Directive

Une **directive** est une classe angular sans template. D'ailleurs si on examine le code de Angular(**ctrl**+**clique** sur component), on verra que la classe "**component**" hérite de la classe "**directive**".

Les **directives** permettent principalement de lier un comportement à un élément d'une page. Une directive comporte un sélecteur qui indique au framework où elle doit être activé. Lorsque angular voit une directive, il instancie la classe correspondante et donne le contrôle de l'élément à cette classe.

Il y a trois types de directives.

- Les composants. (component)
- Les directives d'attribut. (généralement représenté par des attributs dans des balises html)
- Les directives structurels. (ajoute, retire ou manipule des éléments html. "*ngIf" et "*ngFor" en sont.)

Directive d'attribut

Pour créer une directive, on peut utiliser le terminal en faisant bien attention d'être dans le dossier de notre projet.

```
ng generate directive border-card
# border-card étant le nom que j'ai donné à ma nouvelle directive
```

Angular/cli nous à fait deux choses, il a créé un (ou deux si spec) fichier(s) et modifié "**app.module.ts**" Si on regarde "**app.module.ts**" on verra qu'il a importé notre nouvelle directive et ajouté aux declarations.

Ensuite ouvrons "border-card.directive.ts" on y trouvera:

- Un import de la classe "Directive" depuis le noyau d'angular.
- Un décorateur utilisant "Directive" à la place de "Component".
    - On notera que là aussi il y a une propriété "selector" qui sert de nom mais que celui ci se trouve entre "[]" car ce sera un attribut.

Pour commencer, on va importer depuis le noyau "**ElementRef**" :

```
import { Directive, ElementRef } from '@angular/core';
// il nous permettra de récupérer l'élément auquel la directive fait référence.
```

Puis nous allons indiquer à notre constructor que notre classe récupère cet élément.

```
constructor(private el: ElementRef) { }
```

Ensuite créons 2 méthodes, "setBorder" et "setShadow":

```
private setShadow(x: number, y: number, blur: number, radius: number, color:
string)
{
    // "this.el.nativeElement" permet d'obtenir l'élément html sur lequel est
appliqué notre directive.
    this.el.nativeElement.style.boxShadow = `${x}px ${y}px ${blur}px ${radius}px
${color}`;
}
private setBorder(size: number, color: string)
{
    this.el.nativeElement.style.border = `${size}px solid ${color}`;
}
```

Puis nous allons appeler ces deux méthodes dans le constructeur.

```
this.setShadow(5,5,10,2,"black");
this.setBorder(2, "black");
```

Enfin pour voir son effet s'appliquer on va ajouter notre directive en attribut.

```
<div class="recette" *ngFor="let rec of recetteList" appBorderCard>
```

Les effets de notre directive sont maintenant visible.

EVENT

Mais si nos directive n'étaient capable que de cela, autant faire du css. On va maintenant ajouter des évènements à notre directive. pour cela on va importer depuis le noyau "**HostListener**":

```
import { Directive, ElementRef, HostListener } from '@angular/core';
// Il nous permettra d'écouter les évènements de notre élément html
```

Ajoutons maintenant deux méthodes précédé du décorateur "**@HostListener()**" Celui ci prendra en argument un évènement JS.

```
@HostListener("mouseenter") onMouseEnter(){}
@HostListener("mouseleave") OnMouseLeave(){}
// les noms des méthodes importe peu si ce n'est pour la relecture
```

La première s'activera quand la sourie entrera sur notre élément:

```
this.setBorder(2, "green");
this.setShadow(5,5,20,2,"green");
```

Et la seconde s'activera quand la souris le quittera, on repassera alors aux valeurs par défaut:

```
this.setShadow(5,5,10,2,"black");
this.setBorder(2, "black");
```

Input

Notre directive est déjà pas mauvaise mais si on souhaite la réutiliser ailleurs, impossible de la personaliser. C'est pour cela que nous ajoutons maintenant à notre import "**Input**" :

```
import { Directive, ElementRef, HostListener, Input } from '@angular/core';
// la classe input va nous permettre de récupérer une valeur depuis le template.
```

Grâce à cela nous allons pouvoir ajouter à notre classe une nouvelle propriété annoté du décorateur "**@Input**":

```
@Input() appBorderCard: string|undefined;
// Sans alias, le nom est celui du selecteur de notre directive, autrement on peut
utiliser :
@Input("appBorderCard") borderColor: string|undefined;
// Avec alias, on place le selecteur de notre directive en argument et on déclare
une nouvelle propriété
```

Nous allons changer notre méthode "**onMouseEnter**" pour qu'elle utilise cette valeur si elle est défini :

```
this.setBorder(2, this.borderColor || "green");
this.setShadow(5,5,20,2,this.borderColor || "green");
```

Maintenant, dans notre html, si nous le laissons comme tel, la bordure sera verte, mais si on souhaite changer la couleur, il nous suffit de l'indiquer :

```
<div class="recette" *ngFor="let rec of recetteList" appBorderCard="red">
```

EXERCICE 3

Consigne dans le fichier "**exercice3.md**";

Correction

```
// En haut du fichier :
type Shadow = [number, number, number, number];
// En début de classe :
private initColor: string = "black";
private defaultColor: string = "green";
private initShadow: Shadow = [5,5,10,2];
private defaultShadow: Shadow = [5,5,20,2];
private sizeBorder: number = 2;

// Dans le constructor et onMouseLeave :
this.setBorder(this.sizeBorder, this.initColor);
this.setShadow(...this.initShadow, this.initColor);
// Dans onMouseEnter :
this.setBorder(this.sizeBorder, this.borderColor || this.defaultColor);
this.setShadow(...this.defaultShadow, this.borderColor || this.defaultColor);

// Il serait aussi possible d'ajouter des propriétés pour la taille de la bordure,
cela afin d'avoir toute les propriétés accessibles au début de la classe.
```