Improving Performance of the UI

David Mann

@MannD | www.HeirloomSoftware.com

Topics

- Async Loading
- Using Pipes correctly
- TrackBy for ngFor
- Caching & Memoizing
- Lifecycle Hooks (part 1)
- Lifecycle Hooks (part 2)

```
<ng-container
  *ngIf="myObservable | async;
    else loader; let cust">
      CustomerID: {{cust.id}}
</ng-container>


<ng-template #loader>
  <loading-component>
  </loading-component>
</ng-template>
```

Async Loading

Pipe Performance

Pure	Impure
Processed only on *pure* changes: • primitive input change • object reference ***Fast*** **Default** ***Single-instance***	

Pipe Performance

Pure	Impure
Processed only on *pure* changes: • primitive input change • object reference	Processed on every change • mouse move • keystr
Fast	Potentially slow!
Default	
Single-instance	

```
@Pipe({
  name: 'myImpurePipe',
  pure: false
})
```

Pipes

All built-in Pipes are *pure*, except:

Splice

JSON

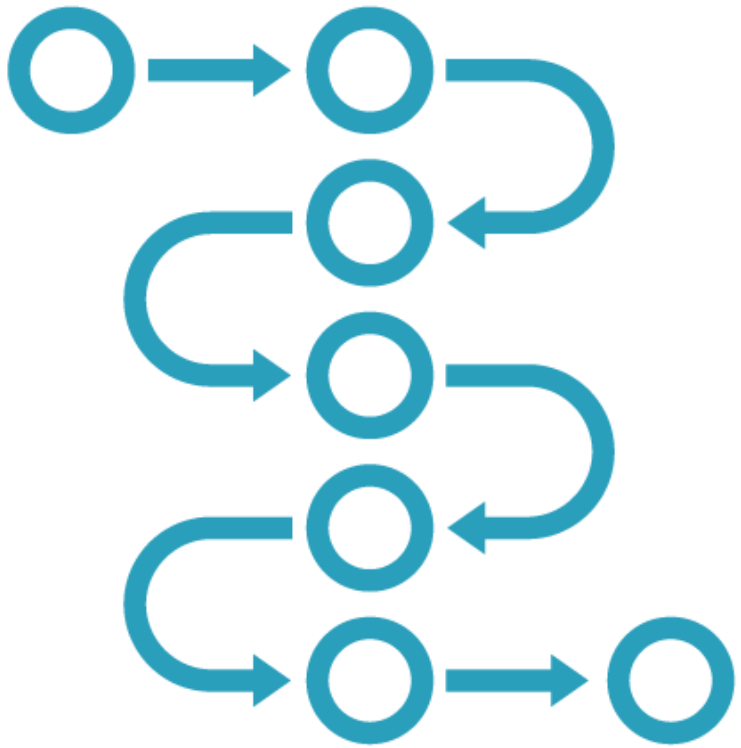
Async

Custom Pipe

Impure?

