain JavaScript

Strongly typed

- TypeScript type definition files (*.d.ts)

Class-based object-orientation

[<https://www.typescriptlang.org/>]



TYPESCRIPT IDE FOR ANGULAR DEVELOPMENT

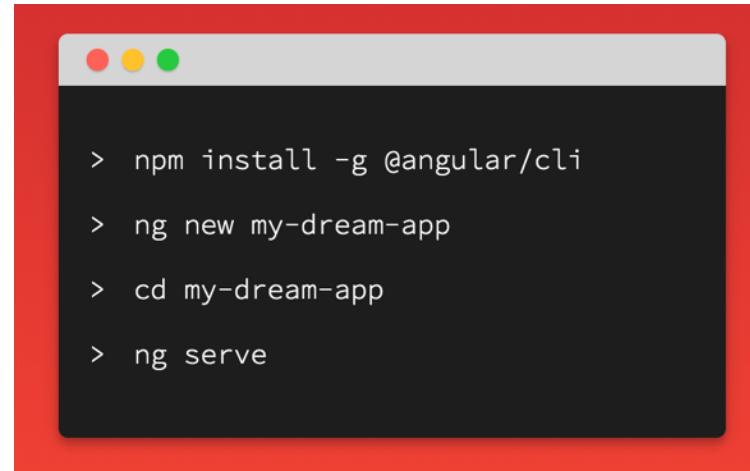
- IntelliJ IDEA
- WebStorm
- Visual Studio Code
- Atom
- Eclipse (Angular IDE by Webclipse)
- ...

[<https://angular.io/resources>]



ANGULAR CLI

- Angular has a Command Line Interface that easily allow to create, develop, test, serve, build your application



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

[<https://cli.angular.io/>]



ANGULAR CLI

ng help - Displays commands and flags

ng new - Creates new Angular application

ng serve - Launches a server

ng generate - Generates file from blueprint

ng test - Runs unit tests using Karma

ng e2e - Runs end to end tests using Protractor

ng build - Compiles into an output directory



class	ng g cl
component	ng g c
directive	ng g d
enum	ng g e
guard	ng g g
interface	ng g i
module	ng g m
pipe	ng g p
service	ng g s



ANGULAR CLI: NG NEW & NG SERVE

```
[root@ITEM-angular $] > ng new my-dream-app
```

```
? Would you like to add Angular routing? Yes
```

```
? Which stylesheet format would you like to use? CSS
```

```
CREATE my-dream-app/README.md (1027 bytes)
```

```
CREATE my-dream-app/angular.json (3822 bytes)
```

```
CREATE my-dream-app/package.json (1311 bytes)
```

```
CREATE my-dream-app/tsconfig.json (435 bytes)
```

```
CREATE my-dream-app/tslint.json (2837 bytes)
```

```
CREATE my-dream-app/.editorconfig (246 bytes)
```

```
CREATE my-dream-app/.gitignore (576 bytes)
```

```
CREATE my-dream-app/src/favicon.ico (5430 bytes)
```

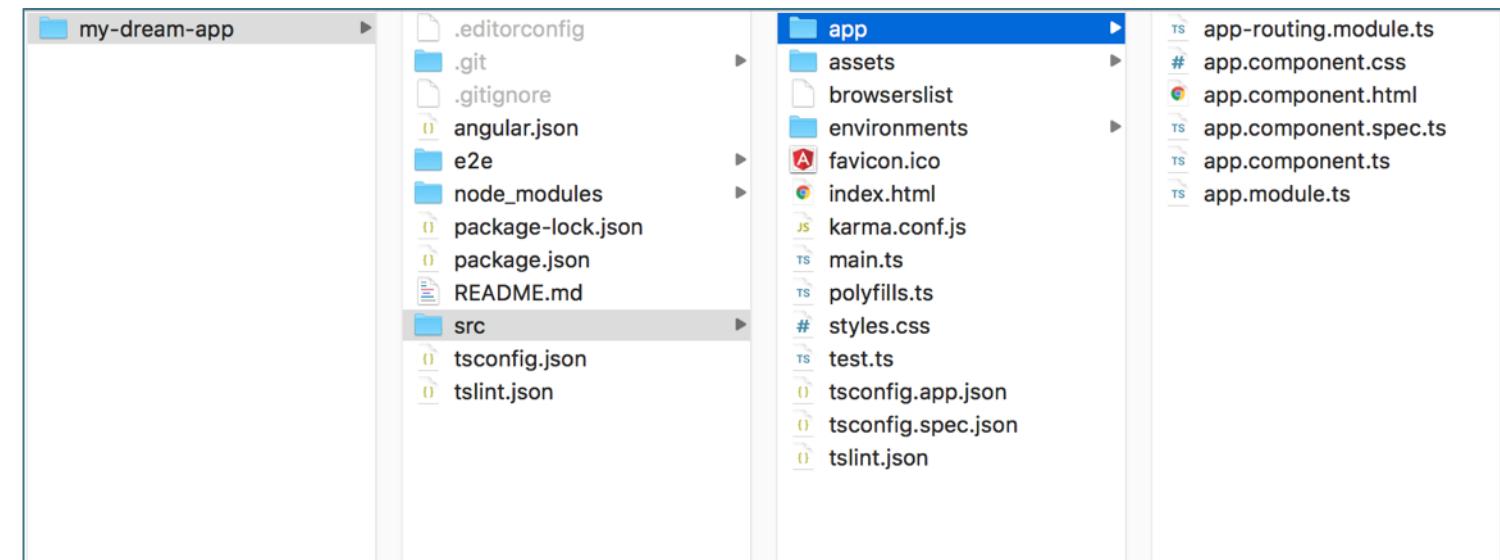
```
CREATE my-dream-app/src/index.html (297 bytes)
```

```
CREATE my-dream-app/src/main.ts (372 bytes)
```

```
[ ... ]
```

```
[root@ITEM-angular $] > ng serve
```

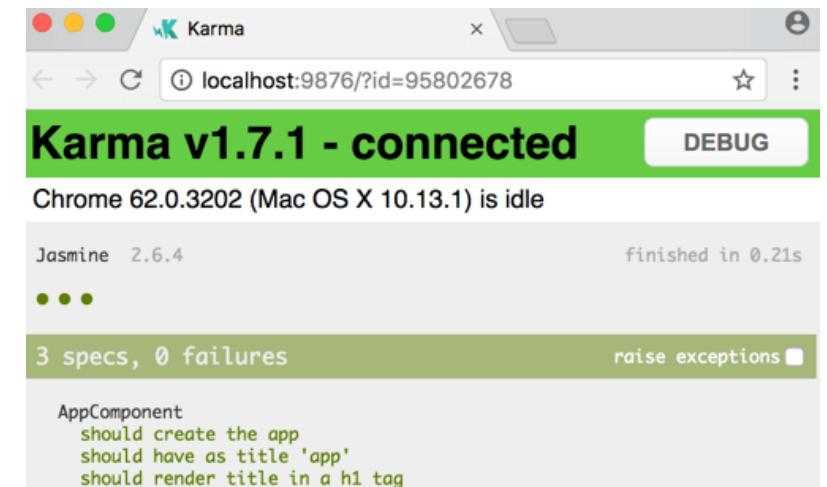
```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
```



ANGULAR TESTING

- The Angular CLI downloads and install everything you need to test an Angular application with the [Jasmine test framework](#).
- The project you create with the CLI is immediately ready to test. Just run the [ng test](#) CLI command.
- `ng test` command builds the app in *watch mode*, and launches the [Karma test runner](#).

```
10% building modules 1/1 modules 0 active
...INFO [karma]: Karma v1.7.1 server started at http://0.0.0.0:9876/
...INFO [launcher]: Launching browser Chrome ...
...INFO [launcher]: Starting browser Chrome
...INFO [Chrome ...]: Connected on socket ...
Chrome ....: Executed 3 of 3 SUCCESS (0.135 secs / 0.205 secs)
```



ANGULAR PACKAGING / DISTRIBUTION / DOCUMENTATION

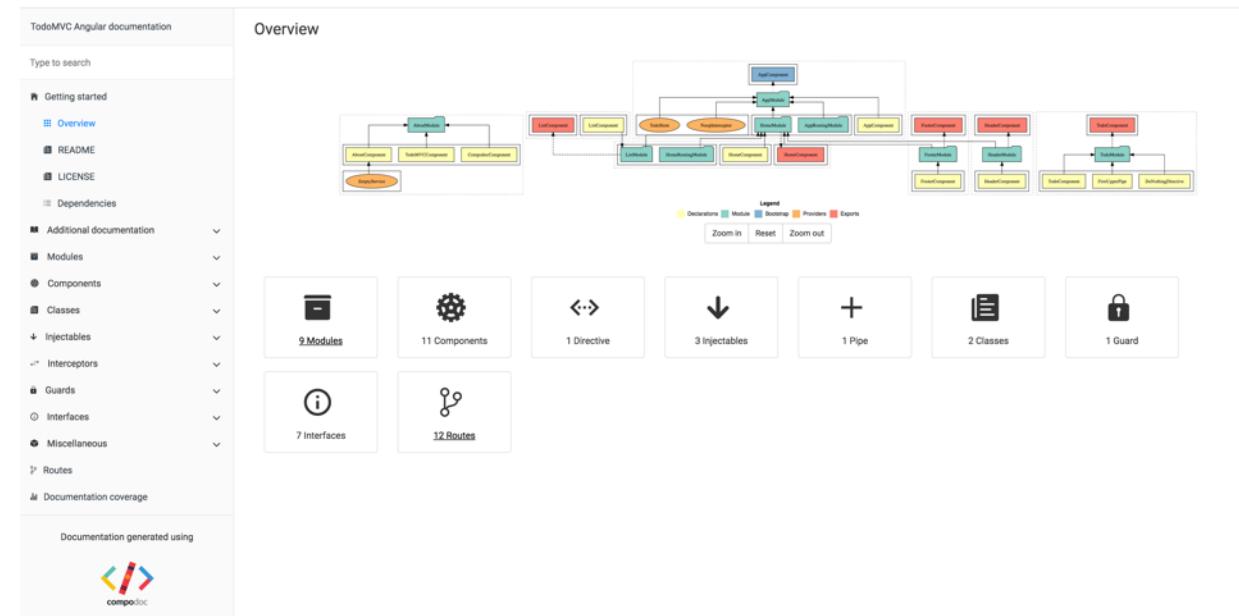
- Angular libraries can be easily build and packaged so to

- Be used in other project:

```
npm install ../example-ng6-lib/dist/example-ng6-lib/example-ng6-lib-0.0.1.tgz
```

