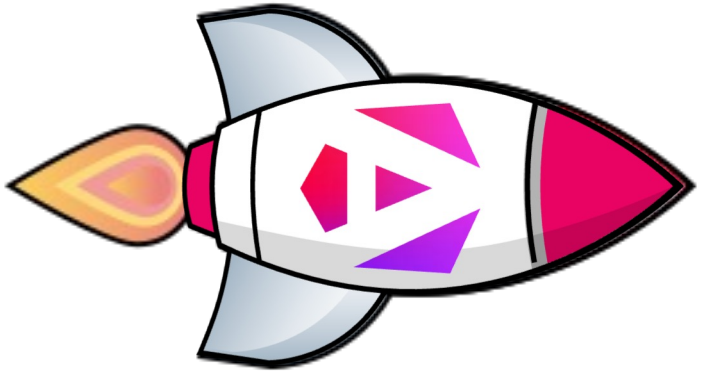
ANGULAR
ARCHITECTS

Runtime Performance

Alexander Thalhammer | @LX_T

Outline - Runtime Performance

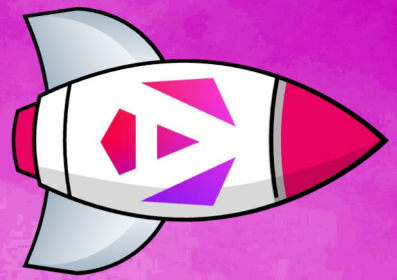


- Change Detection (= Synchronization)
- Runtime Best Practices

Change Detection

- Out of bound change detection
- Zone pollution by 3rd party libs
- view template (.html) optimization
  - with state or flags
  - with Angular Pipes
- Going zoneless

