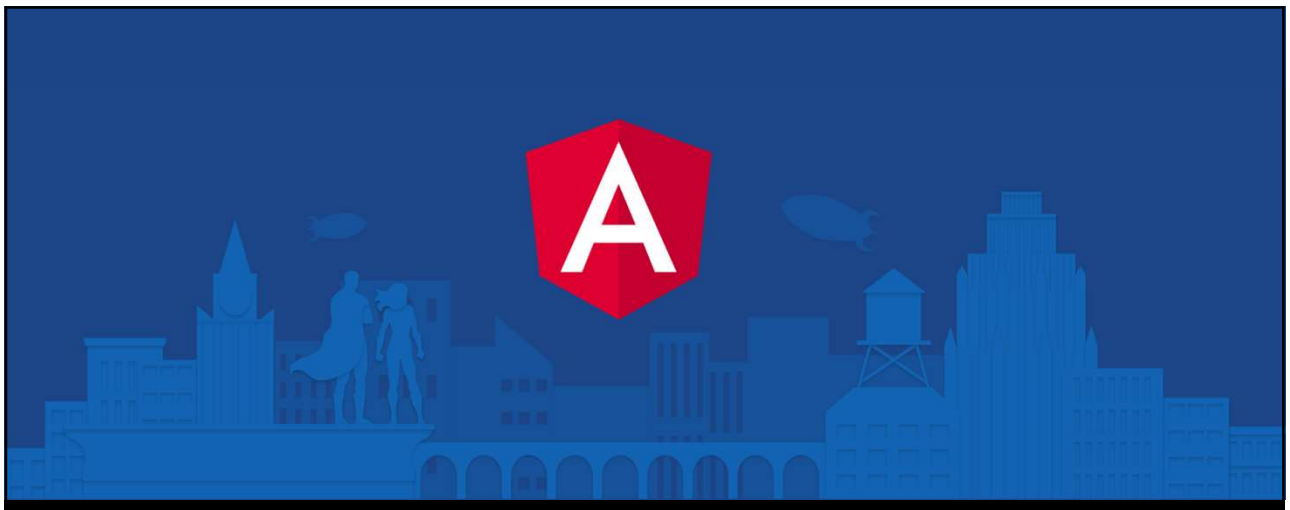


Eigene Komponenten mit Angular entwickeln

Michael Zikes
SOFTWAREarchitekt.at

Inhalt

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

Datenbindung näher betrachtet

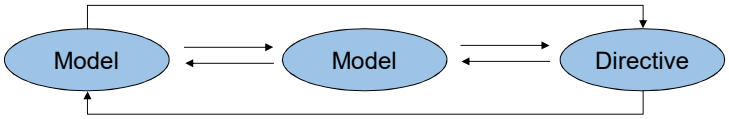
Performance

Komponenten

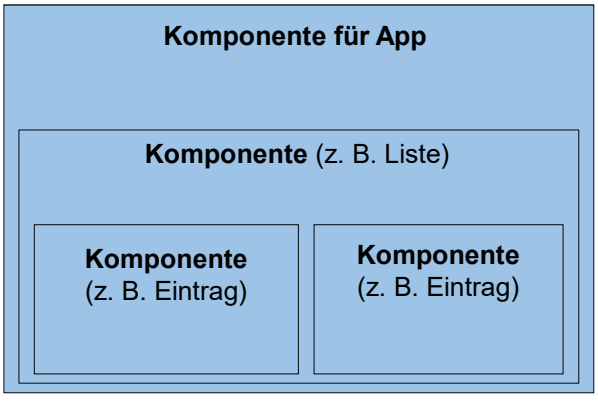
Vorhersagbarkeit

Architektur-Ziele von Angular

Data-Binding in AngularJS 1.x



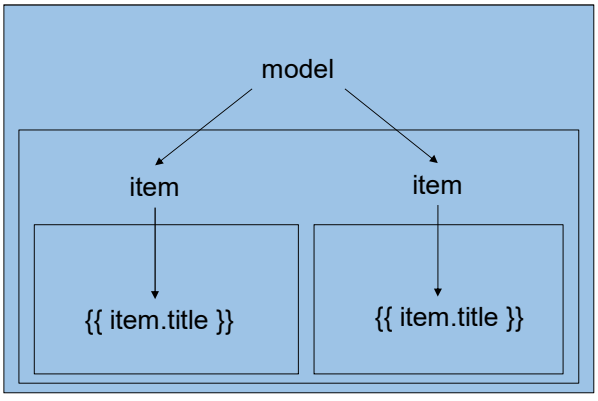
Komponenten-Baum in Angular 2+



Regeln für Property-Bindings

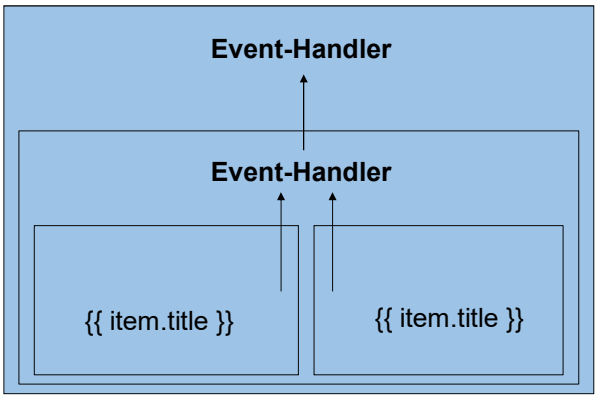
- Daten fließen von oben nach unten (top/down)
  - Parent kann Daten an Children weitergeben
  - Children können keine Daten an Parent weitergeben
- Abhängigkeits-Graph ist ein Baum
- Angular benötigt nur einen Digest um Baum mit GUI abzugleichen

Property-Binding



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

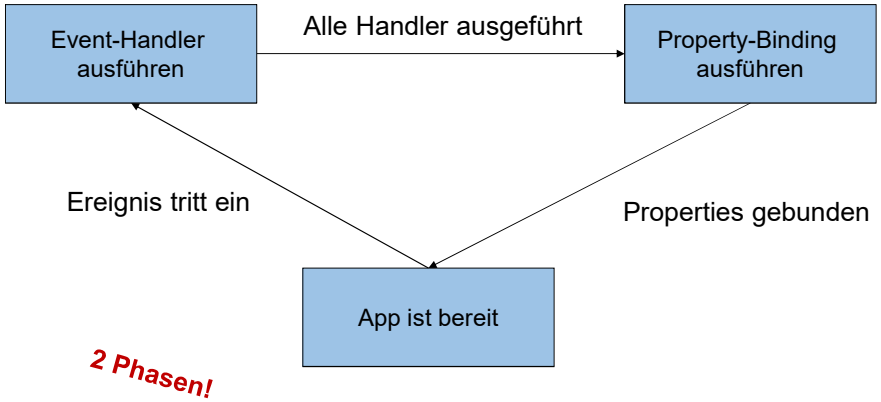
Event-Bindings (One-Way, Bottom/Up)



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

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

Property- und Event-Bindings



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

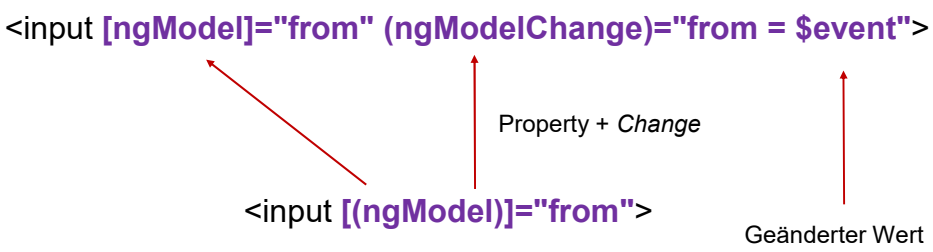
Recap

- 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

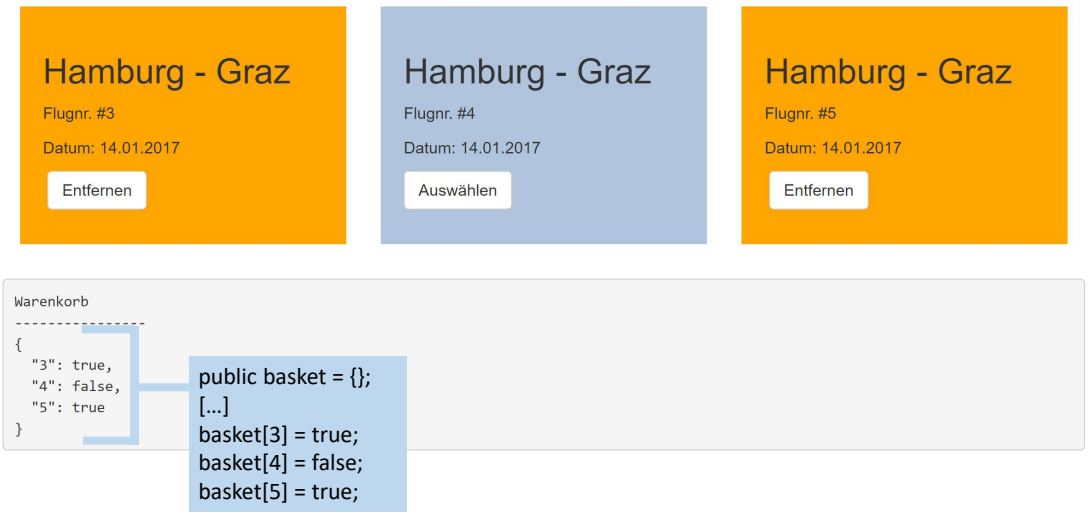


SOFTWARE *architekt.at*

Komponenten mit Bindings

Beispiel: flight-card

Beispiel: flight-card



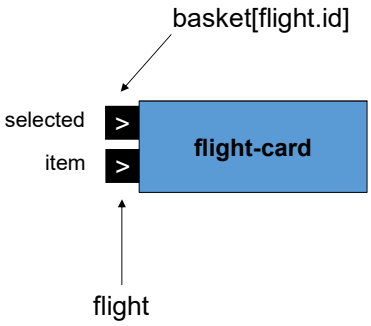
Beispiel: flight-card in flight-search.html

```
<div *ngFor="let f of flight">

   <flight-card [item]="f" [selected]="basket[f.id]">
   </flight-card>

</div>
```


flight-card



Beispiel: flight-card

```
@Component({
	selector: 'flight-card',
	templateUrl: './flight-card.component.html'
})
export class FlightCardComponent {

	[…]

}
```


Beispiel: flight-card

```
export class FlightCard {

	@Input() item: Flight;
	@Input() selected: boolean;

	select() {
		this.selected = true;
	}

	deselect() {
		this.selected = false;
	}
}
```

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()">Select</button>
        <button *ngIf="selected" (click)="deselect()">Deselect</button>
    </p>
</div>
```


Komponente registrieren

```
@NgModule({
    imports: [
        CommonModule, FormsModule, SharedModule
    ],
    declarations: [
       AppComponent, FlightSearchComponent, FlightCardComponent
    ],
    providers: [
       // FlightService
    ],
    bootstrap: [
       AppComponent
    ]
})
export class AppModule {
}
```

DEMO

SOFTWARE architekt.at

Event-Bindings

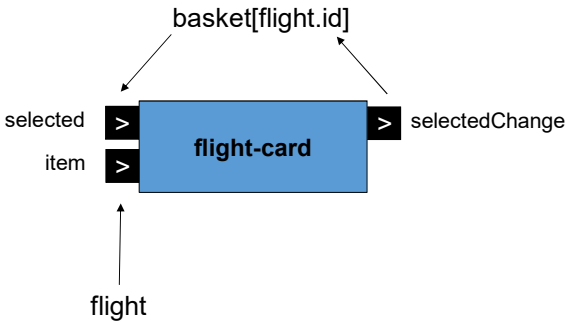
flight-card mit Event *selectedChange*

```
<div *ngFor="let f of flights">

   <flight-card [item]="f"
                [selected]="basket[f.id]"
                (selectedChange)="basket[f.id] = $event">
   </flight-card>

</div>
```


flight-card



Beispiel: flight-car

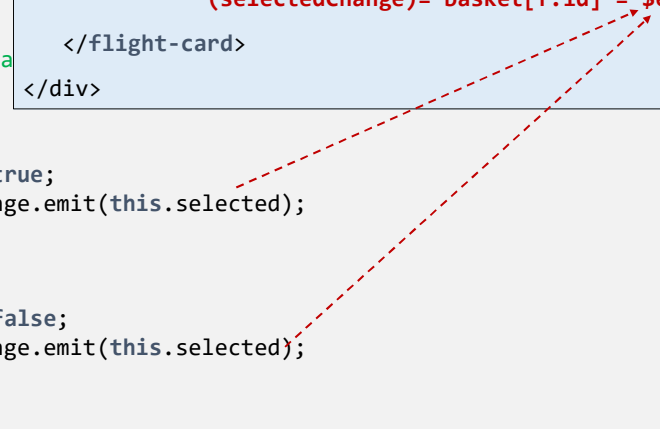
```
export class FlugCard {

        @Input() item: Flight;
        @Input() selected: boolea
        @Output() selectedChange

        select() {
                this.selected = true;
                this.selectedChange.emit(this.selected);
        }

        deselect() {
                this.selected = false;
                this.selectedChange.emit(this.selected);
        }
}
```

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



Gedankenexperiment

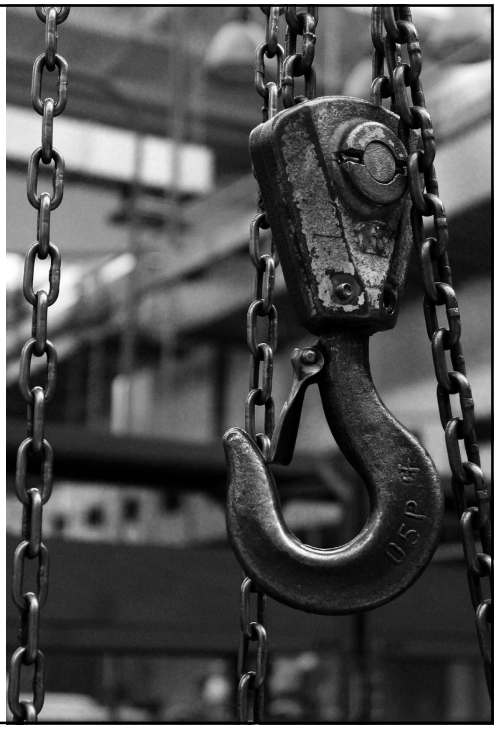
- Was wäre, wenn <flug-card> Use-Case-Logik übernehmen würde?
  - z. B. mit Backend kommuniziert
- Nachvollziehbarkeit?
- Anzahl Zugriffe ==> Performance?
- Wiederverwendbarkeit?

Smart vs. Dumb Components

Smart Component	Dumb
• Use Case Steuerung • Meist Container	• Unabhängig von Use Case • Wiederverwendbar • Meist Blatt

SOFTWARE architekt.at

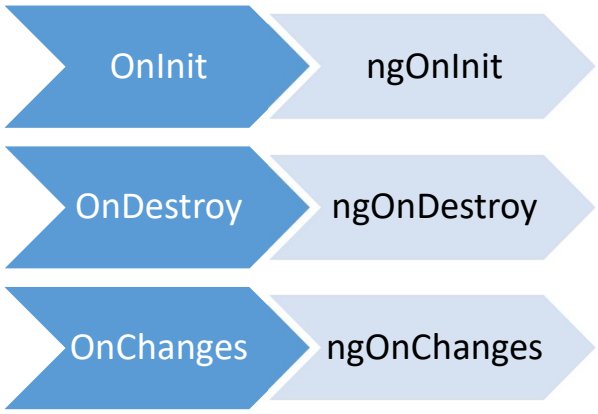
Life Cycle Hooks



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 MyComponent implements OnChanges, OnInit {

    @Input() someData;

    ngOnInit() {
        […]
    }

    ngOnChanges() {
        […]
    }
}
```


DEMO