# **III-Directives**



Dans l'architecture Angular, les directives sont finalement partout.

Les Components sont des directives à la seule différence qu'ils possèdent une fonctionnalité de templating. En TypeScript, une directive est une classe à laquelle on applique le décorateur @Directive.

Il existe deux sortes de directives :

- Les directives structurelles : Elles ont pour but de modifier le DOM en ajoutant, enlevant ou replaçant un élément du DOM. (exemple: nglf, ngFor)
- Les directives d'attribut: Elles ont pour but de modifier l'apparence ou le comportement d'un élément (exemple: ngStyle)

LES DIRECTIVES (\*nglf et \*ngFor)

Les directives sont des instructions intégrées dans le DOM que vous utiliserez presque systématiquement quand vous créerez des applications Angular.

Dans ce cours, nous parlerons des **directives structurelles.** Ce sont des directives qui, comme leur nom l'indique, modifient la structure du document.

Nous verrons deux d'entre elles (il en existe d'autres) :

- \*ngIf : pour afficher des données de façon conditionnelle
- \*ngFor: pour itérer des données dans un array, par exemple.

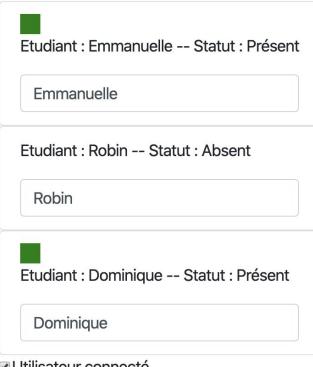
#### \*nglf

Un component auquel on ajoute la directive \*ngIf="condition" ne s'affichera que si la condition est vraie (elle retourne la valeur true où la variable mentionnée est définie et non-nulle), comme un statement if classique.

Pour une démonstration simple, ajoutez une <div> rouge qui ne s'affichera que si la personne est présente :

#### Résultat:

### Liste



#### \*ngFor

Lorsque l'on ajoute la directive \*ngFor="let obj of myArray" à un component, Angular itérera l'array myArray et affichera un component par objet obj.

Créons un tableau directement dans AppComponent (voir image de droite)

Nous avons un array avec trois objets, chaque objet ayant une propriété name et une propriété status.

```
TS app.component.ts > 😭 AppComponent > /\!\!/ presenceTab
import { Component, Input } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
export class AppComponent {
  title = 'mon-premier-projet';
  isAuth = true;
  onClique() {
    alert('Cliqué !');
  presenceTab = [
      name: 'Emmanuelle',
      status: 'Présent'
      name: 'Robin',
      status: 'Absent'
    },
      name: 'Dominique',
      status: 'Présent'
  1;
```

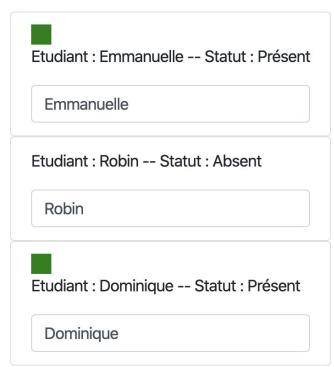
#### > app.component.html

```
src > app > () app.component.html > () div.container > () div.row > () div.container - () () div.container > () div.container > () div.container - () () div.contain
```

Le statement let element of presenceTab, comme dans une boucle for classique, itère pour chaque élément element (nom arbitraire) de l'array presenceTab. Après cette directive, vous pouvez maintenant utiliser l'objet element, dont vous connaissez la forme, à l'intérieur de cette balise HTML. Vous pouvez donc utiliser le property binding, et y passer les propriétés name et status de cet objet.

Remarque: N'oubliez pas l'astérisque devant ces directives, qui signifie à Angular de les traiter comme directives structurelles!

## Résultat : Liste



☑ Utilisateur connecté