Angular Components

Hosted by Alex Thalhammer

Outline

- Take a closer look on data binding
  - Property binding with @Input()
  - Event binding with @Output()
- Use component bindings
- Life Cycle Hooks

Data binding

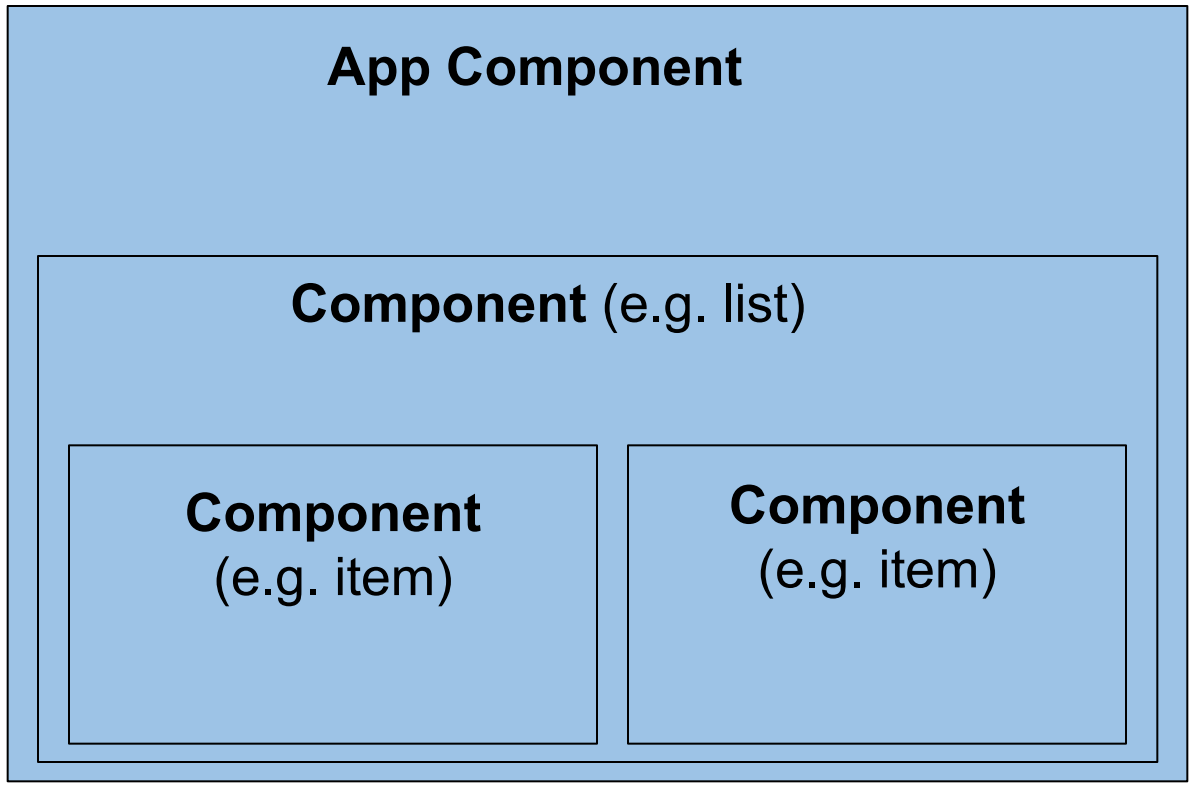
Performance

Components

Predictability

Architecture goals in Angular

Component tree in Angular 2+



