ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Data Binding and Change Detection

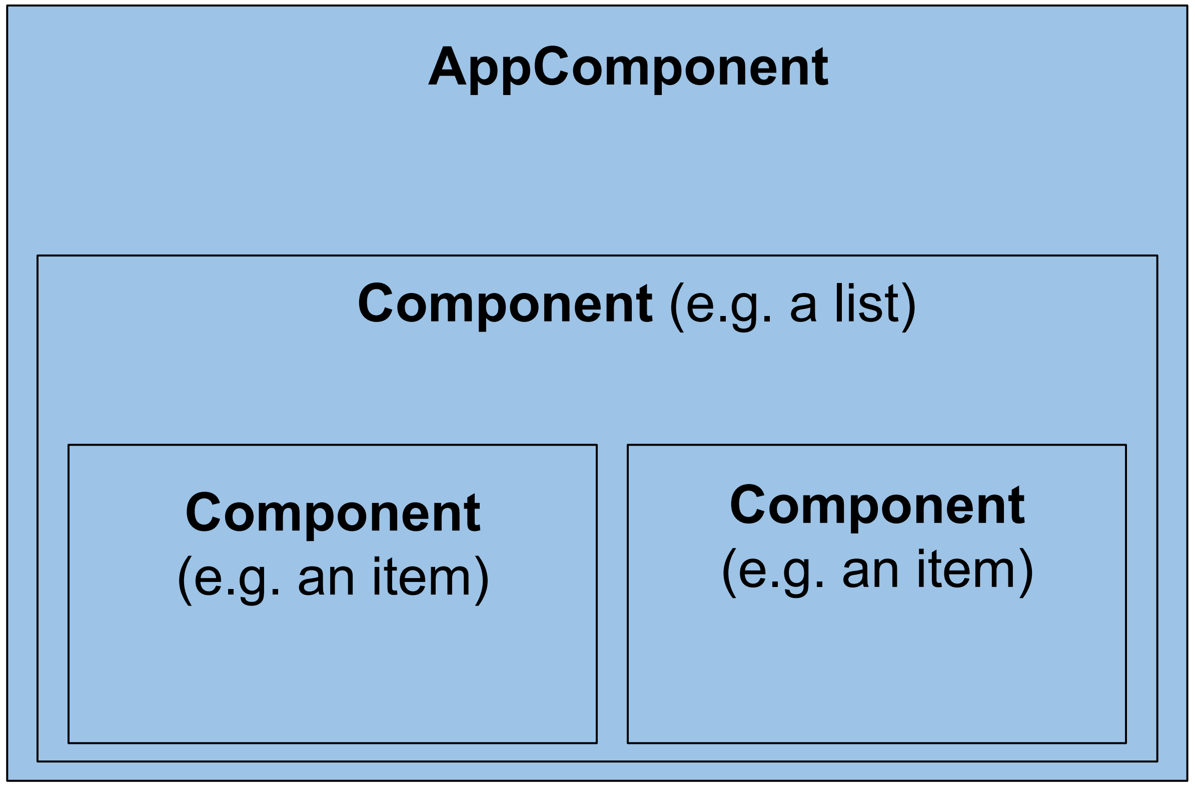
Alex Thalhammer

Contents

- How does data binding work (underneath the covers)?
- Performance-Tuning with OnPush