Shouldn’t do a lot

Iterating Collections

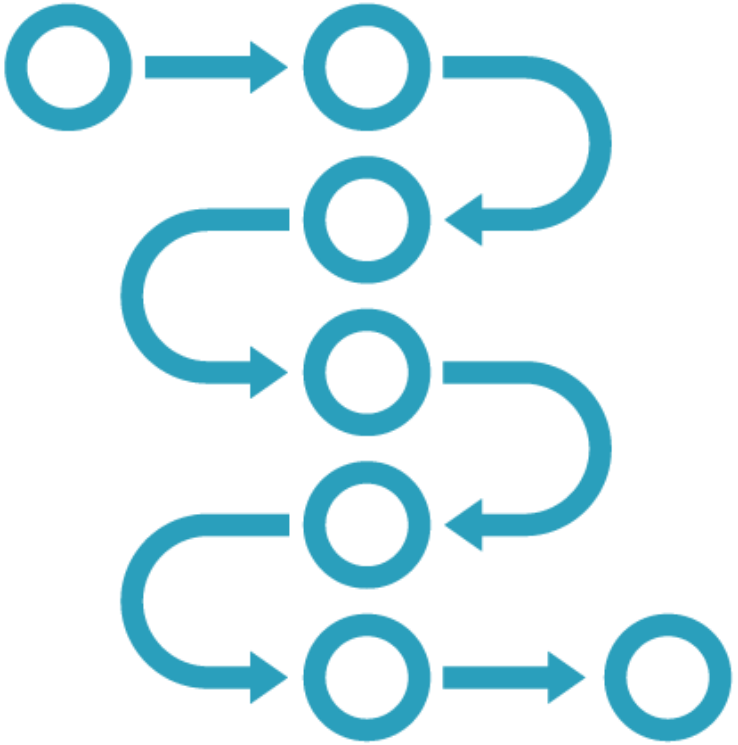


ngFor

```
list: any[] = [
  {name: 'one', id: 1},
  {name: 'two', id: 2},
  {name: 'three', id: 3},
  {name: 'four', id: 4},
  {name: 'five', id: 5}
 ];
```

```
updateList() {
  this.list = [
    {name: 'one', id: 1},
    {name: 'three', id: 3},
    {name: 'five', id: 5},
    {name: 'seven', id: 7},
    {name: 'nine', id: 9}];
}
```

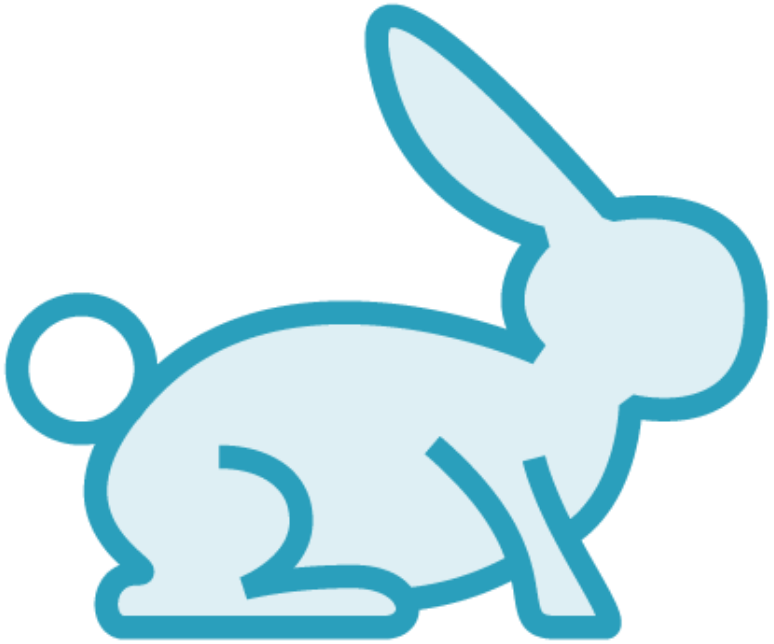
Iterating Collections



ngFor

trackBy

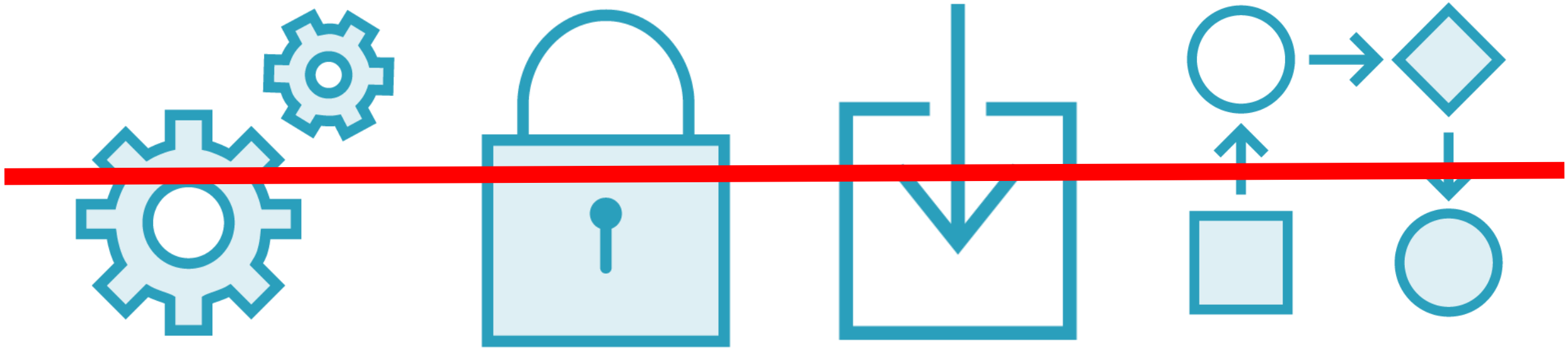
```
<div *ngFor="let item of list; trackBy: trackByFunc">

trackByFunc(idx: number, itm: any) {
  return itm.id;
}
```