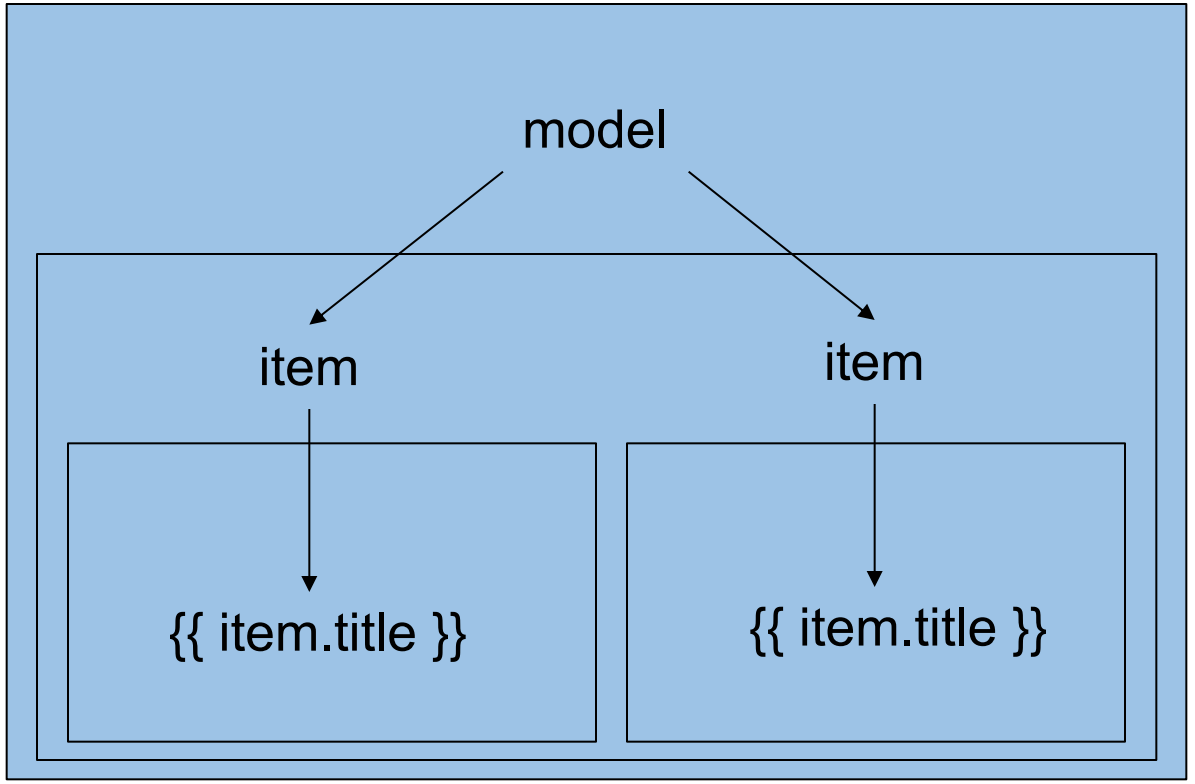
Rules for property binding

- Data can only be passed from top to bottom (top/down)
  - Parent can pass data to children
  - Children cannot pass data to parent (we need events for that)

- Dependency graph is a tree

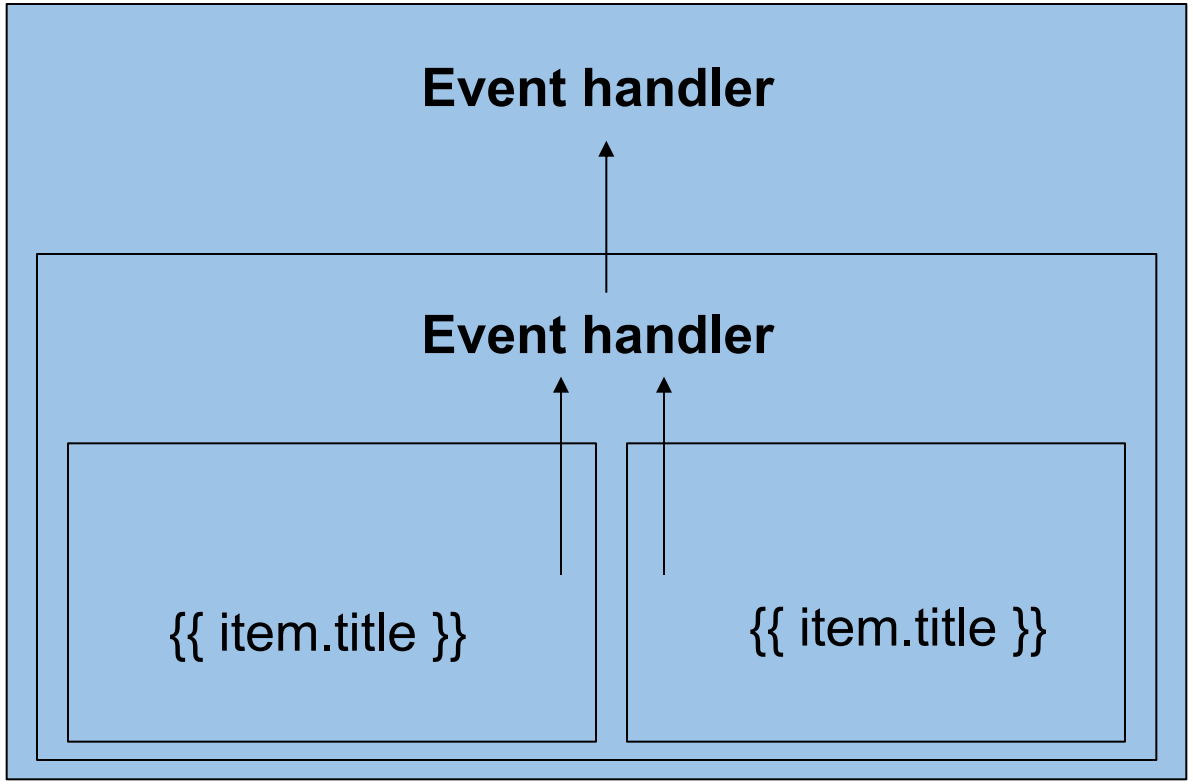
- Angular just takes a digest to compare tree with DOM