- Shared by publishing it on NPM

- Create documentation



[<https://compodoc.github.io/compodoc-demo-todomvc-angular/overview.html>]

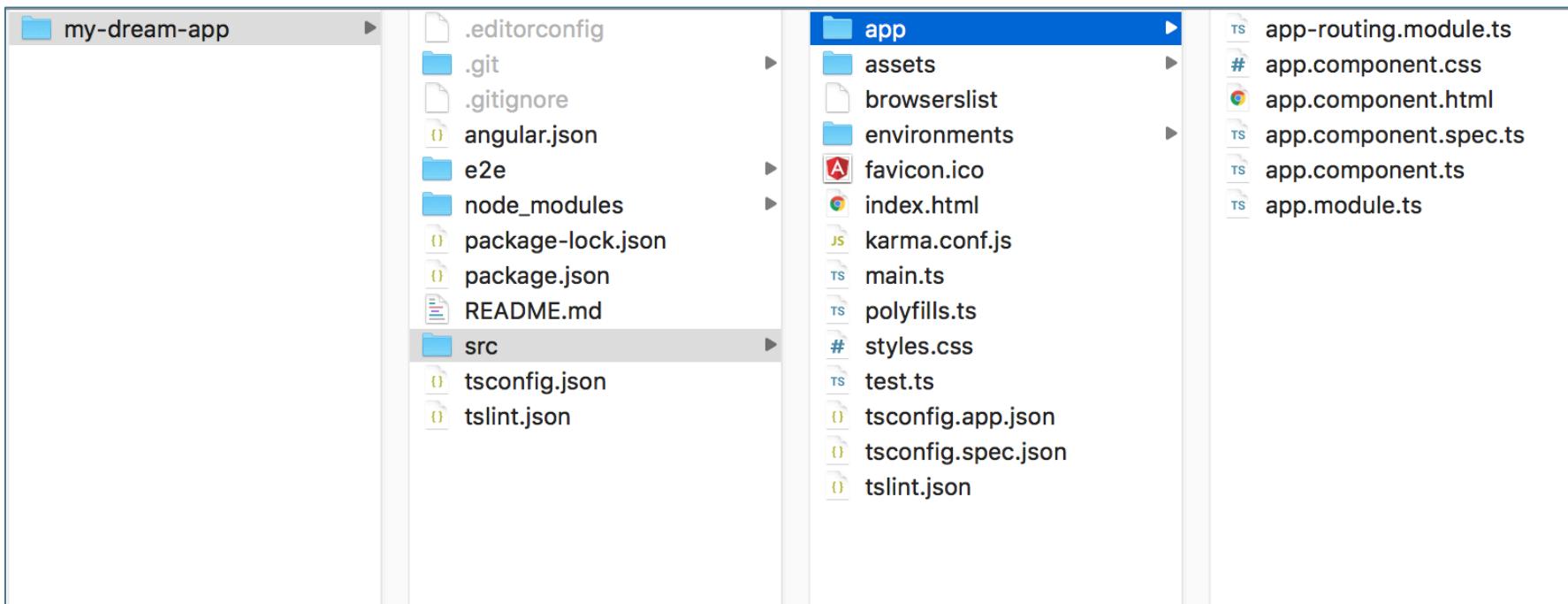


ANGULAR SERVER SIDE RENDERING

- By default Angular uses client-side rendering which is a problem when your website should be listed in search engine results.
- Server-side rendering for Angular applications can be achieved with Angular Universal. Angular Universal is a technology that renders Angular applications on the server.
 - Facilitate web crawlers through [search engine optimization \(SEO\)](#)
 - Improve performance on mobile and low-powered devices
 - Show the first page quickly with a [first-contentful paint \(FCP\)](#)



ANGULAR APP ANATOMY



ANGULAR APP ANATOMY: KEYWORDS

- **MODULES**

An NgModule declares a compilation context for a set of components that is dedicated to an application domain, a workflow, or a closely related set of capabilities. Every Angular app has a *root module*, conventionally named AppModule, which provides the bootstrap mechanism that launches the application.

- **COMPONENTS**

Every Angular application has at least one component, the *root component* that connects a component hierarchy with the page document object model (DOM). Each component defines a class that contains application data and logic, and is associated with an HTML *template* that defines a view to be displayed in a target environment.

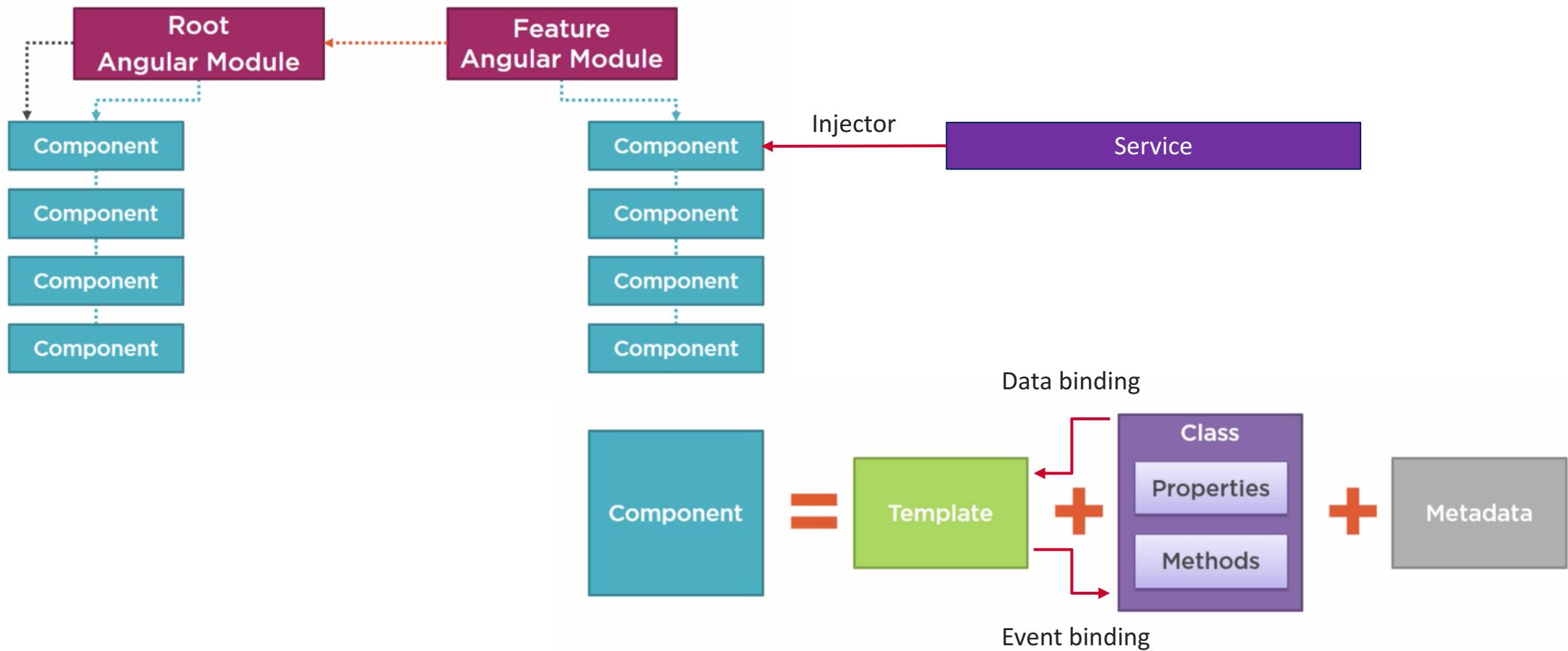
- Templates and views
- Component metadata
- Data binding
- Directives
- Pipes

- **SERVICES (dependency injection)**

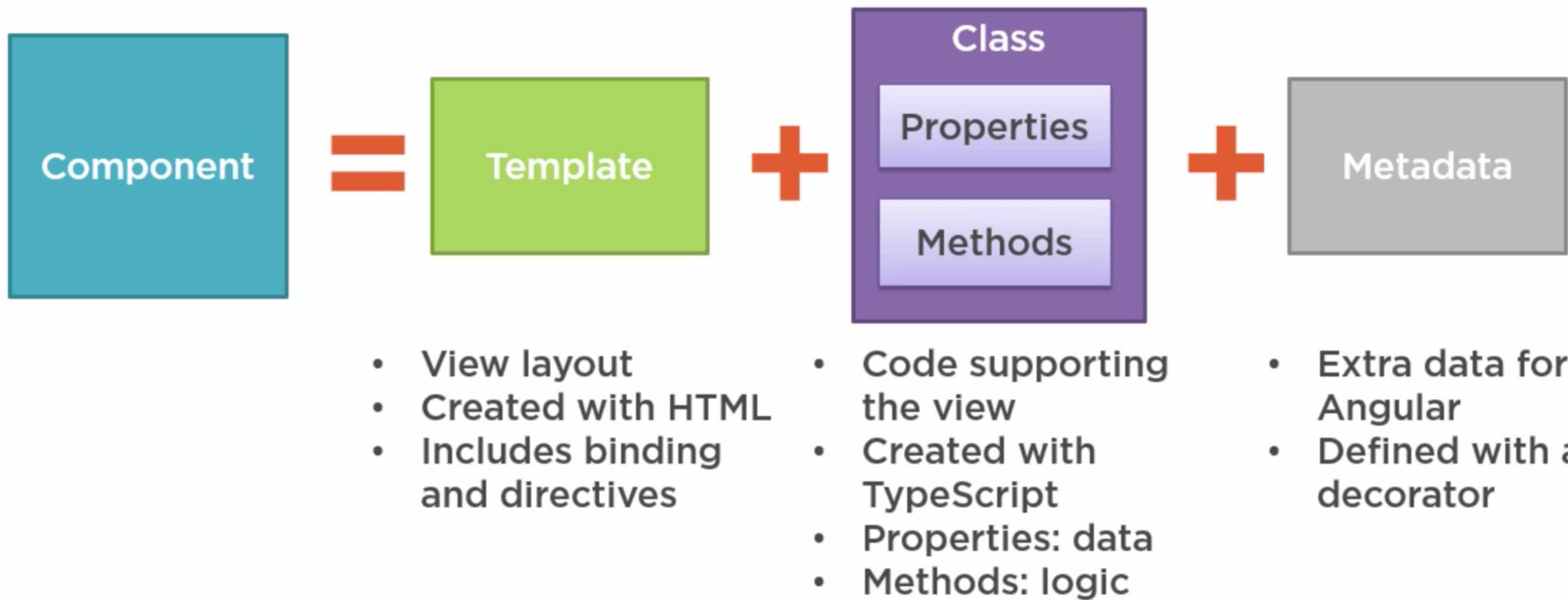
For data or logic that isn't associated with a specific view, and that you want to share across components, you create a *service* class. D.I. is a coding pattern in which a class asks for dependencies from external sources rather than creating them itself.



