# Directives





# Directives

Outline

- What are directive?
- Classification
- Structural Directives
  - The NgIf Directive
  - The NgSwitch Directive
  - The NgFor Directive
- Attribute Directive
  - The NgClass Directive
  - The NgStyle Directive
- Conclusion
- Q&A

# Directives

- Angular directives **help** in **manipulating the DOM**.
- Angular provides a number of built-in directives, which are **attributes** we add to our HTML elements that give us **dynamic behavior**
- By using directives, we can change the **appearance**, **behavior** or a **layout** of the DOM



97

# Classification of Directives

- Angular directives can be **classified** into the following:
- **Structural** Directives
  - NgIf NgSwitch and NgFor
- **Attribute** Directives
  - NgClass and NgStyle
- **Component** Directives
  - They contain the details of how the component should be processed, instantiated and used at run-time
  - Beyond the scope of this course to discuss



98

# The NgIf Directive

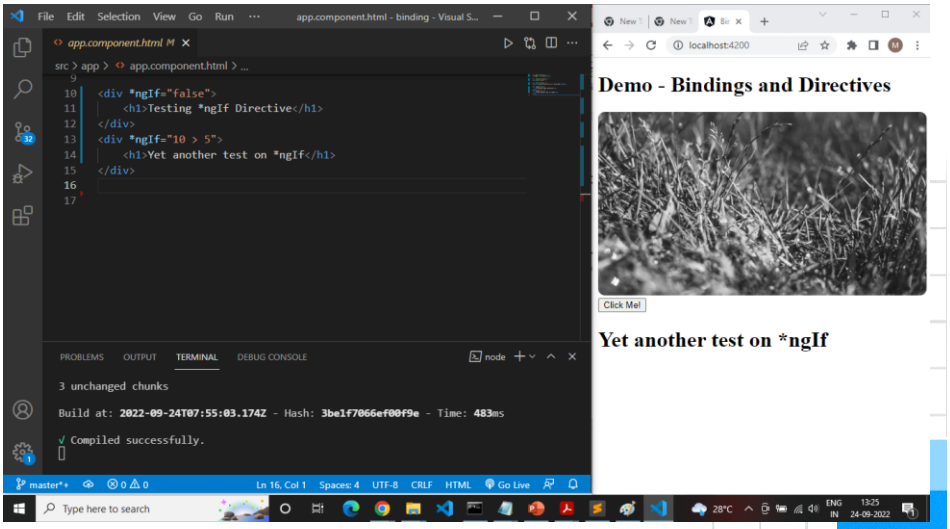
- Used when you want to **display** or **hide** an element based on a **condition**.
- The **condition** is determined by the result of the expression that you pass
- If the result of the expression returns a **false** value, the element will be removed from the DOM.

```
<div *ngIf="false">
    <h1>Testing *ngIf Directive</h1>
</div>
<div *ngIf="10 > 5">
    <h1>Yet another test on *ngIf</h1>
</div>
<div *ngIf="myFunc()">
    <!-- displayed if myFunc returns TRUE -->
</div>
```





# The NgIf Directive



# The NgSwitch Directive

- Sometimes you need to render different elements depending on a given condition.
- When you run into this situation, you could use **ngIf** several times
- For cases like this, Angular introduces the **ngSwitch** directive.
- Similar to the **switch** statement.
- Once we have the result then we can:
  - Describe the known results, using the **ngSwitchCase** directive
  - Handle all the other unknown cases with **ngSwitchDefault**





# The NgSwitch Directive

- Similar to the **switch** statement.
- Code snippet

```
<div class="container" [ngSwitch]="theDow">
    <div *ngSwitchCase=0>Sunday</div>
    <div *ngSwitchCase=1>Monday</div>
    <div *ngSwitchCase=2>Tuesday</div>
    <div *ngSwitchCase=3>Wednesday</div>
    <div *ngSwitchCase=4>Thursday</div>
    <div *ngSwitchCase=5>Friday</div>
    <div *ngSwitchCase=6>Saturday</div>
    <div *ngSwitchDefault>Invalid Day of the Week</div>
</div>
```

- Having the **ngSwitchDefault** element is optional.