Property binding



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

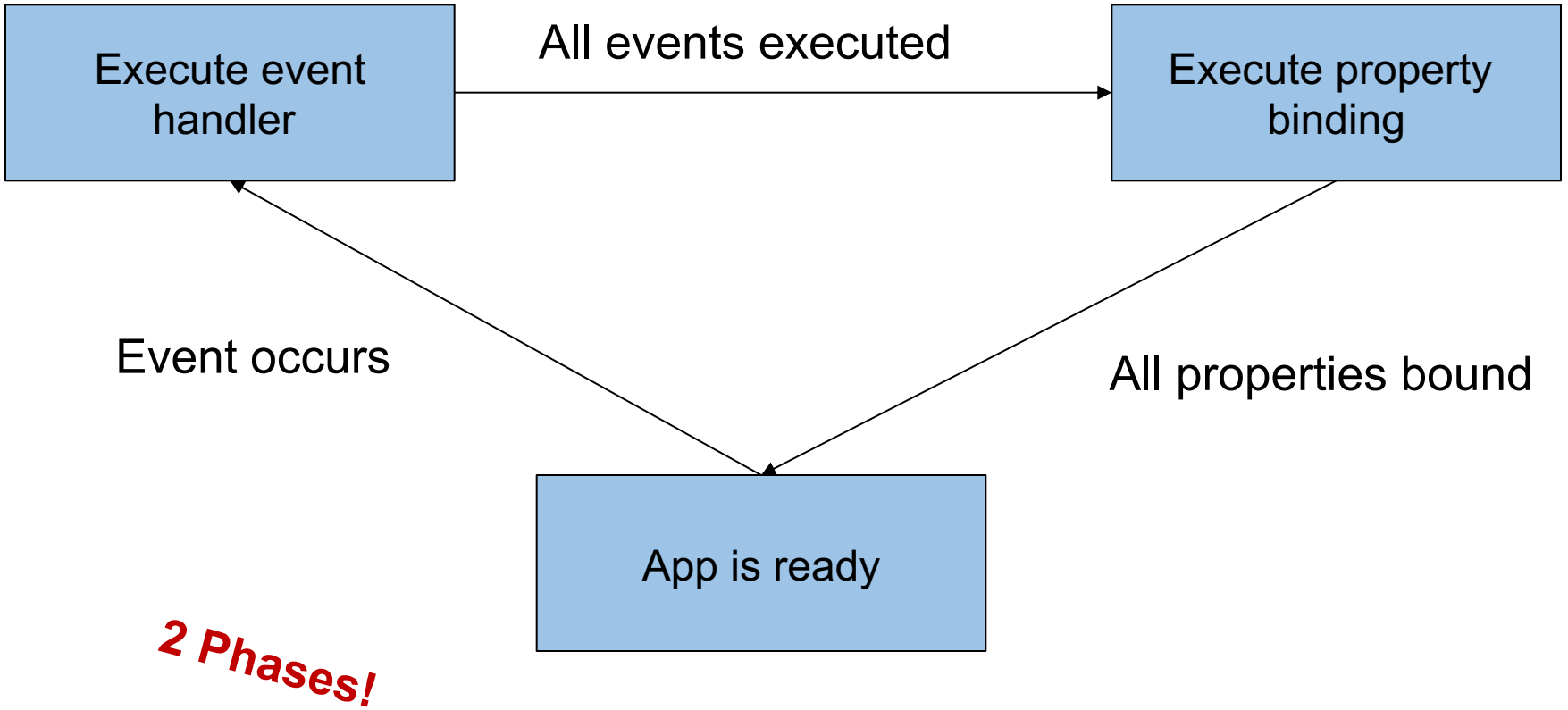
Event bindings (one way, bottom/up)



