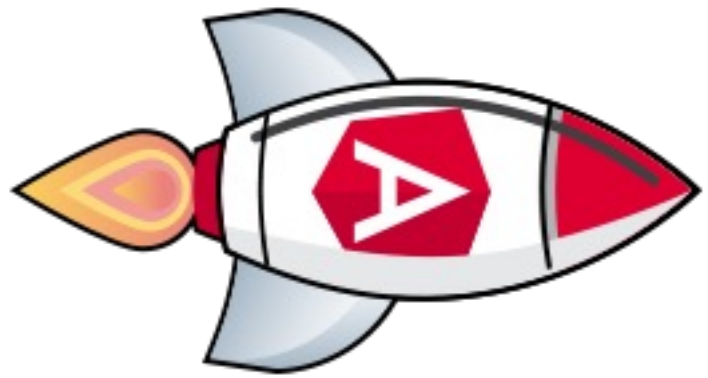
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Angular Runtime Performance Optimization

Hosted by Alex Thalhammer



Outline

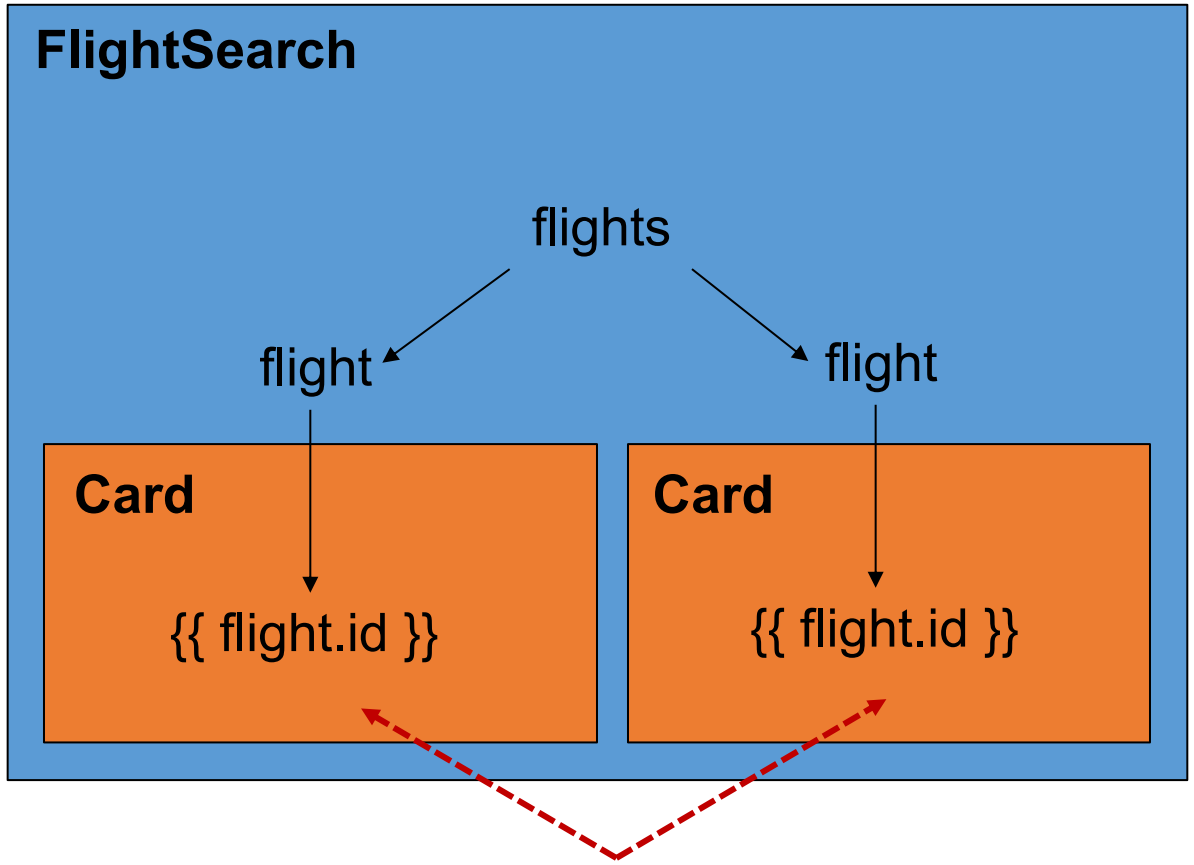
1. Out of bound change detection
2. Zone pollution by 3rd party libs
3. Optimization with state or flags
4. Optimization with Angular Pipes
5. Avoid large component trees
6. Use trackBy in ngFor if possible
7. Use Spinners and preview thumbs
8. Optimistic updates
9. Bonus: Unsubscribing RxJS subscriptions

#1: Out of bound change detection

- Problem: *Local state change triggers change detection in other comps*
- Identify: Use the infamous blink() or the Angular DevTools Profiler
  - E.g. Input field keydown triggers change detection in other components
- Solution: ChangeDetectionStrategy.OnPush as default

Performance-Tuning with OnPush

OnPush



Angular just checks when “notified”

"Notify" about change?

- Change bound data (@Input)
  - OnPush: Angular just compares the object reference (like ngOnChanges)
  - e. g. oldFlight !== newFlight
- Raise event / output within the component
- Notify a bound observable with the async pipe
  - {{ flights$ | async }}
- Trigger it manually
  - Don't do this at home ;-)
  - Use this.cdr.markForCheck()

Activate OnPush

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

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

#1: detectChanges() vs markForCheck()

- Use cdr.**detectChanges() to trigger CD immediately** 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

- Use cdr.**markForCheck() to mark for check in next CD cycle** if you're using **OnPush** and you're bypassing the ChangeDetectionStrategy by mutating some data or you've updated the model inside a **setTimeout**

DEMO – ChangeDetection

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

#2: Zone pollution by 3rd party libs (charts)

#2: 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
  - setTimeout()

- Solution: Run outside of NG Zone
  - Inject (private ngZone: NgZone)
  - Call this.ngZone.runOutsideAngular(() => doStuff)
  - https://angular.io/guide/change-detection-zone-pollution

- Alternative: Using cdr.detach() for components

#2: ChangeDetectorRef API, once more

detectChanges	• Runs Change Detector for the component and its children • It runs CD once also for the component which is detached from the component tree
markForCheck	• It marks component and all parents up to root as dirty • In next cycle Angular runs CD for marked components
reattach	• Re-attaches the component in the change detection tree • If parent component's CD is detached, it won't help, so make sure to run markForCheck with reattach
detach	• Detaches the component from the change detection tree • Bindings will also not work for the component with detached CD
checkNoChanges	• Changes the component and its children and throws error if change detected

DEMO – Zone Pollution

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Lab

Runtime Performance – Change Detection

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

#3: Optimization with state or flags

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

#4: Optimization with Angular Pipes

- Problem: *Redundant calculations for content or formatting*
- Identify: Methods being executed in string interpolations in the template or similar things slowing change detection cycles
- Solution: Use (pure) Angular Pipes

#5: 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>

#6: Using trackBy in ngFor

- Problem: *Angular will replace all items in ***ngFor** upon changes*
- Identify: Easy - search for "*ngFor"
- Solution: Use the trackBy function

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

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

#7: 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!)

Spinners & Preview Thumbs

Twitter / Insta / …

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

#8: 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)

Optimistic Updates

E.g. Like Buttons

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE ARCHITECT

#9: 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

#9: Closing Subscriptions

- Explicitly

  let subscription = observable$.subscribe(…);

  // subscription.add(observableTwo$.subscribe(…)) // also possible
  subscription?.**unsubscribe()**;

- Implicitly
  - observable$.pipe(**takeUntil(otherObservable)**).subscribe(…);
  - observable$.pipe(**takeWhile(boolean)**).subscribe(…);

  } **last operator!**

- Implicitly with async-Pipe in Angular

  {{ observable$ | **async** }}

- Automatic by Angular
  - Angular Router Params

DEMO – Unsubscribing

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Lab

Further Runtime Performance

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE ARCHITECT

Recap

1. **Out of bound change detection**
2. **Zone pollution by 3rd party libs**
3. Optimization with state or flags
4. **Optimization with Angular Pipes**
5. Avoid large component trees
6. Use trackBy in ngFor if possible
7. Use Spinners and preview thumbs
8. Optimistic updates
9. **Bonus: Unsubscribing RxJS subscriptions**

References

- Minko Gechev (@mgechev) for Angular on YouTube
  - https://www.youtube.com/watch?v=FjyX_hkscII
  - https://www.youtube.com/watch?v=f8sA-i6gkGQ
- Resolving Zone Pollution
  - https://angular.io/guide/change-detection-zone-pollution
- Angular Performance Optimization using Pure Pipe
  - https://www.youtube.com/watch?v=YsOf90RZfss
- Angular CDK Scrolling Comp
  - https://material.angular.io/cdk/scrolling/overview