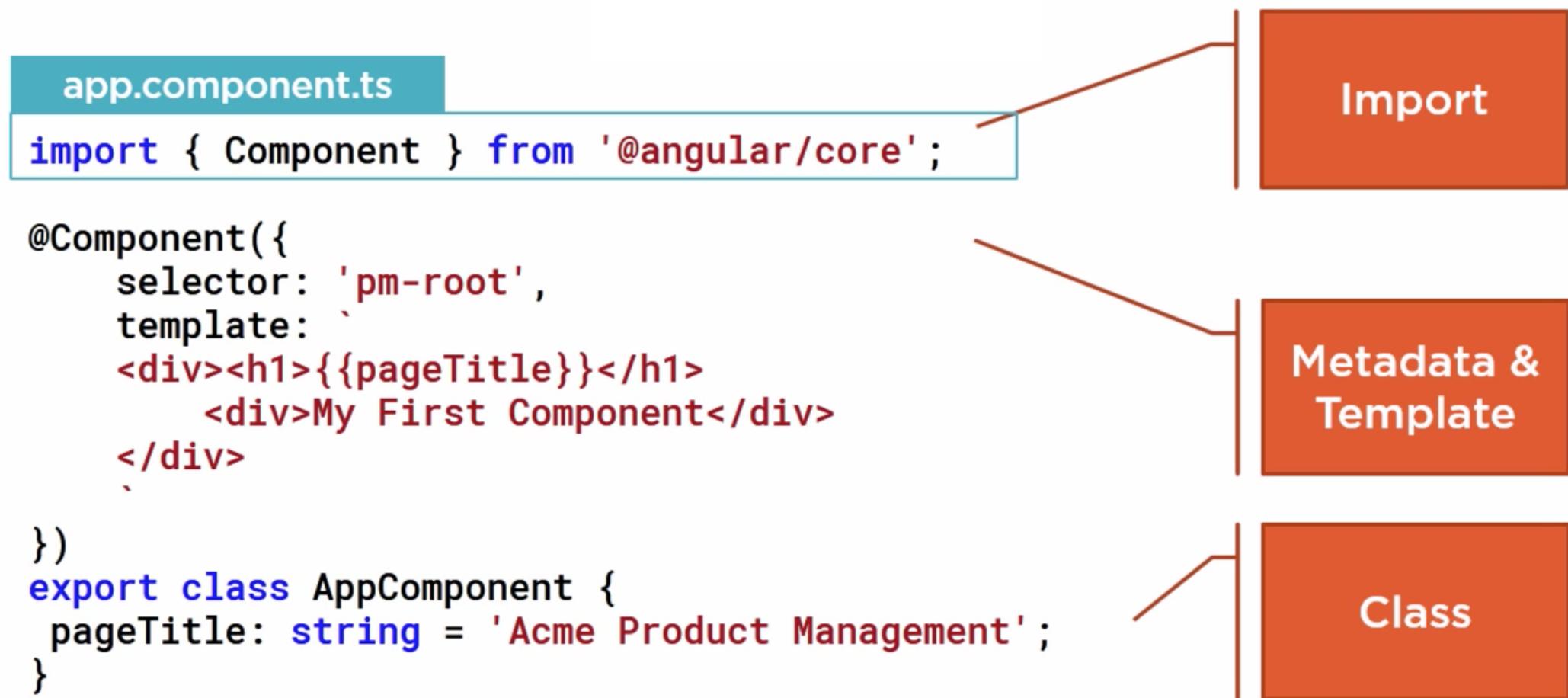
ANGULAR APP ANATOMY



ANGULAR COMPONENT



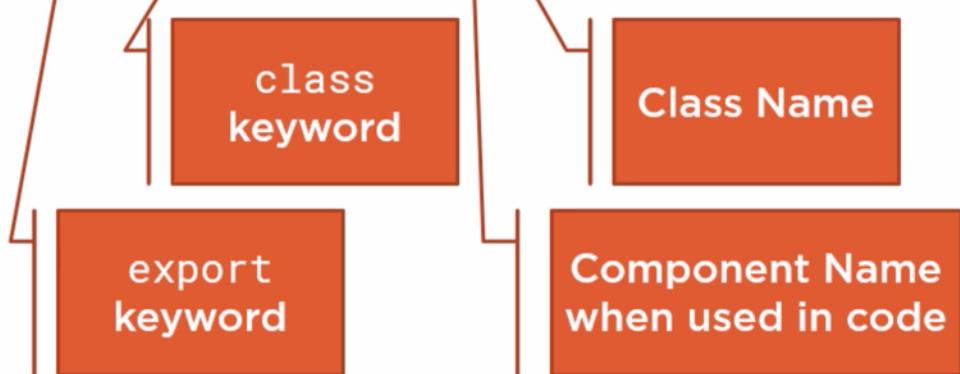
ANGULAR COMPONENT: FILE.TS



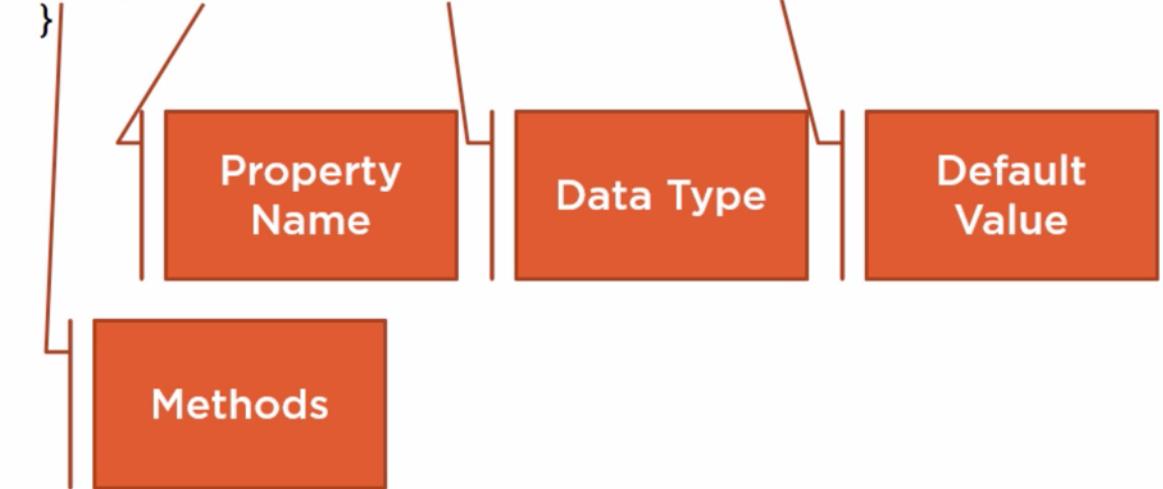
ANGULAR COMPONENT: CLASS

app.component.ts

```
export class AppComponent {  
  pageTitle: string = 'Acme Product Management';  
}
```



```
export class AppComponent {  
  pageTitle: string = 'Acme Product Management';  
}
```



ANGULAR COMPONENT: CLASS

Clear name

- Use PascalCasing
- Append "Component" to the name

export keyword

Data in properties

- Appropriate data type
- Appropriate default value
- camelCase with first letter lowercase

Logic in methods

- camelCase with first letter lowercase

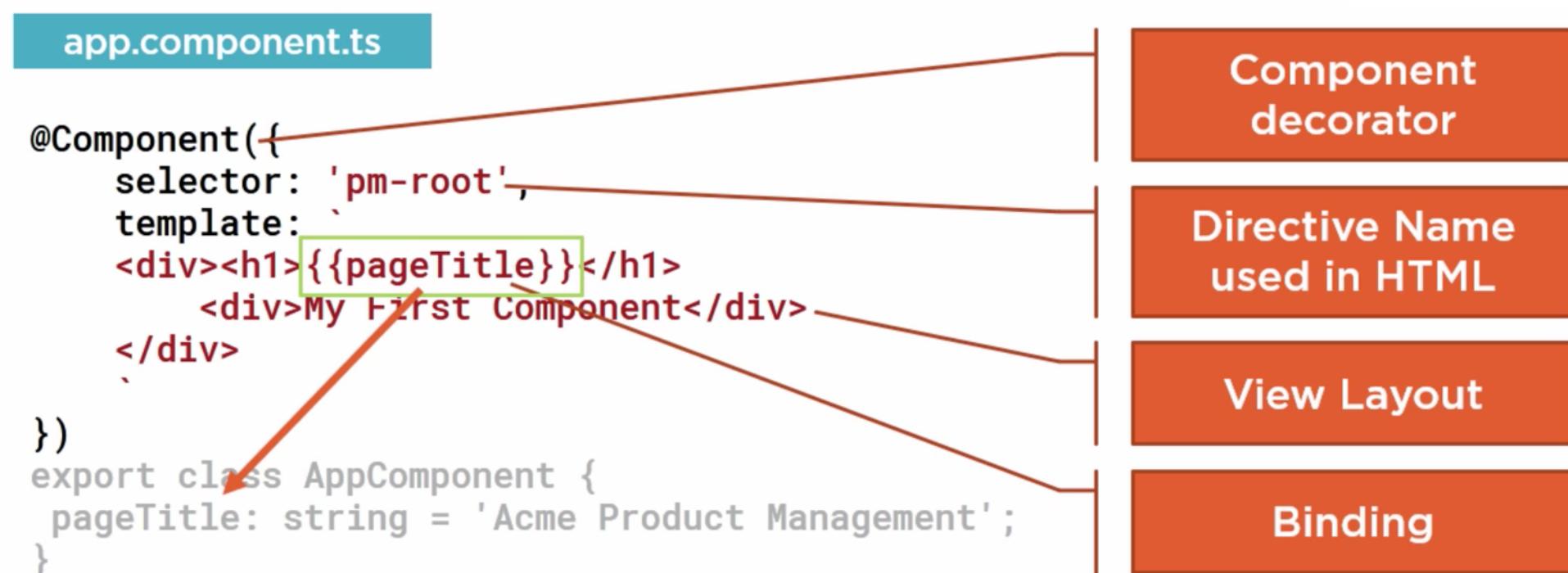


ANGULAR COMPONENT: DECORATOR

A function that adds **metadata** to a class, its members, or its method arguments.

