Basic Built-in Directives in Angular

Index

Introduction

NgIf

- Directive used when you want to display or hide an element based on a condition.
- Some examples are:

```
<div *ngIf="false"></div>

<div *ngIf="a > b"></div>

<div *ngIf="myFunc()"></div>
```

NgSwitch

```
<div class="container" [ngSwitch]="myVar">
    <div *ngSwitchCase="'A'">Var is A</div>
    <div *ngSwitchCase="'B'">Var is B</div>
    <div *ngSwitchCase="'C'">Var is C</div>
    <div *ngSwitchCase="'A'">Var is A (again)!</div>  // you can do it multiple time
    <div *ngSwitchDefault>Var is something else</div>
</div>
```

NgStyle

```
<div [style.background-color]="'yellow'">
    Uses fixed yellow background
</div>
```

```
<div [ngStyle]="{color: 'white', 'background-clor':'blue'}">
    Uses fixed white text on blue background
</div>
```

NgClass

- The NgClass directive, represented by a ngClass attribute in your HTML template, allows you to dinamically set and change the CSS classes for a given DOM element.

```
.bordered {
    border: 1px dashed black;
    background-color: #eee;
}
```

```
<div [ngClass]="{bordered: false}">This is never  bordered</div>
<div [ngClass]="{bordered: true}">This is always bordered</div>
```

NgFor

The role of this directive is to repeat a given DOM element (or a collection of DOM elements) and pass ana element of the array on each iteration

```
*ngFor="let item of items"
```

Usage example:

```
this.cities = ['Miami', 'Sao Paulo', 'New York'];

<div class="ui list" *ngFor="let c of cities">
    <div class="item">{{c}}</div>
</div>
```

Angular/Typescript code:

```
this.people = [
    {name: 'Anderson', age: 35, city: 'Sao Paulo'},
    {name: 'John', age: 12, city: 'Miami'},
    {name: 'Peter', age: 22, city: 'New York'}
];
```

Template code:

```
<table class="ui celled table">
    <thead>
        <tr>
            <th>Name</th>
            <th>Age</th>
            <th>City</th>
        </tr>
    </thead>
    <tr *ngFor="let p of people">
        <td>{{p.name}}</td>
        <td>{{p.age}}</td>
        <td>{{p.city}}</td>
    </tr>
</table>
```

You can use also nested array:

```
<div *ngFor="let item of peopleByCity">
<h2 class="ui header">{{ item.city }}</h2>

<table class="ui celled table">
    <thead>
        <tr>
            <th>Name</th>
            <th>Age</th>
        </tr>
    </thead>
    <tr *ngFor="let p of item.people">
        <td>{{p.name}}</td>
        <td>{{p.age}}</td>
    </tr>
    <table>
</div>
```

You can also add an index to the list:

```
<div class="ui list" *ngFor="let c of cities; let num = index">
    <div class="item">
        {{num+1}} - {{c}}
    </div>
</div>
```

NgNonBindable

We use ngNonBindable when we want tell Angular not to compile or bind a particular section of our page.

```
<div class='ngNonBindableDemo'>
    <span class='bordered'>{{content}}</span>
    <span class="pre" ngNonBindable>
    &larr; This is what {{content}} rendered </span>
</div>
```

Conclusion

In Angular we can combine these simple directives to create dynamic and powerful apps. However, the directives help us OUTPUT dynamic data, not accept user interaction.