Cheatography

Angular 5 Cheat Sheet

by jcron via cheatography.com/52116/cs/14170/

Binding

One Way binding	`<p>{{ descri ption }}< /p>`
Two Way Binding	`<input [(ngMo del )]= " use r.n ame " />`
Property Binding	`<img [src]= " use r.i mag eUR L" />`
Attribute Binding	`<button [attr.a ri a-l abe l]= " sub mit " >Su bmi t</ but ton>`
Class Binding	`<div [class.se lec ted ]="S ele cte d">S ele cte d</ div>`
ngClass	`<div [ngCla ss] ="{' sel ected' : isSele cte d() }">S ele ctable Item</ d iv>`
ngClass	`<div [ngCla ss] ="ge tCl ass es( )">I tem </d iv>`
Style Binding	`<button [style.co lo r ] ="i sSe lected ? 'red' : 'white'">`
ngStyle	`<div [ngSty le] ="{' bac kgr oun d-i mage': 'url(/ pat h/t o/i mg.j pg )' } " >It em< /di v>`
ngStyle	`<div [ngSty le] ="se tSt yle s() ">  {{cust omer.name}} </d iv>`
Component Binding	`<my -ap p-c omp onent [user] ="ac tiv eUs er"> </m y-a pp- com pon ent>`
Event Binding	`<button (click )="s ubm it( )">S ubm it< /bu tto n>`
$event	`<input [value ]="n ame "  (input )="n ame =$e ven t.t arg et.v al ue">`

Angular Lifecycle Hooks (in sequence)

ngOnChanges()	Called before `ngOnInit` and when any input properties change.
ngOnInit()	Called once after the *first* `ngOnCh anges`.
ngDoCheck()	Called during every change detection run, immediately after `ngOnCh anges` and `ngOnInit`.
ngAfterContentInit()	Called once after *first* `ngDoCheck`. *Component only hook*
ngAfterContentChecked()	Called after the `ngAfte rCo nte ntInit` and after every *subsequent* `ngDoCheck`. *Component only hook*
ngAfterViewInit()	Called *once* after the *first* `ngAfte rCo nte ntC he cked`. *Component only hook*
ngAfterViewChecked()	Called after the `ngAfte rVi ewInit` and every *subsequent* `ngAfte rCo nte ntC hecked`. *Component only hook*

Angular Lifecycle Hooks (in sequence) (cont)

ngOnDestroy()	Called just before Angular destroys the directive/component

Pipes

Upper Case	`{{user.name | upperc ase }}`
Title Case	`{{user.name | titlec ase}}`
Lower Case	`{{user.name | lowerc ase }}`
Date	`{{orderDate | date:' me d i um '}}`
Date Format	`{{orderDate | date:' yM M M d' '}}`
Currency	`{{price | curren cy} }`
Percent	`{{taxes | percen t: ' 1.1 -1' }}`
Number	`{{value | number :' 1.1 - 2'} }`
JSON Debug	`{{user | json}}`
Slice	`{{dataArray | slice: 1:3}}`

Structural Directives

ngIf - Removes or recreates a portion of the DOM tree based on the `s howSe ction` expression.	`<se ction *ngIf= " sho wSe cti on ">`
ngFor - Turns the `li` element and its contents into a template, and uses that to instantiate a view for each item in list.	`<li *ngFor ="let item of list">`
ngSwitch - Conditionally swaps the contents of the div by selecting one of the embedded templates based on the current value of `condit ion Exp re ssion`.	`<div [ngSwi tch ]="c ond iti on Swit chD efa ult >...</ ng- tem`

```
import { Common Module } from '@angu lar /co mmon' ;
```

Class Decorators

@Component({...})	Declares class is a component and provides metadata.
@Directive({...})	Declares class is a directive and provides metadata.
@Pipe({...})	Declares class is a pipe and provides metadata.
@Injectable({...})	Declares this class has dependencies that should be injected into the constructor on instance creation.

```
import { Directive, ... } from '@angular/core';
```

@Directive Configuration

selector: '.button:-not(a)`	Specifies a CSS selector that identifies this directive within a template. Supported selectors include element, [attribute], .class, and :not(). Does not support parent-child relationship selectors.
providers: [MyService, { provide: ... }]	List of dependency injection providers for this directive and its children.

```
@Directive({ property1: value1, ... })
```

@Component Configuration

moduleId: module.id	If set, the templateUrl and styleUrl are resolved relative to the component.
viewProviders: [MyService, { provide: ... }]	List of dependency injection providers scoped to this component's view.

@Component Configuration (cont)

template: 'Hello {{name}}'	Inline template
templateUrl: 'my-component.html'	External template URL
styles: ['.primary {color: red}']	Inline template styles
styleUrls: ['my-component.css']	External template styles URL

```
@Component extends @Directive, so the @Directive configuration applies to components.
```