Prefixed with an @.

Angular provides built-in decorators.



ANGULAR COMPONENT: DECORATOR

Component decorator

- Prefix with @; Suffix with ()

selector: Component name in HTML

- Prefix for clarity

template: View's HTML

- Correct HTML syntax



ANGULAR COMPONENT: IMPORT

Before we use an external function or class, we define where to find it

import statement

import allows us to use exported members from external ES modules

Import from a third-party library, our own ES modules, or from Angular

```
import { Component } from '@angular/core';  
  
@Component({  
  selector: 'pm-root',  
  template:  
    <div><h1>{{pageTitle}}</h1>  
      <div>My First Component</div>  
    </div>  
})  
export class AppComponent {  
  pageTitle: string = 'Acme Product Management';  
}
```

@angular/
core

@angular/
animate

@angular/
http

@angular/
router

Angular is modular:

<https://www.npmjs.com/~angular>

import keyword

Angular library
module name

Member name



ANGULAR COMPONENT: COMPONENT AS A DIRECTIVE

Create component in CLI:

```
ng g c [component name]
```

app.component.ts

```
@Component({  
  selector: 'pm-root',  
  template:  
    <div><h1>{{pageTitle}}</h1>  
    1      <pm-products></pm-products>  
    </div>  
})  
export class AppComponent { }
```

product-list.component.ts

```
@Component({  
  selector: 'pm-products',  
  templateUrl:  
    './product-list.component.html'  
})  
export class ProductListComponent { }
```

app.module.ts

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



ANGULAR COMPONENT: TEMPLATE HTML

Inline Template

```
template:  
"<h1>{{pageTitle}}</h1>"
```

Inline Template

```
template:  
'  
<div>  
  <h1>{{pageTitle}}</h1>  
  <div>  
    My First Component  
  </div>  
</div>  
'
```

Linked Template

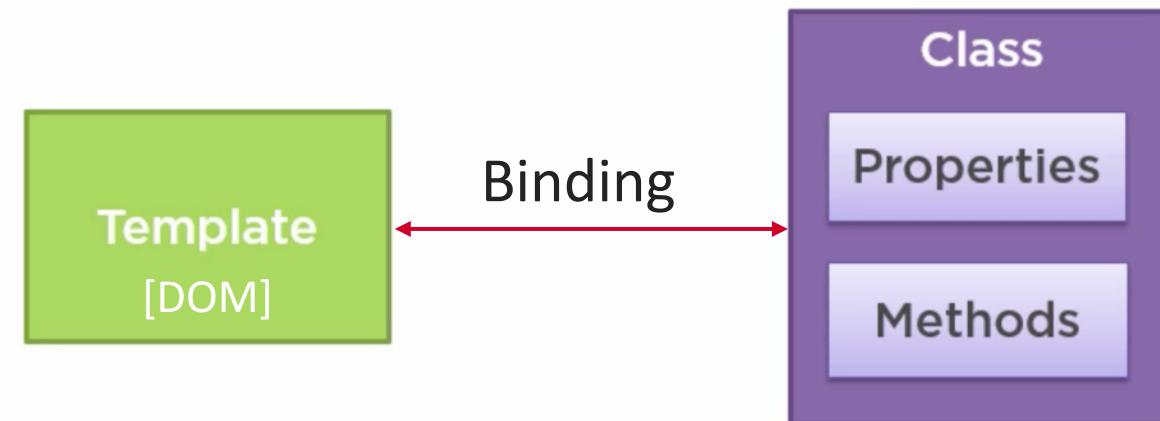
```
templateUrl:  
'./product-list.component.html'
```

ES 2015
Back Ticks

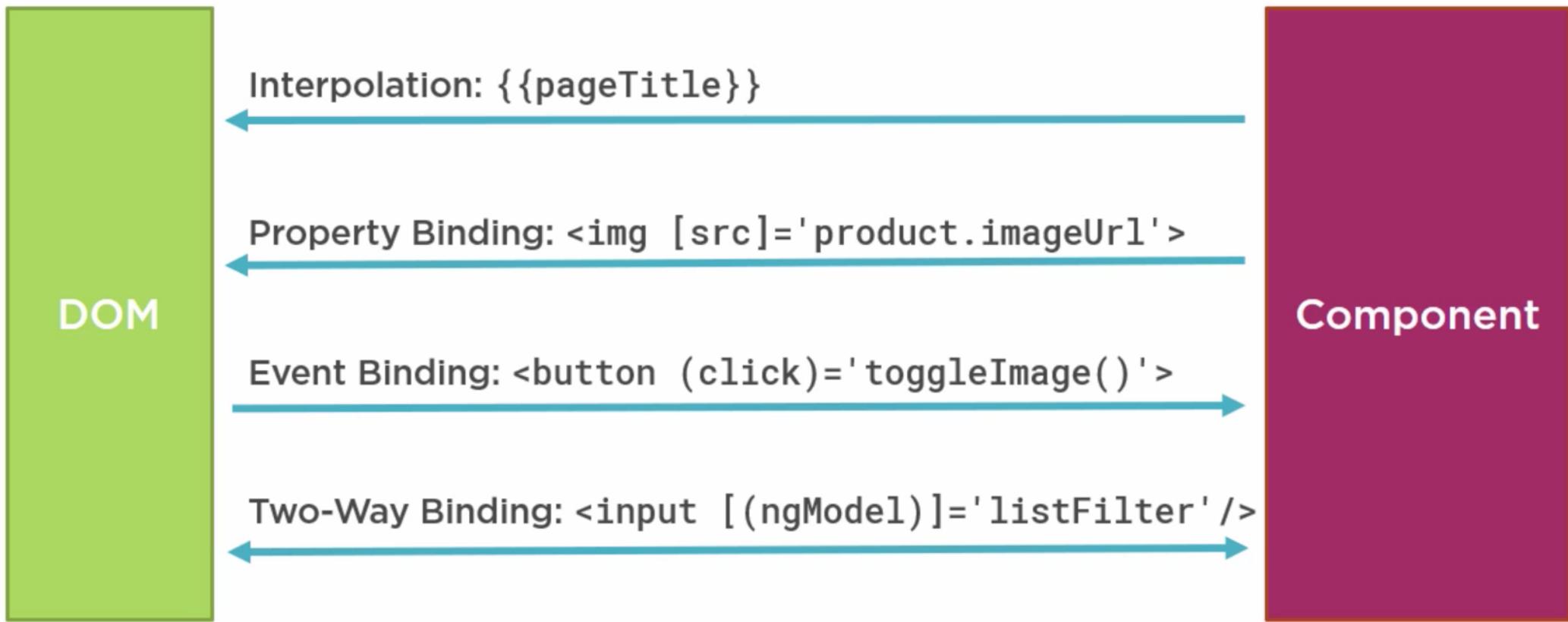


ANGULAR COMPONENT: BINDING

Coordinates communication between the component's class and its template and often involves passing data.



ANGULAR COMPONENT: DATA BINDING



ANGULAR COMPONENT: DATA BINDING - INTERPOLATION

Template

```
<h1>{{pageTitle}}</h1>
{{'Title: ' + pageTitle}}
{{2*20+1}}
{{'Title: ' + getTitle()}}
<h1[innerText]={{pageTitle}}></h1>
```

Class

```
export class AppComponent {
  pageTitle: string =
    'Acme Product Management';
  getTitle(): string {...};
}
```



ANGULAR COMPONENT: DATA BINDING – PROPERTY BINDING

```
<img [src]='product.imageUrl'>  
  
<img src={{product.imageUrl}}>  
  
<img src='http://openclipart.org/{{product.imageUrl}}'>
```

```
<td>  
    <img [src]='product.imageUrl'  
          [title]='product.productName'  
          [style.width.px]='imageWidth'  
          [style.margin.px]='imageMargin'>  
</td>
```



ANGULAR COMPONENT: DATA BINDING – EVENT BINDING

Template

Class

```
<h1>{{pageTitle}}</h1>
<img [src]='product.imageUrl'>
<button (click)='toggleImage()'>
```

()
Target Event

''
Template Statement



ANGULAR COMPONENT: DATA BINDING – TWO WAY BINDING

App.module

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ArticlesComponent } from './articles/articles.component';

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

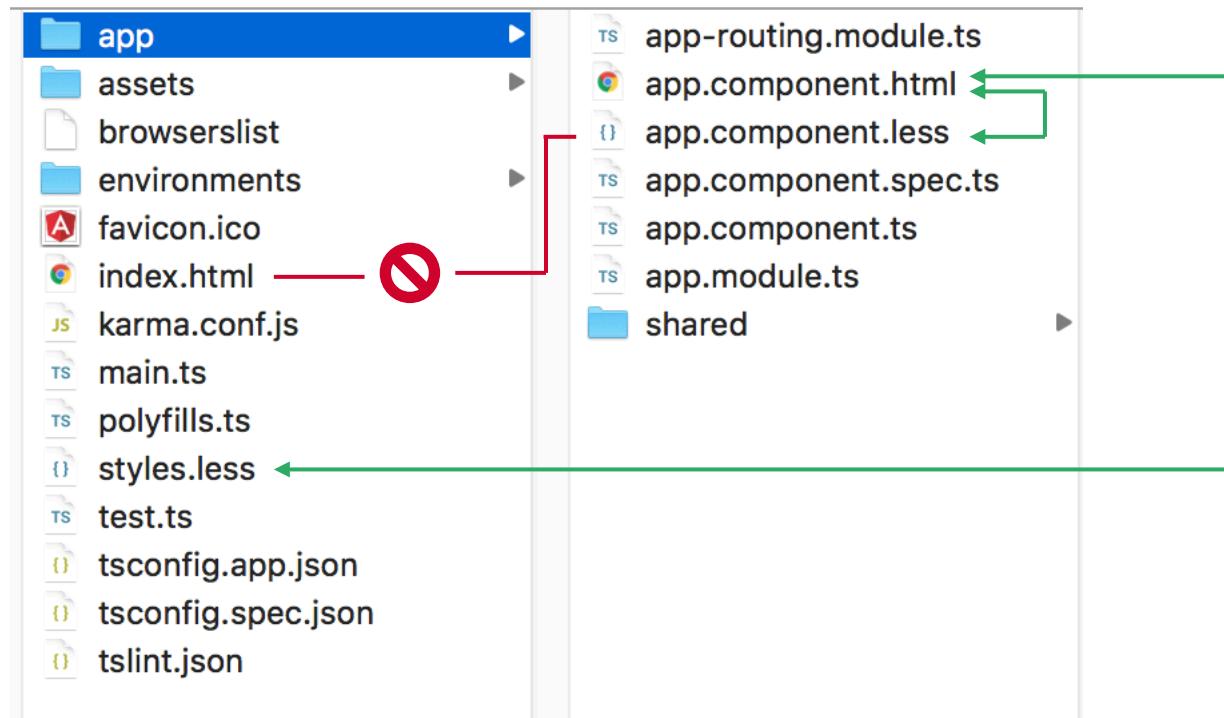
Template

```
<div class="alert alert-secondary text-center">
  <label>Grid: col-xs-</label><input [(ngModel)]='columns' type="number" />
</div>
```

