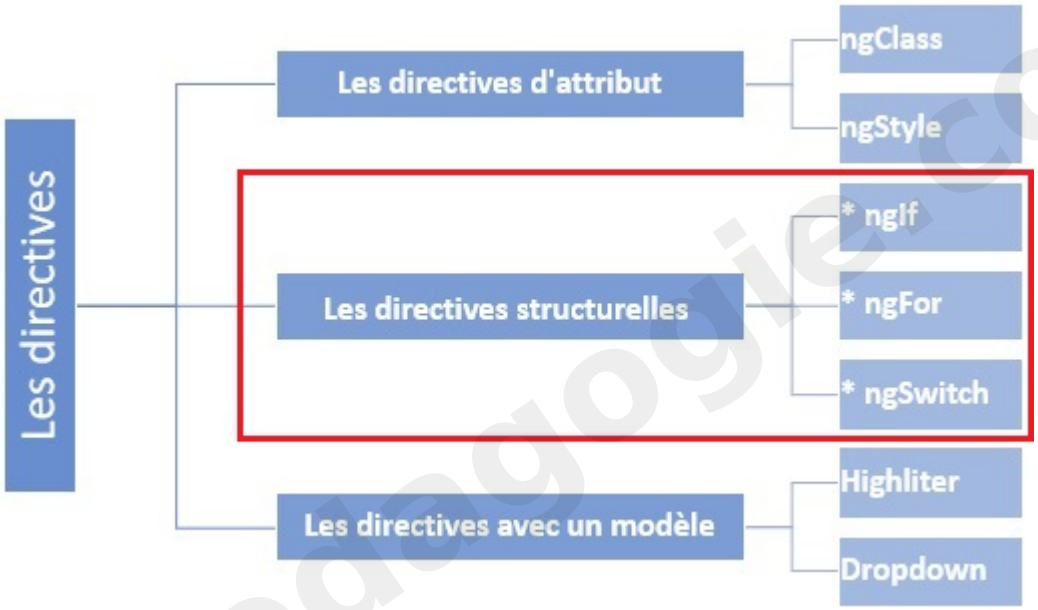
Les directives structurelles

1. **Objectifs**
   - Apprendre à utiliser les directives structurelles.
   - Etre capable d'utiliser *ngIf , *ngFor et *ngSwitch dans Angular

2. **Définition**



   - Les directives structurelles interagissent avec le conteneur de vue actuel et modifient la structure du code DOM / HTML.
   - Elles sont nommées directives structurelles car elles modifient la structure du DOM. Par exemple (`*ngIf`, `*ngFor`, `*ngSwitch`).
   - Les directives structurelles ont été supprimées dans la section des éléments. Ces directives sont utilisées pour manipuler et changer la structure des éléments DOM. Les directives structurelles ont un signe étoile (*) devant la directive.
   - **Directive *ngIf**: La `*ngIf` nous permet d'ajouter/supprimer un élément DOM.
   - **Directive *ngSwitch**: Le `*ngSwitch` nous permettra d'ajouter/supprimer un élément DOM. Elle est identique à l'instruction **switch de C#**.
   - **Directive *ngFor**: La directive `*ngFor` est utilisée pour répéter une partie du modèle HTML une fois par chaque élément d'une liste d'itération (Collection).

apcpedagogie.com

3. ## Directive *ngIf

   - `NgIf` est une directive comportementale qui nous permet de basculer un modèle basé sur une instruction conditionnelle.

   - **Syntaxe**

     - La syntaxe de la directive `ngIf` ressemble à ceci:

     ```
     <div *ngIf="condition">Contenu à rendre lorsque la condition est vraie.</div>
     ```

   - Une version étendue de ceci ressemblerait à ceci:

   ```
   <ng-template [ngIf]="condition">
     <div>Contenu à rendre lorsque la condition est vraie.</div>
   </ng-template>
   ```

4. **Utilisation de `NgIf` d'Angular**

   - La syntaxe de `NgIf` est agréable et simple, nous pouvons simplement le déclarer sur un élément ou un composant et le laisser opérer sa magie.
   - Cette instruction conditionnelle serait évaluée de la même manière que notre JavaScript évaluerait une **`if (condition) {}instruction`**, convertissant la valeur que vous fournissez en une valeur véridique ou fausse et le rendu en conséquence.

