

Binding		Angular Lifecycle Hooks (in sequence) (cont)	
One Way binding	<code><p>{{ description }}</p></code>	ngOnDestroy()	Called just before Angular destroys the directive/component
Two Way Binding	<code><input [(ngModel)] = "userName" /></code>	Pipes	
Property Binding	<code></code>	Upper Case	<code>{{user.name uppercase}}</code>
Attribute Binding	<code><button [attribute] = "submit" /></code>	Title Case	<code>{{user.name titlecase}}</code>
Class Binding	<code><div [class.selected] = "selected">Selected</div></code>	Lower Case	<code>{{user.name lowercase}}</code>
ngClass	<code><div [ngClass] = '{selected: isSelected}'>SelectableItem</div></code>	Date	<code>{{orderDate date: 'medium'}}</code>
ngClass	<code><div [ngClass] = 'getClasses()'>Item</div></code>	Date Format	<code>{{orderDate date: 'yMMMd'}}</code>
Style Binding	<code><button [style.color] = 'isSelected ? 'red' : 'white'></code>	Currency	<code>{{price currency}}</code>
ngStyle	<code><div [ngStyle] = '{background-image: url(/path/to/image.jpg)}'>Item</div></code>	Percent	<code>{{taxes percent: '1.1 -1'}}</code>
ngStyle	<code><div [ngStyle] = 'setStyles()'></code>	Number	<code>{{value number: '1.1 -2'}}</code>
Component Binding	<code><my-app-component [user] = 'activeUser'></code>	JSON Debug	<code>{{user json}}</code>
Event Binding	<code><button (click) = 'submit()'>Submit</button></code>	Slice	<code>{{dataArray slice: 1:3}}</code>
\$event	<code><input [value] = 'name' (input) = 'name'></code>	Structural Directives	
Angular Lifecycle Hooks (in sequence)		ngIf - Removes or recreates a portion of the DOM tree based on the expression.	<code><section *ngIf = "showSection"></code>
ngOnChanges()	Called before ngOnInit and when any input properties change.	ngFor - Turns the list element and its contents into a template, and uses that to instantiate a view for each item in list.	<code><li *ngFor = "let item of list"></code>
ngOnInit()	Called once after the first ngOnChanges.	ngSwitch - Conditionally swaps the contents of the div by selecting one of the embedded templates based on the current value of condition Expression.	<code><div [ngSwitch] = 'condition'>...</div></code>
ngDoCheck()	Called during every change detection run, immediately after ngOnChanges and ngOnInit.	<code>import { CommonModule } from '@angular/common';</code>	
ngAfterContentInit()	Called once after first ngDoCheck. <i>Component only hook</i>		
ngAfterContentChecked()	Called after the ngAfterContentInit and after every subsequent ngDoCheck. <i>Component only hook</i>		
ngAfterViewInit()	Called once after the first ngAfterContentInit and checked. <i>Component only hook</i>		
ngAfterViewChecked()	Called after the ngAfterViewInit and every subsequent ngAfterContentChecked. <i>Component only hook</i>		

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({ selector: 'button', providers: [MyService] })
```

@Component Configuration

moduleId: module.id	If set, the templateUrl and styleUrls 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.



By **jcron**
cheatography.com/jcron/

Not published yet.
Last updated 3rd January, 2018.
Page 2 of 2.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>