Class



ANGULAR COMPONENT: GLOBAL STYLES AND COMPONENTS STYLES



ANGULAR PIPE

Transforming properties before display with pipes:

Built-in pipes

- date
- number, decimal, percent, currency
- json, slice
- etc

Custom pipes



Pipe character |

Pipe name

Pipe parameters

- Separated with colons

```
 {{ product.productCode | lowercase }}  
 <img [src]='product.imageUrl'  
       [title]='product.productName | uppercase'>  
  
 {{ product.price | currency | lowercase }}  
 {{ product.price | currency:'USD':'symbol':'1.2-2' }}
```



ANGULAR PIPE: CUSTOM PIPE

Create a pipe in CLI:

```
ng g p [pipe name]
```

Module

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

Template

```
<td>{{ product.productCode | convertToSpaces:'-' }}</td>
```

Pipe

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

@Pipe({
  name: 'convertToSpaces'
})
export class ConvertToSpacesPipe
  implements PipeTransform {

  transform(value: string,
            character: string): string{
    }
}
```

```
transform(value: string, character: string): string {
}
```



ANGULAR PIPE: CUSTOM PIPE

Create a custom pipe

Import Pipe and PipeTransform

Create a class that implements
PipeTransform

- export the class

Write code for the Transform method

Use a custom pipe

Import the custom pipe

Add the pipe to the declarations array of
an Angular module

Any template associated with a component
that is also declared in that Angular
module can use that pipe

Use the Pipe in the template

- Pipe character
- Pipe name
- Pipe arguments (separated with colons)

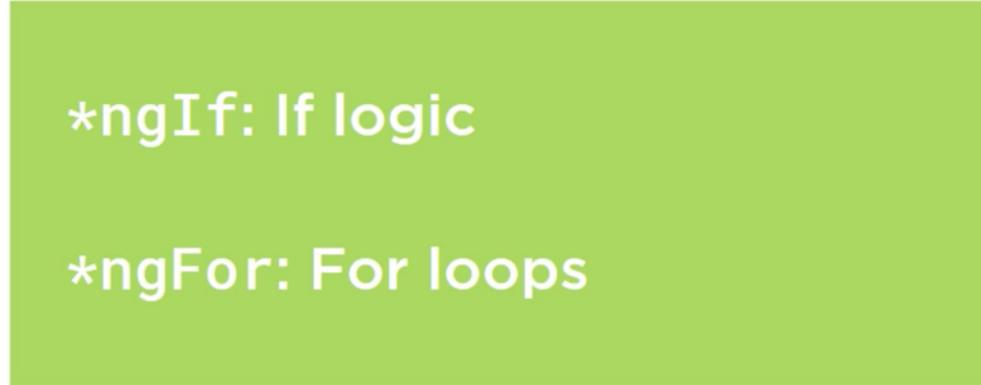


ANGULAR DIRECTIVES

Custom HTML element or attribute used to power up and extend our HTML.

- Custom
- Built-In

- **Structural
Directives**



*ngIf: If logic

*ngFor: For loops



ANGULAR DIRECTIVES: STRUCTURAL DIRECTIVES

```
<tr *ngFor='let product of products'>
  <td></td>
  <td>{{ product.productName }}</td>
  <td>{{ product.productCode }}</td>
  <td>{{ product.releaseDate }}</td>
  <td>{{ product.price }}</td>
  <td>{{ product.starRating }}</td>
</tr>
```

Template
input variable

```
<p>
  <span class="badge badge-secondary">{{articleN.tag}}</span>
  <span *ngIf='articleN.like' class="badge badge-success">Liked!</span>
</p>
```

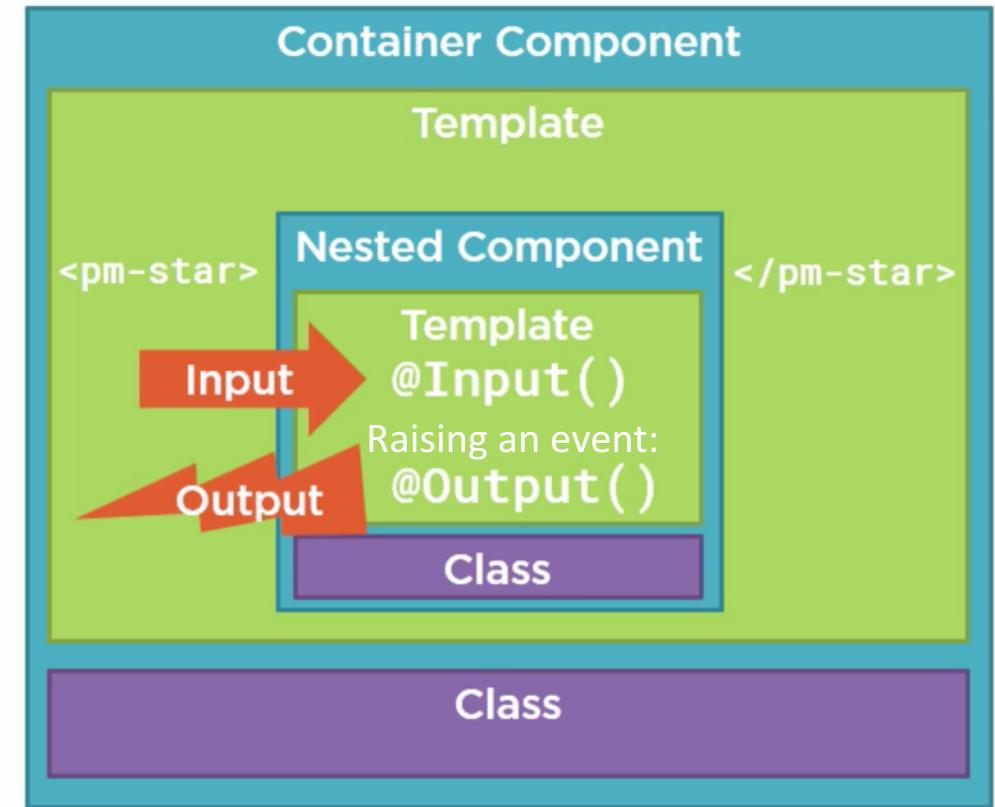


ANGULAR COMPONENT: NESTED COMPONENT

Its template only manages a fragment of a larger view

It has a selector

It optionally communicates with its container



ANGULAR COMPONENT: NESTED COMPONENT

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {
  onNotify(message: string): void { }
}
```

product-list.component.html

```
<td>
  <pm-star [rating]='product.starRating'
            (notify)='onNotify($event)'>
  </pm-star>
</td>
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateUrl: './star.component.html'
})
export class StarComponent {
  @Input() rating: number;
  starWidth: number;
  @Output() notify: EventEmitter<string> =
    new EventEmitter<string>();
```

```
onClick() {
  this.notify.emit('clicked!');
}
}
```

star.component.html

```
<div (click)='onClick()'>
  ... stars ...
</div>
```



ANGULAR COMPONENT: NESTED COMPONENT

Passing data to a nested component (`@Input`)

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {
  onNotify(message: string): void { }
}
```

product-list.component.html

```
<td>
  <pm-star [rating]='product.starRating'
            (notify)='onNotify($event)'>
  </pm-star>
</td>
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateUrl: './star.component.html'
})
export class StarComponent {
  @Input() rating: number;
  starWidth: number;
  @Output() notify: EventEmitter<string> =
    new EventEmitter<string>();

  onClick() {
    this.notify.emit('clicked!');
  }
}
```

star.component.html

```
<div (click)='onClick()'>
  ... stars ...
</div>
```



ANGULAR COMPONENT: NESTED COMPONENT

Passing data from a nested component (**@Output**) and emitting an event

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {
  onNotify(message: string): void { }
}
```

product-list.component.html

```
<td>
  <pm-star [rating]='product.starRating'
            (notify)='onNotify($event)'>
  </pm-star>
</td>
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateUrl: './star.component.html'
})
export class StarComponent {
  @Input() rating: number;
  starWidth: number;
  @Output() notify: EventEmitter<string> =
    new EventEmitter<string>();

  onClick() {
    this.notify.emit('clicked!');
  }
}
```

star.component.html

```
<div (click)='onClick()'>
  ... stars ...
</div>
```



ANGULAR COMPONENT: NESTED COMPONENT

Nested component

Input decorator

- Attached to a property of any type
- Prefix with @; Suffix with ()

Output decorator

- Attached to a property declared as an EventEmitter
- Use the generic argument to define the event payload type
- Use the new keyword to create an instance of the EventEmitter
- Prefix with @; Suffix with ()