Data Binding

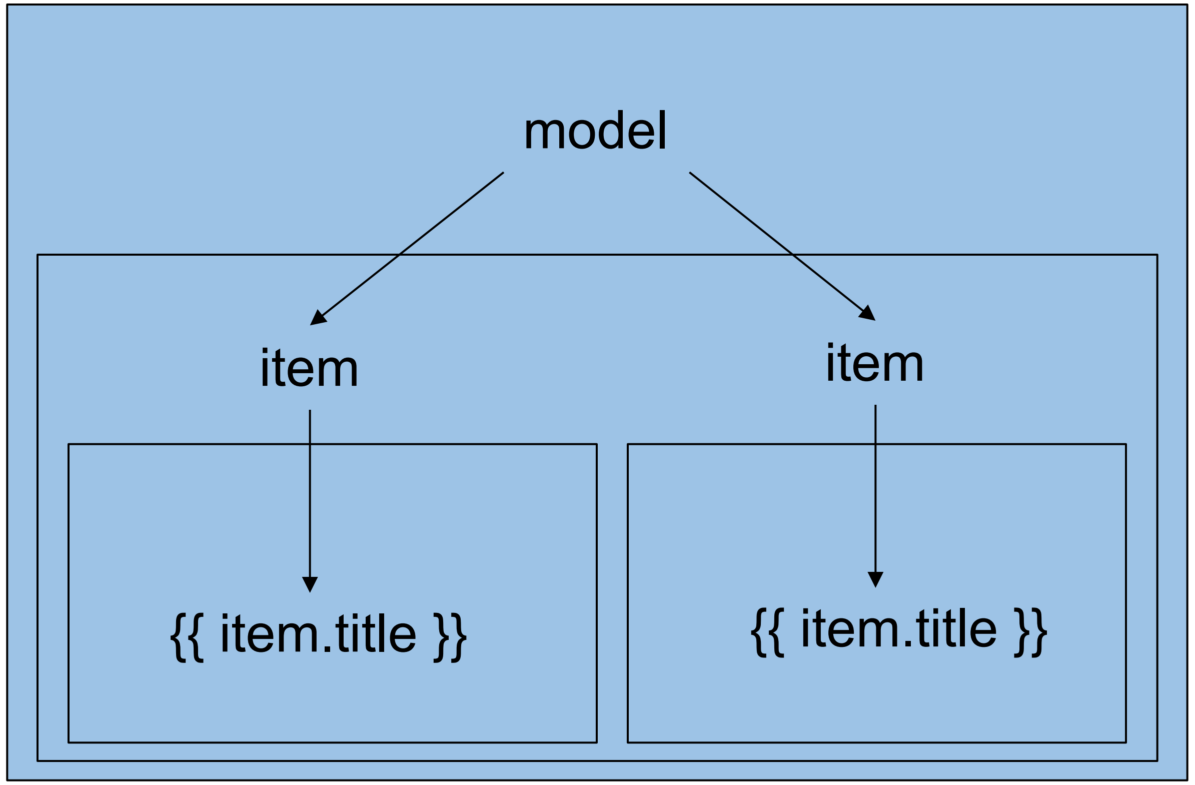
Component Tree in Angular 2+



Rules for Property-Bindings

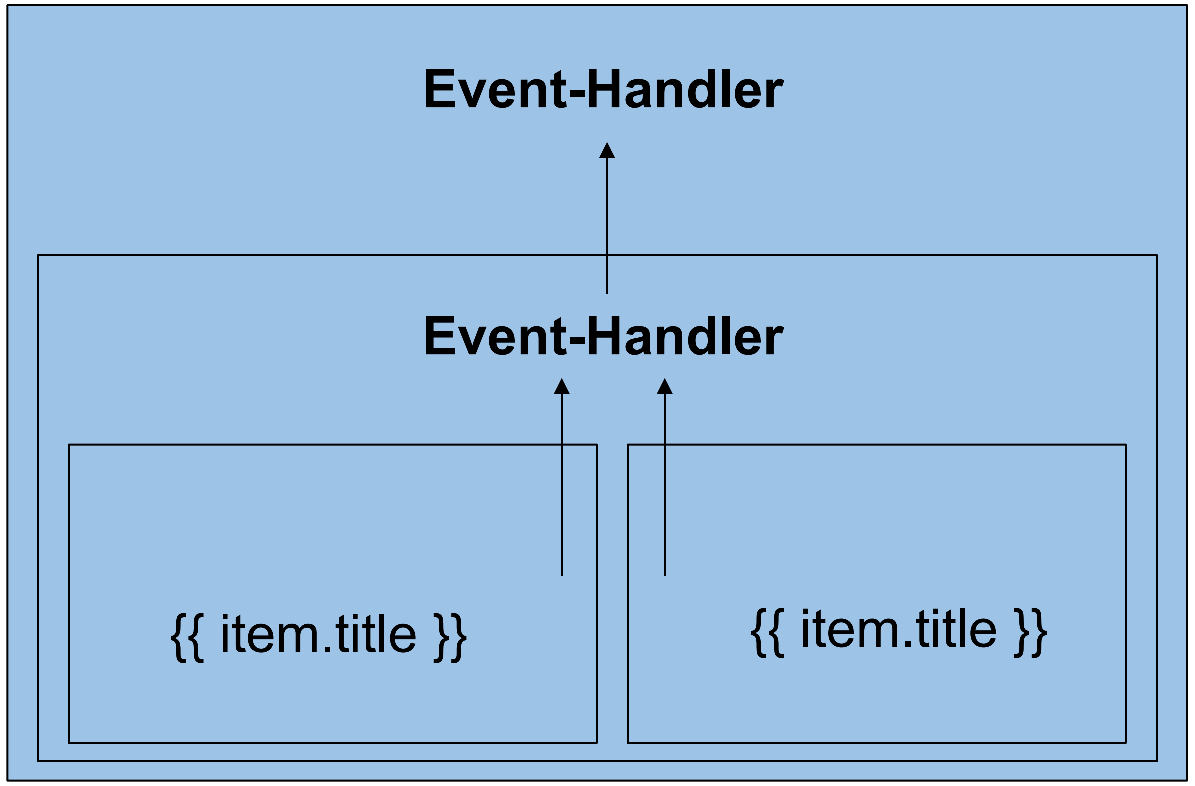
- Data flows top/down
  - Parent can send data to children
  - Children **cannot** send data to parent
- Dependency graph is a tree
- Angular only needs one "digest"