Event bindings (one way, bottom/up)

- But: Events can trigger data change → Property Binding

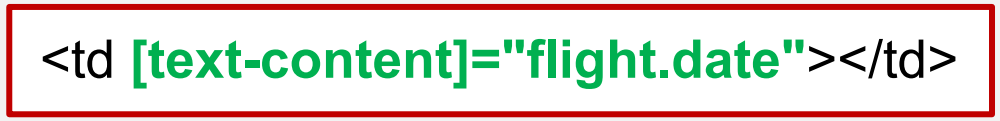
Property and event bindings



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>{{flight.from}}</td>
      <td>{{flight.to}}</td>
      <td><a href="#" (click)="selectFlight(flight)">Select</a></td>
   </tr>
</table>
```



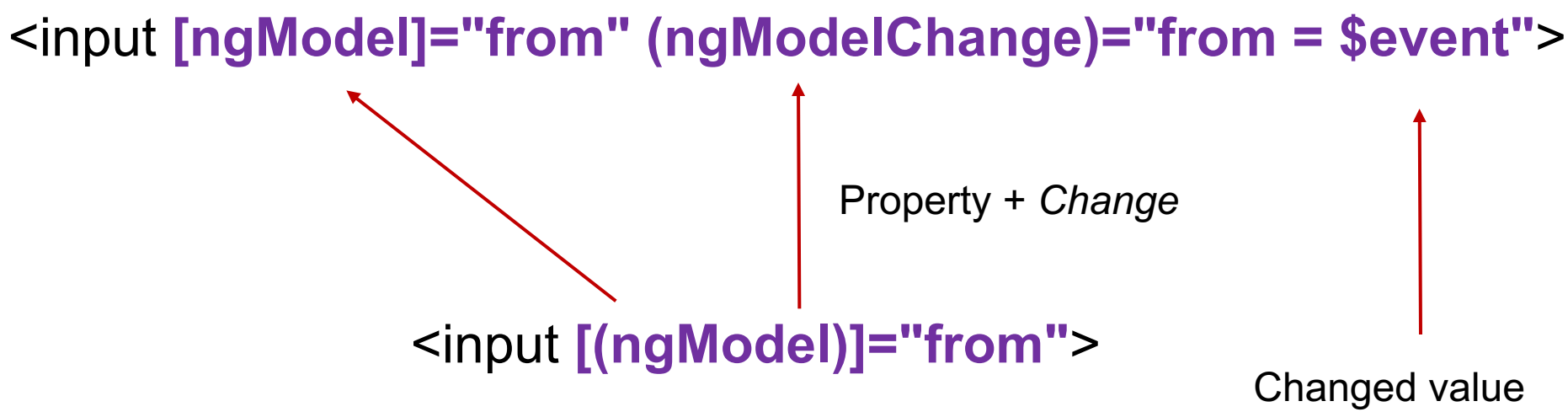
Recap

- Property binding: one way; top/down
- Event binding: one way; bottom/up
- Two way bindings?
- Two way = property binding + event binding

Property binding + event binding

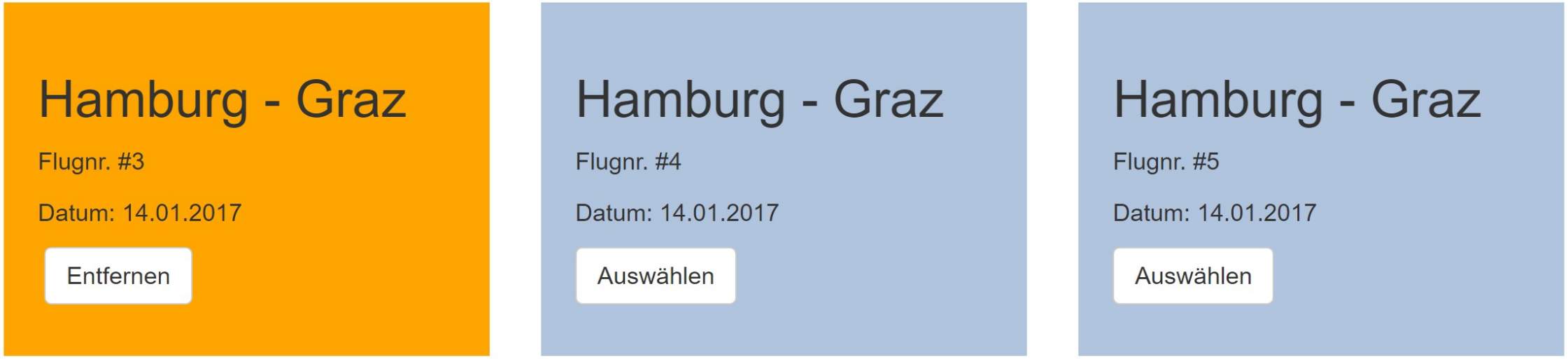
```