Code that doesn’t run

Caching

Caching Adds Complexity



Cache Management

Security

Storage

Extra Logic

Memoizing

Lifecycle Hooks

1x		

Lifecycle Hooks

constructor 1x	**OnChanges**	

Lifecycle Hooks

constructor	OnChanges	OnInit
1x		1x

Lifecycle Hooks

constructor 1x	OnChanges	OnInit 1x
DoCheck		

Lifecycle Hooks

constructor 1x	OnChanges	OnInit 1x
DoCheck	AfterContentInit	

Lifecycle Hooks

constructor 1x	OnChanges	OnInit 1x
DoCheck	AfterContentInit 1x	

Lifecycle Hooks

constructor 1x	OnChanges	OnInit 1x
DoCheck	AfterContentInit 1x	AfterContentChecked

Lifecycle Hooks

constructor 1x	OnChanges	OnInit 1x
DoCheck	AfterContentInit 1x	AfterContentChecked
AfterViewInit 1x		

Lifecycle Hooks

constructor 1x	OnChanges	OnInit 1x
DoCheck	AfterContentInit 1x	AfterContentChecked
AfterViewInit 1x	AfterViewChecked	

Lifecycle Hooks

constructor 1x	OnChanges	OnInit 1x
DoCheck	AfterContentInit 1x	AfterContentChecked
AfterViewInit 1x	AfterViewChecked	OnDestroy 1x

Lifecycle Hook Playground

Lifecycle Hooks

(Most recent events on top) Clear Log

ngAfterContentChecked @ 19:18:50.79
ngDoCheck @ 19:18:50.79
ngAfterViewChecked @ 19:18:49.883
ngAfterContentChecked @ 19:18:49.881
ngDoCheck @ 19:18:49.881
ngAfterViewChecked @ 19:18:49.665
ngAfterContentChecked @ 19:18:49.659
ngDoCheck @ 19:18:49.659
ngAfterViewChecked @ 19:18:49.644
ngAfterViewInit @ 19:18:49.644
ngAfterContentChecked @ 19:18:49.634
ngAfterContentInit @ 19:18:49.634
ngDoCheck @ 19:18:49.634
ngOnInit @ 19:18:49.634
ngOnChanges @ 19:18:49.634
ctor @ 19:18:49.628

Change Bound Value

Current Value: 0

Inject Template

Change Local Value

Current Value: 0

Move mouse around in here and watch hooks fly...

constructor 19:18:49:628	ngOnChanges 19:18:49:634	ngOnInit 19:18:49:634
ngDoCheck 19:18:50:079	ngAfterContentInit 19:18:49:634	ngAfterContentChecked 19:18:50:079
ngAfterViewInit 19:18:49:644	ngAfterViewChecked 19:18:49:883	ngOnDestroy 19:18:49:627

```
ngOnInit(): void {
  this.ngOnInit_LastRun  = new Date();
  this.logEntry('ngOnInit', this.ngOnInit_LastRun);
}

export class HooksWidgetComponent implements IWidget,
 OnChanges, OnInit, DoCheck, AfterContentInit, AfterContentChecked,
 AfterViewInit, AfterViewChecked, OnDestroy {
```

Hook Events

Interfaces

Lifecycle Hooks

constructor	OnChanges	OnInit
DoCheck	AfterContentInit	AfterContentChecked
AfterViewInit	AfterViewChecked	OnDestroy