Property Binding



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

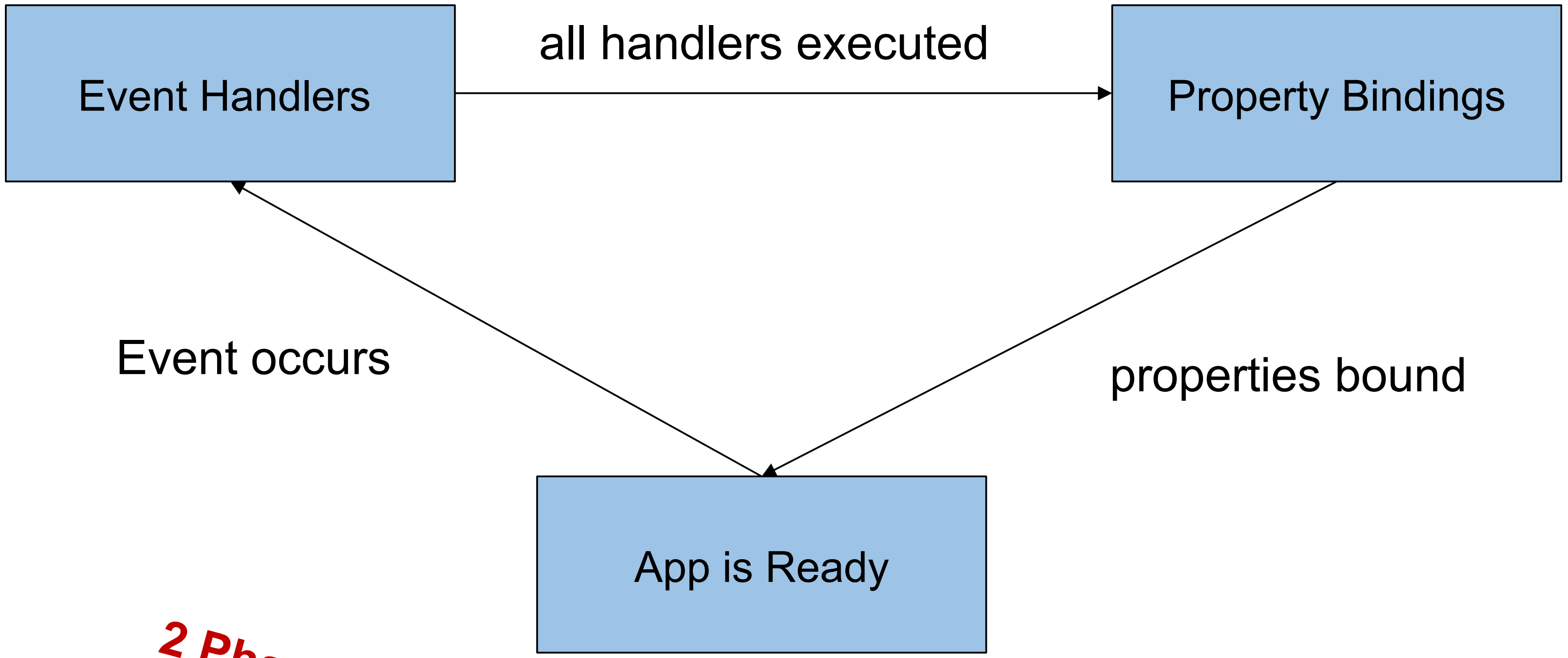
Event Bindings (One-Way, Bottom/Up)