<input [ngModel]="from" (ngModelChange)="update($event)">
```

Property und Event-Bindings

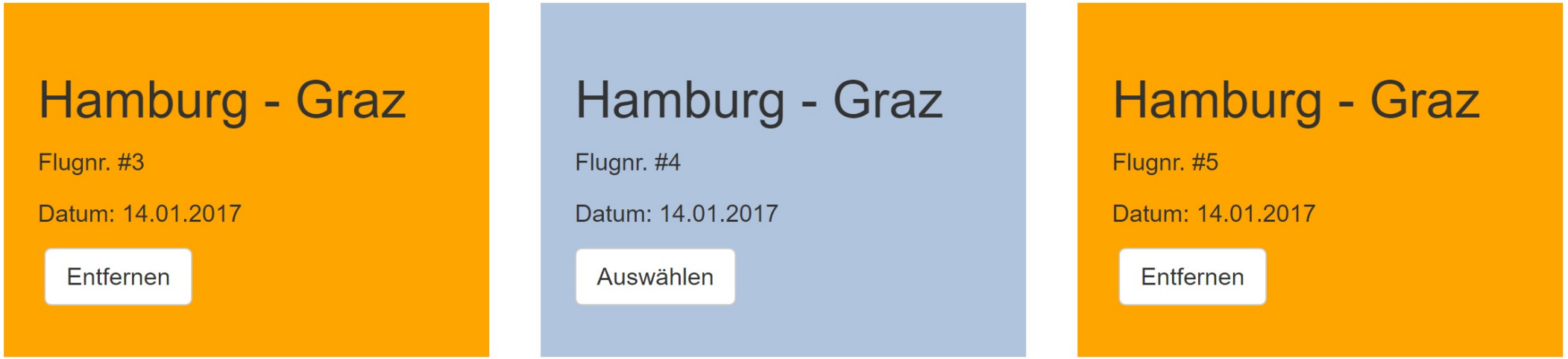


Components data bindung

Example: flight-card



Example: flight-card




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

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

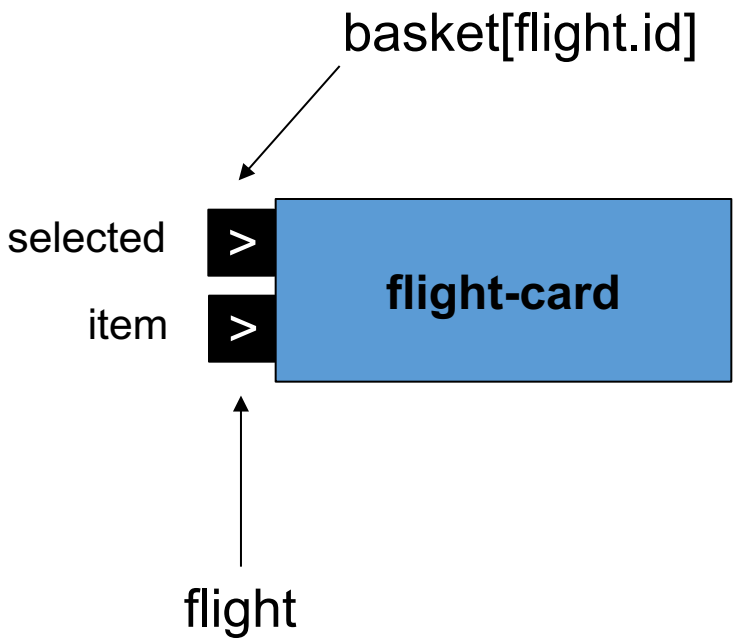
Example: flight-card in flight-search.html

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

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

</div>
```

flight-card



Example: flight-card

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

        […]

