



Angular Komponenten

Hosted by Alex Thalhammer

Inhalt

- Datenbindung näher betrachtet
- Komponenten mit Bindings
- Life Cycle Hooks



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Datenbindung näher betrachtet



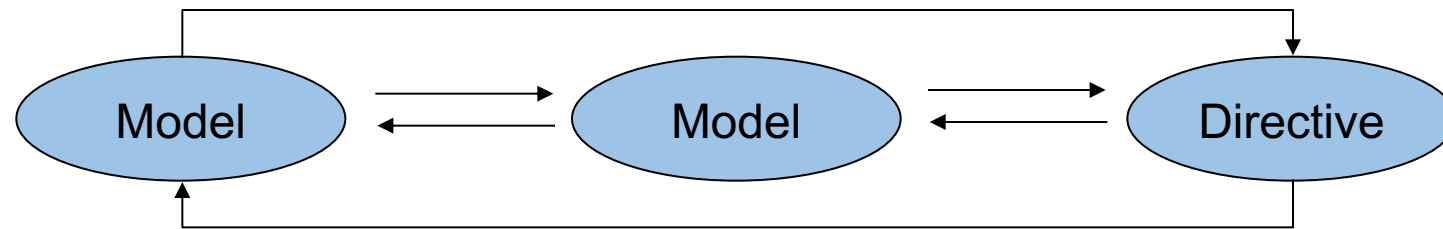
Performance

Komponenten

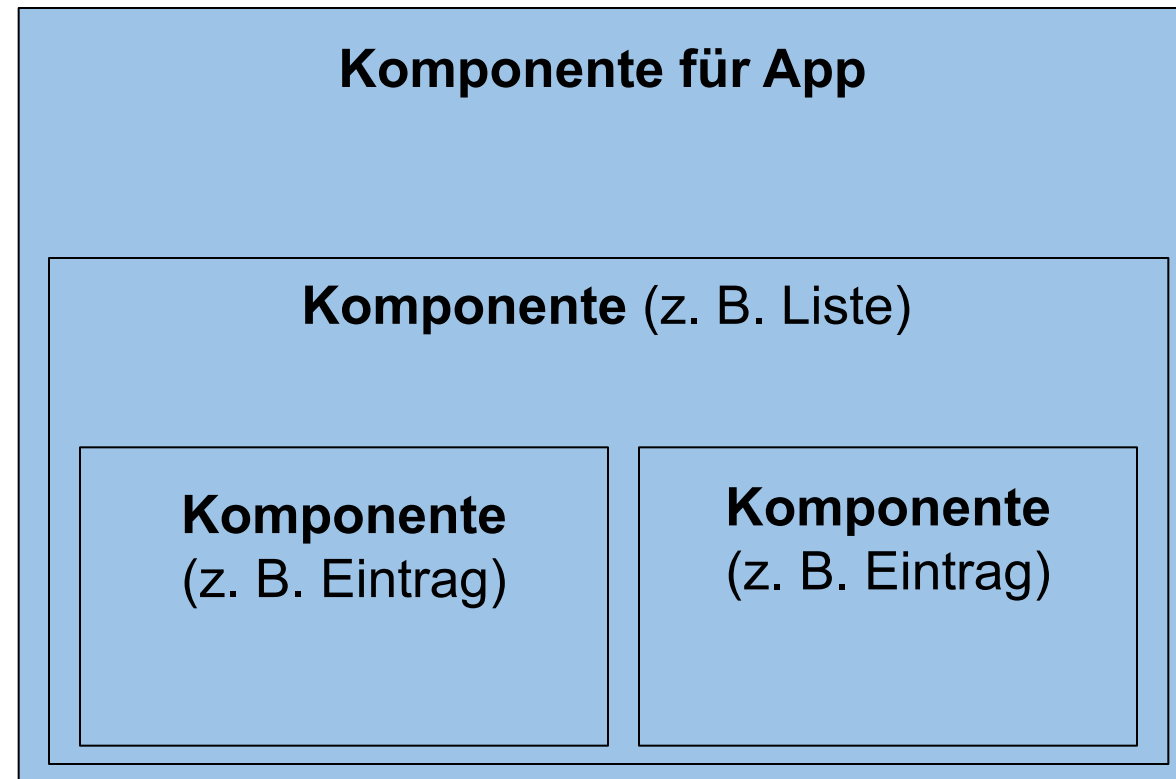
Vorhersagbarkeit

Architektur-Ziele von Angular