Event Bindings (One-Way, Bottom/Up)

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

Property- and Event-Bindings



2 Phases!

View

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

<table>
    <tr *ngFor="let flight of flights">
        <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>
```

```
<td [text-content]="flight.date"></td>
```

DEMO

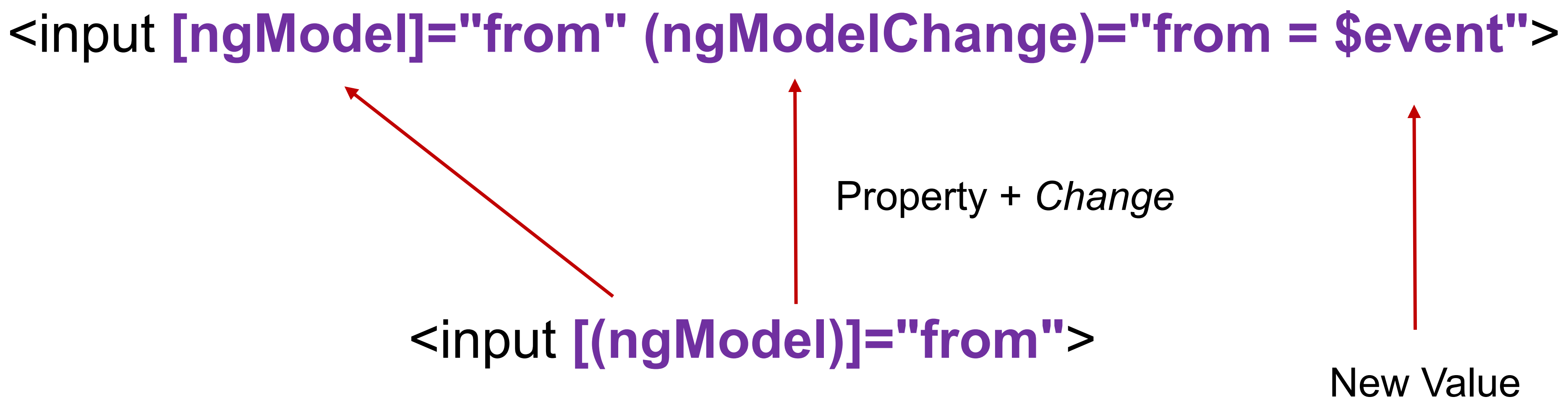
Recap

- 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



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

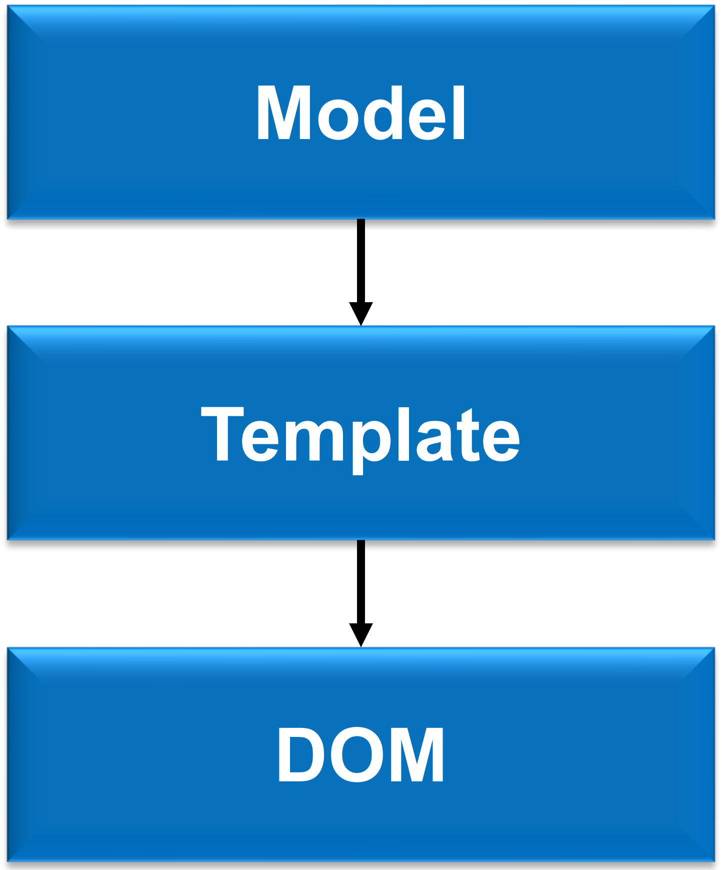