Container component

Use the directive

- Directive name -> nested component's selector

Use property binding to pass data to the nested component

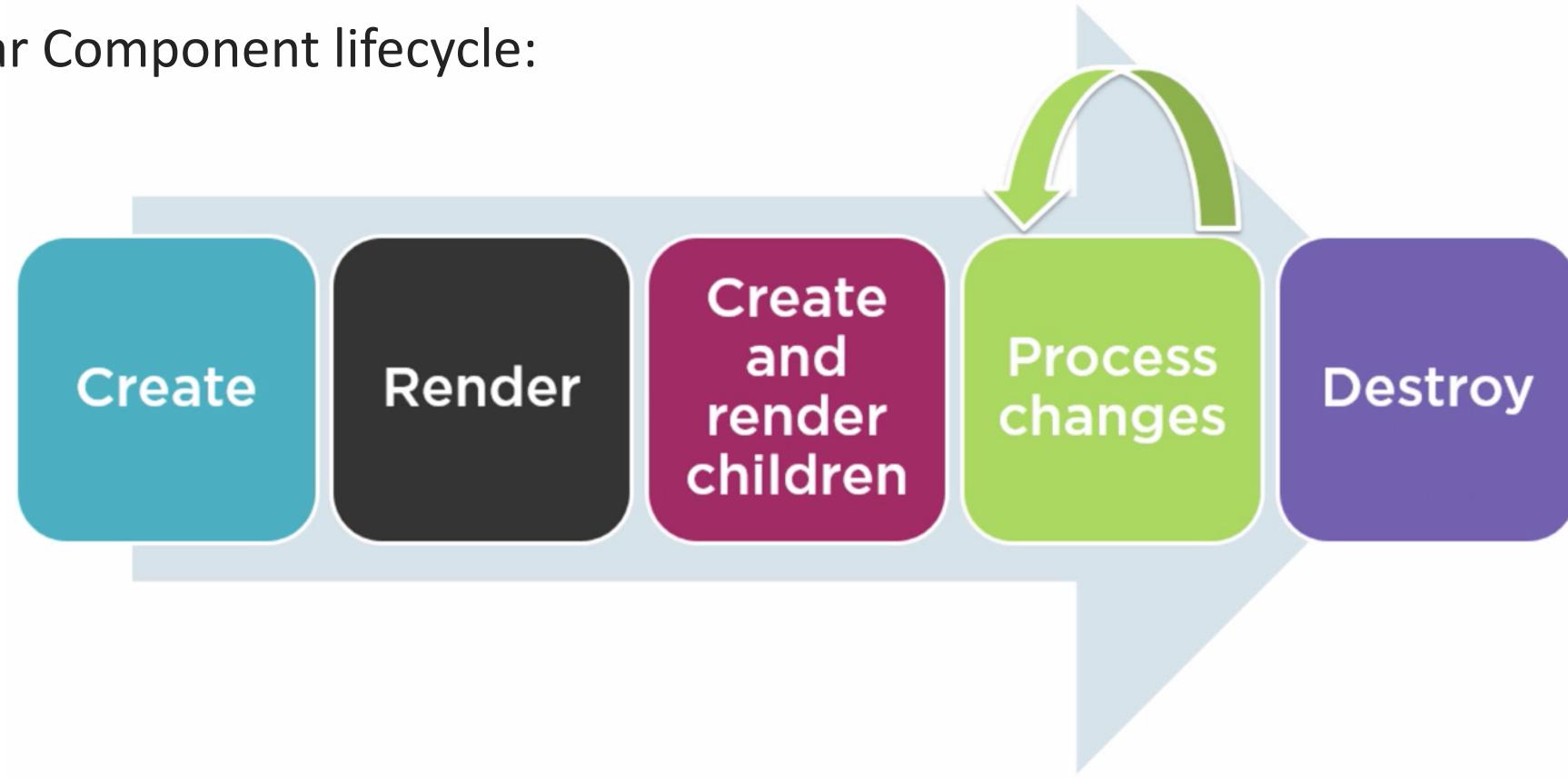
Use event binding to respond to events from the nested component

- Use \$event to access the event payload passed from the nested component



ANGULAR COMPONENT: LIFECYCLE HOOKS

Angular Component lifecycle:



ANGULAR COMPONENT: LIFECYCLE HOOKS

Angular Component lifecycle hooks (<https://angular.io/guide/lifecycle-hooks>)

`ngOnChanges()`

`ngOnInit()`

`ngDoCheck()`

[`ngAfterContentInit\(\)`](#)

`ngAfterContentChecked()`

[`ngAfterViewInit\(\)`](#)

`ngAfterViewChecked()`

`ngOnDestroy()`

OnInit: Perform component initialization, retrieve data

OnChanges: Perform action after change to input properties

OnDestroy: Perform cleanup



ANGULAR COMPONENT: LIFECYCLE HOOKS

```
2 import { Component, OnInit } from '@angular/core';
1 export class ProductListComponent
    implements OnInit {
    pageTitle: string = 'Product List';
    showImage: boolean = false;
    listFilter: string = 'cart';
    products: IProduct[] = [...];
3
3     ngOnInit(): void {
        console.log('In OnInit');
    }
}
```

Import the lifecycle hook interface

Implement the lifecycle hook interface

Write code for the hook method



ANGULAR STRONG TYPING AND INTERFACES

Interface

A **specification** identifying a related set of properties and methods.

A class commits to supporting the specification by **implementing** the interface.

Use the interface as a **data type**.

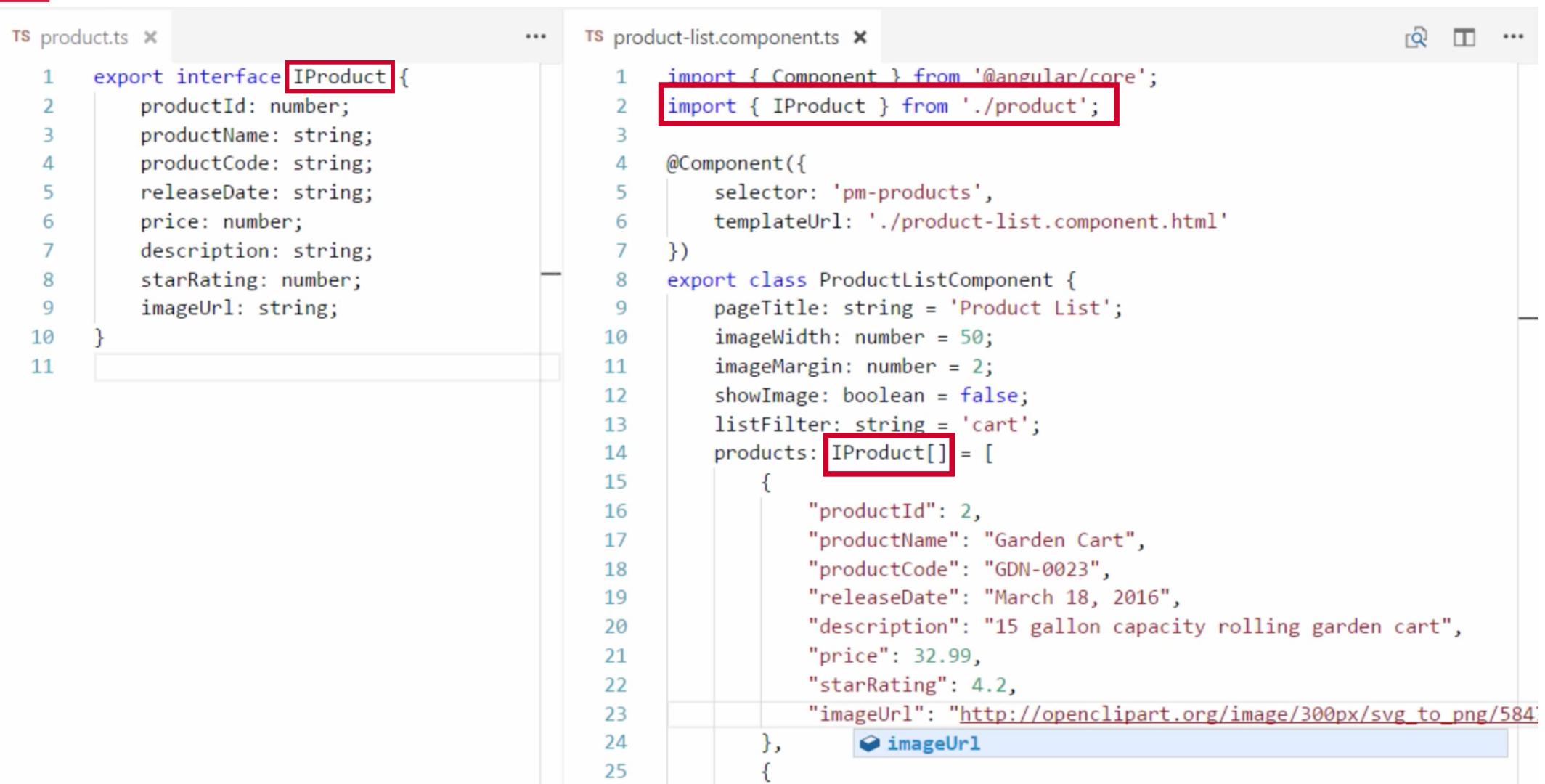
Create an interface via CLI:

```
ng g i article
```

```
export interface IProduct {
  productId: number;
  productName: string;
  productCode: string;
  releaseDate: Date;
  price: number;
  description: string;
  starRating: number;
  imageUrl: string;
  calculateDiscount(percent: number): number;
}
```



ANGULAR STRONG TYPING AND INTERFACES



```
TS product.ts x ...
1 export interface IProduct {
2   productId: number;
3   productName: string;
4   productCode: string;
5   releaseDate: string;
6   price: number;
7   description: string;
8   starRating: number;
9   imageUrl: string;
10 }
11

TS product-list.component.ts x ...
1 import { Component } from '@angular/core';
2 import { IProduct } from './product';
3
4 @Component({
5   selector: 'pm-products',
6   templateUrl: './product-list.component.html'
7 })
8 export class ProductListComponent {
9   pageTitle: string = 'Product List';
10  imageWidth: number = 50;
11  imageMargin: number = 2;
12  showImage: boolean = false;
13  listFilter: string = 'cart';
14  products: IProduct[] = [
15    {
16      "productId": 2,
17      "productName": "Garden Cart",
18      "productCode": "GDN-0023",
19      "releaseDate": "March 18, 2016",
20      "description": "15 gallon capacity rolling garden cart",
21      "price": 32.99,
22      "starRating": 4.2,
23      "imageUrl": "http://openclipart.org/image/300px/svg_to_png/584"
24    },
25    {
26      "productId": 3,
27      "productName": "Kitchen Cart",
28      "productCode": "KDN-0023",
29      "releaseDate": "April 12, 2016",
30      "description": "30 gallon capacity rolling kitchen cart",
31      "price": 42.99,
32      "starRating": 4.5,
33      "imageUrl": "http://openclipart.org/image/300px/svg_to_png/584"
34    }
35  ]
36 }
```



ANGULAR: ASYNCHRONOUS CALLS

In angular it involves:

- Services
- HttpClient Module
- Observables and reactive extensions



ANGULAR: SERVICES

A service:

- is typically a **class** with a **narrow, well-defined purpose**.
So It should do something really specific
- Is used for feature that:
 - Are independent from any particular component
 - Provide shared data or logic across components)
- It hasn't templates because it is concerning just with logic or data



ANGULAR: SERVICES

To use a service:

- Create it
- Just import it: *dependency injection* allows you to use it without creating a new instance (services are *singletons*, so there is just one instance shared)
- *Registration* defines where a Service is available (using a hierachic dependency injection)