}
```

Example: flight-card

```
export class FlightCard {

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

	select(): void {
		this.selected = true;
	}

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

Template

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

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

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

Register component

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

Event bindings

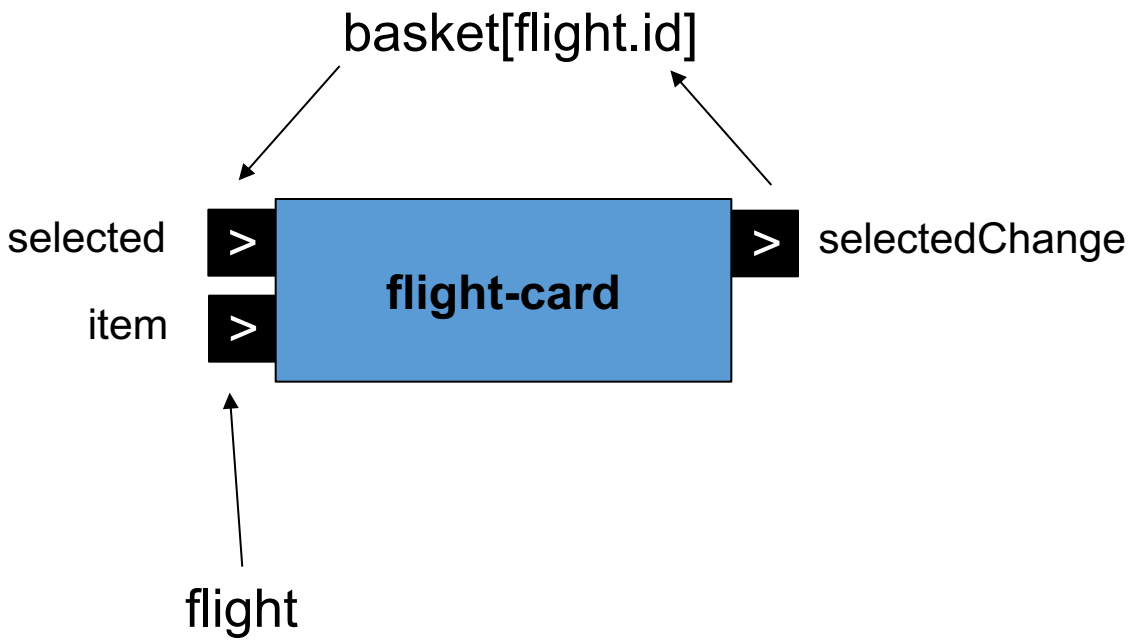
Example: flight-card event *selectedChange*