5. **Exemples d'utilisation**

   - **Afficher/masquer un objet**
     - Si l'expression associée à la directive est "`false`" alors l'élément et son contenu sont retirés du DOM (ou jamais ajoutés).
     - L'exemple suivant montre comment afficher ou masquer un bouton suivant la valeur de la variable **afficher**

   ```
   //src/app.component.html
   <button *ngIf="afficher">Afficher *ngIf</button>
   //src/app.component.ts
      export class AppComponent {
        afficher = true;
        title = 'lesdirectives';
   ```

apcpedagogie.com

```
}
```

6. **`*ngIf` et les opérateurs logiques**

   - La directive **ngIf** prend également en charge les opérateurs logiques, tout comme une instruction `if` régulière le ferait.
     - ***ngIf** avec l'opérateur **Not(!) (le NON logique)**
       - Pour inverser la valeur booléenne d'une variable, nous pouvons utiliser l'opérateur not (!). Le bloc marqué est alors affiché lorsque la variable est fausse.

```
@Component({
  selector: 'my-app',
  template: `
    <div *ngIf="!admin">
     Bonjour <strong>Mohamed Ali</strong>
    </div>
  `,
})
export class AppComponent  {
  admin = true;
}
```

   - ***ngIf** avec l'opérateur **&& (ET logique)**
     - Pour enchaîner deux ou plusieurs instructions utiliser l'opérateur AND (&&). Tout comme en JavaScript, la déclaration n'est vraie que lorsque toutes les valeurs chaînées sont également vraies.
     - Le bloc marqué n'est donc affiché que lorsque toutes les valeurs sont vraies.

```
@Component({
  selector: 'my-app',
  template: `
    <input type="checkbox" [(ngModel)]="selected"/>
    <input type="checkbox" [(ngModel)]="selected2" />

    <span *ngIf="selected && selected2">
      M'afficher lorsque les deux cases sont cochées :)
    </span>
  `,
})
```

apcpedagogie.com

7. ***ngIf** avec l'opérateur **|| (OU logique)**
    - Tout comme l'opérateur **AND**, l'opérateur **OR** est utilisé pour enchaîner une ou plusieurs instructions.
    - La différence avec l'opérateur **OR** est qu'une seule instruction doit être vraie pour que l'instruction globale soit également vraie.

```
@Component({
  selector: 'angular-app',
  template: `
  <div *ngIf="a == 6 || b == 9">
    La valeur de a est 6 ou la valeur de b est 9
  </div>
  `,
})
export class AppComponent {
  a = 6;
  b = 9;
}
```

8. ## Directive *ngIf Else

    - Le ***ngIf Else** se comporte de manière très similaire à une instruction JavaScript **`if (condition) { } else { }`**.
    - Tout comme nous sommes habitués à d'autres langages de programmation, la directive angulaire **ngIf** nous permet également de déclarer un bloc **else**. Ce bloc est affiché si l'instruction définie dans le bloc principal prut être fausse.

```
<div *ngIf="isLoggedIn; else loggedOut">
 Bienvenue à nouveau.
</div>
<ng-template #loggedOut>
  S'il vous plaît, connectez-vous.
</ng-template>
```

9. ## Directive *ngFor

    - `*ngFor` est une directive de modèle intégrée qui facilite l'itération sur quelque chose comme un tableau ou un objet et crée un modèle pour chaque élément.

apcpedagogie.com

- **Utilisation**

```
<ul>
  <li *ngFor="let utilisateur of utilisateur">{{ utilisateur.name }}</li>
</ul>
```

- `let utilisateur` créer une variable locale qui sera disponible dans le modèle.
- `of utilisateur` signifie que nous allons itérer sur les utilisateurs itérables qui devraient être mis à disposition dans notre composant.
- Le caractère `*` avant `ngFor` crée un modèle parent. Il s'agit d'un raccourci vers la syntaxe suivante: `template="ngFor let item of items"`

10. Directive *ngSwitch

- `NgSwitch` est une directive de modèle Angular 9 intégrée, et elle fonctionne de la même manière que l'instruction **`switch` de JavaScript**. Cette directive permet d'afficher ou de masquer les modèles basés sur l'instruction conditionnelle.
- La directive `NgSwitch` requiert `*ngSwitchCase` et une directive `*ngSwitchDefault` pour obtenir le résultat des multiples possibilités.

- **Syntaxe**

```
<container_element [ngSwitch]="switch_expression">
    <inner_element *ngSwitchCase="match_expresson_1">...</inner_element>
    <inner_element *ngSwitchCase="match_expresson_2">...</inner_element>
    <inner_element *ngSwitchCase="match_expresson_3">...</inner_element>
    <inner_element *ngSwitchDefault>...</element>
</container_element>
```

11. **Utilisation**

Directives	La description
ngSwitch	C'est une directive structurelle, toutes les valeurs possibles définies à l'intérieur. Il obtient la valeur de commutation basée sur l'instruction correspondante parmi plusieurs possibilités.

apcpedagogie.com

Directives	La description
ngSwitchCase	L'instruction ngSwitchCase est déclarée dans la directive ngSwitch avec la valeur spécifique.
ngSwitchDefault	Cette instruction s'exécute lorsque l'expression ne correspond à aucune des valeurs définies.

12. **Exemple**

    - Dans l'exemple ci-dessous, nous essaierons de comprendre l'expression `NgSwitch` de manière simple. Afin de définir la directive `ngSwitch` dans le modèle HTML angulaire. Ajoutez la directive `[ngSwitch]` avec la valeur souhaitée sur le conteneur principal:

```
<ul [ngSwitch]="superhero">

</ul>
```

Sommaire du cours Angular

Source : https://apcpedagogie.com/les-directives-structurelles · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com