```
<input [(ngModel)]="from">
```

DEMO

Change Detection in Angular

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

DOM Rendering



Change Detection

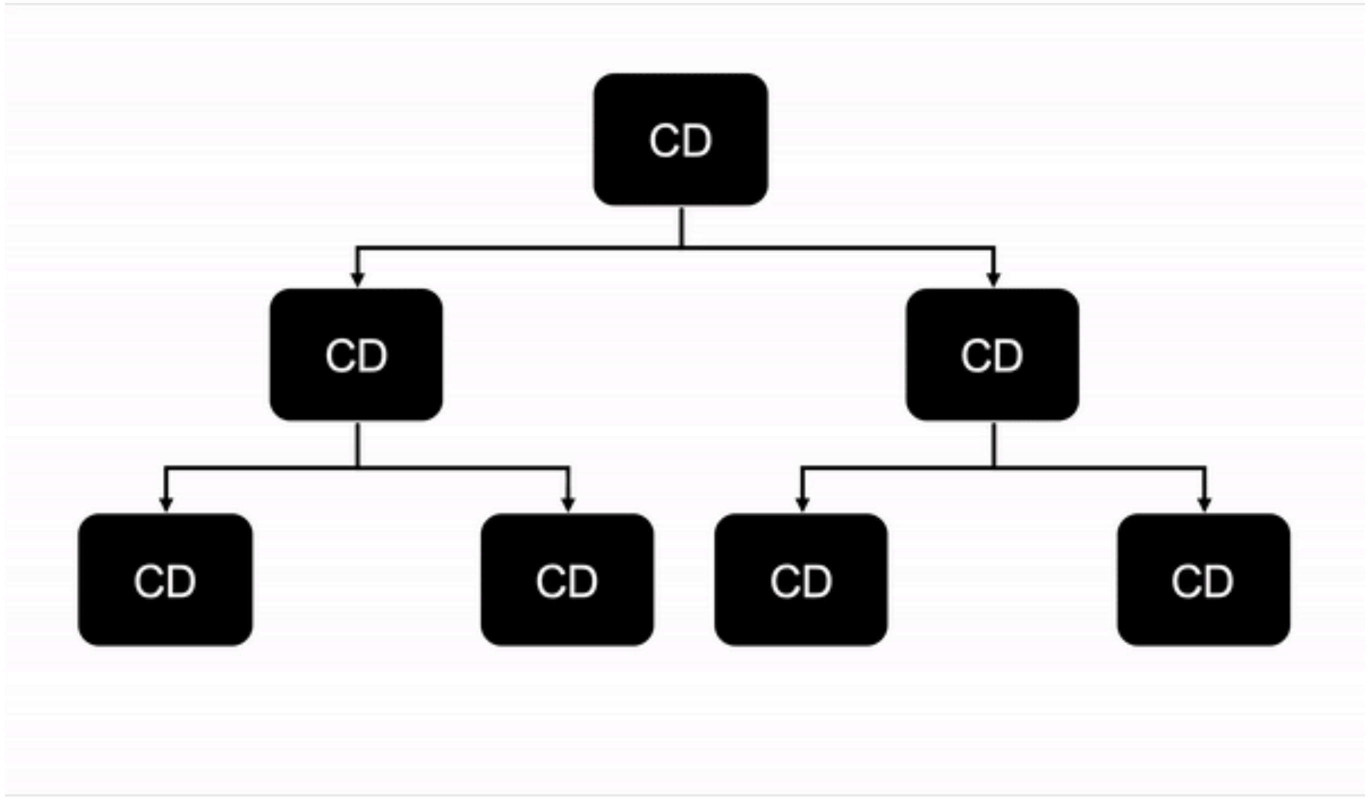
- 1.) User or App changes the model (e.g. @Input() Binding)

- 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)

DEMO

Change Detection – From Root To Leaves



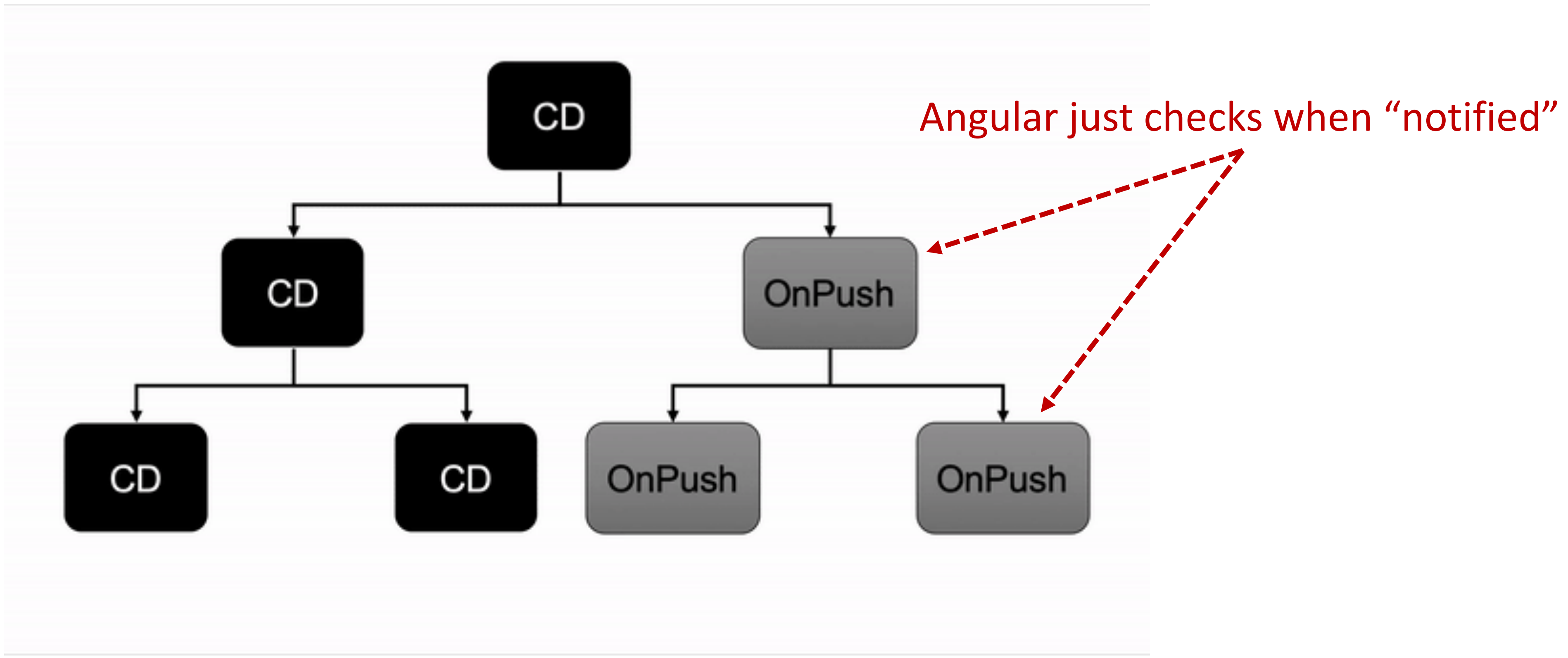
https://mokkapps.de/blog/the-last-guide-for-angular-change-detection-you-will-ever-need/

Performance-Tuning with OnPush

ANGULAR ARCHITECTS INSIDE KNOWLEDGE