Root Injector

Service is available throughout the application

Recommended for most scenarios

Component Injector

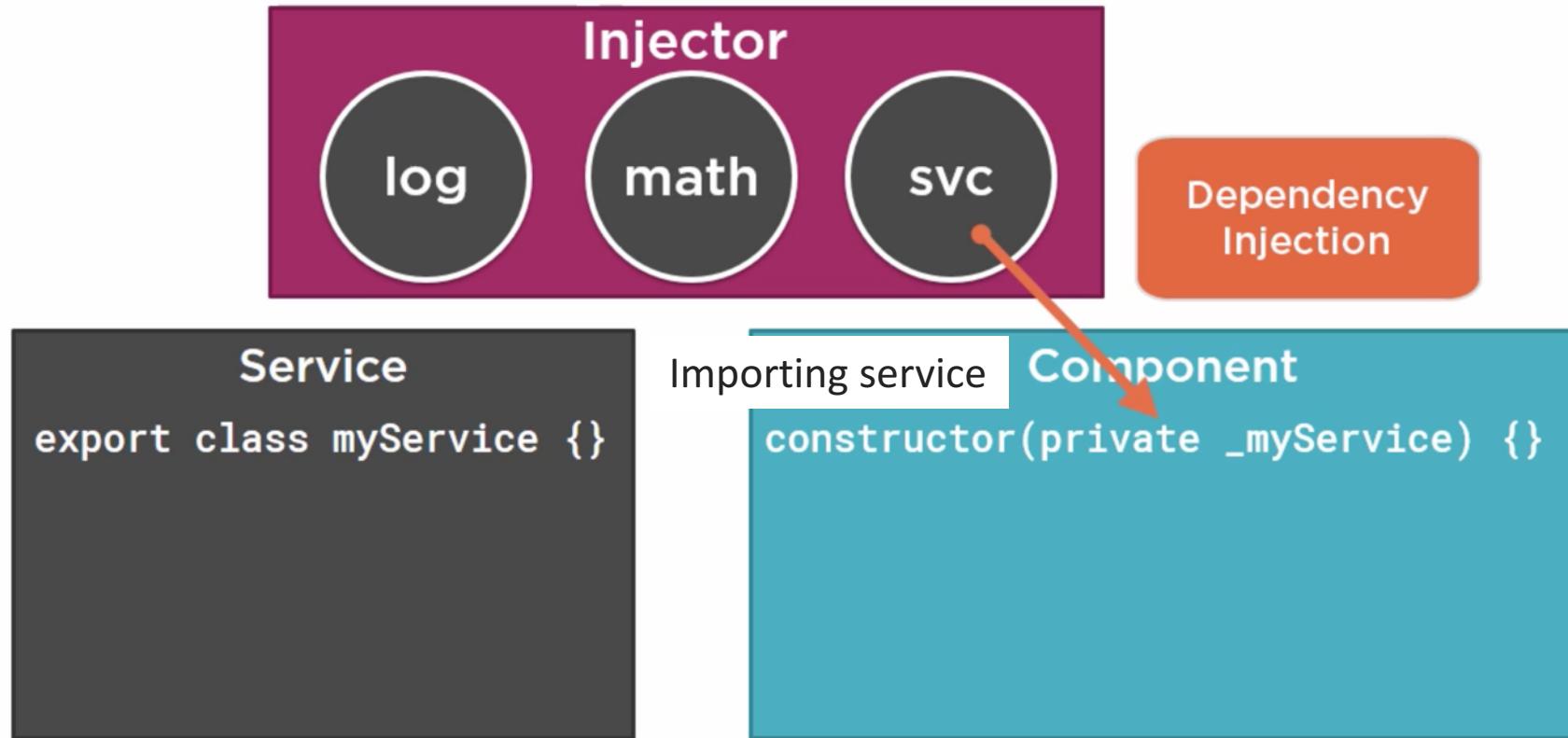
Service is available ONLY to that component and its child (nested) components

Isolates a service used by only one component

Provides multiple instances of the service



ANGULAR: SERVICES



ANGULAR: SERVICE CREATION AND REGISTRATION

ng generate service [serviceName]

It use injector, so it imports it

```
product.service.ts
import { Injectable } from '@angular/core'

@Injectable({
  providedIn: 'root'
})
export class ProductService {
  getProducts(): IProduct[] {
  }
}
```

`product.service.ts`

`import { Injectable } from '@angular/core'`

`@Injectable({
 providedIn: 'root'
})`

`export class ProductService {

 getProducts(): IProduct[] {
 }

}`

Root injector, but it could be
a Module or a Component
providedIn: 'HeroModule'

PascalCase
Word "Service" appended



ANGULAR: SERVICE COMPONENT REGISTRATION

product-list.component.ts

```
...
import { ProductService } from './product.service';

@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {

  constructor(private productService: ProductService) {
  }

}
```

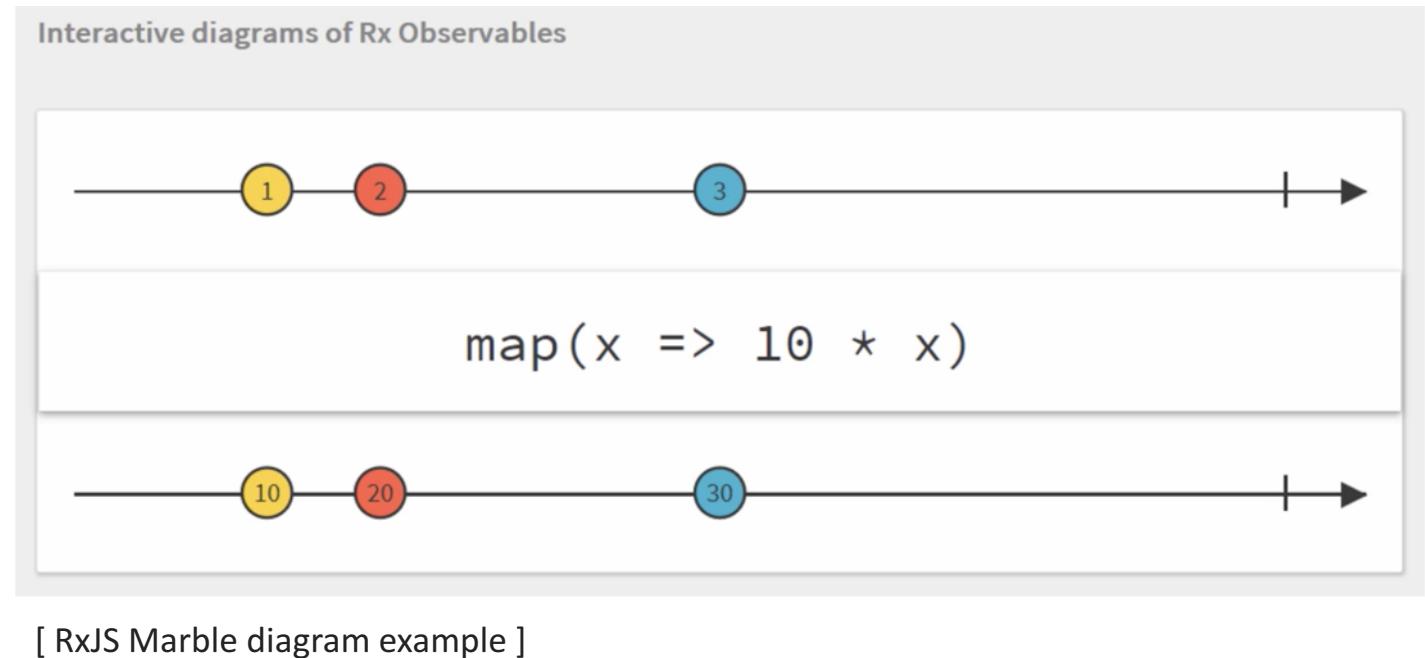
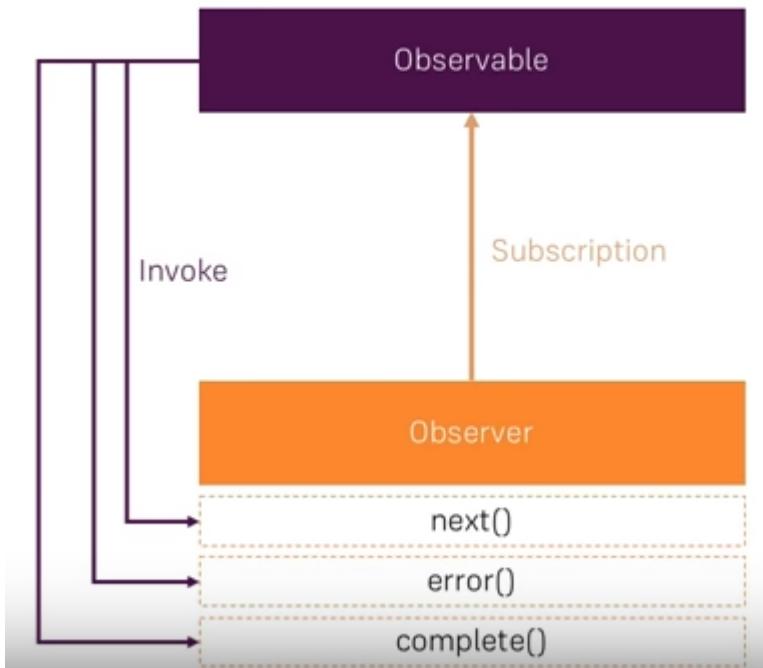


ANGULAR: HTTP CLIENT AND OBSERVABLES

- AngularJS had a `$http` built in service, based on Promise
- Angular has instead an Observable approach, based on RxJS library
- Observable
 - is an element of **Reactive Programming** (a programming paradigm concerned with data streams and the propagation of changes)
 - usually is an object wrapping some data source (stream of data source that continuously emits data over time)
 - emits events
 - subscribes an Observer, that observes events and do something
- Observer is there to execute some piece of code whenever we receive some data, error or if observable reports that this is done.



ANGULAR: HTTP CLIENT AND OBSERVABLES



ANGULAR: PROMISE VS OBSERVABLE

Promise	Observable
Provides a single future value	Emits multiple values over time
Not lazy	Lazy
Not cancellable	Cancellable
	Supports map, filter, reduce and similar operators



ANGULAR: SERVICE + HTTP REQUEST + OBSERVABLES

Registering the Http Service Provider

app.module.ts

```
...  
import { HttpClientModule } from '@angular/common/http';  
  
@NgModule({  
  imports: [  
    BrowserModule,  
    FormsModule,  
    HttpClientModule ],  
  declarations: [  
    AppComponent,  
    ProductListComponent,  
    ConvertToSpacesPipe,  
    StarComponent ],  
  bootstrap: [ AppComponent ]  
})  
export class AppModule { }
```

Sending an Http Request (Basic data load)

product.service.ts

```
...  
import { HttpClient } from '@angular/common/http';  
import { Observable } from 'rxjs';  
  
@Injectable({  
  providedIn: 'root'  
})  
export class ProductService {  
  private apiUrl = 'www.myWebService.com/api/products';  
  
  constructor(private http: HttpClient) { }  
  
  getProducts(): Observable<IProduct[]> {  
    return this.http.get<IProduct[]>(this.apiUrl);  
  }  
}
```

Exception Handling

product.service.ts

```
...  
import { HttpClient, HttpErrorResponse } from '@angular/common/http';  
import { Observable } from 'rxjs';  
import { catchError, tap } from 'rxjs/operators';  
  
...  
  
getProducts(): Observable<IProduct[]> {  
  return this.http.get<IProduct[]>(this.apiUrl).pipe(  
    tap(data => console.log('All: ' + JSON.stringify(data))),  
    catchError(this.handleError)  
  );  
}  
  
private handleError(err: HttpErrorResponse) {
```

It isn't an Angular Pipe,
but RxJS pipe!

It performs a side effect
for every emission on the
source Observable



ANGULAR: BASIC ROUTING

- The Angular Router is an optional service that presents for a **given URL** a **component view**.
- It is not part of the Angular core.

```
[  
  { path: 'products', component: ProductListComponent },  
  { path: 'products/:id', component: ProductDetailComponent },  
  { path: 'welcome', component: WelcomeComponent },  
  { path: '', redirectTo: 'welcome', pathMatch: 'full' },  
  { path: '**', component: PageNotFoundComponent }  
]
```

App.module

- The service injects the component in directive (in app.component):
<router-outlet></router-outlet>
- It needs a base in index.html : <base href='/' />
- Simply activated by attribute routerLink



ANGULAR: BASIC ROUTING

```
src/index.html (base-href)
<base href="/">
```

It toggles css classes for active RouterLink bindings based on the current RouterState

```
src/app/app.component.html
<h1>Angular Router</h1>
<nav>
  <a routerLink="/crisis-center" routerLinkActive="active">Crisis Center</a>
  <a routerLink="/heroes" routerLinkActive="active">Heroes</a>
</nav>
<router-outlet></router-outlet>
```

RouterModule configuration set on import

```
src/app/app.module.ts (import)

import { RouterModule, Routes } from '@angular/router';

src/app/app.module.ts (excerpt)

const appRoutes: Routes = [
  { path: 'crisis-center', component: CrisisListComponent },
  { path: 'hero/:id', component: HeroDetailComponent },
  {
    path: 'heroes',
    component: HeroListComponent,
    data: { title: 'Heroes List' }
  },
  { path: '', redirectTo: '/heroes', pathMatch: 'full' },
  { path: '**', component: PageNotFoundComponent }
];

@NgModule({
  imports: [
    RouterModule.forRoot(
      appRoutes,
      { enableTracing: true } // <-- debugging purposes only
    )
    // other imports here
  ],
  ...
})
export class AppModule { }
```



ANGULAR: BASIC ROUTING

The diagram illustrates the flow of data from a user interaction to the corresponding Angular routing and component logic.

Browser Screenshot: A screenshot of a browser window titled "Acme Product Management". The address bar shows "localhost:4200/products". The navigation bar includes links for "Home" and "Product List". The "Product List" link is highlighted with a blue box and has a green callout pointing to its HTML representation: "Product List".

Router Configuration: A green box contains the route definition: "{ path: 'products', component: ProductListComponent }".

Component Implementation: A callout from the component code points to the "router-outlet" element in the template. The component code is shown in a blue box:

```
product-list.component.ts
import { Component } from '@angular/core';

@Component({
  templateUrl: './product-list.component.html'
})
export class ProductListComponent { }
```



ANGULAR: ROUTES SETUP

No leading slash

App.module

```
[  
  { path: 'products', component: ProductListComponent },  
  { path: 'products/:id', component: ProductDetailComponent },  
  { path: 'welcome', component: WelcomeComponent },  
  { path: '', redirectTo: 'welcome', pathMatch: 'full' },  
  { path: '**', component: PageNotFoundComponent }]
```

Default route

Wildcard root

*Never in first position

*Order matters!



ANGULAR: KEY ROUTER TERMS

Router Part	Meaning
Router	Displays the application component for the active URL. Manages navigation from one component to the next.
RouterModule	A separate NgModule that provides the necessary service providers and directives for navigating through application views.
Routes	Defines an array of Routes, each mapping a URL path to a component.
Route	Defines how the router should navigate to a component based on a URL pattern. Most routes consist of a path and a component type.
RouterOutlet	The directive (<code><router-outlet></code>) that marks where the router displays a view.
RouterLink	The directive for binding a clickable HTML element to a route. Clicking an element with a <code>routerLink</code> directive that is bound to a <i>string</i> or a <i>link parameters array</i> triggers a navigation.
RouterLinkActive	The directive for adding/removing classes from an HTML element when an associated <code>routerLink</code> contained on or inside the element becomes active/inactive.
ActivatedRoute	A service that is provided to each route component that contains route specific information such as route parameters, static data, resolve data, global query params, and the global fragment.



ANGULAR: ROUTING

Passing parameters to a Route

App.module

```
[  
  { path: 'products', component: ProductListComponent },  
  { path: 'products/:id', component: ProductDetailComponent },  
  { path: 'welcome', component: WelcomeComponent },  
  { path: '', redirectTo: 'welcome', pathMatch: 'full' },  
  { path: '**', component: PageNotFoundComponent }  
]
```



ANGULAR: PASSING PARAMETERS TO A ROUTE

app.module.ts

```
{ path: 'products/:id', component: ProductDetailComponent }
```

product-list.component.html

```
<td>
  <a [routerLink]="/products", product.productId">
    {{product.productName}}
  </a>
</td>
```

product-detail.component.ts

```
import { ActivatedRoute } from '@angular/router';

constructor(private route: ActivatedRoute) {
  console.log(this.route.snapshot.paramMap.get('id'));
}
```



ANGULAR: SINGLE ITEM DATA FROM A SERVICE

articles.component.html

```
<a [routerLink]=["/articles" , articleN.id ]"><img class="img-fluid rounded" src="{{articleN.picture}}></a>
```

article.service.ts

```
getArticle(id: number): Observable<IArticle | undefined> {
  return this.getArticles().pipe(
    map((articles: IArticle[]) => articles.find(p => p.id === id))
  );
}
```

```
| ngFirstApp/src/app/articles/article-detail.component.ts 📄
| @@ -1,7 +1,4 @@
| import { Component, OnInit } from '@angular/core';
| import { ActivatedRoute, Router } from '@angular/router';

| import { IArticle } from './article';
| @@ -14,13 +11,31 @@ import { ArticleService } from './article.service';
|
| export class ArticleDetailComponent implements OnInit {

+   errorMessage = '';
+   article: IArticle | undefined;

+   constructor(private route: ActivatedRoute,
+               private articleService: ArticleService
+             ) {
+
+     }

+   ngOnInit() {
+     const param = this.route.snapshot.paramMap.get('id');
+     if (param) {
+       const id = +param;
+       this.getArticle(id);
+     }
+   }

+   getArticle(id: number) {
+     this.articleService.getArticle(id).subscribe(
+       article => this.article = article,
+       error => this.errorMessage = <any>error);
+   }
}
```



ANGULAR: ACTIVATE ROUTING WITH CODE

```
<button class='btn btn-secondary' (click)='onBack()'>  
    Back  
</button>
```

product-detail.component.ts

```
import { Router } from '@angular/router';  
...  
constructor(private router: Router) { }  
  
onBack(): void {  
    this.router.navigate(['/products']);  
}
```



ANGULAR: PROTECTING ROUTES WITH GUARDS

[CanActivate](#)

Interface that a class can implement to be **a guard deciding if a route can be activated**.

If all guards return true, navigation will continue. If any guard returns false, navigation will be cancelled. If any guard returns a [UrlTree](#), current navigation will be cancelled and a new navigation will be kicked off to the [UrlTree](#) returned from the guard.

[CanActivateChild](#)

Interface that a class can implement to be **a guard deciding if a child route can be activated**.

If all guards return true, navigation will continue. If any guard returns false, navigation will be cancelled. If any guard returns a [UrlTree](#), current navigation will be cancelled and a new navigation will be kicked off to the [UrlTree](#) returned from the guard.

[CanDeactivate](#)

Interface that a class can implement to be **a guard deciding if a route can be deactivated**.

If all guards return true, navigation will continue. If any guard returns false, navigation will be cancelled. If any guard returns a [UrlTree](#), current navigation will be cancelled and a new navigation will be kicked off to the [UrlTree](#) returned from the guard.

[CanLoad](#)

Interface that a class can implement to be **a guard deciding if children can be loaded**.



ANGULAR: PROTECTING ROUTES WITH GUARDS

Ng g g articles/article-detail

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, Router } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ArticleDetailGuard implements CanActivate {

  constructor(private router: Router) { }

  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {
    /**
     * const id = +next.url[1].path;

     if (isNaN(id) || id < 1) {
       alert('Invalid article Id');
       this.router.navigate(['/articles']);
       return false;
     }
     return true;
    /**
   }
}
```

```
ngFirstApp/src/app/app.module.ts 📄

@@ -12,6 +12,7 @@ import { AddPrefixPipe } from './shared/add-prefix.pipe';
import { LikeBtnComponent } from './shared/like-btn/like-btn.component';
import { WelcomeComponent } from './welcome/welcome.component';
import { ArticleDetailComponent } from './articles/article-detail.component';
+ import { ArticleDetailGuard } from './articles/article-detail.guard';

@NgModule({
  @NgModule({
@@ -31,7 +32,9 @@ import { ArticleDetailComponent } from './articles/article-detail.component';
  RouterModule.forRoot([
    {path: 'welcome' , component : WelcomeComponent },
    {path: 'articles' , component : ArticlesComponent },
-   {path: 'articles/:id' , component : ArticleDetailComponent },//send 'id' parameter
+   {path: 'articles/:id' ,
+     canActivate: [ArticleDetailGuard], //Guard
+     component : ArticleDetailComponent },//send 'id' parameter
    {path: '' , redirectTo : 'welcome' , pathMatch : 'full' }, //default
    {path: '**' , redirectTo : 'welcome' , pathMatch : 'full' } /** wildcard selected if no p
  ]),
}
```