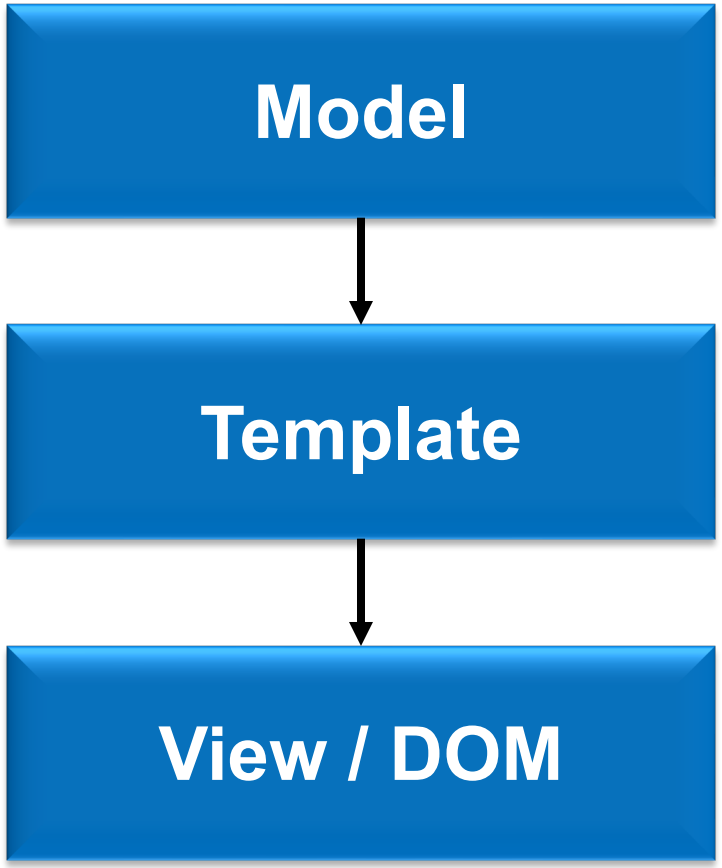
Agenda



Change Detection in Angular

ANGULAR
ARCHITECTS

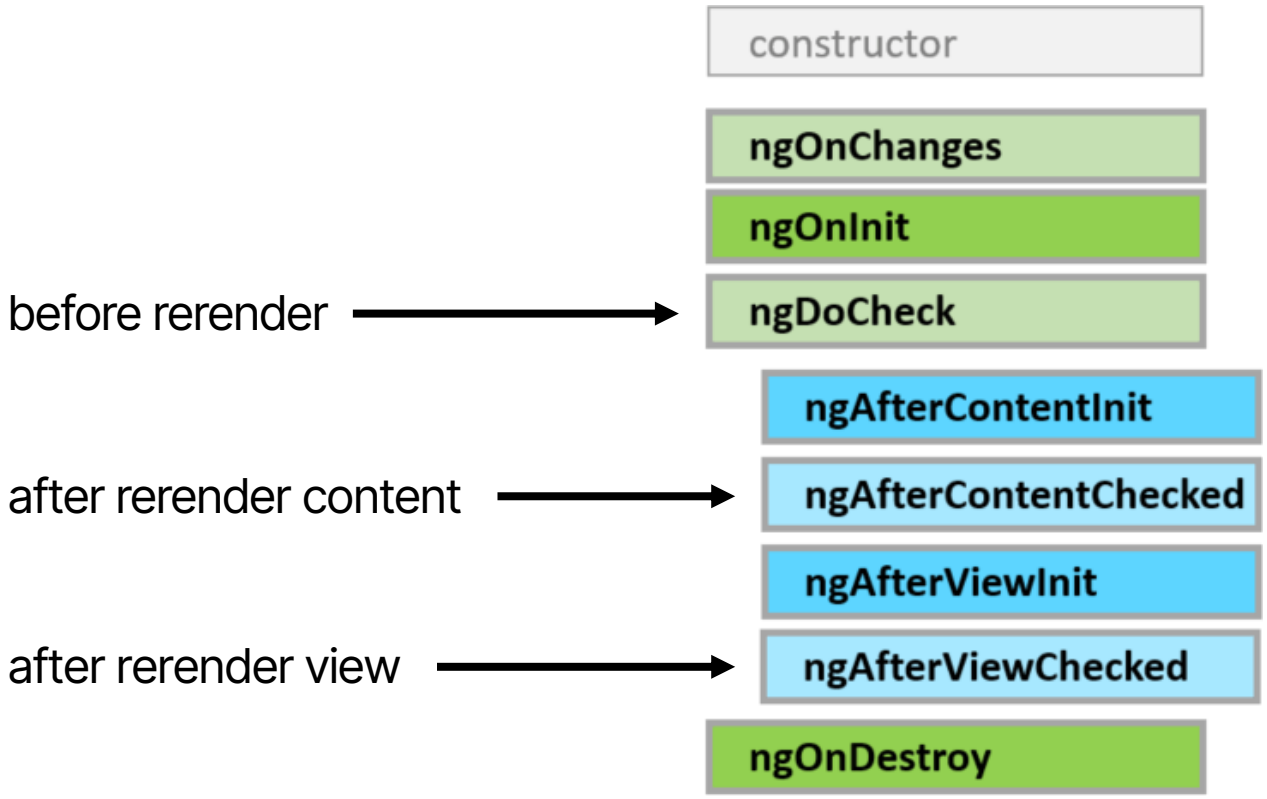
DOM Rendering



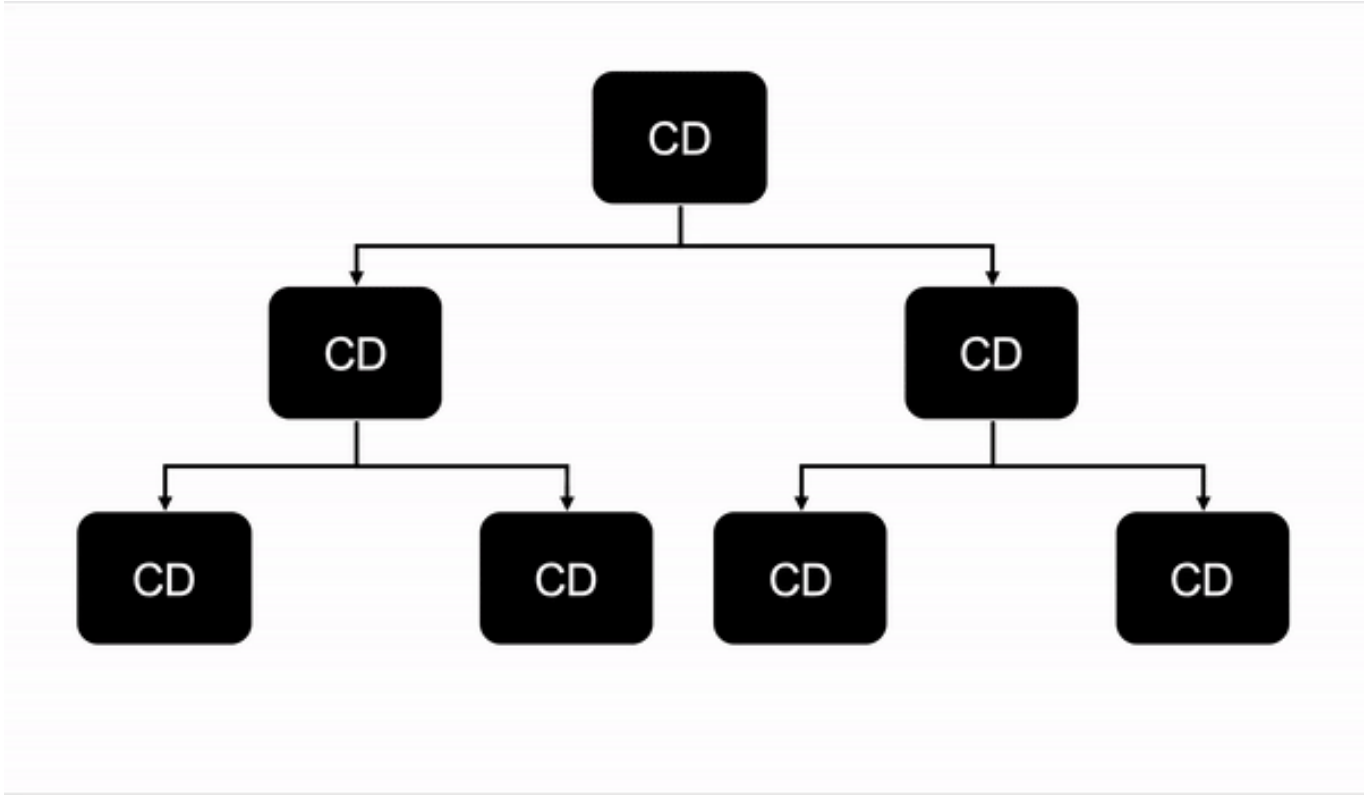
Change Detection

- 1.) User or App changes the model (e.g. input, blur or click)
- 2.) NG CD runs for **every component** (from root to leaves)
- 3.) Check / rerrender the component's view (DOM)

