ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Data Binding and Change Detection

Alex Thalhammer

Contents

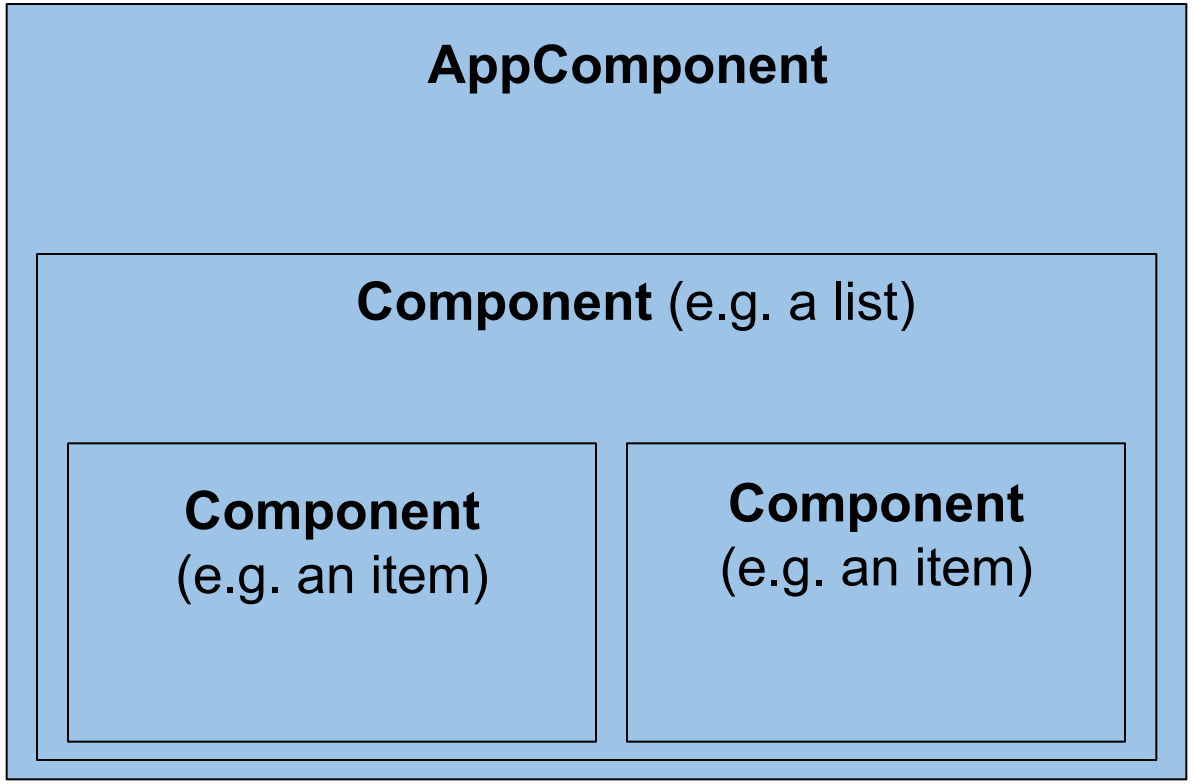
- How does data binding work (underneath the covers)?
- Smart vs Dumb Components
- Performance-Tuning with ChangeDetection OnPush

