Angular 4

Directives and Pipes

Directives

Directives power up the HTML.

It is used to attach behavior to elements in the DOM

There are three kinds of directives in Angular 4

Component Directives

- Directive with a template

Structural Directives

- Directive used to change the DOM Layout

Attribute Directives

- Directive used to change the appearance or behavior of an

Demo

ComponentDirective

Structural Directives

Structural directives modifies the structure or layout of a view by adding, removing or manipulating elements and their children.

'*' in front of the directive name marks the directive as a structural directive.

In angular we have three built-in structural directives

- **ngIf** : ngIf directive inserts or removes an element based on a truthy/falsey condition.
- **ngFor** : ngFor directive is used to iterate an array of items
- **ngSwitch**: ngSwitch directive is used to conditionally swap DOM structure on template based on an expression.

Demo

ngIf-Directive

ngFor-Directive

ngSwitch-Directive

Attribute Directives

Attribute directives alter the appearance or behavior of an existing element.

In templates they look like regular HTML attributes.

Some important angular in-built attribute directives are :

- **ngModel** : Implements two-way data binding, which modifies the behavior of an existing element (typically an <input>) by setting its display value property and responding to change events.
- **ngStyle** : Changes the style based on a result of expression evaluation.
- **ngClass** : Conditionally adds and removes CSS classes on an HTML element based on an expression's evaluation result

Demo

NgStyle-Directive

NgClass-Directive

@HostBinding and @HostListener

Host property decorators are used to bind a host element to a component or directive.

@HostBinding is used to bind the host element property to a directive property. Angular automatically checks host property bindings during change detection. If a binding changes, it will update the host element of the directive.

@HostListener will listen to the event emitted by host element, declared with @HostListener.

Demo

CustomDirective

Pipe

Pipes are used to transform displayed values within a template.

Pipes transform bound properties before they are displayed

A pipe takes in data as input and transforms it to a desired output.

To pass an argument to a pipe in the HTML form, pass it with a colon after the pipe (for multiple arguments, simply append a colon after each argument)

Angular gives us several built-in pipe like lowercase, date, number, decimal, percent, currency, json, slice etc

Angular provides a way to create custom pipes as well.

Built-in Pipes

Format Pipes

- DatePipe
- UpperCasePipe
- LowerCasePipe
- CurrencyPipe
- PercentPipe
- JsonPipe
- TitleCasePipe

Array pipes

- SlicePipe

Async pipes

- AsyncPipe

Custom Pipes

A pipe is a class decorated with pipe metadata

The pipe class implements a transform method

- Takes an input value and an optional array of parameter strings
- Returns the transformed value

```
import {PipeTransform,Pipe} from 'angular2/core';

@Pipe({  name : 'customPipe'})

export class ExponentialStengthPipe implements
PipeTransform{
     transform(value:number,args:string[]):any {

          return  Math.pow(value,parseInt(args[0] || ‘1’, 10));
               }
}
```

Demo

WorkingWithPipes

CustomPipe

Summary

Component Directives is a directive with a template.

Directives can't be bootstrapped

Structural directives change the DOM layout by adding and removing DOM elements.

Attribute directives changes the appearance or behavior of a DOM element.

In order to use *ngModel* in the application components, we need to compulsorily add FormsModule in the Imports array of Application Module class

ngNonBindable tells the Angular not to compile or bind a particular section of a DOM.

Using *ngStyle* directive we can set CSS properties for the DOM element from Angular expressions

Summary

ngClass directive allows us to dynamically set and change the CSS classes for a given DOM element

Pipes are used to transform displayed values within a template.

The pipe class implements a transform method which takes an input value and an optional array of parameter strings which returns the transformed value