Change Detection – Check Components



Change Detection – From Root To Leaves

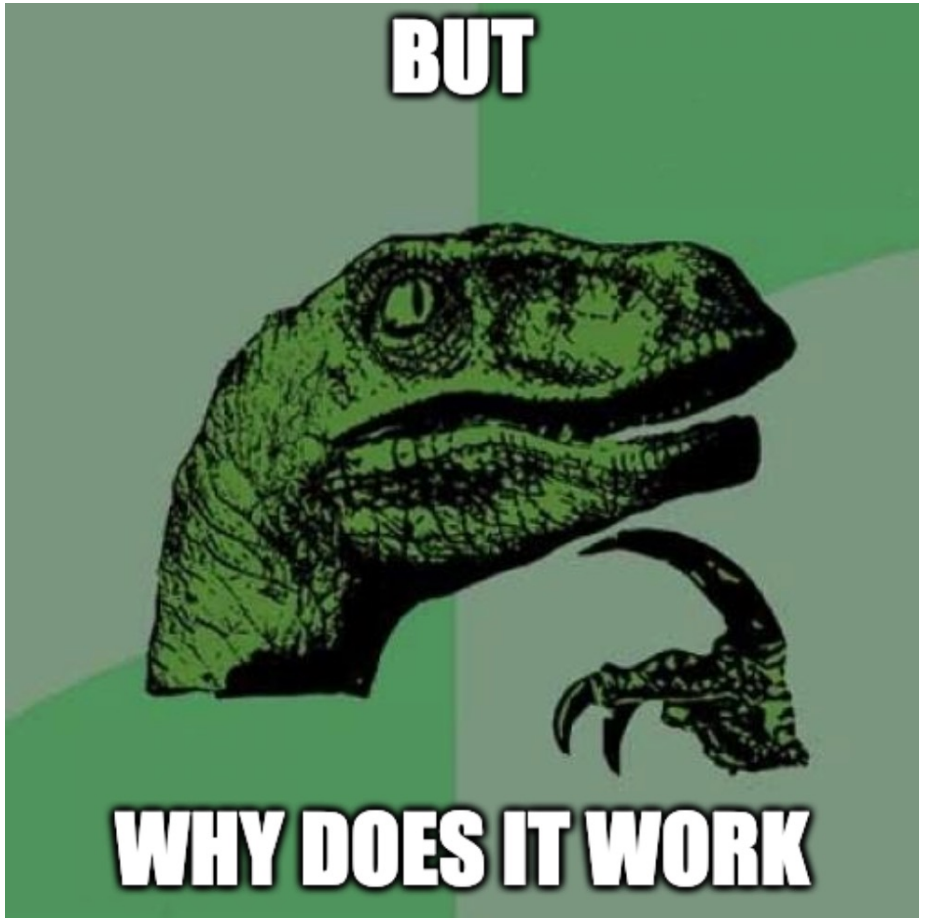


Img src: https://mokkapps.de/blog/the-last-guide-for-angular-change-detection-you-will-ever-need/

Digression – how is zone.js CD triggered?

- Many browser events (click, blur, keyup, etc.) patched
- XMLHttpRequests (AJAX / HttpClient) patched
- setTimeout() and setInterval()
  - often used as a "hack" to trigger CD
  - meaning: "I have no clue how this works"
- Websockets

Change Detection



Demo

Default Strategy & Blink

ANGULAR
ARCHITECTS

Out of bound change detection

- Problem: Local state change triggers CD in other comps
  - E.g. Input field keydown triggers CD in parent/child components