Data-Binding in AngularJS 1.x



Komponenten-Baum in Angular 2+

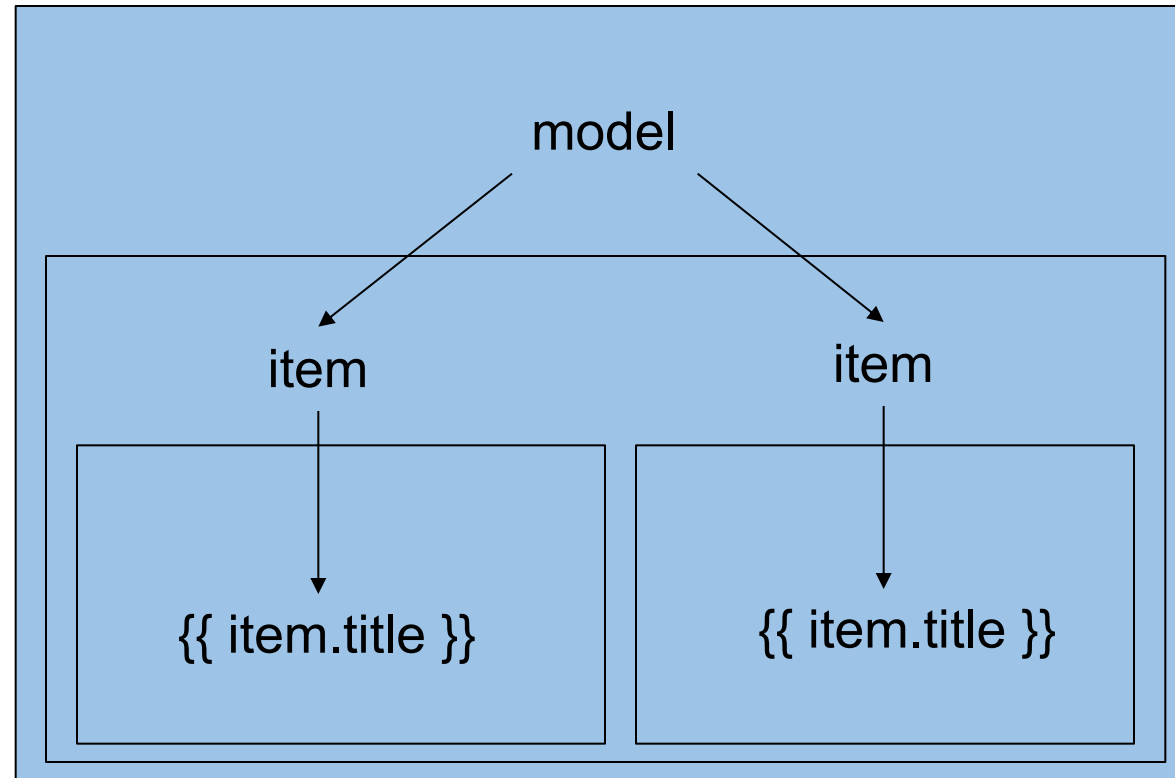


Regeln für Property-Bindings

- Daten fließen von oben nach unten (top/down)
 - Parent kann Daten an Children weitergeben
 - Children können keine Daten an Parent weitergeben
- Abhängigkeits-Graph ist ein Baum
- Angular benötigt nur einen Digest um Baum mit GUI abzugleichen

