



Angular directives

GFT INTERNAL TRAINING

INNOVATE. TRANSFORM. DELIVER.

Directives in angular

- **components** —> directives with a template
- **attribute directives** —> change the appearance or behavior of an element, component, or another directive
- **structural directives** —> change the DOM layout by adding and removing DOM elements

<https://angular.io/guide/attribute-directives>

<https://angular.io/guide/structural-directives>

attribute directive example ngStyle

app.component.html x

<https://angular-hn7wtw.stack>

```
1 <hello name="{{ name }}"></hello>
2 <p [ngStyle]="{color: 'red' }">
3   Start editing to see some magic happen :)
4 </p>
5
6
```

Hello Angular!

Start editing to see some magic happen :)

attribute directive example ngClass

Description ↗

The CSS classes are updated as follows, depending on the type of the expression evaluation:

- `string` - the CSS classes listed in the string (space delimited) are added,
- `Array` - the CSS classes declared as Array elements are added,
- `Object` - keys are CSS classes that get added when the expression given in the value evaluates to a truthy value, otherwise they are removed.

```
<some-element [ngClass]='first second'>...</some-element>
```

```
<some-element [ngClass]='['first', 'second']'>...</some-element>
```

```
<some-element [ngClass]='{'first': true, 'second': true, 'third': false}'>...</some-element>
```

```
<some-element [ngClass]="stringExp|arrayExp|objExp">...</some-element>
```

```
<some-element [ngClass]='{'class1 class2 class3' : true}'>...</some-element>
```

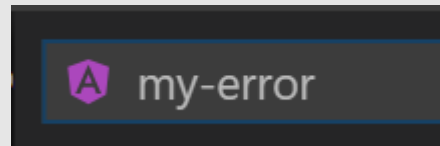
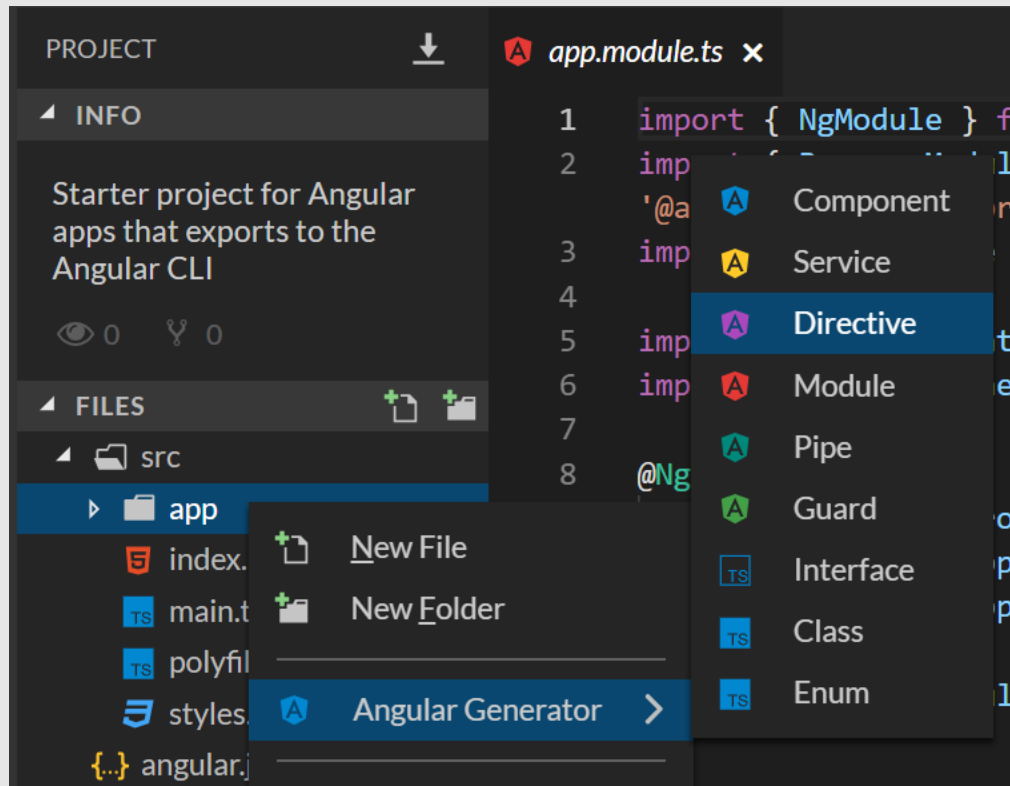
Structural examples

