Cheatography

Angular 5 Cheat Sheet

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

Binding

One Way binding	`<p>{{ description }}</p>`
Two Way Binding	`<input [(ngModel)]="user.name" />`
Property Binding	`<img [src]="user.imageURL" />`
Attribute Binding	`<button [attr.aria-label]="submit">Submit</button>`
Class Binding	`<div [class.selected]="Selected">Selected</div>`
ngClass	`<div [ngClass]="{'selected' : isSelected()}">Selectable Item</div>`
ngClass	`<div [ngClass]="getClasses()">Item</div>`
Style Binding	`<button [style.color]="isSelected ? 'red' : 'white'">`
ngStyle	`<div [ngStyle]="{'background-image': 'url(/path/to/img.jpg)'}">Item</div>`
ngStyle	`<div [ngStyle]="setStyles()"> {{customer.name}} </div>`
Component Binding	`<my-app-component [user]="activeUser"></my-app-component>`
Event Binding	`<button (click)="submit()">Submit</button>`
$event	`<input [value]="name" (input)="name=$event.target.value">`

Angular Lifecycle Hooks (in sequence)

ngOnChanges()	Called before `ngOnInit` and when any input properties change.
ngOnInit()	Called once after the *first* `ngOnChanges`.
ngDoCheck()	Called during every change detection run, immediately after `ngOnChanges` and `ngOnInit`.
ngAfterContentInit()	Called once after *first* `ngDoCheck`. *Component only hook*
ngAfterContentChecked()	Called after the `ngAfterContentInit` and after every *subsequent* `ngDoCheck`. *Component only hook*
ngAfterViewInit()	Called *once* after the *first* `ngAfterContentChecked` *Component only hook*

Angular Lifecycle Hooks (in sequence) (cont)

ngAfterViewChecked()	Called after the `ngAfterViewInit` and every *subsequent* `ngAfterContentChecked` *Component only hook*
ngOnDestroy()	Called just before Angular destroys the directive/component

Pipes

Upper Case	`{{user.name | uppercase}}`
Title Case	`{{user.name | titlecase}}`
Lower Case	`{{user.name | lowercase}}`
Date	`{{orderDate | date:'medium'}}`
Date Format	`{{orderDate | date:'yMMMd''}}`
Currency	`{{price | currency}}`
Percent	`{{taxes | percent:'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 `showSection` expression.	`<section *ngIf="showSection">... </section>`
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"> {{item.name}}</li>`
ngSwitch - Conditionally swaps the contents of the div by selecting one of the embedded templates based on the current value of `conditionExpression`.	`<div [ngSwitch]="conditionExpression"> <ng-template [ngSwitchCase]="case1Exp">... </ng-template> <ng-template ngSwitchCase="case2LiteralString"> ...</ng-template> <ng-template ngSwitchDefault>...</ng-template> </div>`

```
import { CommonModule } from '@angular/common';
```

Cheatography

Angular 5 Cheat Sheet

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

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.

@Component Configuration (cont)

viewProviders: [MyService, { provide: ... }]	List of dependency injection providers scoped to this component's view.
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.
```