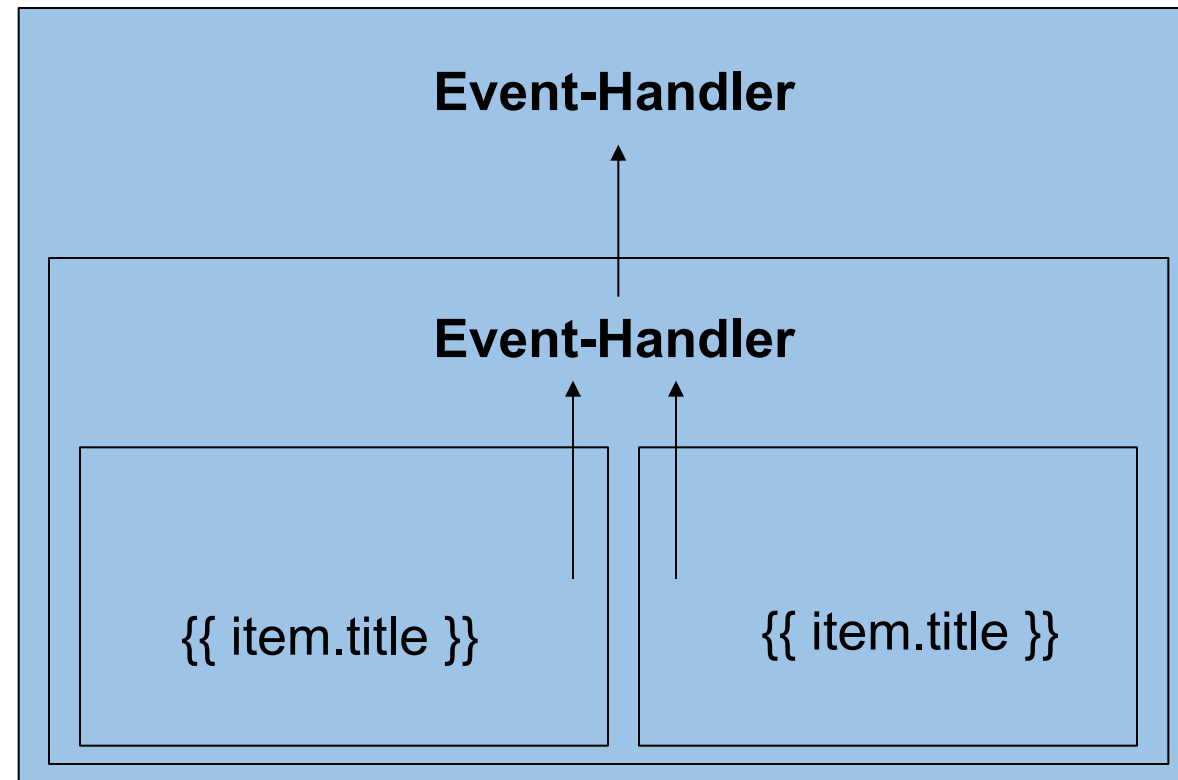


Property-Binding



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

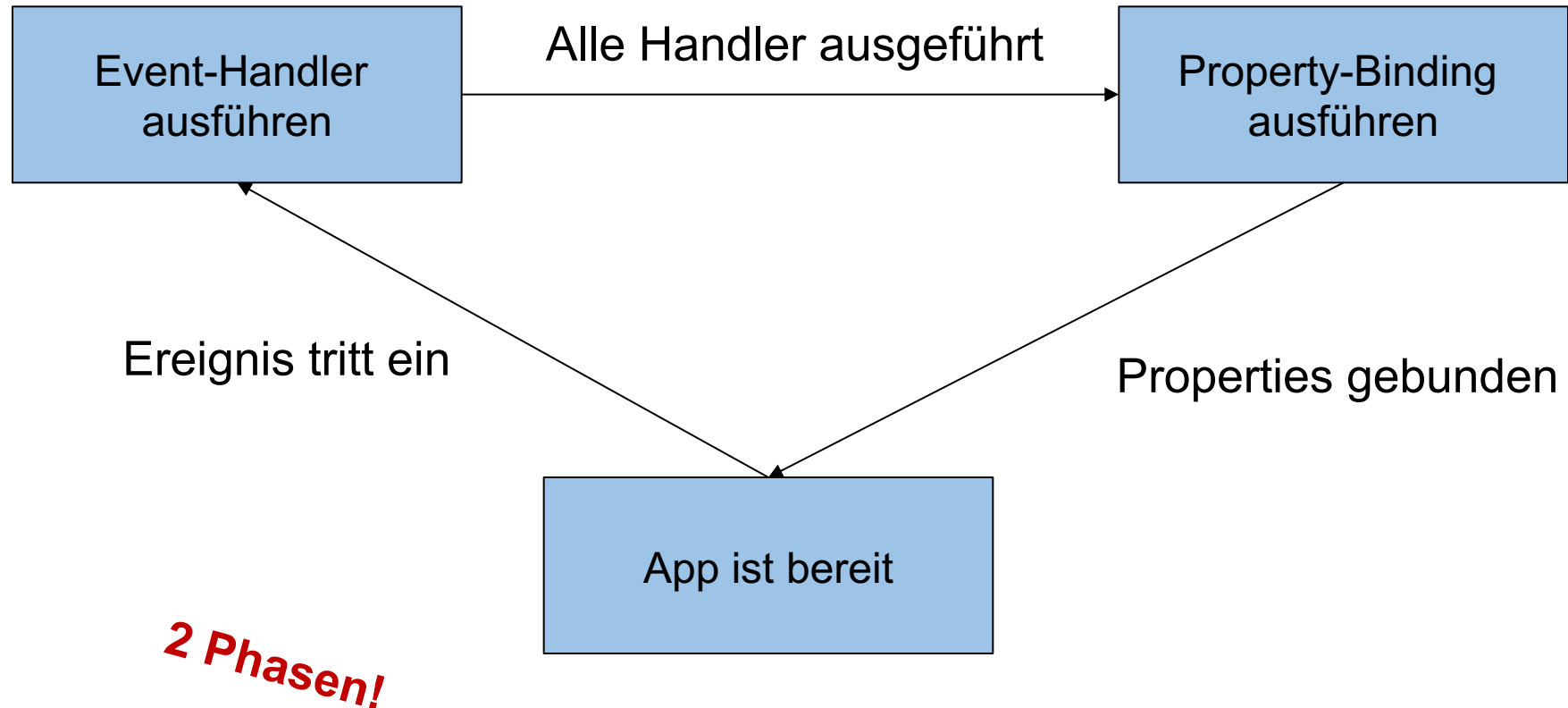
Event-Bindings (One-Way, Bottom/Up)



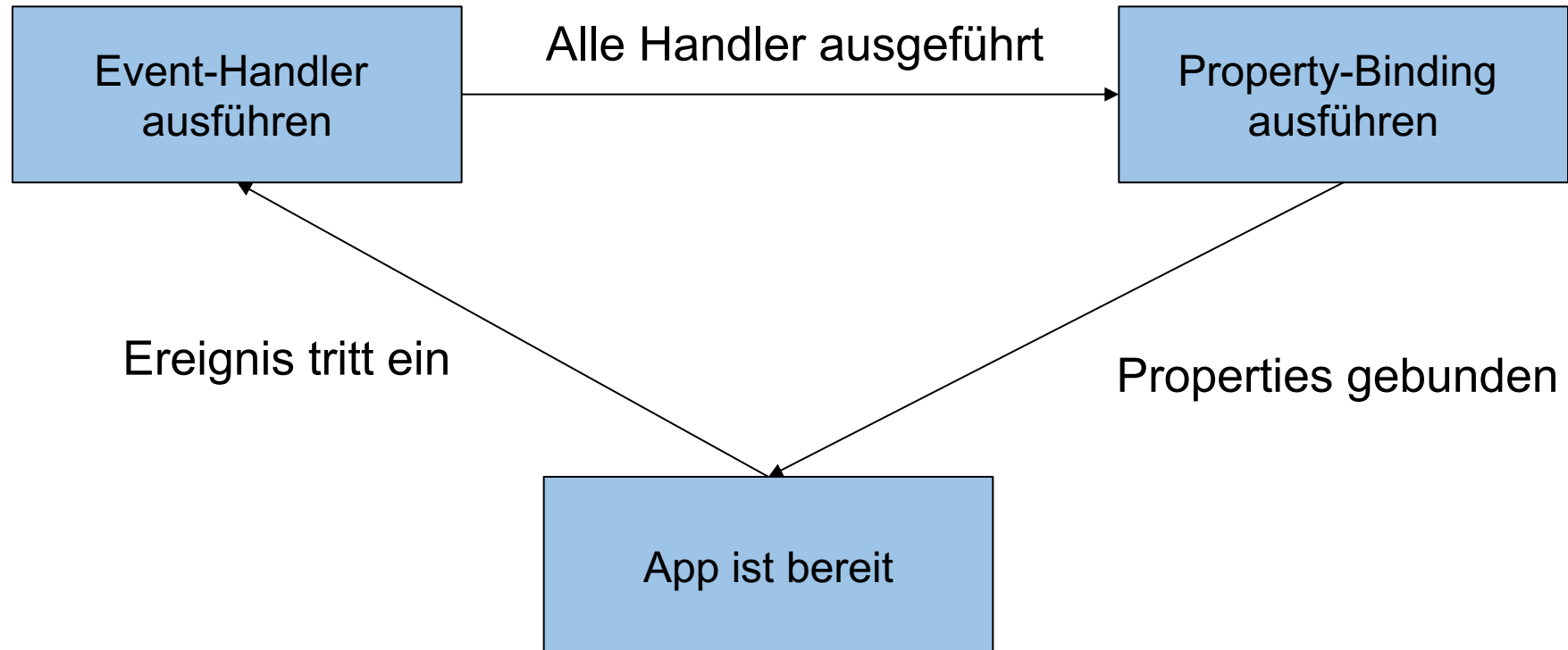
Event-Bindings (One-Way, Bottom/Up)

- Kein Digest um Events zu versenden
- Aber: Events können Daten ändern → Property Binding

Property- und Event-Bindings



Property- und Event-Bindings



View

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

```
<table>  
  <tr *ngFor="let flight of flights">  
    <td>{{flight.id}}</td>  
    <td>{{flight.date}}</td> ← - - - - - > <td [text-content]="flight.date"></td>  
    <td>{{flight.from}}</td>  
    <td>{{flight.to}}</td>  
    <td><a href="#" (click)="selectFlight(flight)">Select</a></td>  
  </tr>  
</table>
```



Recap

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

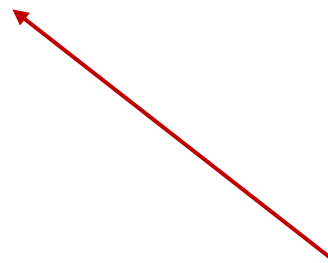


Property und Event-Bindings

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

Property und Event-Bindings

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



Property + *Change*

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



Geänderter Wert





Komponenten mit Bindings

Beispiel: flug-card

Hamburg - Graz

Flugnr. #3

Datum: 14.01.2017

Entfernen

Hamburg - Graz

Flugnr. #4

Datum: 14.01.2017

Auswählen

Hamburg - Graz

Flugnr. #5

Datum: 14.01.2017

Auswählen

Beispiel: flug-card

Hamburg - Graz

Flugnr. #3

Datum: 14.01.2017

Entfernen

Hamburg - Graz

Flugnr. #4

Datum: 14.01.2017

Auswählen

Hamburg - Graz

Flugnr. #5

Datum: 14.01.2017

Entfernen

Warenkorb

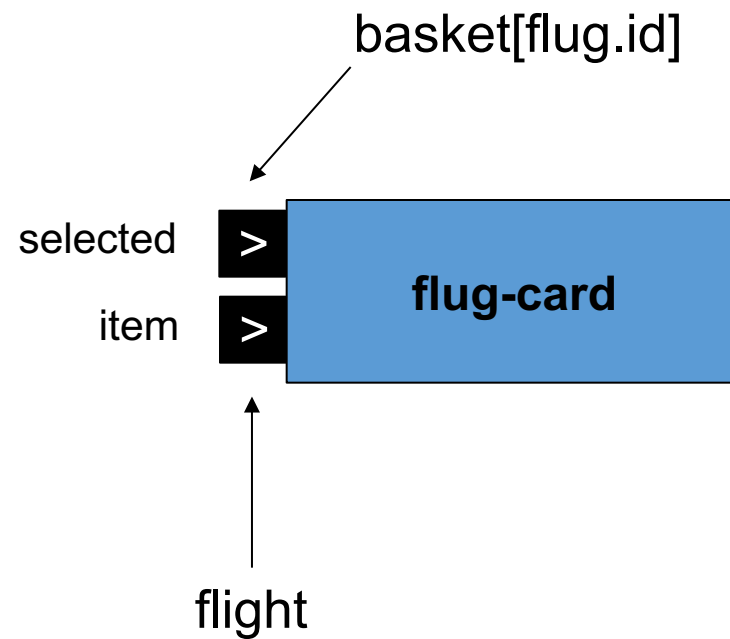
```
{  
  "3": true,  
  "4": false,  
  "5": true  
}
```

```
public basket = {};  
[...]  
basket[3] = true;  
basket[4] = false;  
basket[5] = true;
```

Beispiel: flug-card in flug-suchen.html

```
<div *ngFor="let f of fluege">  
  
  <flug-card [item]="f" [selected]="basket[f.id]">  
  </flug-card>  
  
</div>
```

flug-card



Beispiel: flug-card

```
@Component({  
  selector: 'flug-card',  
  templateUrl: './flug-card.html'  
})  
export class FlugCard {  
  
  [...]  
  
}
```



Beispiel: flug-card

```
export class FlugCard {  
  
    @Input() item: Flug;  
    @Input() selected: boolean;  
  
    select() {  
        this.selected = true;  
    }  
  
    deselect() {  
        this.selected = false;  
    }  
}
```



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Template

```
<div style="padding:20px;"
    [ngStyle]="{ 'background-color':
        (selected) ? 'orange' : 'lightsteelblue' }" >

    <h2>{{item.from}} - {{item.to}}</h2>
    <p>Flugnr. #{{item.id}}</p>
    <p>Datum: {{item.date | date:'dd.MM.yyyy'}}</p>

    <p>
        <button *ngIf="!selected" (click)="select()">Auswählen</button>
        <button *ngIf="selected" (click)="deselect()">Entfernen</button>
    </p>
</div>
```



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

Komponente registrieren

```
@NgModule({  
  imports: [  
    CommonModule, FormsModule, SharedModule  
  ],  
  declarations: [  
    AppComponent, FlugSuchenComponent, FlugCardComponent  
  ],  
  providers: [  
    FlugService  
  ],  
  bootstrap: [  
    AppComponent  
  ]  
})  
export class AppModule {  
}
```



DEMO





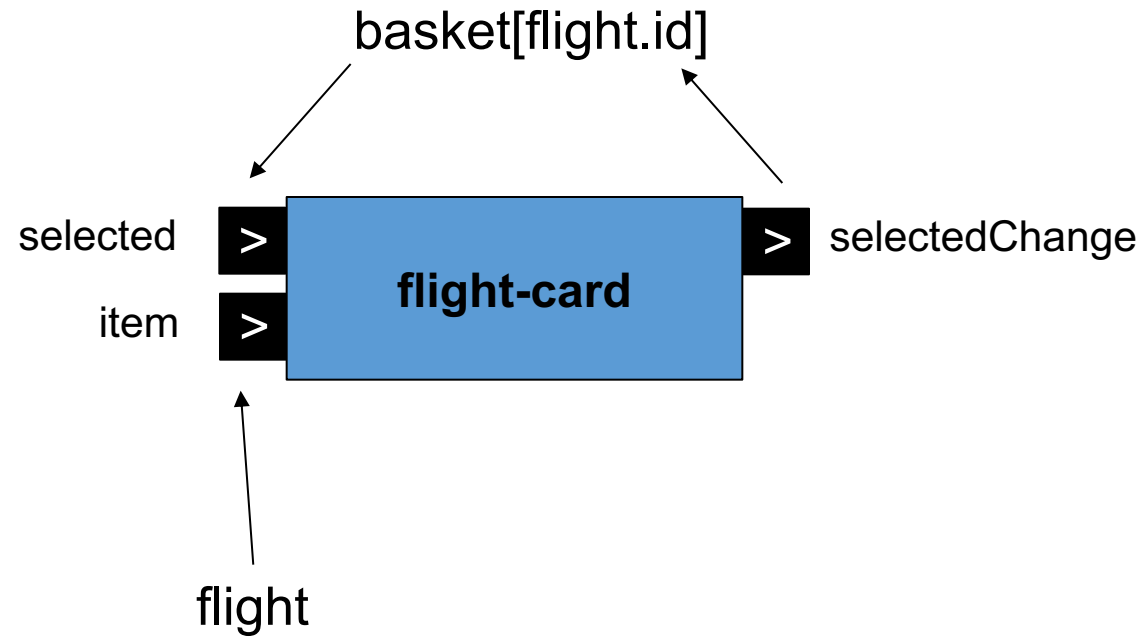
Event-Bindings

flug-card mit Event *selectedChange*

```
<div *ngFor="let f of fluege">  
  
  <flug-card [item]="f"  
    [selected]="basket[f.id]"  
    (selectedChange)="basket[f.id] = $event">  
  
  </flug-card>  
  
</div>
```



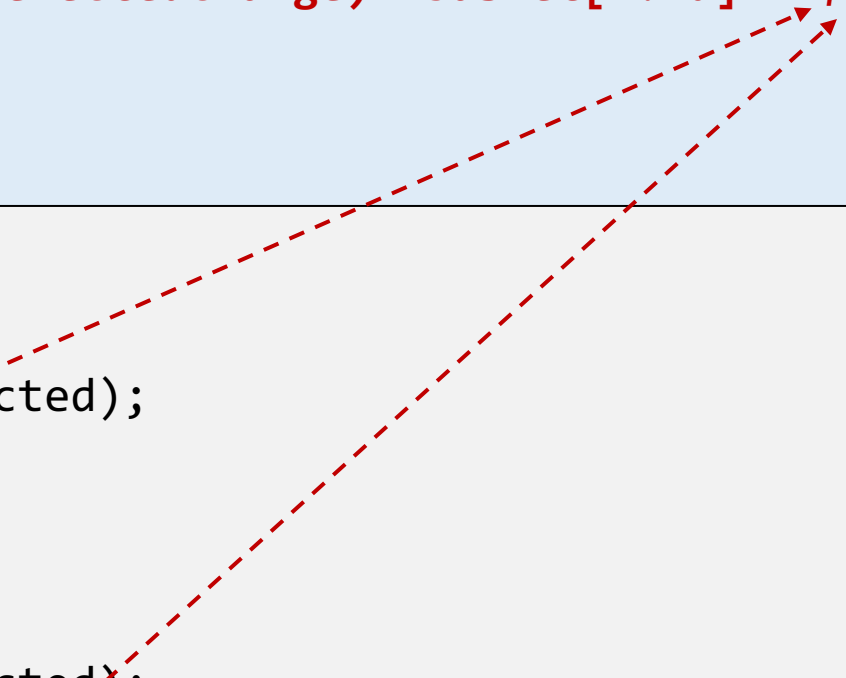
flight-card



Beispiel: flug-card

```
export class FlugCard {  
  
  @Input() item: Flug;  
  @Input() selected: boolean;  
  @Output() selectedChange  
  
  select() {  
    this.selected = true;  
    this.selectedChange.next(this.selected);  
  }  
  
  deselect() {  
    this.selected = false;  
    this.selectedChange.next(this.selected);  
  }  
}
```

```
<div *ngFor="let f of fluege">  
  <flug-card [item]="f"  
    [selected]="basket[f.id]"  
    (selectedChange)="basket[f.id] = $event">  
  </flug-card>  
</div>
```



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT

DEMO



LAB



Gedankenexperiment

- Was wäre, wenn <flug-card> Use-Case-Logik übernehmen würde?
 - z. B. mit Backend kommuniziert
- Nachvollziehbarkeit?
- Anzahl Zugriffe ==> Performance?
- Wiederverwendbarkeit?

Smart vs. Dumb Components

Smart Component

- Use Case Steuerung
- Meist Container

Dumb

- Unabhängig von Use Case
- Wiederverwendbar
- Meist Blatt



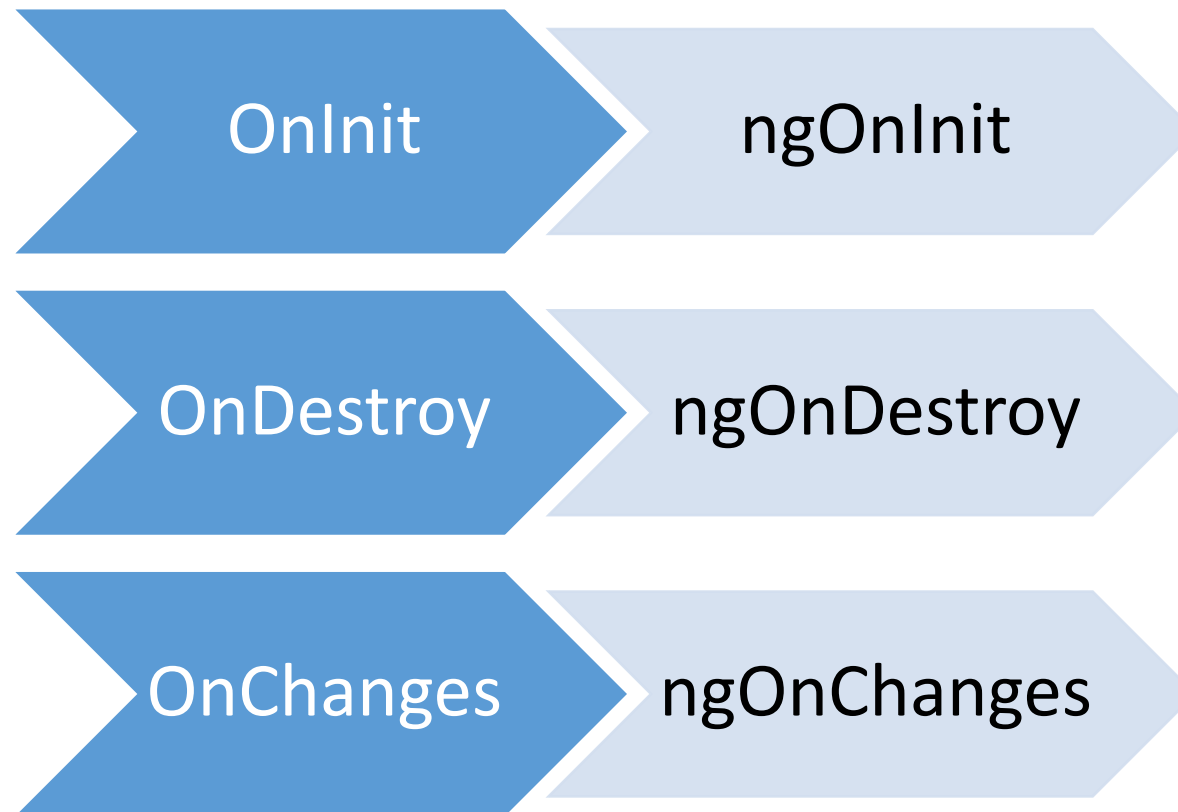
Life Cycle Hooks



Was sind Life Cycle Hooks

- Methoden in Komponenten
- Werden zu bestimmten Zeitpunkten von Angular aufgerufen

Life-Cycle-Hooks (Auswahl)



Nutzung

```
@Component({  
  selector: 'my-component',  
  [...]  
})  
export class Component implements OnChanges, OnInit {  
  
  @Input() someData;  
  
  ngOnInit() {  
    [...]  
  }  
  
  ngOnChanges() {  
    [...]  
  }  
}
```



DEMO



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



SOFTWARE
ARCHITECT