# The NgFor Directive

- This directive is to **repeat a given DOM element** (or a collection of DOM elements)
- It passes an element of the array on each iteration.
- Similar to the **for** statement.

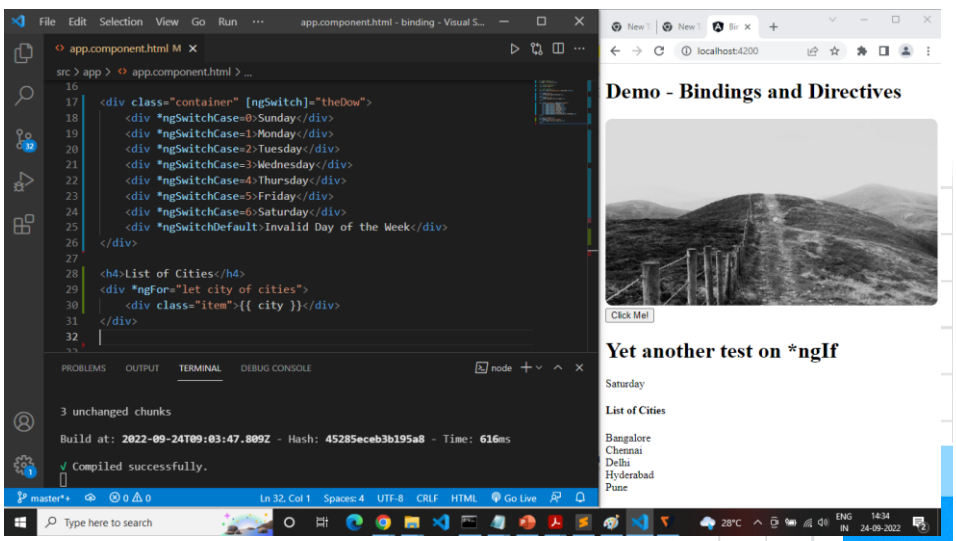
```
export class AppComponent {
  title = 'Bindings and Directives';
  imgUrl = 'https://picsum.photos/640/360';
  theDow = new Date().getDay();
  cities: string[] = ['Bangalore', 'Chennai', 'Delhi', 'Hyderabad',

  handleClick() {
    console.log('Clicked the button...');
  }
}
```

```
<h4>List of Cities</h4>
<div *ngFor="let city of cities">
    <div class="item">{{ city }}</div>
</div>
```





# The NgFor Directive



# The NgClass Directive

- Represented by a **ngClass** attribute in HTML template
- Allows to dynamically set and change the CSS classes for a given DOM element.
- Let's assume we have a CSS class called **bordered** as follows:

```
.bordered {
    border: 2px dashed red;
    background-color: bisque;
}
```



105

# The NgClass Directive

- Look into the two ** ** elements below:

```
<h4>The NgClass Directive</h4>
<div [ngClass]="{bordered: false}">
    This is never bordered
</div>
<div [ngClass]="{bordered: true}">
    This is always bordered
</div>
```

**The NgClass Directive**

This is never bordered

This is always bordered



106

# The NgStyle Directive

- Represented by a **ngStyle** attribute in HTML template
- Allows to dynamically set and change the CSS style for a given DOM element.
- Let's assume we have a method **getColor()** which return's a random color

```
getColor(): string {
  return ( Math.random() > 0.5 ) ? "red" : "green";
}
```





# The NgStyle Directive

- Now we want to set the background color of a DOM element based on the **getColor()** return value.
- Then we can think of using the **ngStyle** directive

```
<h4>The NgStyle Directive</h4>
<p [ngStyle]="{backgroundColor: getColor()}">
    The background of this paragraph will be
    either in RED or GREEN color, based on the
    random selection.
</p>
```

**The NgStyle Directive**

The background of this paragraph will be either in RED or GREEN color, based on the random selection.


Mohamed Mukthar Ahmed

109