Data Binding

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

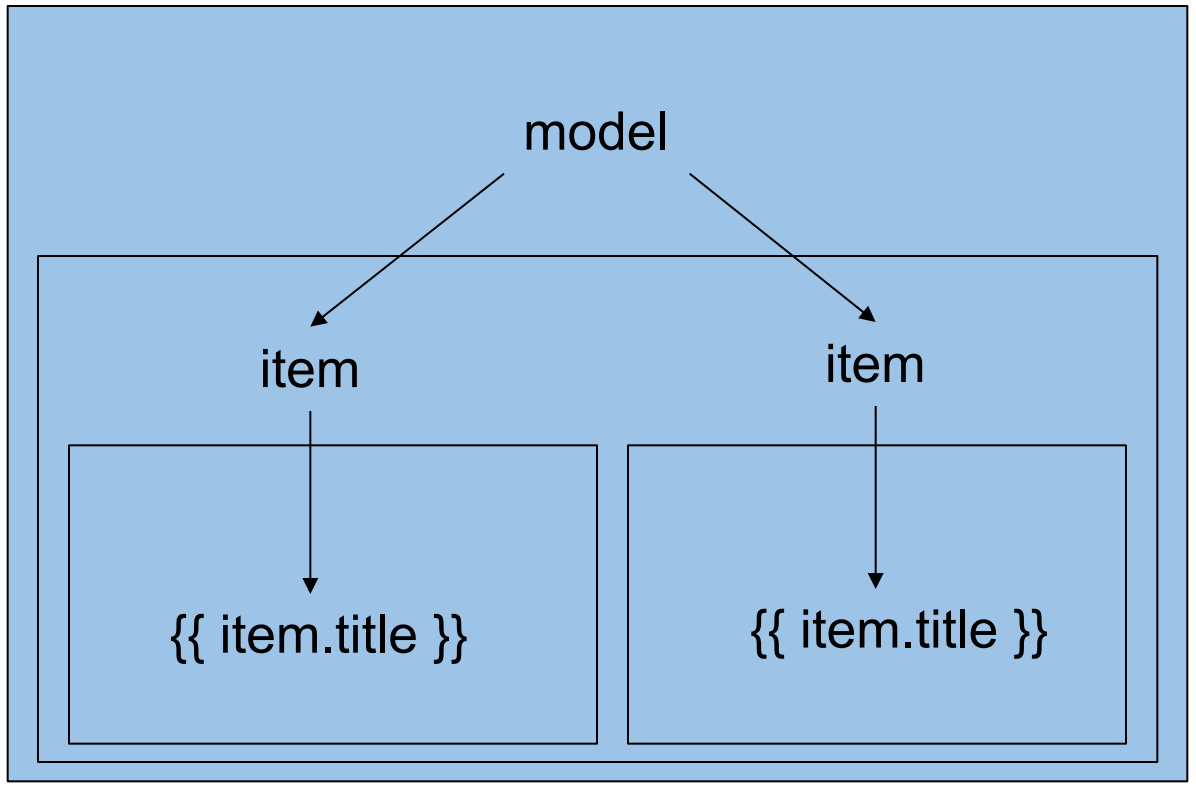
Component Tree in Angular 2+



Rules for Property-Bindings

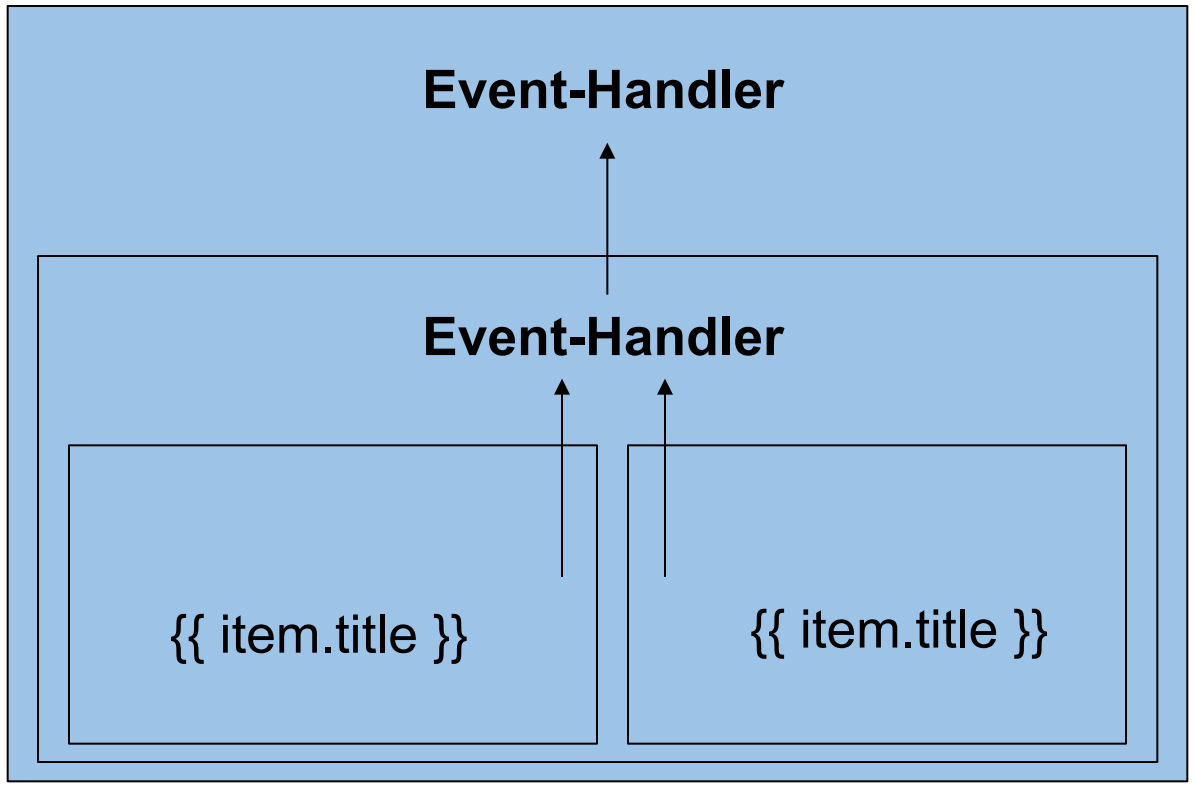
- Data flows top/down
  - Parent can send data to children
  - Children **cannot** send data to parent
- Dependency graph is a tree
- Angular only needs one "digest"