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

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

</div>
```

flight-card



Example: flight-ca

```
export class FlightCard {

        @Input() item: Flight;
        @Input() selected: boole
        @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 flights">
   <flight-card [item]="f"
                [selected]="basket[f.id]"
                (selectedChange)="basket[f.id] = $event">
   </flight-card>
</div>
```

DEMO

LAB

Thought experiment

- What would you think about our <flug-card> controlling some use case logic?
  - e.g. communicate with API
- Number of requests ==> Performance?
- Traceability?
- Reusability?

Smart vs. Dumb Components

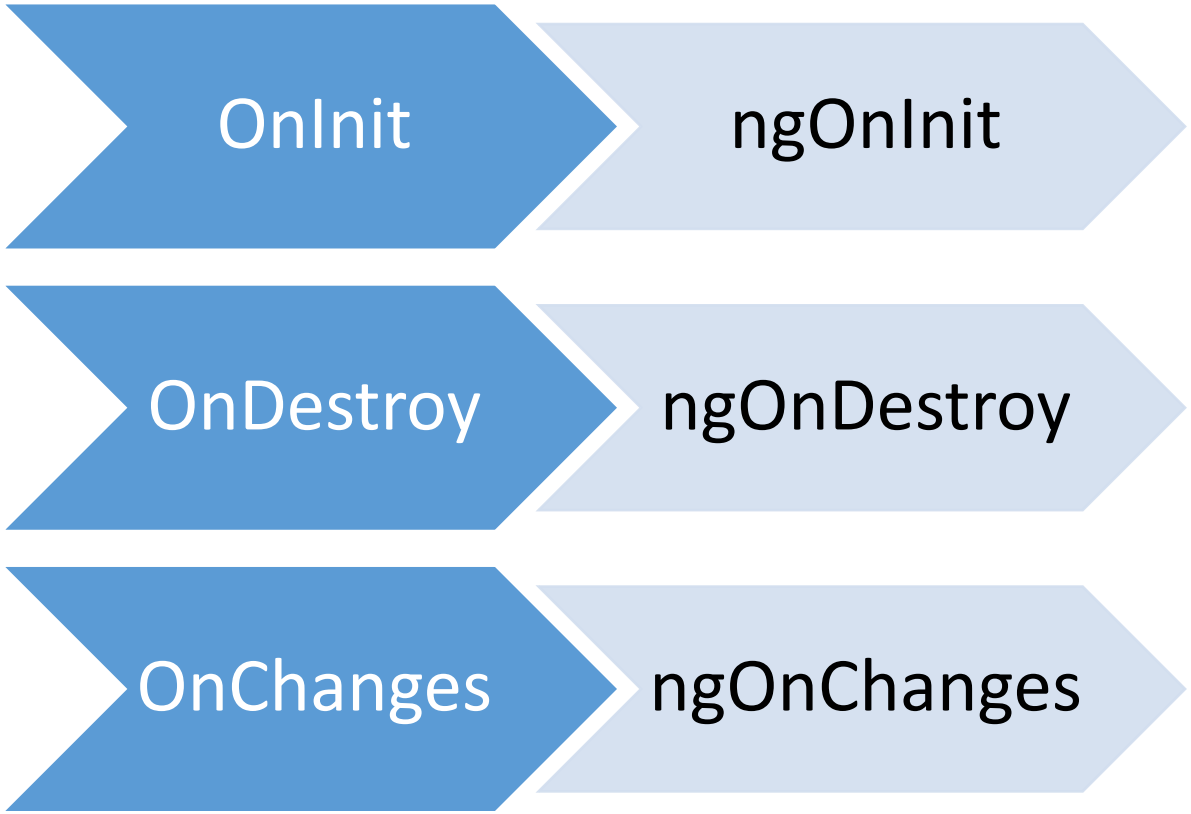
Smart Component	Dumb
• Use Case controller • Container	• Independent of Use Case • Reusable • Leave

Life Cycle Hooks

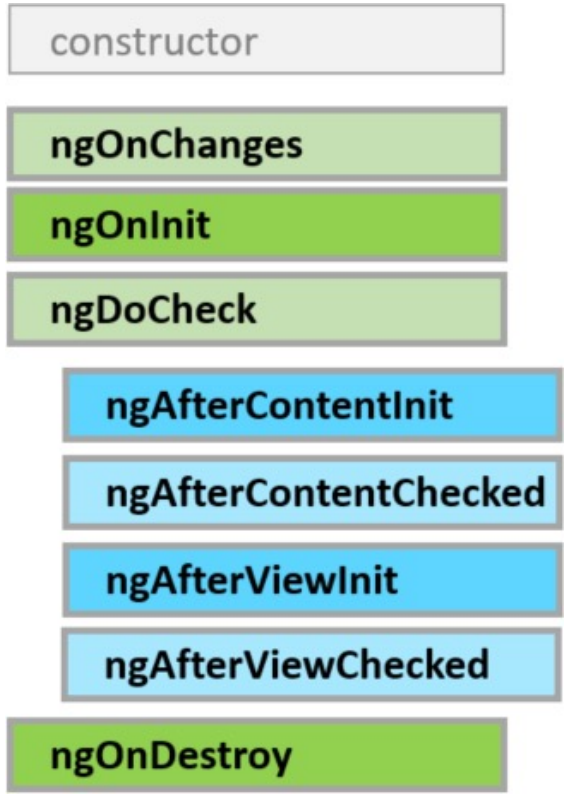
What are life cycle hooks?

- Built in methods in our components
- Will be called at a certain time by Angular

Life-Cycle-Hooks (selection)



Life-Cycle-Hooks (all, in order)



Nutzung

```
@Component({
    selector: 'my-component',
    […]
})
export class Component implements OnChanges, OnInit {

    @Input() someData;

    ngOnInit(): void {
        […]
    }

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

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT