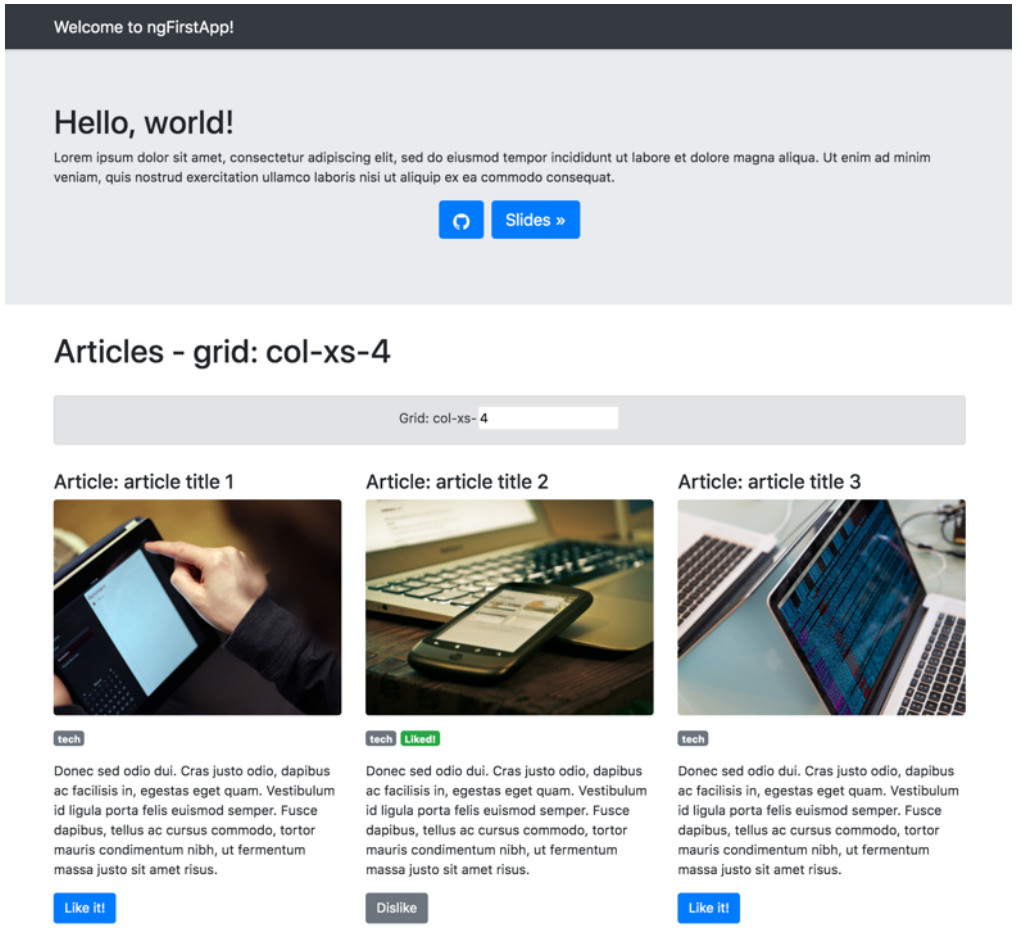
AN INTRODUCTION TO ANGULAR

May 2019

sopra steria

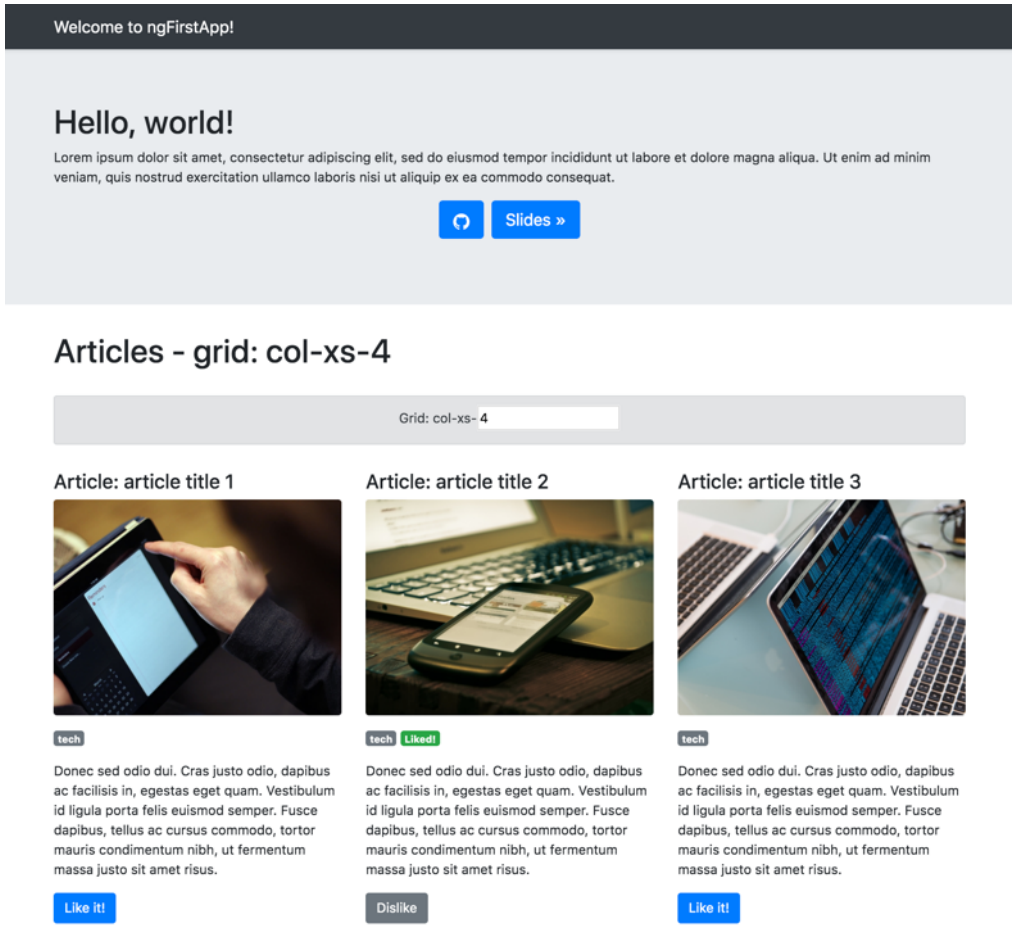
OVERVIEW, OBJECTIVES

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

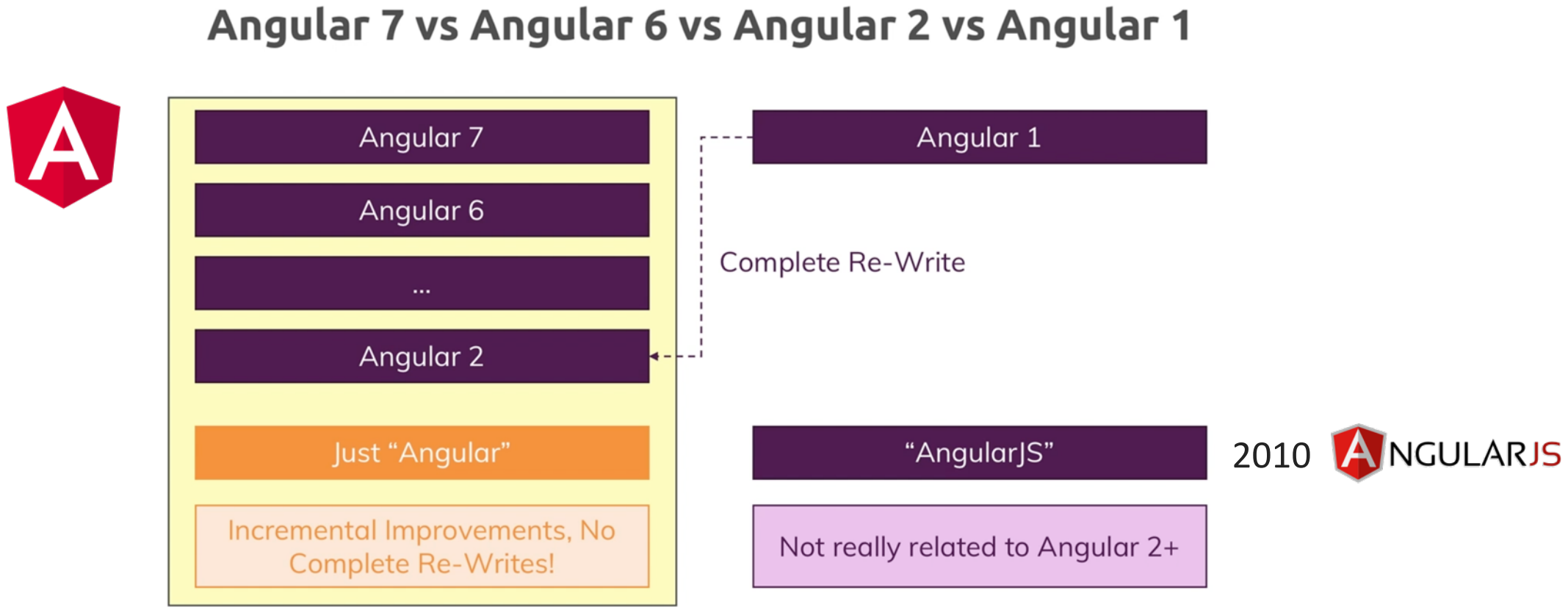


ANGULAR COURSE REPOSITORY

- REPOSITORY: https://github.com/fdelfelice-sopra/an-introduction-to-angular
- COMMITS LIST: https://github.com/fdelfelice-sopra/an-introduction-to-angular/commits/firstApp



ANGULARJS IS NOT ANGULAR:



[https://angular.io/]

[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/]

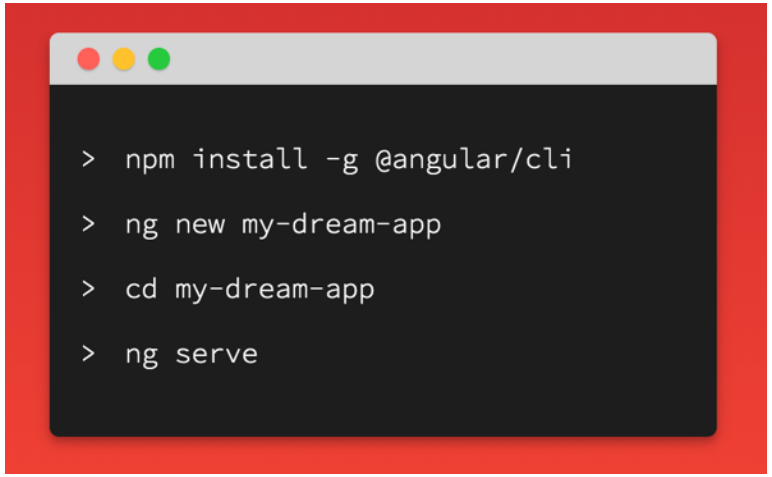
TYPESCRIPT IDE FOR ANGULAR DEVELOPMENT

- 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



[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

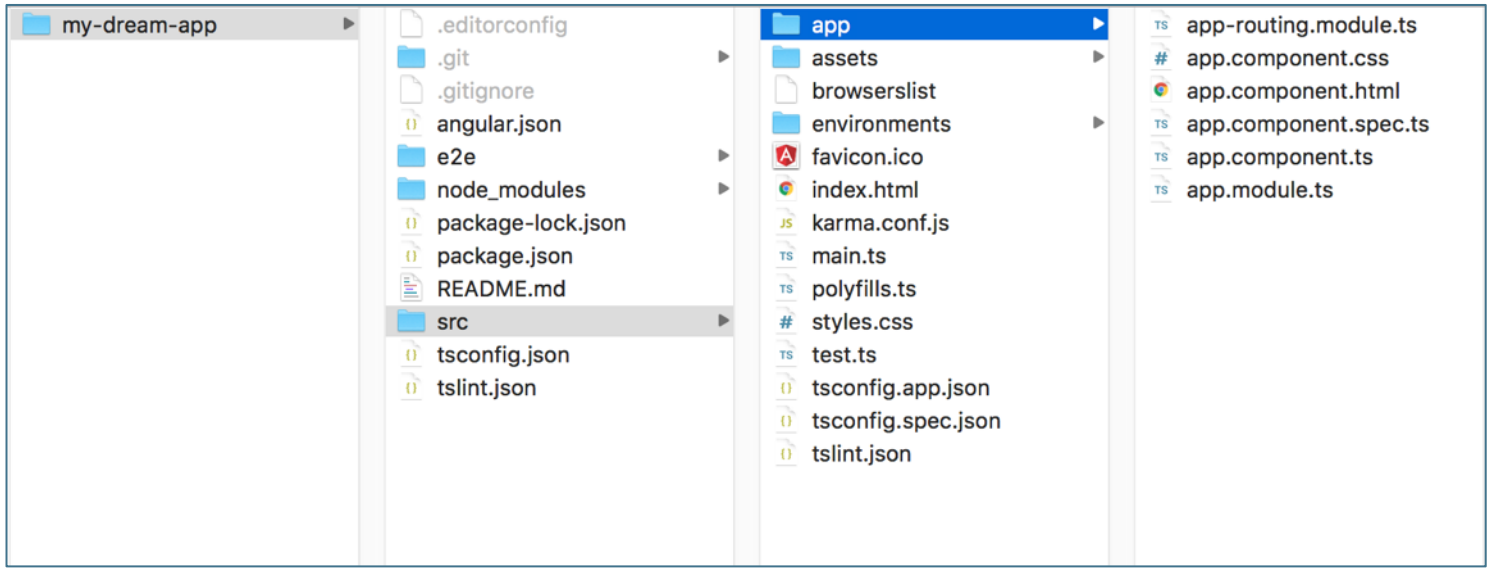
ng e2e - Runs end to end tests using Protractor

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

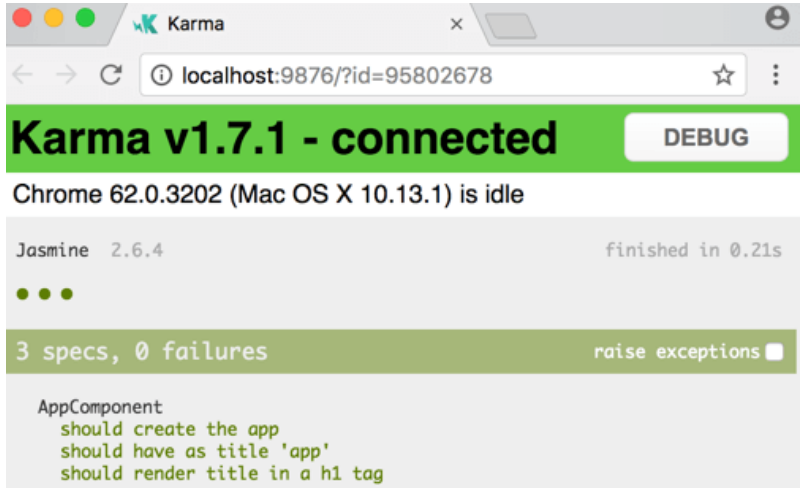
```
[root@ITEM-angular $] > ng new my-dream-app
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE my-dream-app/README.md (1027 bytes)
CREATE my-dream-app/angular.json (3822 bytes)
CREATE my-dream-app/package.json (1311 bytes)
CREATE my-dream-app/tsconfig.json (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)
```

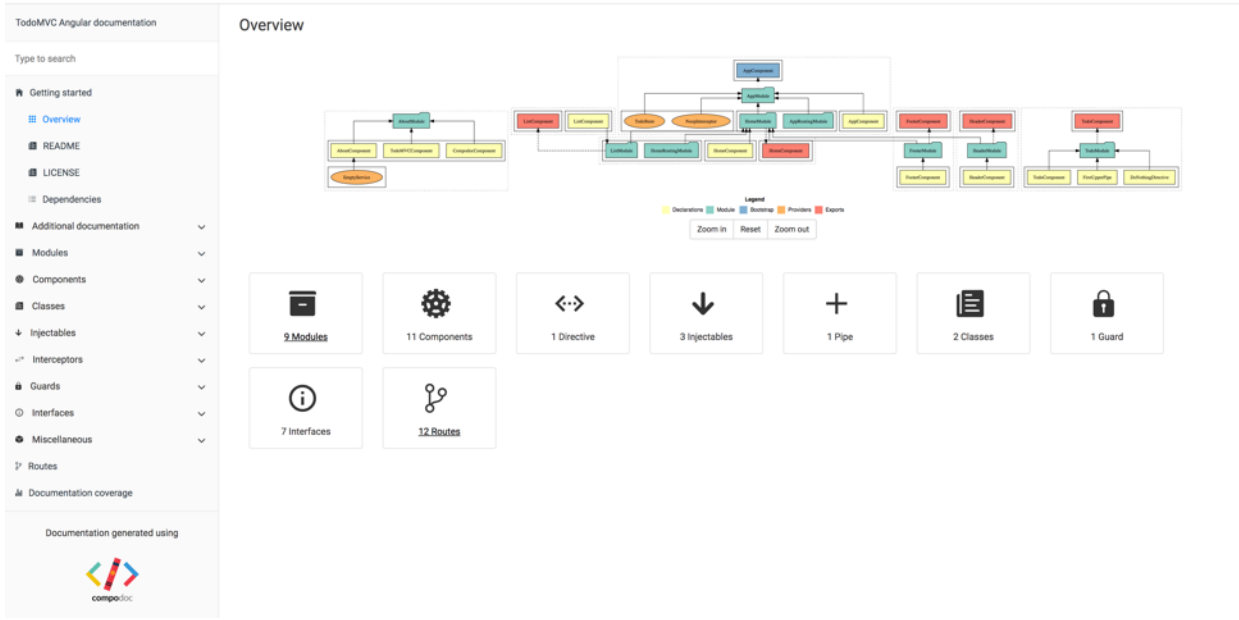


ANGULAR PACKAGING / DISTRIBUTION / DOCUMENTATION

- 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

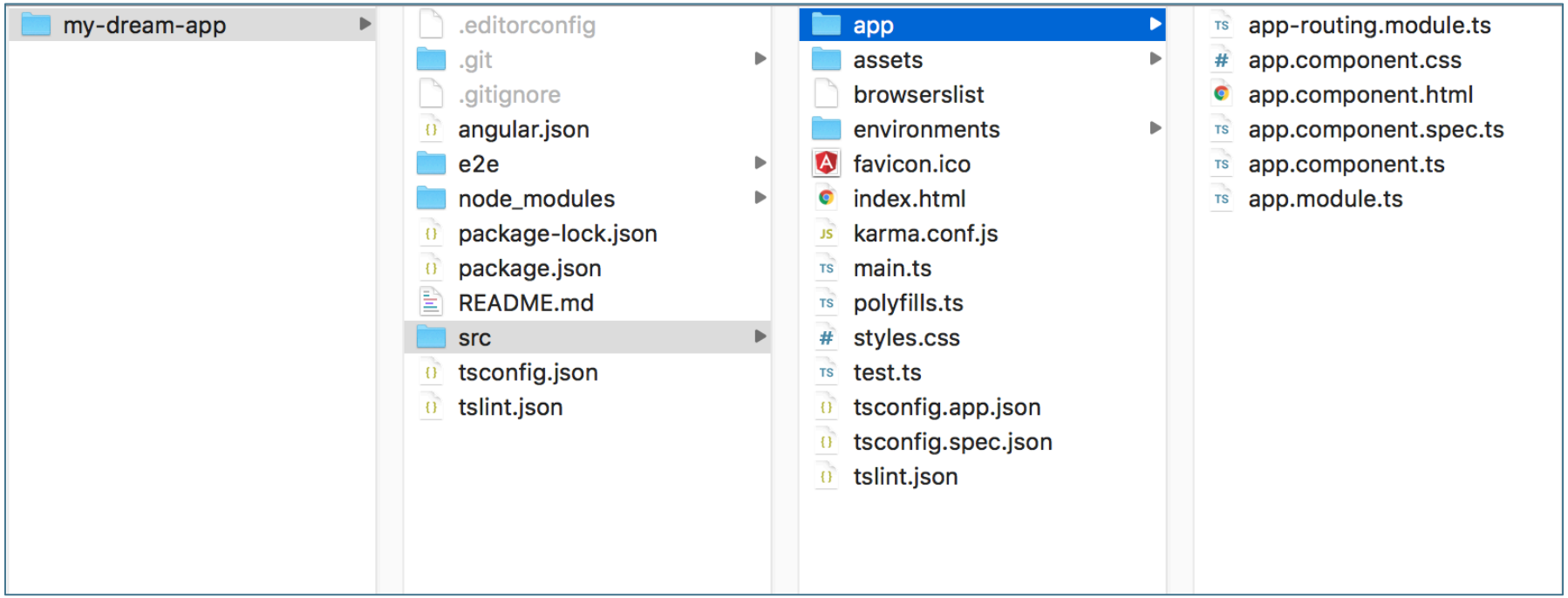


[https://compodoc.github.io/compodoc-demo-todomvc-angular/overview.html]

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

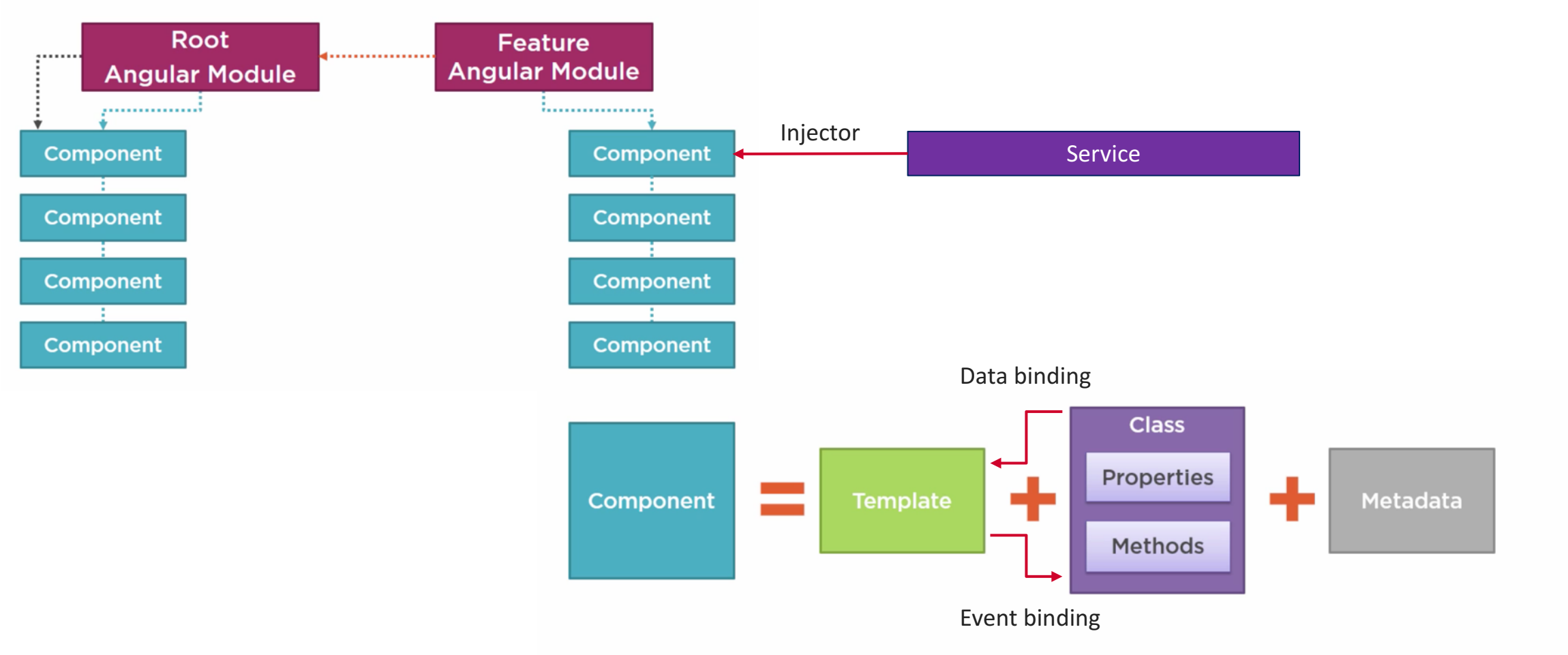
  Every Angular application has at least one component, the *root component* that connects a component hierarchy with the page document object model (DOM). Each component defines a class that contains application data and logic, and is associated with an HTML *template* that defines a view to be displayed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

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

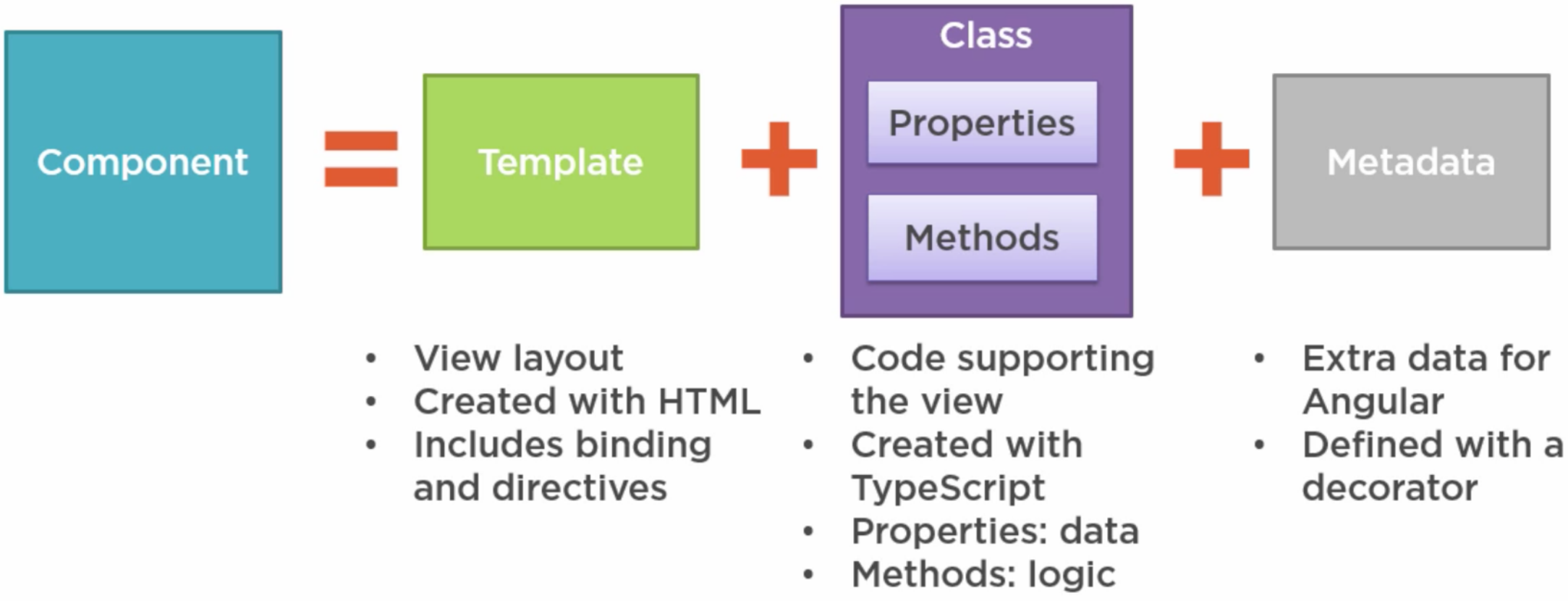
- ## SERVICES (dependency injection)

  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

app.component.ts

```
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';
}
```

Import

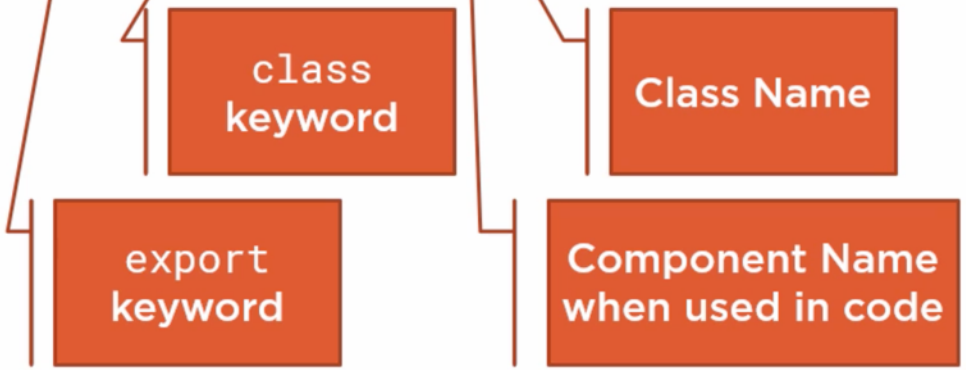
Metadata & Template

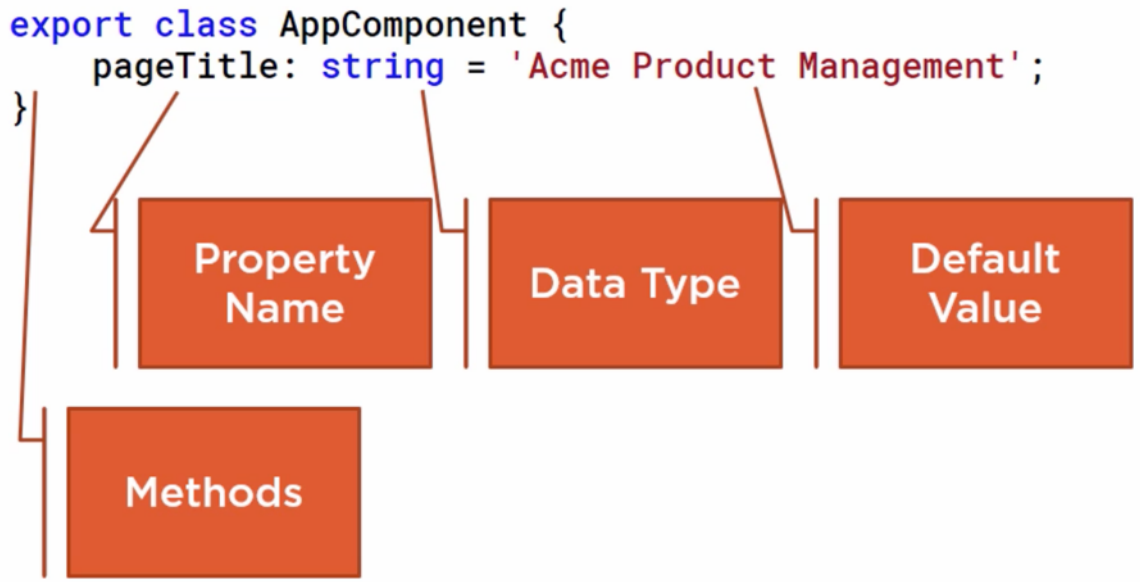
Class

ANGULAR COMPONENT: CLASS

app.component.ts

```
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

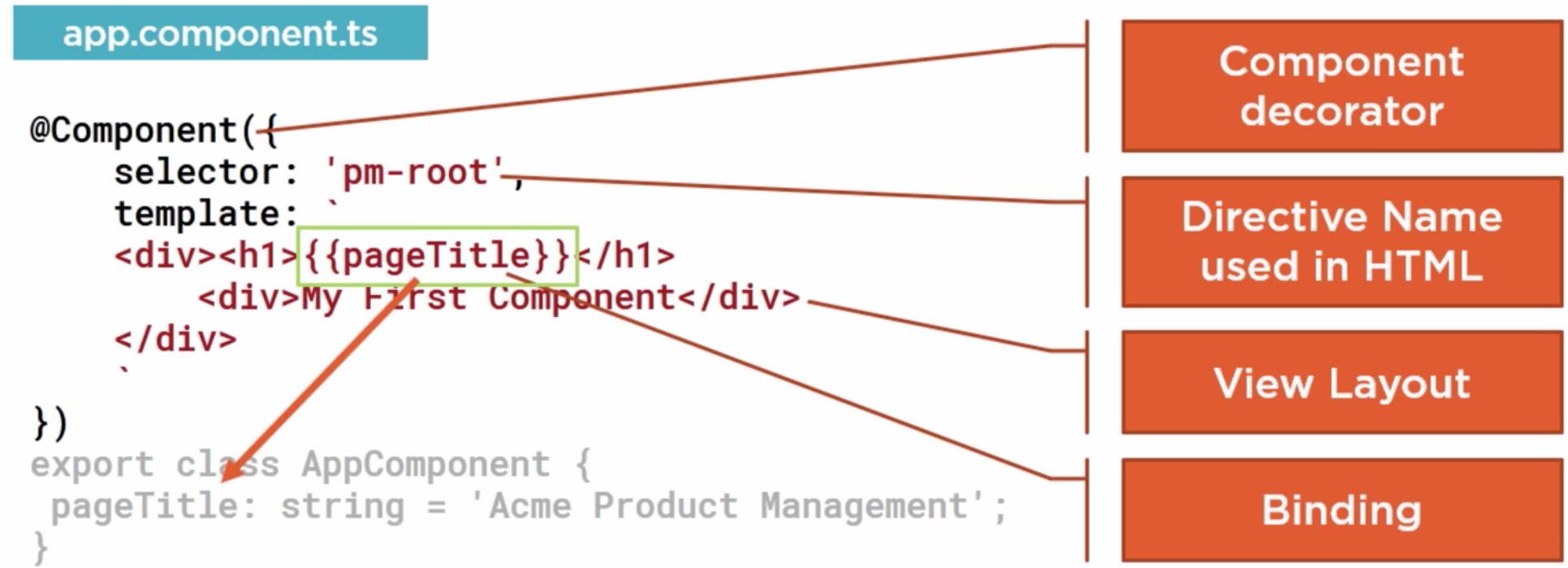
A function that adds metadata to a class, its members, or its method arguments.

Prefixed with an @.

Angular provides built-in decorators.

app.component.ts

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



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

Angular is modular:


```
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';
}
```

- import keyword
- Angular library module name
- Member name

ANGULAR COMPONENT: COMPONENT AS A DIRECTIVE

Create component in CLI:

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

product-list.component.ts

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

app.component.ts

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

1

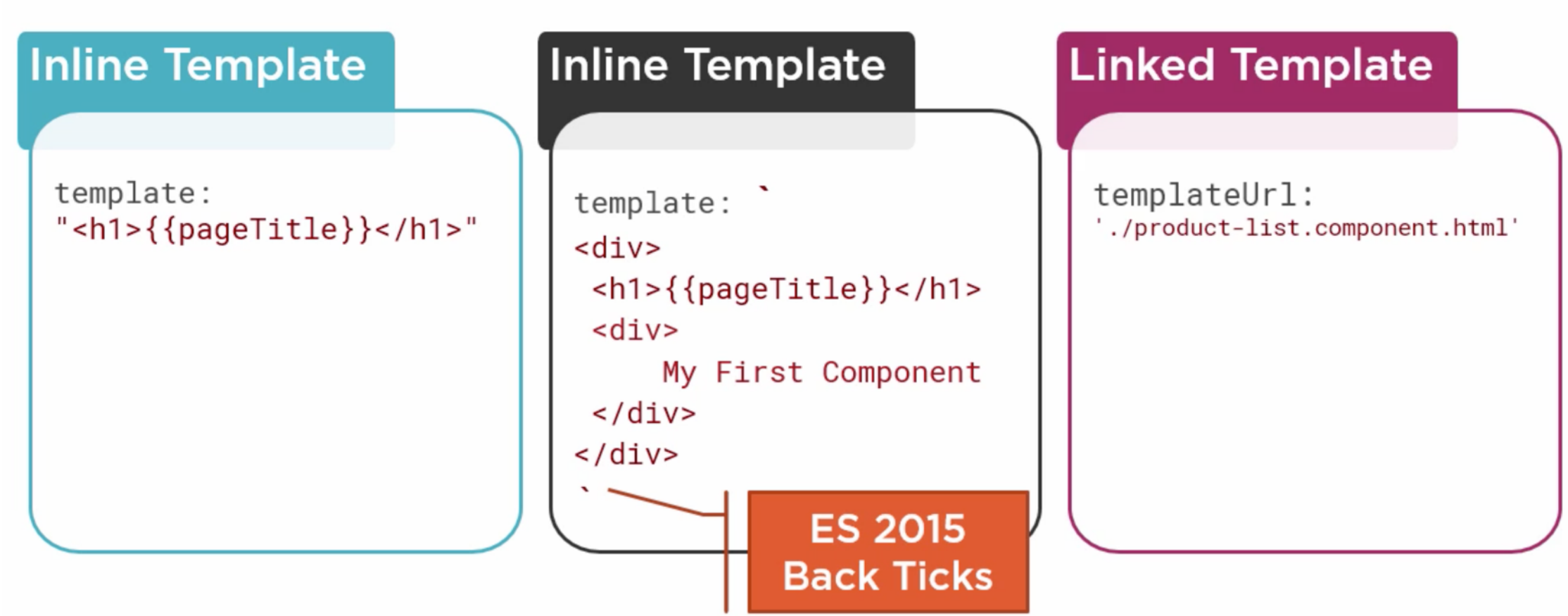
app.module.ts

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

2

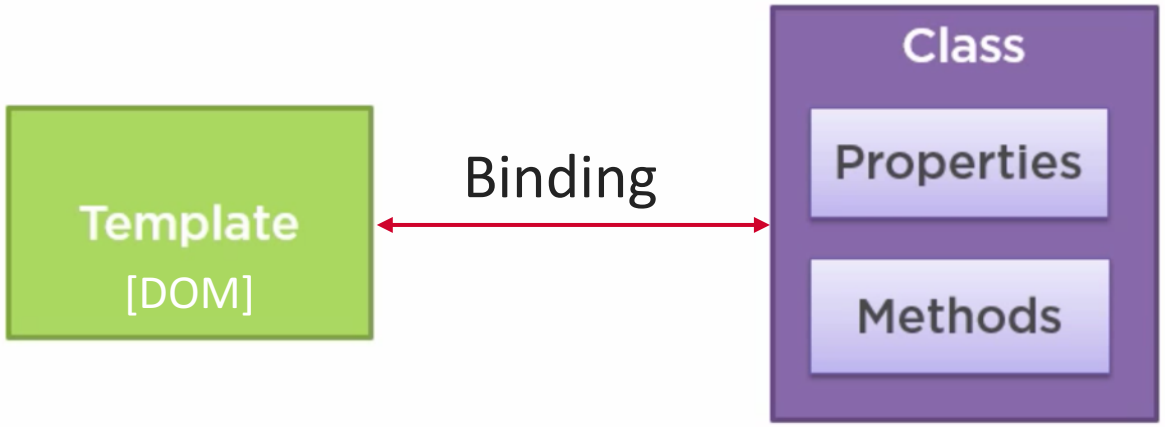
https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/82e5124f8347800de73e0a534b68170b820b7333

ANGULAR COMPONENT: TEMPLATE HTML

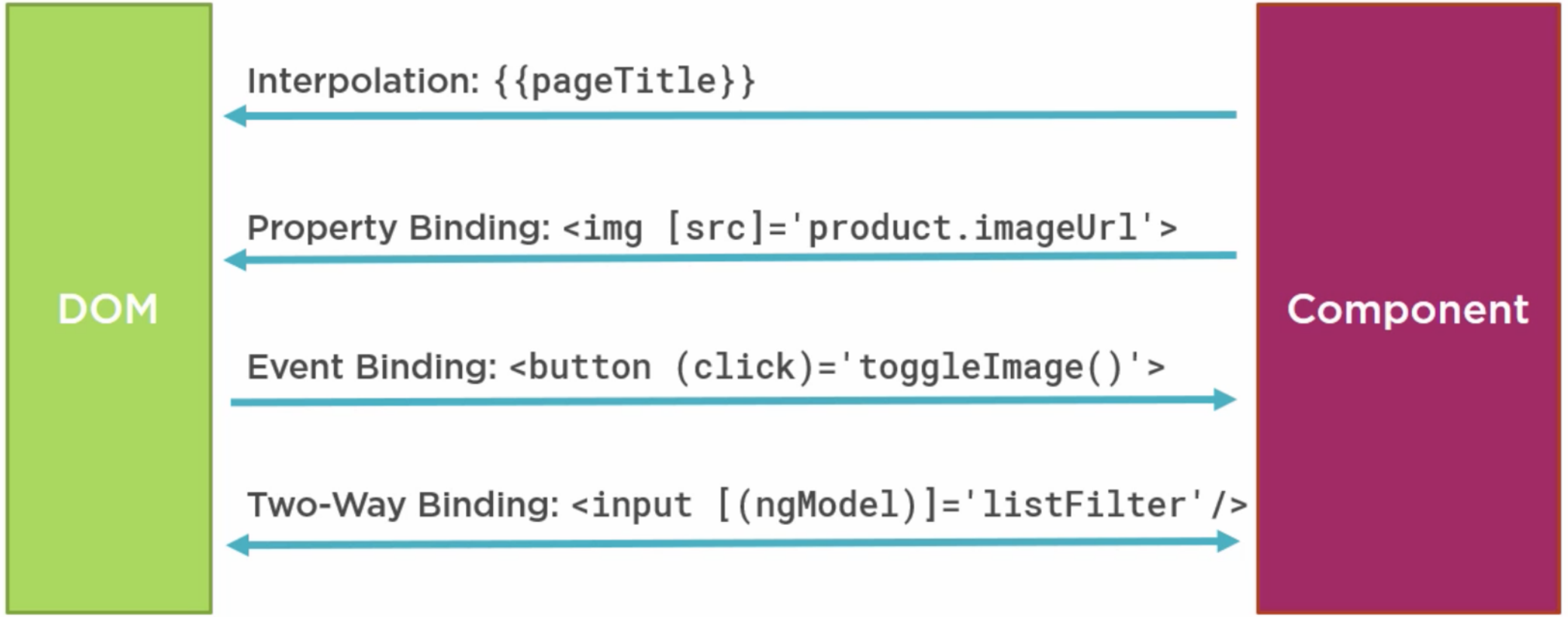


ANGULAR COMPONENT: BINDING

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 {...};
}
```

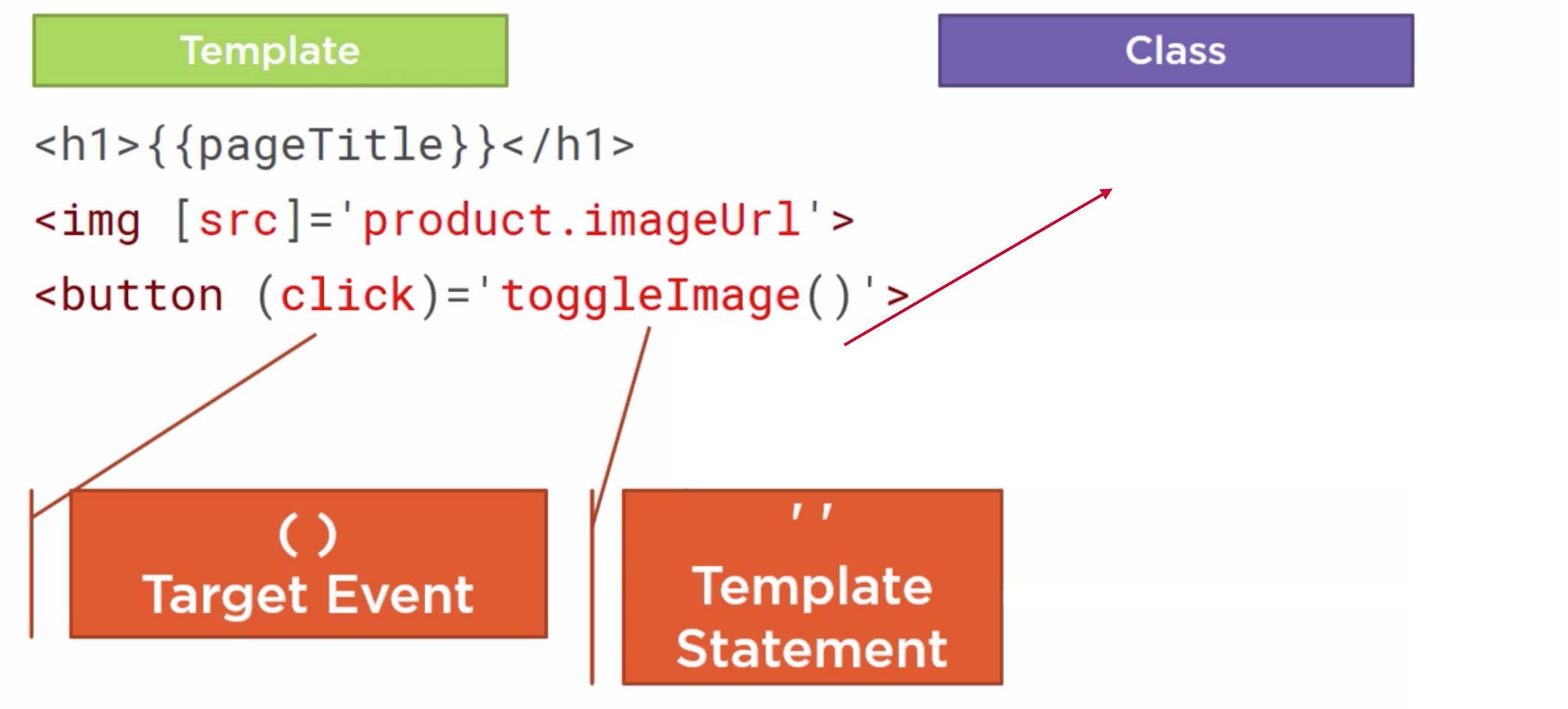

ANGULAR COMPONENT: DATA BINDING – PROPERTY BINDING

```
<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>
```


https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/bed235c3060ded50d654deea8be9e6a842a4ef14

ANGULAR COMPONENT: DATA BINDING – EVENT BINDING



https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/59e9148315e2b508ac398116d33b540416ad3339

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 { }
```

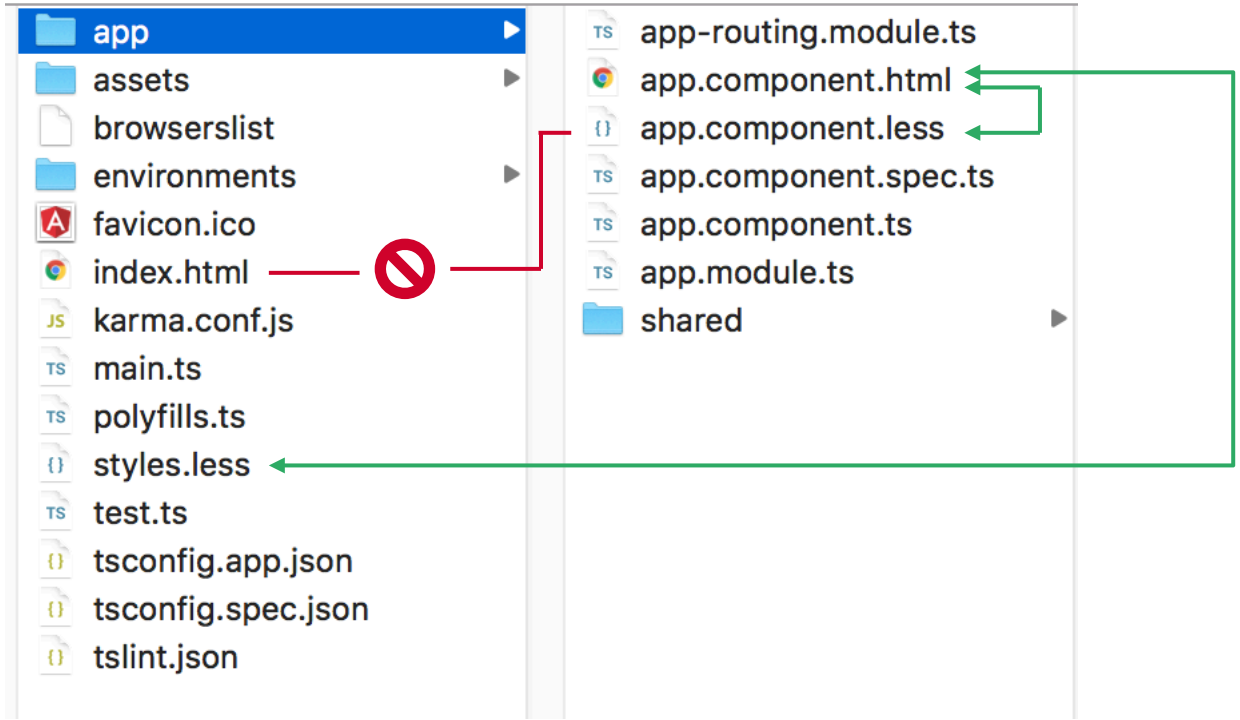
Class

Template

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


https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/874d437ade5464a72b09ee659fd366f7167ff752

ANGULAR COMPONENT: GLOBAL STYLES AND COMPONENTS STYLES



https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/0f34cf3690fbe6fad32689bfff4b4d7d4f0eedcd

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' }}
```


https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/8c7ff0f7b704bc5057dac80244f2f695cb2fc8b4

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 { }
```

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

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

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

Template

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

Pipe

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

  }
```


https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/879becbf45ddbe39e618072ebd4a3cb72d5759ba

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)

https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/879becbf45ddbe39e618072ebd4a3cb72d5759ba

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>
```

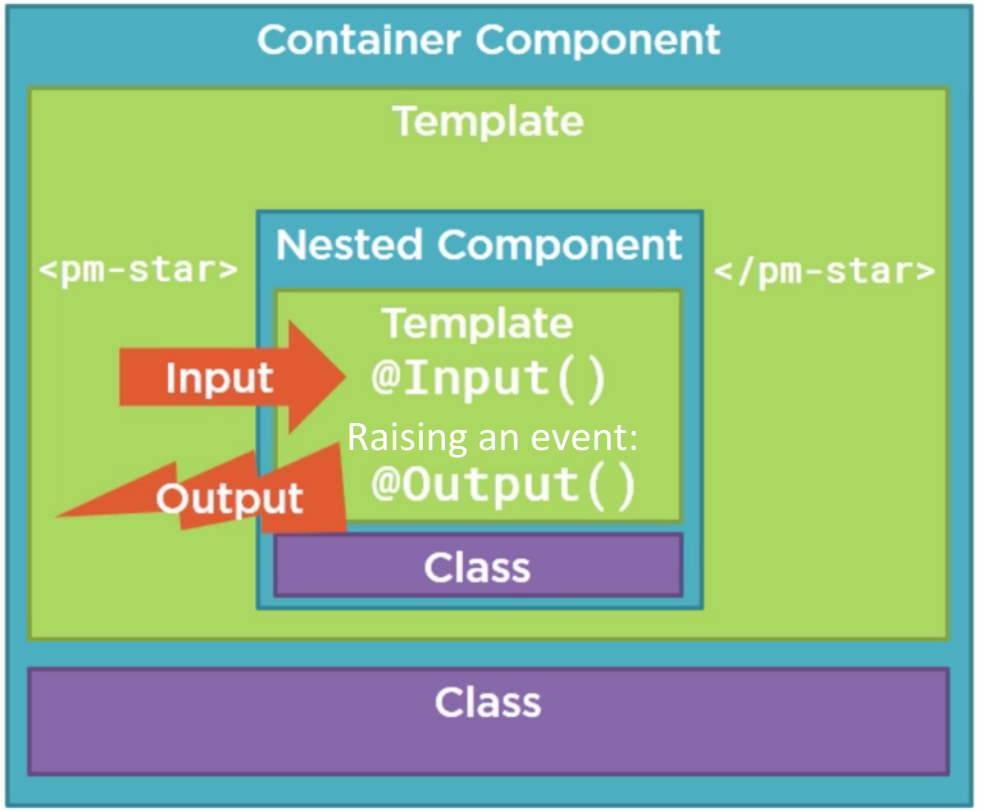

https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/51ed53439a6931ce501d4fea8d459ff46dd41bda

ANGULAR COMPONENT: NESTED COMPONENT

Its template only manages a fragment of a larger view

It has a selector

It optionally communicates with its container



https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/36ca255ba522fac0828a57686aff7f19ec287369

ANGULAR COMPONENT: NESTED COMPONENT

product-list.component.ts

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

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!');
 }
}
```

product-list.component.html

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

star.component.html

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


https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/36ca255ba522fac0828a57686aff7f19ec287369

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 { }
}
```

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!');
 }
}
```

product-list.component.html

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

star.component.html

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


https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/36ca255ba522fac0828a57686aff7f19ec287369

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 { }
}
```

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!');
 }
}
```

product-list.component.html

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

star.component.html

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


https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/36ca255ba522fac0828a57686aff7f19ec287369

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

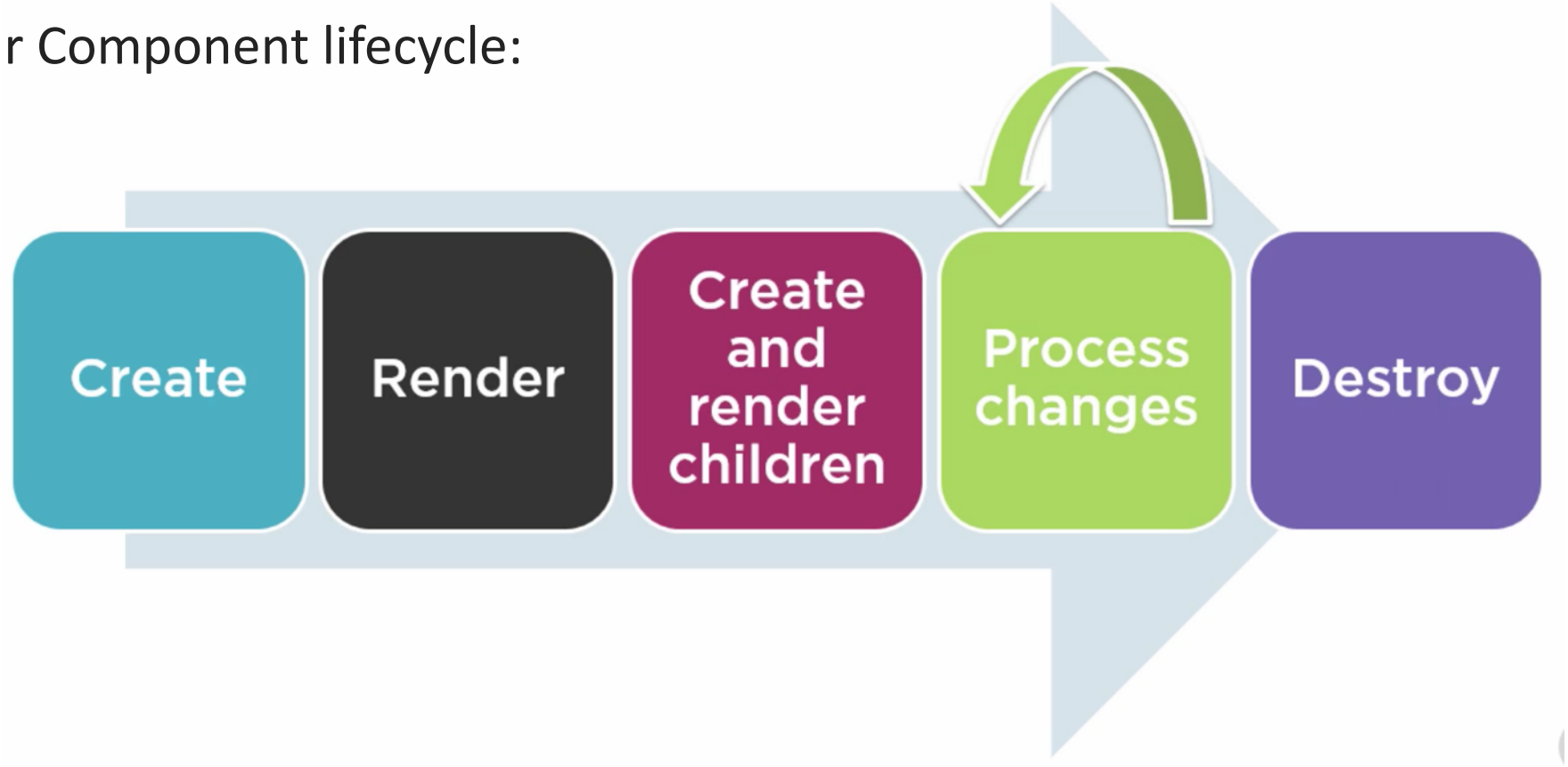
Use event binding to respond to events from the nested component

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

https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/36ca255ba522fac0828a57686aff7f19ec287369

ANGULAR COMPONENT: LIFECYCLE HOOKS

Angular Component lifecycle:



https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/ed1bba425ec8d671aa54366c98505ccab711f632

ANGULAR COMPONENT: LIFECYCLE HOOKS

Angular Component lifecycle hookes (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

https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/ed1bba425ec8d671aa54366c98505ccab711f632

ANGULAR COMPONENT: LIFECYCLE HOOKS

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

  export class ProductListComponent
1                  implements OnInit {
    pageTitle: string = 'Product List';
    showImage: boolean = false;
    listFilter: string = 'cart';
    products: IProduct[] = […];

3   ngOnInit(): void {
          console.log('In OnInit');
    }
  }
```

Import the lifecycle hook interface

Implement the lifecycle hook interface

Write code for the hook method

https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/ed1bba425ec8d671aa54366c98505ccab711f632

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.

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

Create an interface via CLI:

```
ng g i article
```


https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/bb829794ae861de70db104d866c37da583c57a71

ANGULAR STRONG TYPING AND INTERFACES

product.ts

```ts
export interface IProduct {
    productId: number;
    productName: string;
    productCode: string;
    releaseDate: string;
    price: number;
    description: string;
    starRating: number;
    imageUrl: string;
}
```

product-list.component.ts

```ts
import { Component } from '@angular/core';
import { IProduct } from './product';

@Component({
    selector: 'pm-products',
    templateUrl: './product-list.component.html'
})
export class ProductListComponent {
    pageTitle: string = 'Product List';
    imageWidth: number = 50;
    imageMargin: number = 2;
    showImage: boolean = false;
    listFilter: string = 'cart';
    products: IProduct[] = [
        {
            "productId": 2,
            "productName": "Garden Cart",
            "productCode": "GDN-0023",
            "releaseDate": "March 18, 2016",
            "description": "15 gallon capacity rolling garden cart",
            "price": 32.99,
            "starRating": 4.2,
            "imageUrl": "http://openclipart.org/image/300px/svg_to_png/584
        },
        {
```

imageUrl

https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/bb829794ae861de70db104d866c37da583c57a71

ANGULAR: ASYNCHRONOUS CALLS

In angular it involves:

- Services
- HttpClient Module
- Obervables 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 indipendent from any particular component
  - Provide shared data or logic across components)
- It hasn’t templates because it is concerning just with logic or data

https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/812505e99cbf1d480bb71742ac43b7dcb56b4b9a

ANGULAR: SERVICES

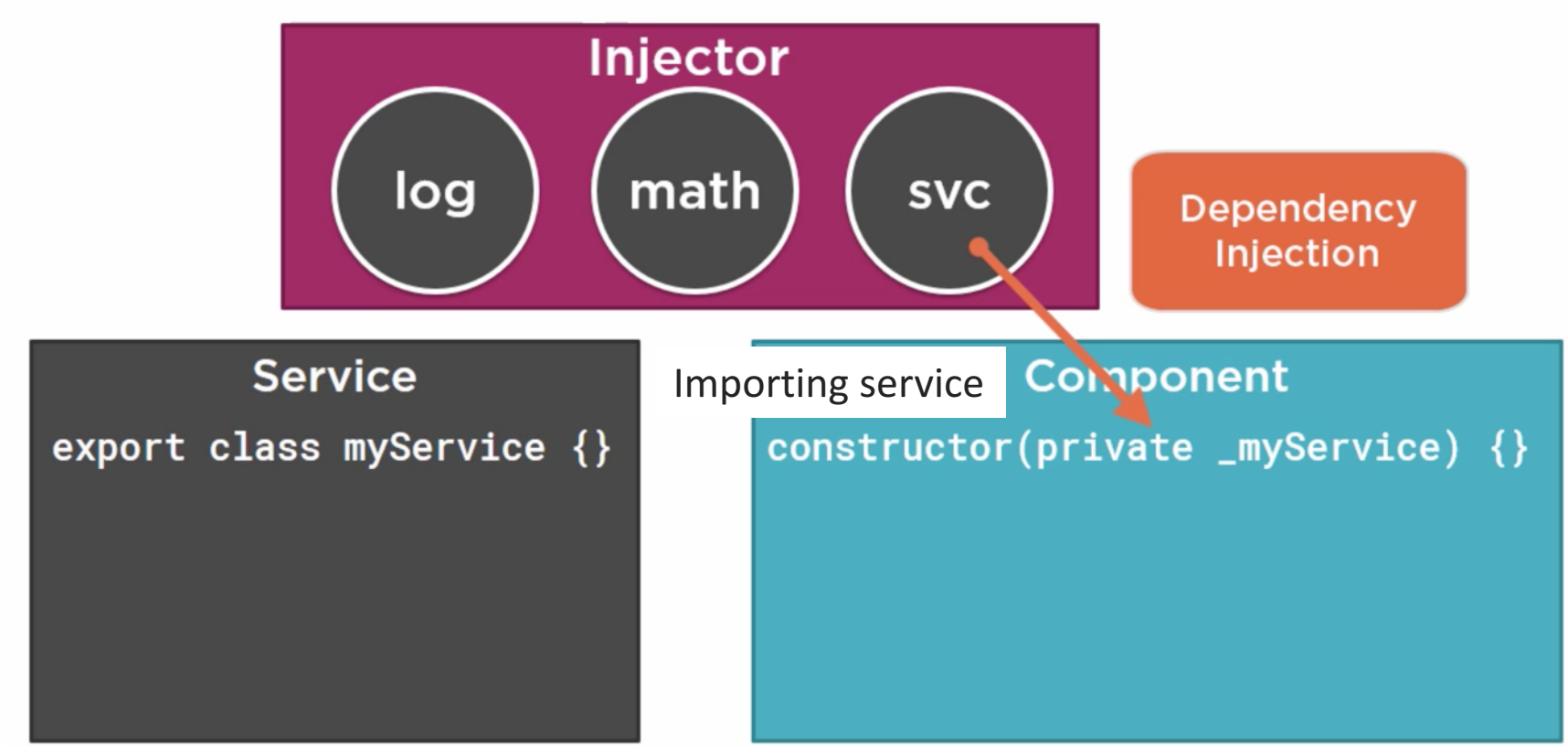
To use a service:

- 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 hierarchic dependency injection)

Root Injector	Component Injector
Service is available throughout the application	Service is available ONLY to that component and its child (nested) components
Recommended for most scenarios	Isolates a service used by only one component
	Provides multiple instances of the service

https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/812505e99cbf1d480bb71742ac43b7dcb56b4b9a

ANGULAR: SERVICES



https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/812505e99cbf1d480bb71742ac43b7dcb56b4b9a

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[] {
  }

}
```

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

PascalCase
Word "Service" appended

https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/812505e99cbf1d480bb71742ac43b7dcb56b4b9a

ANGULAR: SERVICE COMPONENT REGISTRATION

product-list.component.ts

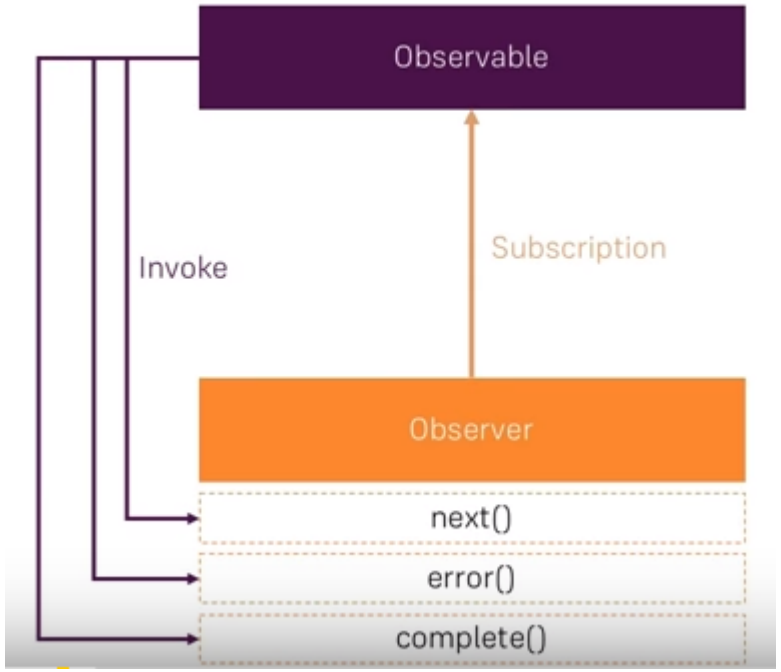
```
...
import { ProductService } from './product.service';

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

 constructor(private productService: ProductService) {
 }

}
```

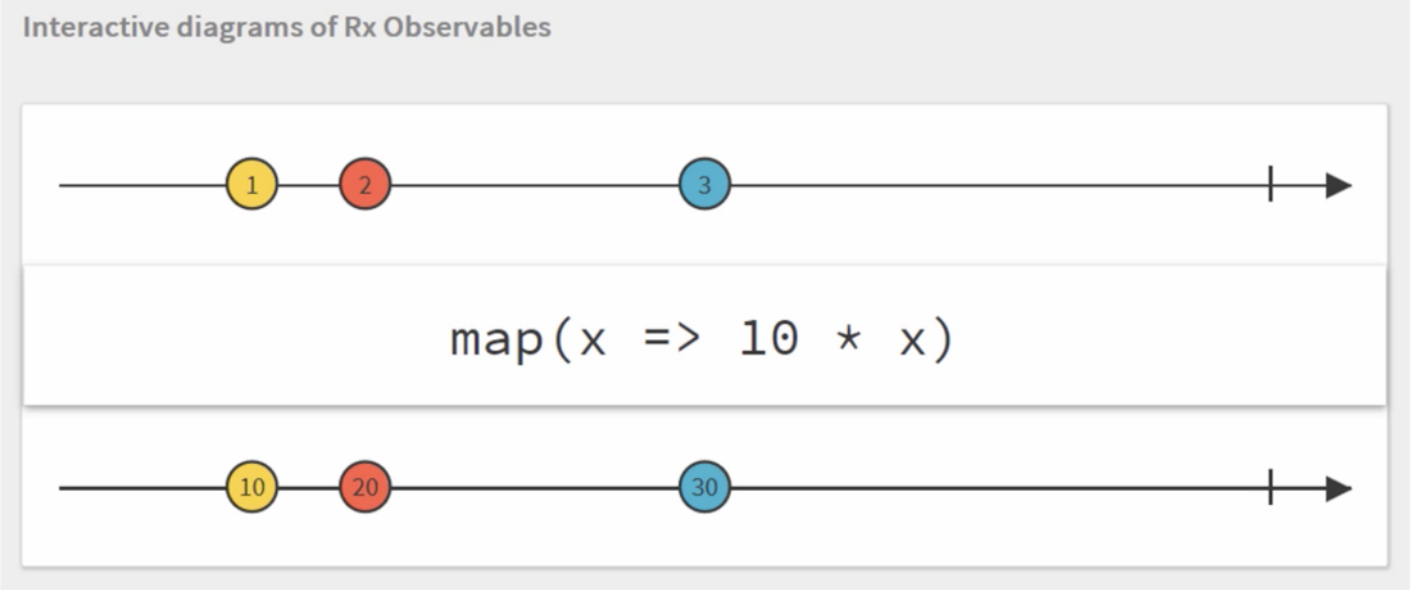

https://github.com/fdelfelice-sopra/an-introduction-to-angular/commit/812505e99cbf1d480bb71742ac43b7dcb56b4b9a

ANGULAR: HTTP CLIENT AND OBSERVABLES

- AngularJS had a $http built in service, based on Promise
- Angular has instead an Observable approch, 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 data)
  - 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





[RxJS Marble diagram example]

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 productUrl = 'www.myWebService.com/api/products';

  constructor(private http: HttpClient) { }

  getProducts(): Observable<IProduct[]> {
   return this.http.get<IProduct[]>(this.productUrl);

  }
}
```

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.productUrl).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