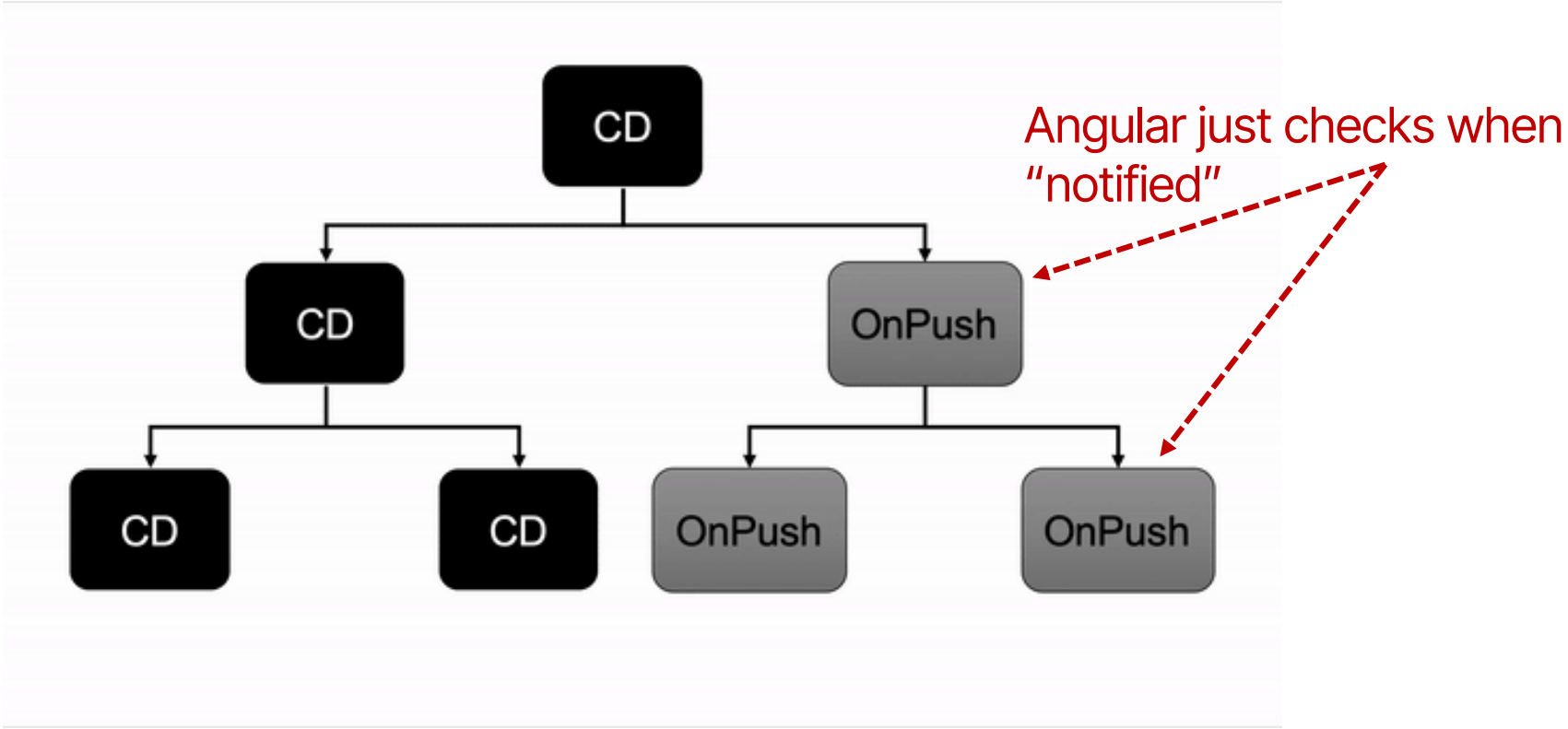
- Identify:
  - Use the (© AngularArchitects) infamous blink() or
  - use the Angular DevTools Profiler
  - console.log() in CD lifecycle hook (e.g. ngDoCheck)

Performance-Tuning with OnPush

Activate OnPush Strategy

```
@Component({
    […]
    changeDetection: ChangeDetectionStrategy.OnPush
})
export class FlightCardComponent {
    […]
    @Input({ required: true }) flight!: Flight;
}
```

Change Detection – OnPush Strategy



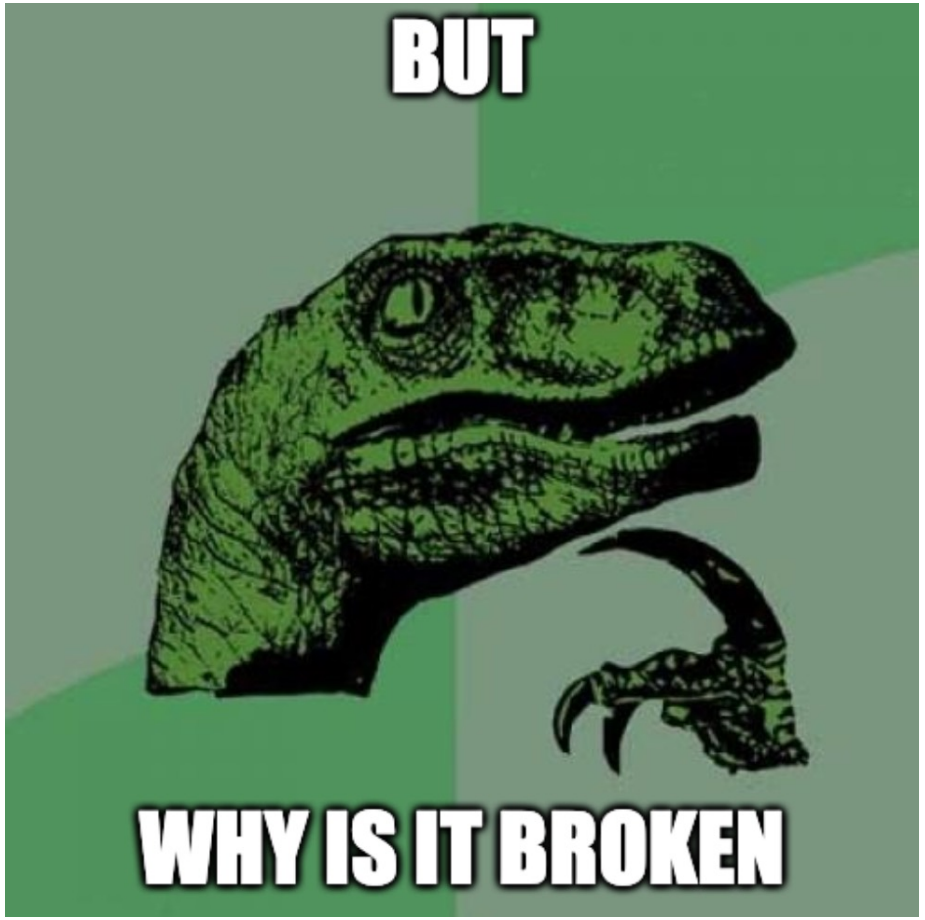
Img src: https://mokkapps.de/blog/the-last-guide-for-angular-change-detection-you-will-ever-need/