- NgIf
- NgFor
- NgSwitch
- ...

creating custom directives

- always use a prefix when naming custom Angular directives to avoid conflicts with any standard HTML attributes
- we also shouldn't use the ng prefix

example custom attribute directive my-error



example custom attribute directive

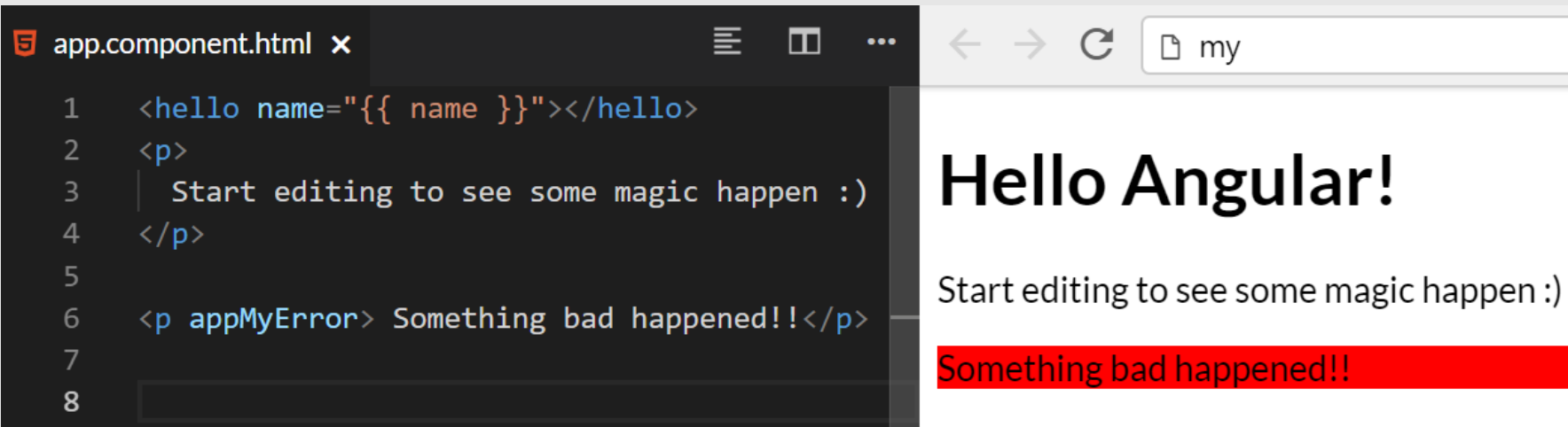
app.module.ts x

```
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { FormsModule } from '@angular/forms';
4
5  import { AppComponent } from './app.component';
6  import { HelloComponent } from './hello.component';
7  import { MyErrorDirective } from './my-error.directive';
8
9  @NgModule({
10     imports:      [ BrowserModule, FormsModule ],
11     declarations: [ AppComponent, HelloComponent, MyErrorDirective ],
12     bootstrap:    [ AppComponent ]
13  })
14  export class AppModule { }
```


example custom attribute directive

```
my-error.directive.ts x
1  import { Directive, ElementRef } from '@angular/core';
2
3  @Directive({
4    selector: '[appMyError]'
5  })
6  export class MyErrorDirective {
7
8    constructor(element: ElementRef) {
9      element.nativeElement.style.background='red';
10    }
11  }
```

example custom attribute directive



The image shows a side-by-side comparison of code and its rendered output. On the left, a code editor window titled 'app.component.html' displays the following HTML code:

```
1 <hello name="{ { name } }"></hello>
2 <p>
3   Start editing to see some magic happen :)
4 </p>
5
6 <p appMyError> Something bad happened!!</p>
7
8
```

On the right, a browser window titled 'my' shows the rendered output. It displays the text 'Hello Angular!' in a large font, followed by 'Start editing to see some magic happen :)'. Below this, the text 'Something bad happened!!' is displayed on a red background, indicating that the custom attribute directive 'appMyError' has been successfully applied and is triggering a visual effect.

Shaping the future of digital business

GFT Internal Technical Training

Eduardo García Ibaseta

eduardo.garcia-ibaseta@gft.com
+34 935 639476

GFT IT Consulting, S.L.

Av. Alcalde Barnils, 69-71

08174 Sant Cugat del Vallès (BARCELONA)