Constructor

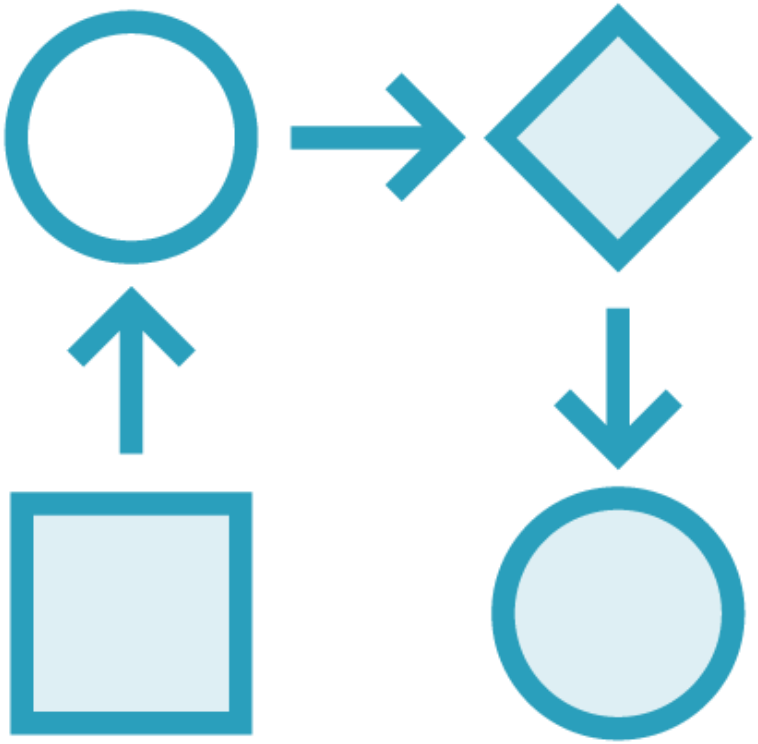
Lean

Easily Testable

Properties = declaration value

@ViewChild/@ContentChild *undefined*

OnChanges



```
▼ boundVal: SimpleChange
    currentValue: 1
    firstChange: false
    previousValue: 0
```

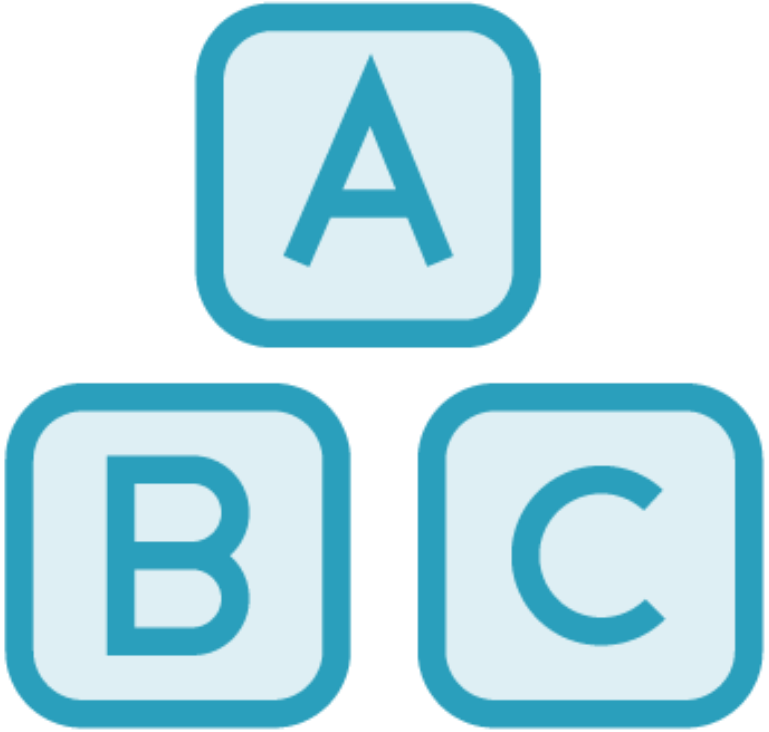
@Input() property changes

SimpleChanges

Input properties available

@ViewChild/@ContentChild set – sort of

OnInit



Initialize instances

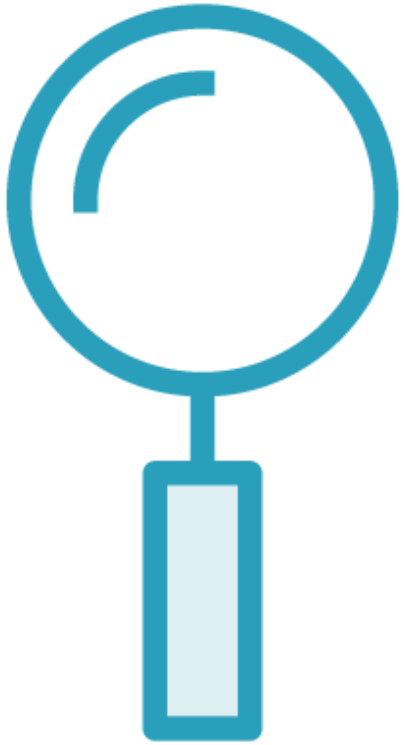
@Input() properties

@ViewChild/@ContentChild

- still not ready

Pre-Directive (in Components)

DoCheck

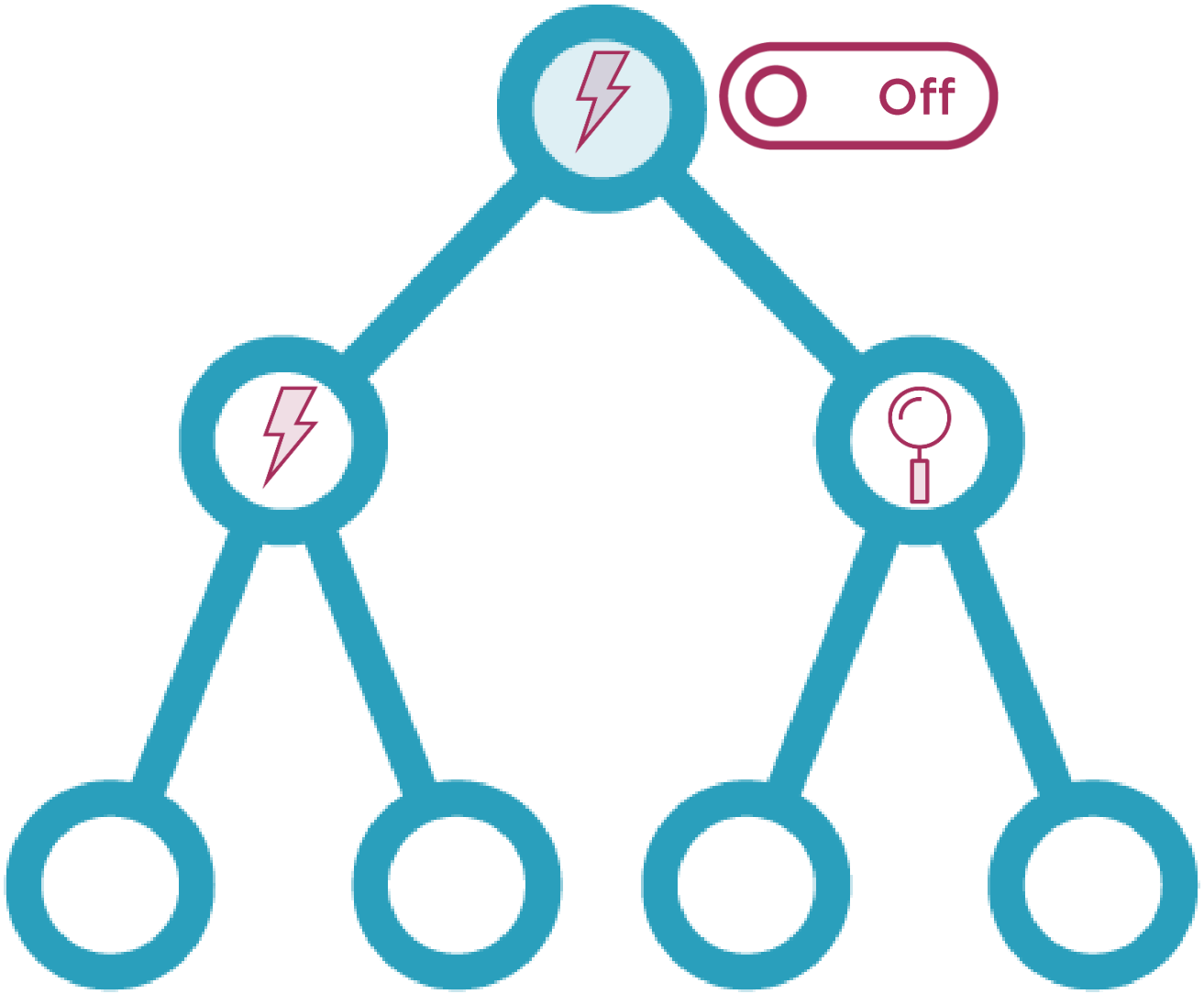


Catch-all

Every **change detection cycle**

- *Including parent & sibling!*

DoCheck



AfterContentInit

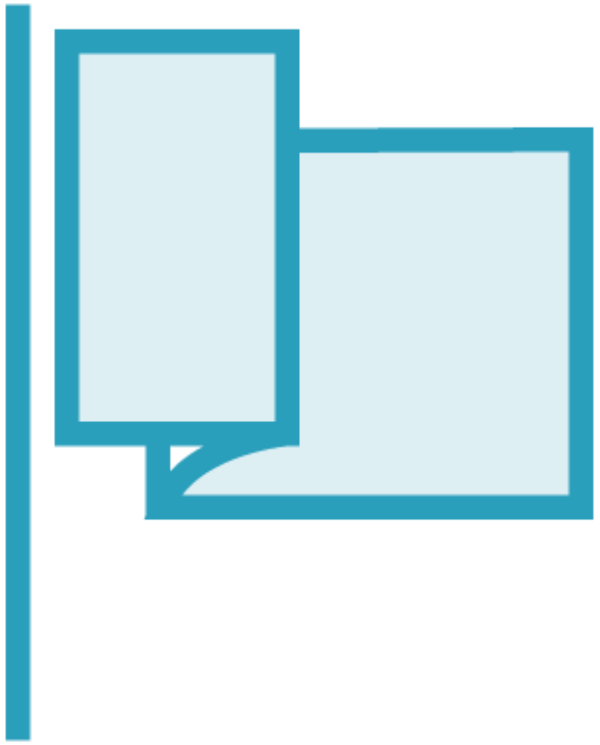
@ContentChild

AfterContentChecked

@ContentChild

Subsequent cycles

AfterViewInit



@ViewChild

AfterViewChecked

@ViewChild

OnDestroy

Cleanup

Demo

Lifecycle Hooks 2

Hook Running: .....

Change Local Primitive Value

Change Bound Primitive Value

Change Bound Object Property

Change Bound Object Ref

constructor
Bound Primitive: **initial value from child declaration**
Bound Object: **{"prop1":"initial value from child declaration","prop2":-1}**
Local Primitive: **initial value**
@ViewChild: **undefined**
@ContentChild: **undefined**

ngOnChanges
Bound Primitive: **Initial value from parent declaration**
Bound Object: **{"prop1":"initial value from parent - 0","prop2":0}**
Local Primitive: **initial value**
@ViewChild: **[object Object]**
@ContentChild: **[object Object]**

SimpleChanges: **{"boundPrimitive":{"currentValue":"Initial value from parent declaration","firstChange":true},"boundObj":{"currentValue":{"prop1":"initial value from parent - 0","prop2":0},"firstChange":true}}**

ngOnInit
Bound Primitive: **Initial value from parent declaration**
Bound Object: **{"prop1":"initial value from parent - 0","prop2":0}**
Local Primitive: **initial value**
@ViewChild: **[object Object]**
@ContentChild: **[object Object]**