Demo

OnPush Strategy

ANGULAR
ARCHITECTS

Change Detection



"Notify" about change?

- 1 Fire event within component or its children (via zone.js)
- 2 Change bound data (**@Input** or input/model signal)
  - OnPush: Angular just compares the object reference!
  - e. g. oldFlight !== newFlight (BTW: like ngOnChanges)
- 3 Emit a bound observable into the **async** pipe | or update a **signal**
  - {{ flights$ | async }} | {{ flights() }}
- 4 Do it manually (cdr.**markForCheck()**)
  - Don't do this at home ;-)
  - But there are reasonable cases (where we can neither use 2 nor 3)
- 5 Attaching or detaching a view using ViewContainerRef
- 6 Bound host or template listener calls

CDR - markForCheck() vs detectChanges()

- Use CDR.markForCheck() to **notify** the CD cycle if using **OnPush**
  - Running up the component tree
  - Useful when you're bypassing the ChangeDetectionStrategy.OnPush e.g. by mutating some data or you've just updated the components model

- Use CDR.detectChanges() to **trigger CD immediately** for this component and it's **children** respecting the its/their CD strategy
  - Running down the component tree
  - Useful when you've updated the model after angular has run it's change detection, or if the update hasn't been in Angular world at all

- For the whole app (from root to leaves) use ApplicationRef.tick()

Set OnPush as default

- Add to angular.json / project.json schematic

```
"@schematics/angular:component": {
  "changeDetection": "OnPush",
  "style": "scss"
},
```

- Add an ESLint rule

```
"@angular-eslint/prefer-on-push-component-change-detection": "warn"
```

- OnPush in every component?
  - well yes, but
  - optional in smart components (and root)

Zone pollution by 3rd party libs (charts)

Demo

Zone Pollution

ANGULAR
ARCHITECTS

Zone pollution by 3rd party libs (charts)

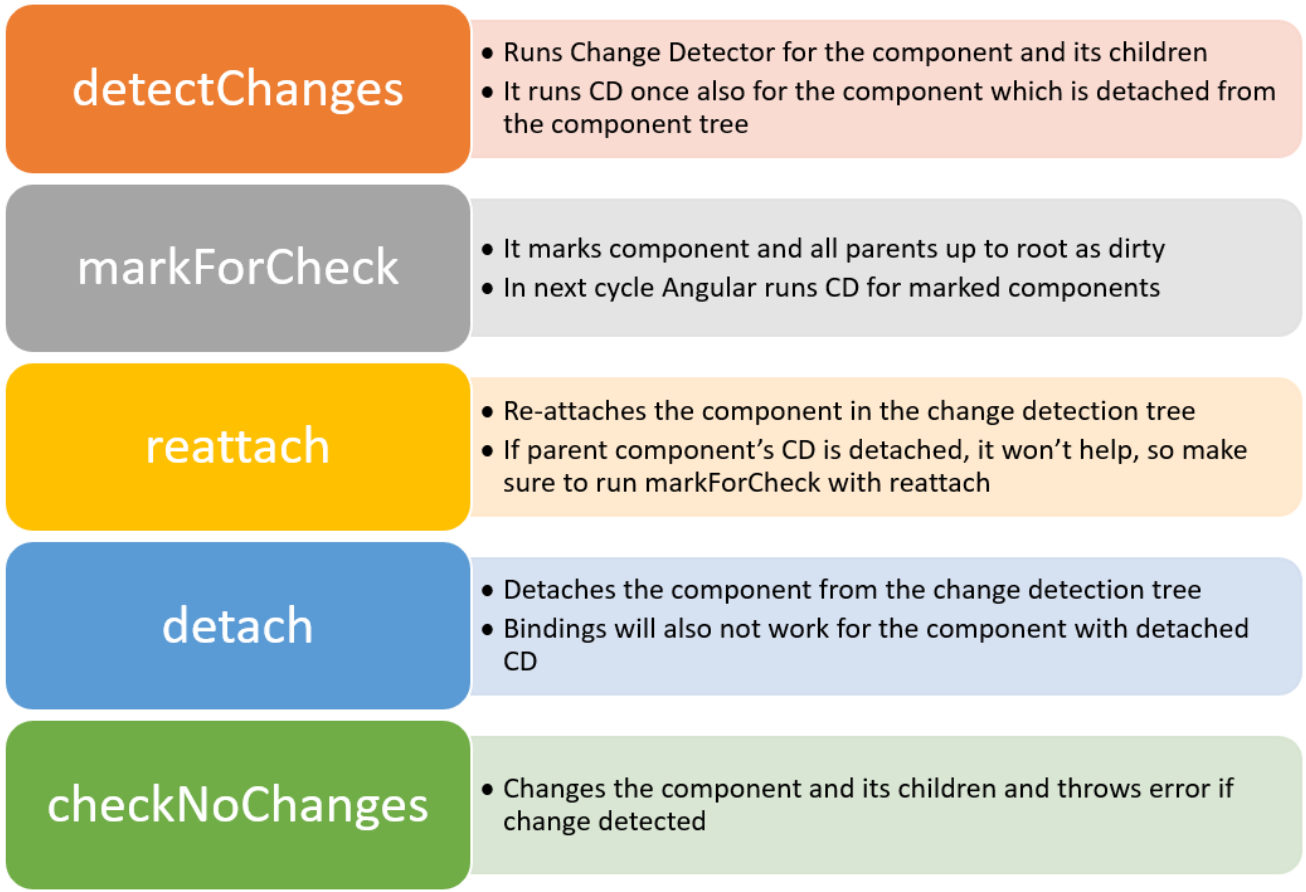
- Problem: Callbacks that trigger redundant change detection cycles
- Identify: Use the infamous blink() or the Angular DevTools Profiler
  - E.g. MouseEvent listeners
  - **requestAnimationFrame()** or
  - **setInterval()**
  - a live watch
- Solution: Run outside of NG Zone
  - Inject (private readonly ngZone: NgZone)
  - Call this.ngZone.runOutsideAngular(() => doStuff)
  - https://angular.io/guide/change-detection-zone-pollution
- Alternative: Using cdr.detach() for components
- Alternative: Get rid of zone.js by going zoneless