Property Binding



[http://victorsavkin.com/post/110170125256/change-detection-in-angular-2]

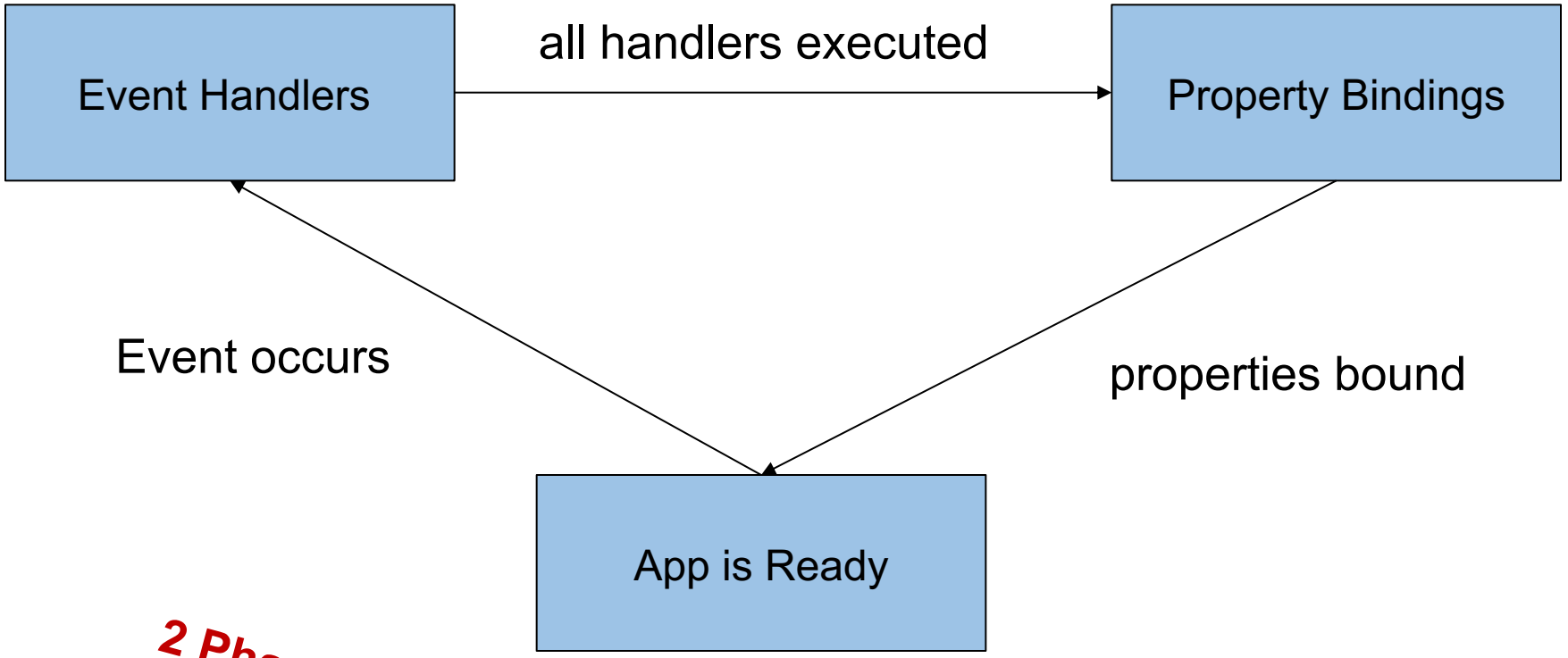
Event Bindings (One-Way, Bottom/Up)



Event Bindings (One-Way, Bottom/Up)

- Cheap: No "digest" needed!
- However: Events can change data → Property Binding

Property- and Event-Bindings



View

```
<button [disabled]="!from || !to" (click)="search()">
   Search
</button>

<table>
   @for (flight of flights; track flight.id) {
      <tr>
         <td>{{flight.id}}</td>
         <td>{{flight.date}}</td>
         <td>{{flight.from}}</td>
         <td>{{flight.to}}</td>
         <td><a href="#" (click)="selectFlight(flight)">Select</a></td>
      </tr>
   }
</table>
```

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

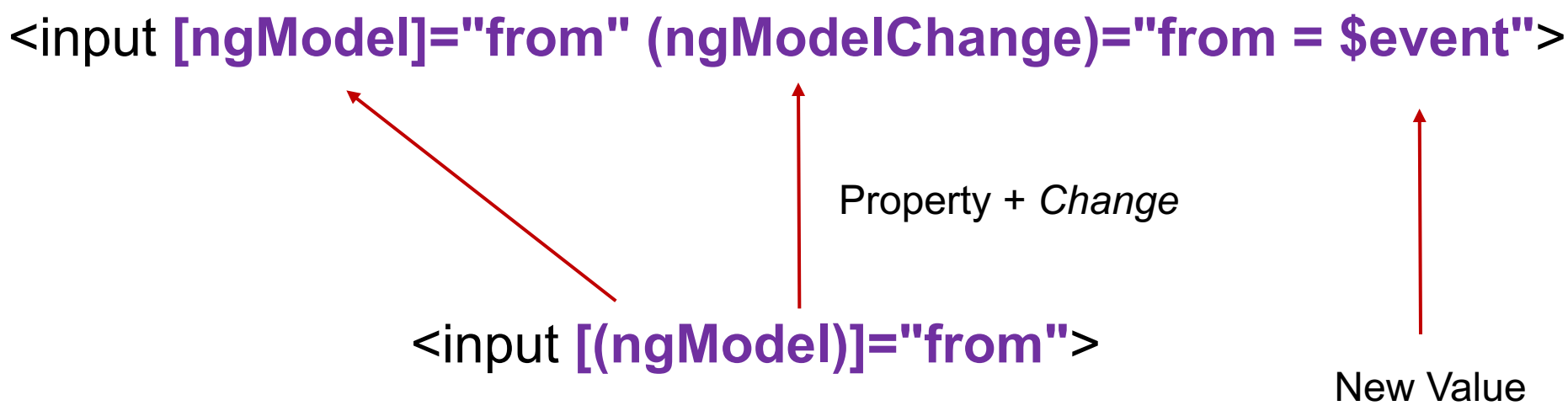
Recap

- Property-Binding: One-Way; Top/Down
- Event-Binding: One-Way; Bottom/Up
- Two-Way-Binding?
- Two-Way = Property-Binding + Event-Binding

Property and Event Bindings

```
<input [ngModel]="from" (ngModelChange)="update($event)">
```

Property and Event Bindings



DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Thought experiment

- What if <app-flight-card> would handle use case logic?
  - e.g. communicate with API
- Number of requests ==> Performance?
- Traceability?
- Reusability?

Smart vs. Dumb Components

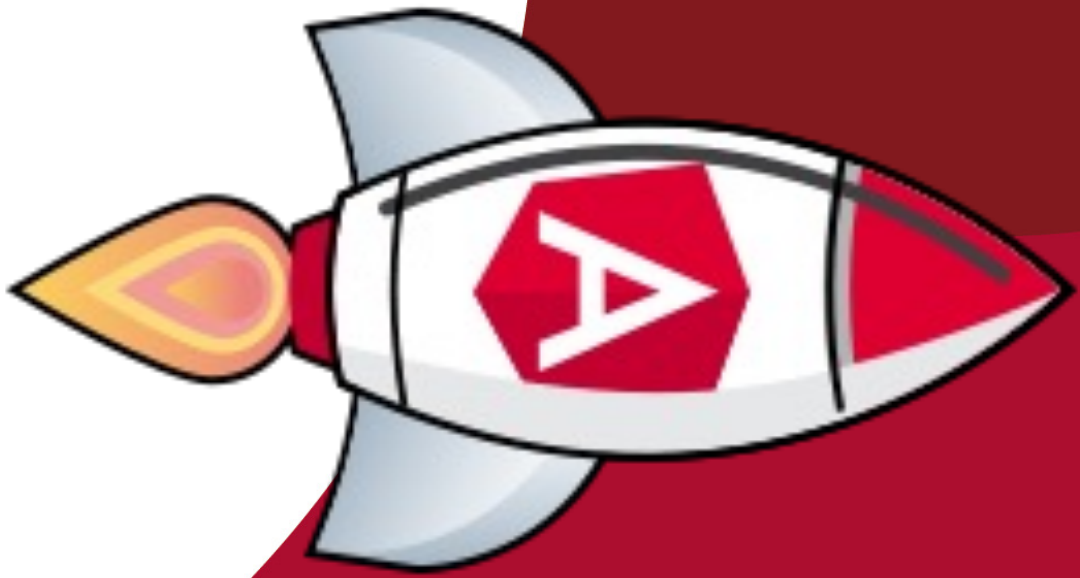
Smart / Controller	Dumb / Presentational
• 1 per feature / use case / route • Business logic • Container	• Independent of Use Case • Reusable • Leave

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

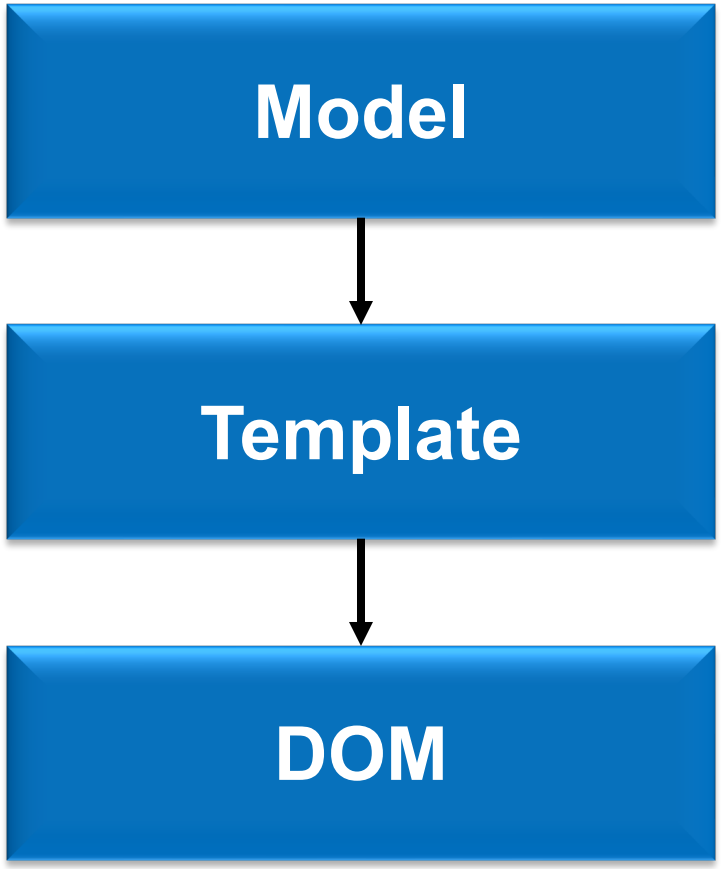
SOFTWARE
ARCHITECT

Change Detection in Angular



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

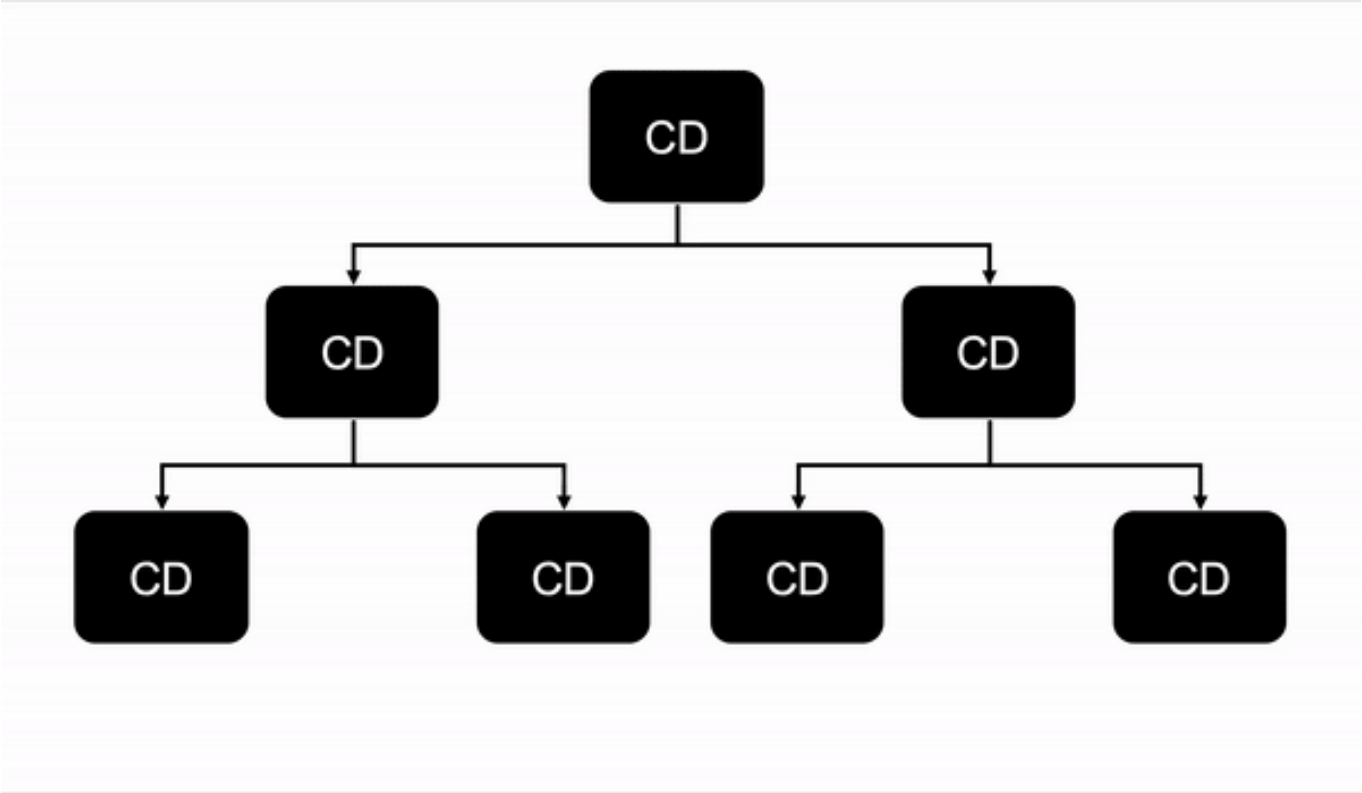
DOM Rendering



Change Detection

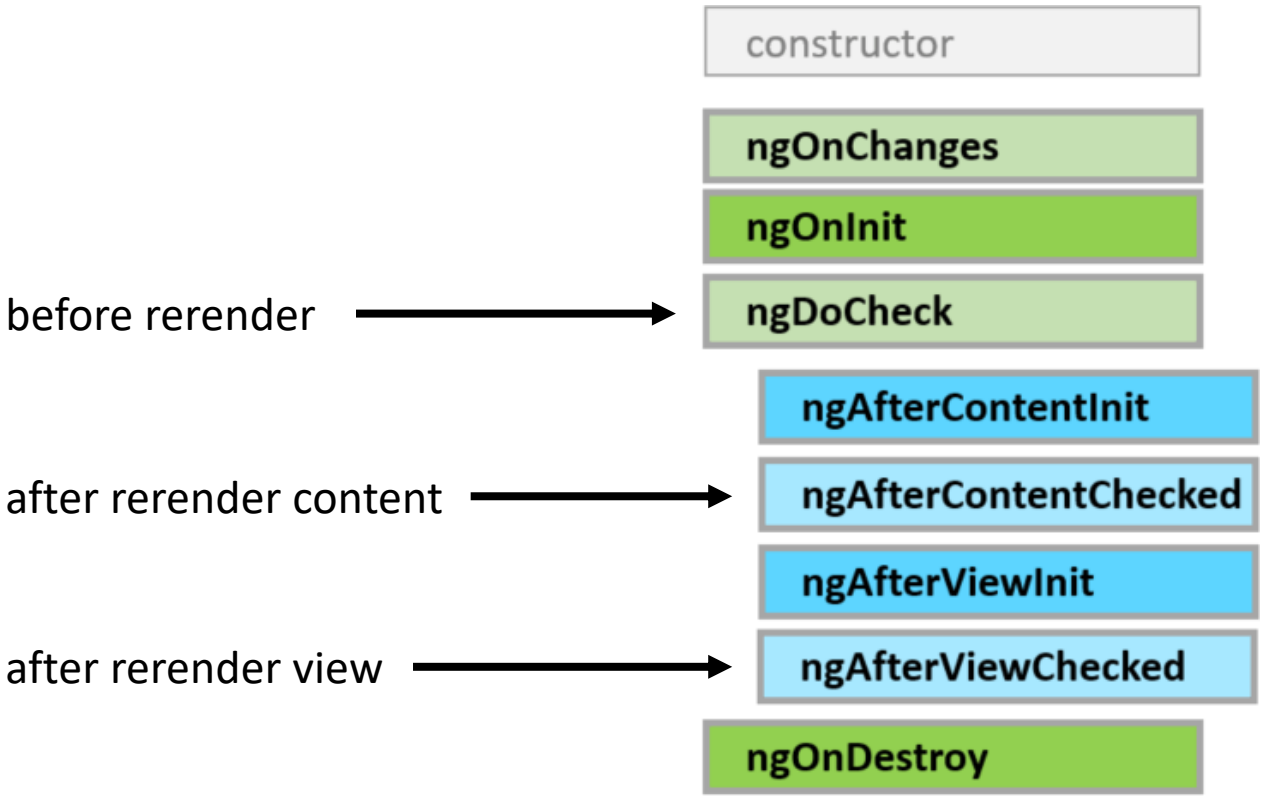
- 1.) User or App changes the model (e.g. input, blur or click)
- 2.) NG CD checks for **every component** (from root to leaves) if the corresponding component model has changes and thus its view (DOM) needs to be updated
- 3.) If yes then update / rerender the component's view (DOM)

Change Detection – From Root To Leaves



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

Change Detection – Rerender Components



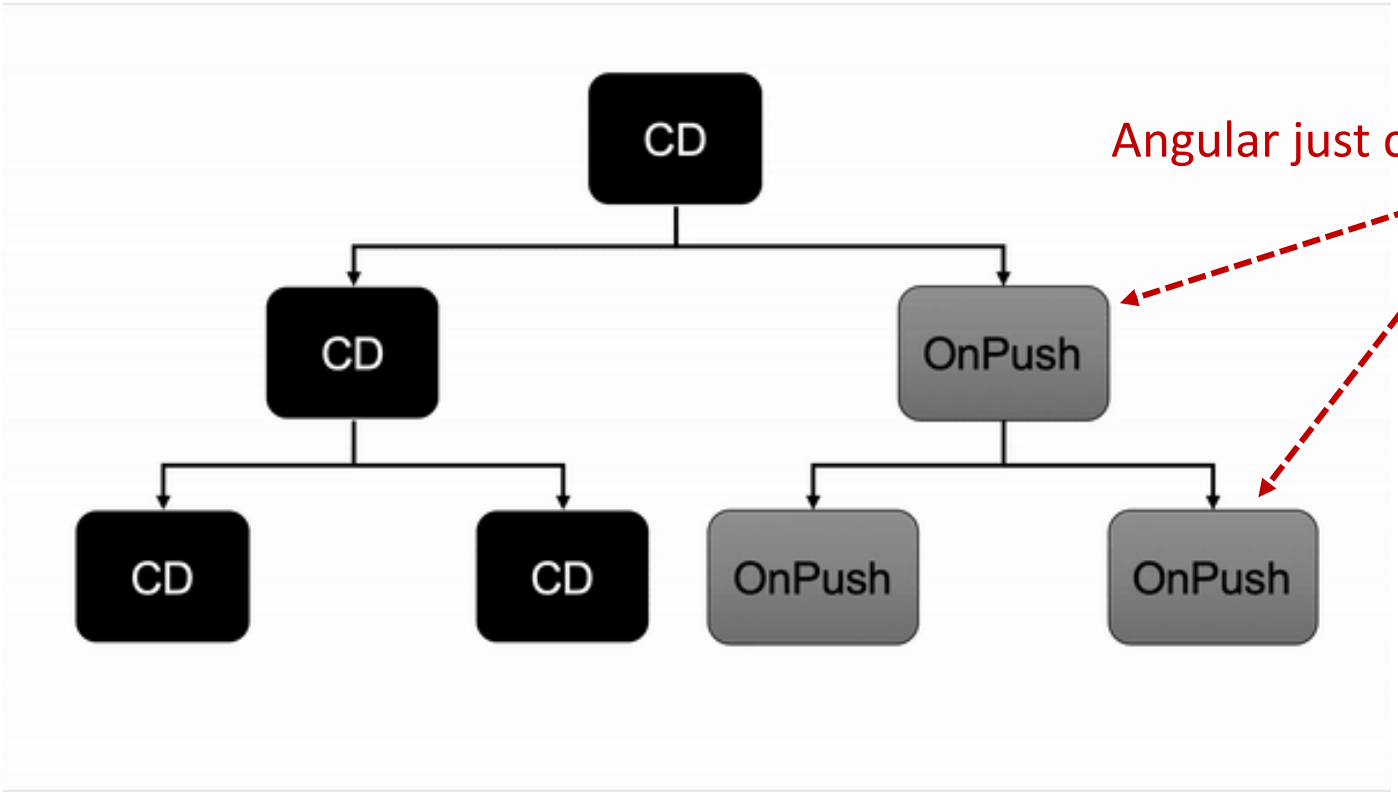
DEMO – ChangeDetection

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Performance-Tuning with OnPush

Change Detection – OnPush Strategy



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

Activate OnPush Strategy

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

"Notify" about change?

- 1 Change bound data (**@Input**)
  - OnPush: Angular just compares the object reference!
  - e. g. oldFlight !== newFlight (BTW: like ngOnChanges)
- 2 Raise event within the component and its children (e.g. @Output)
- 3 Emit in a bound observable into the async pipe | or update a signal
  - {{ **flights$ | async** }} | {{ **flightsSignal()** }}
- 4 Do it manually (cdr.markForCheck())
  - Don't do this at home ;-)
  - But there are reasonable cases (where we can neither use 1 nor 3)

CDR - markForCheck() vs detectChanges()

- Use CDR.**markForCheck() to notify the next CD cycle** if using **OnPush**
  - 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 view and it's children respecting the its/their CD strategy
  - 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 Application you can to ApplicationRef.tick()

DEMO – OnPush

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

LAB

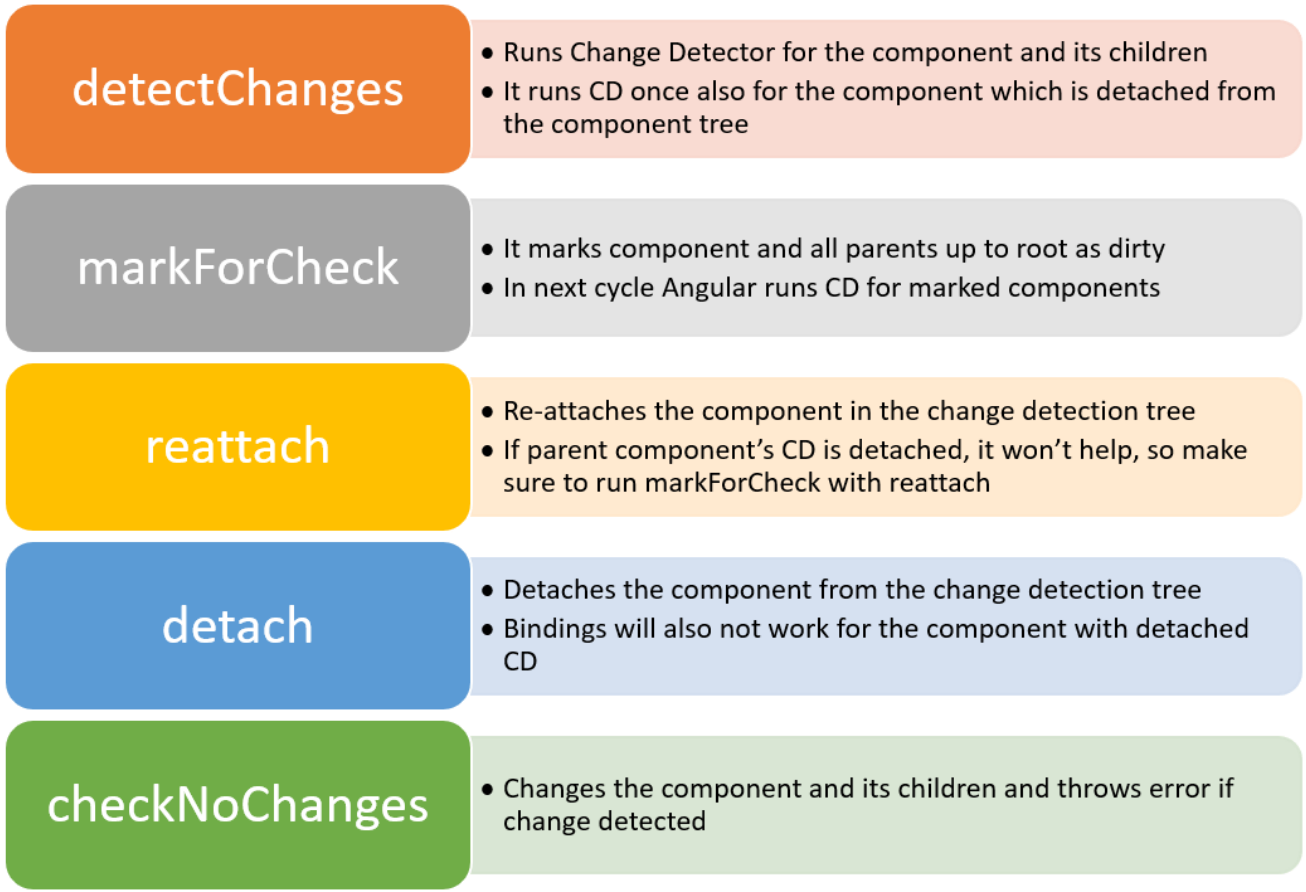
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Set OnPush as default

- Add to angular.json schematic / project.json generator
- Add a lint rule

One more thing: ChangeDetectorReference



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

Summary

- Event Bindigs → Property Bindings
  - Two-way bindings
- CD Strategy OnPush (as default)
  - ChangeDetectorReference

For a performance deep dive

Check out my special workshop

https://www.angulararchitects.io/schulungen/angular-performance-workshop/