ngDoCheck
Bound Primitive: **Initial value from parent declaration**
Bound Object: **{"prop1":"initial value from parent - 0","prop2":0}**
Local Primitive: **initial value**
@ViewChild: **[object Object]**
@ContentChild: **[object Object]**

ngAfterContentInit
Bound Primitive: **Initial value from parent declaration**
Bound Object: **{"prop1":"initial value from parent - 0","prop2":0}**
Local Primitive: **initial value**
@ViewChild: **[object Object]**
@ContentChild: **[object Object]**

ngAfterContentChecked
Bound Primitive: **Initial value from parent declaration**
Bound Object: **{"prop1":"initial value from parent - 0","prop2":0}**
Local Primitive: **initial value**
@ViewChild: **[object Object]**
@ContentChild: **[object Object]**

ngAfterViewInit
Bound Primitive: **Initial value from parent declaration**
Bound Object: **{"prop1":"initial value from parent - 0","prop2":0}**
Local Primitive: **initial value**
@ViewChild: **[object Object]**
@ContentChild: **[object Object]**

ngAfterViewChecked
Bound Primitive: **Initial value from parent declaration**
Bound Object: **{"prop1":"initial value from parent - 0","prop2":0}**
Local Primitive: **initial value**
@ViewChild: **[object Object]**
@ContentChild: **[object Object]**

ngOnDestroy

Key Takeaways

- Async Loading
- Using Pipes correctly
- TrackBy for ngFor
- Caching & Memoizing
- Lifecycle Hooks (part 1)
- Lifecycle Hooks (part 2)

Next Up

Security