SOFTWARE ARCHITECT

Change Detection – OnPush Strategy



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

"Notify" about change?

- Change bound data (@Input)
  - OnPush: Angular just compares the object reference!
  - e. g. oldFlight !== newFlight (BTW: like ngOnChanges)
- Raise event within the component and its children (e.g. @Output)
- Emit in a bound observable into the async pipe
  - {{ flights$ | async }}
- Do it manually (cdr.markForCheck())
  - Don't do this at home ;-)
  - Try to avoid this – but there are reasonable cases

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()

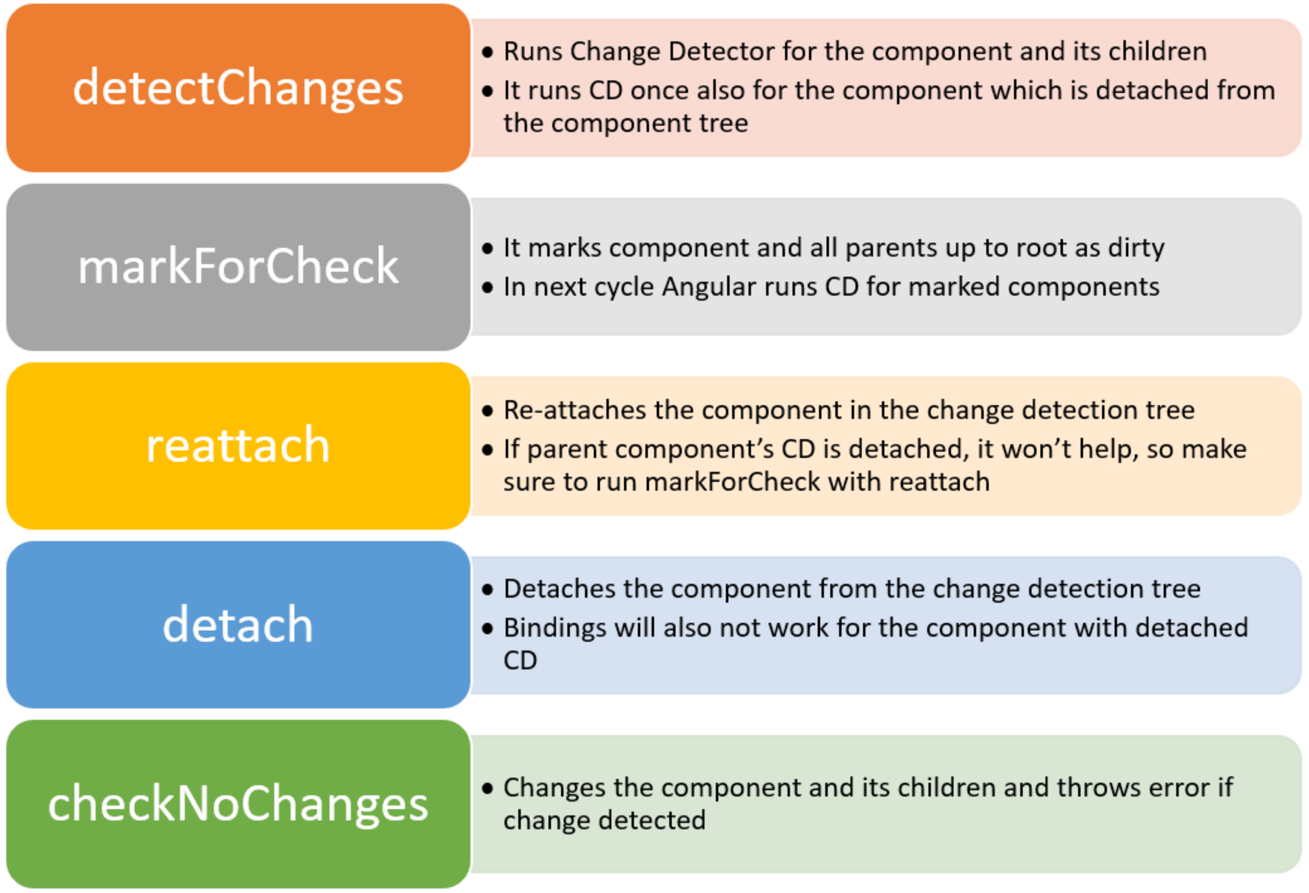
Activate OnPush Strategy

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

DEMO

LAB

One more thing: ChangeDetectorReference



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

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

For a performance deep dive

Check out my special workshop

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

Summary

- Event Bindigs → Property Bindings
  - Two-way bindings
- CD Strategy OnPush
  - ChangeDetectorReference