Demo

Fixed Zone Pollution

ANGULAR
ARCHITECTS

ChangeDetectorRef API, once more



Img src: https://www.telerik.com/blogs/simplifying-angular-change-detection/

Optimization with state or flags

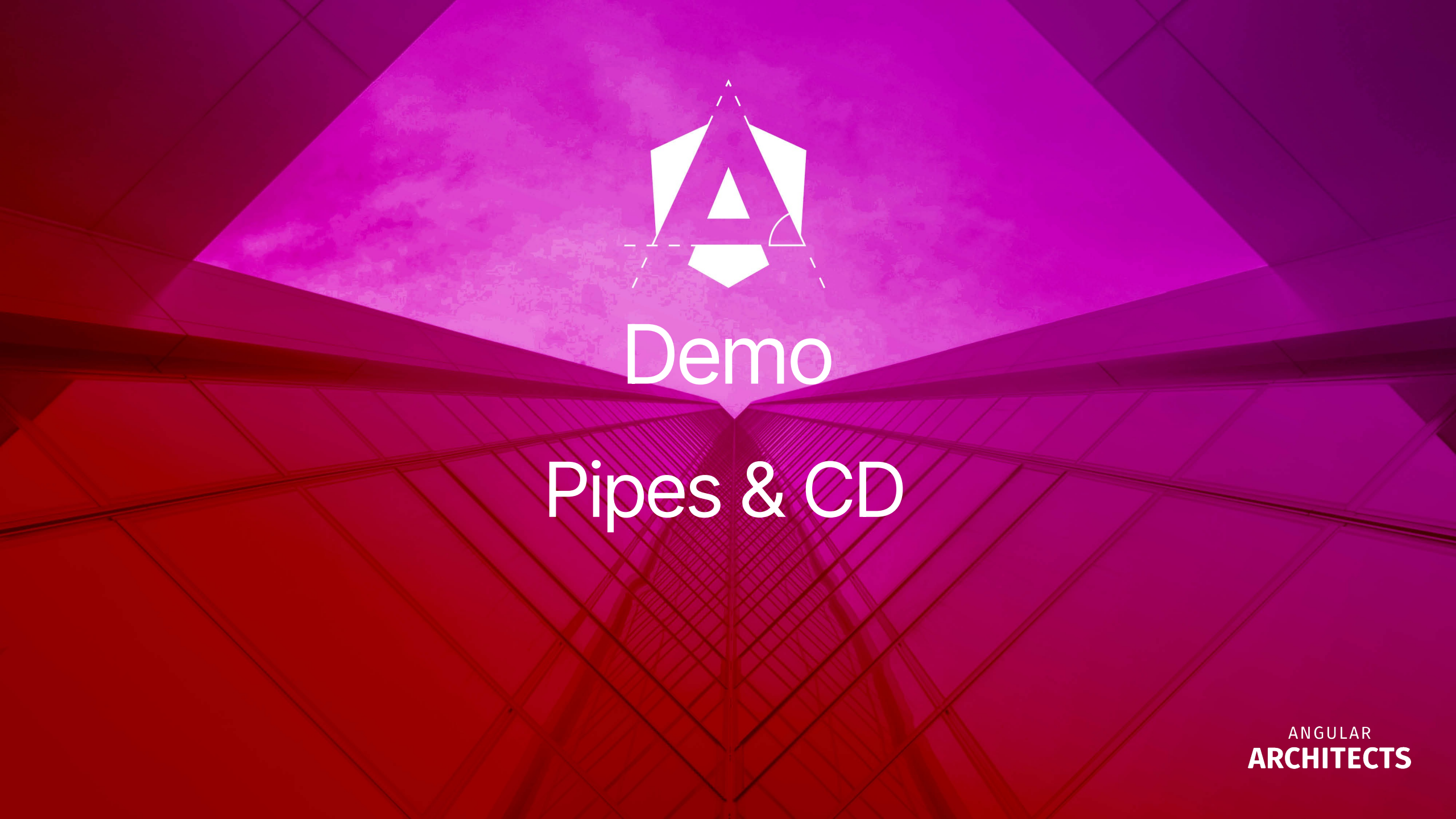
- Problem: Redundant calculations for conditions
- Identify: Methods being executed in **@if** (*ngIf) statements
- Solution:
  - Use StateManagement like subjects or
  - use signals or
  - use boolean flags or strings, that only change when they should

Optimization with Angular Pipes

– Problem: Redundant calculations / transforming / formatting

– Identify: Methods in html templates

– Solution: Use (pure) Angular Pipes



Demo

Pipes & CD

ANGULAR
ARCHITECTS

Zoneless Angular

- No zone.js event bindings

- Need to make sure to **notify** Angular about changes
  - (see notification options above: 2, 3 or 4)
  - To trigger Change Detection and thus DOM updates

Demo

Going Zoneless

ANGULAR
ARCHITECTS

Change Detection

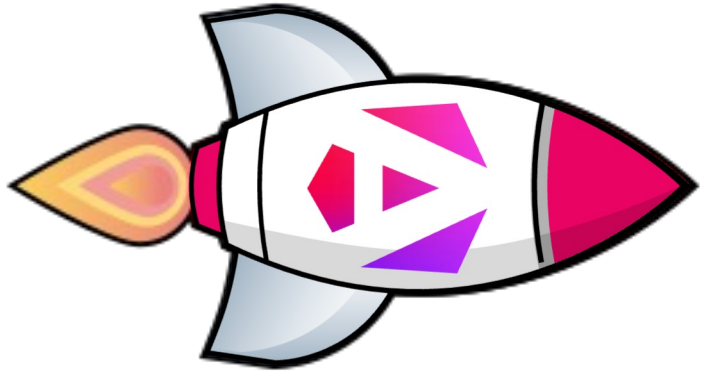
- **Out of bound change detection**
- **Zone pollution by 3rd party libs**
- HTML template optimization
  - with state or flags
  - with **Angular Pipes**
- Going zoneless

Recap

References

- Minko Gechev (@mgechev) for Angular on YouTube
  - https://www.youtube.com/watch?v=FjyX_hkscII
  - https://www.youtube.com/watch?v=f8sA-i6gkGQ
  - New in NG 17, 18+:
    https://www.youtube.com/watch?v=2M17gRQbgfI
- Resolving Zone Pollution
  - https://angular.io/guide/change-detection-zone-pollution
- Angular Performance Optimization using Pure Pipe
  - https://www.youtube.com/watch?v=YsOf90RZfss

Outline - Runtime Performance

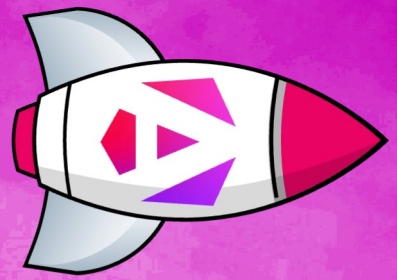


- Change Detection
- Runtime Best Practices

Runtime Best Practices

- Large @for loops
  - Using track in @for
  - Avoid large component trees
- UX improvements
  - Use spinners / preview thumbs / skeletons
  - Optimistic updates
- Bonus: RxJS Subscription Best Practices

Agenda



Handling large @for loops

ANGULAR
ARCHITECTS

Migrate to NG 17 Control Flow

- The future is here ☺
- May look a bit awkward at first sight
  - but it has (a lot) **performance** benefits and
  - on top of that it make things **easier**

- Easy migration

```
ng generate @angular/core:control-flow
```

- Make sure to add
  - @empty / @else
  - improve track @for

NG 17 Control Flow benchmark

Duration in milliseconds ± 95% confidence interval (Slowdown

Name Duration for...	vanillajs	angular-cf-nozone-v17.0.2	vue-v3.4.21	angular-cf-v17.0.2	angular-ngfor-v17.0.2	react-hooks-v18.2.0
Implementation notes	772					
Implementation link	code	code	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	36.2 ±0.5 (1.03)	44.2 ±0.3 (1.26)	44.2 ±0.5 (1.26)	44.8 ±0.5 (1.27)	45.6 ±0.4 (1.30)	45.8 ±0.3 (1.30)
replace all rows updating all 1,000 rows. (5 warmup runs).	39.5 ±0.3 (1.03)	51.2 ±0.3 (1.33)	48.5 ±0.5 (1.26)	54.4 ±0.4 (1.41)	54.9 ±0.3 (1.43)	54.8 ±0.3 (1.42)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	16.8 ±0.2 (1.07)	17.4 ±0.2 (1.11)	19.6 ±0.3 (1.25)	17.5 ±0.3 (1.11)	17.7 ±0.4 (1.13)	20.8 ±0.3 (1.32)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	2.9 ±0.2 (1.07)	3.9 ±0.2 (1.44)	4.3 ±0.2 (1.59)	3.9 ±0.1 (1.44)	3.9 ±0.1 (1.44)	5.1 ±0.2 (1.89)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	18.1 ±0.2 (1.01)	19.9 ±0.4 (1.11)	20.7 ±0.3 (1.16)	20.0 ±0.3 (1.12)	170.1 ±1.2 (9.50)	166.3 ±1.2 (9.29)

https://krausest.github.io/js-framework-benchmark/current.html

Using track in @for (*ngFor)

– Problem: Angular replaces items in @for (*ngFor) upon changes

– Identify: Easy - search for "@for (*ngFor)"

– Solution: Use the **track** function (*previously trackBy*)

```
<li *ngFor="let dashboard of dashboards; trackBy: trackByDashboardId"

 trackByDashboardId(index: number, item: Dashboard): number {
   return item.id;
 }
```

Using track in @for

- Automatically required

```
@for (flight of flights; track flight.id) {
   [...]
} @empty {
  No flights found.
}
```

Demo

@for track

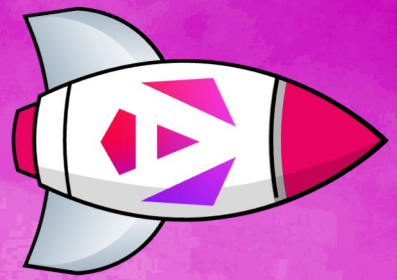
Avoid large component trees

- Problem: Too many (100+) components are loaded
- Identify: Lots of components slowing down frame rate
- Solution: On demand component rendering
  - E.g. Pagination or Angular CDKs <cdk-virtual-scrolling-component>

Demo

Virtual Scrolling

ANGULAR
ARCHITECTS



Other UX improvements

Spinners & Preview Thumbs

Twitter / Insta / ...

Use spinners and preview thumbs

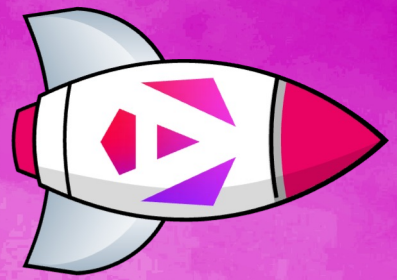
- Problem: App waits for backend before showing content
- Identify: Waiting for API data to show a view (page)
- Solution: Show view (page) immediately
  - Show spinners to indicate data is still loading
  - Even more sophisticated: show preview images (used everywhere on big platforms!)

Optimistic Updates

E.g. Like Buttons

Optimistic Updates

- Problem: App waits for backend for confirmations
- Identify: Spinner showing when clicking on save
- Solution: Confirm action immediately
  - Go back in case of an error (e.g. no network)
  - But maybe not a good idea for all user flow ☺



RxJS Subscription Best Practices

ANGULAR
ARCHITECTS

Why asynchronicity?

Asynchronous operations (API requests)

Interactive behavior (user input)

Websockets

Server Send Events (Push)

Why do we (always!) need to unsubscribe?

Avoid side effects

Avoid memory leaks

Also for HttpClient's get / post ...

Manage your RxJS subscriptions

- Problem: Components create subscriptions without closing them
- Identify: .subscribe() without .unsubscribe() or other methods
- Solution: Unsubscribe from all Observables in your App
  - Except Angular Router Params

RxJS Subscription Management

- Explicitly with reference
  - readonly subscription = observable$.subscribe(...); // field initializer
    // subscription?.add(otherObservable$.subscribe(...)); // also possible since V6
    subscription?**.unsubscribe()**; // ngOnDestroy
- Implicitly with take until
  - ~~observable$.pipe(takeUntil(otherObservable)).subscribe(...);~~
  - observable$.pipe(**takeUntilDestroyed()**).subscribe(...);

last operator!

- Implicitly with async Pipe managed by Angular or using a Signal
  - {{ observable$ | **async** }} → **also triggers a cdr.markForCheck for OnPush ☺**
- Automatically managed by Angular
  - Router Params / ParamMap (only 1 I know where unsubscribing is not needed)

Where / when do we subscribe?

– 1 **Field initializer or constructor**

– 2 **If** @Input(s) needed → **ngOnInit** hook (needs destroyRef)

– 3 Elsewhere (needs injected destroyRef)

Demo

RxJS Subscription Management

ANGULAR
ARCHITECTS

Runtime Best Practices

- Large @for loops
  - Using **track** in @for
  - **Avoid** large component trees
- UX improvements
  - Use spinners / preview thumbs / skeleton
  - Optimistic updates
- Bonus: **RxJS Subscription Management**

Recap

References

- Angular CDK Scrolling Comp
  - https://material.angular.io/cdk/scrolling/overview

ANGULAR
ARCHITECTS

Lab 04 Runtime Best Practices

OnPush Strategy / track / Virtual Scrolling