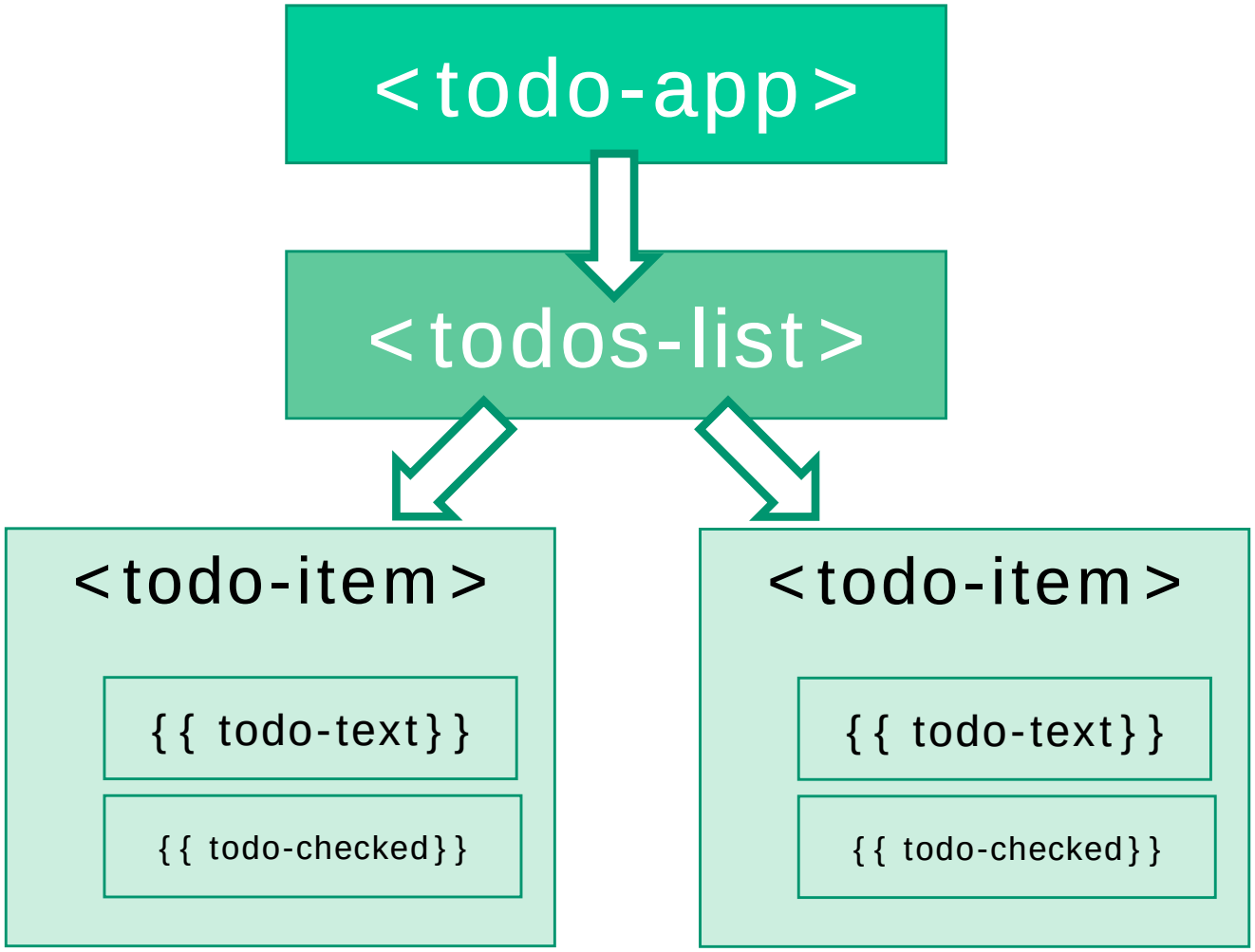
Angular 2
Module 4 – Component Trees

Peter Kassenaar –
info@kassenaar.com

Angular-app: Tree of components



Application as a tree of components

- Meerdere components?
    1. Separaat ontwikkelen
    2. Via DI invoegen
    3. Via HTML insluiten in de `parent`-component

- Herhaal deze stappen voor alle benodigde componenten

1. Detailcomponent toevoegen

```
// city.detail.ts
import { Component } from 'angular2/core';

@Component({
    selector: 'city-detail',
    template: `
    <h2>City details</h2>
        <ul class="list-group">
            <li class="list-group-item">Naam: [naam van stad]</li>
            <li class="list-group-item">Provincie: [provincie]</li>
            <li class="list-group-item">Highlights: [highlights]</li>
        </ul>
    `
})

export class CityDetail{

}
```

Nieuwe selector

Nog in te vullen

2. Injection in Module

```
// Angular Modules
…

// Custom Components
import {AppComponent} from './app.component';
import {CityDetail} from './city.detail';
import {CityService} from "./city.service";

// Module declaration
@NgModule({
   imports     : [BrowserModule, HttpModule],
   declarations: [AppComponent, CityDetail],
   bootstrap   : [AppComponent],
   providers   : [CityService]
})
export class AppModule {
}
```

Nieuwe component

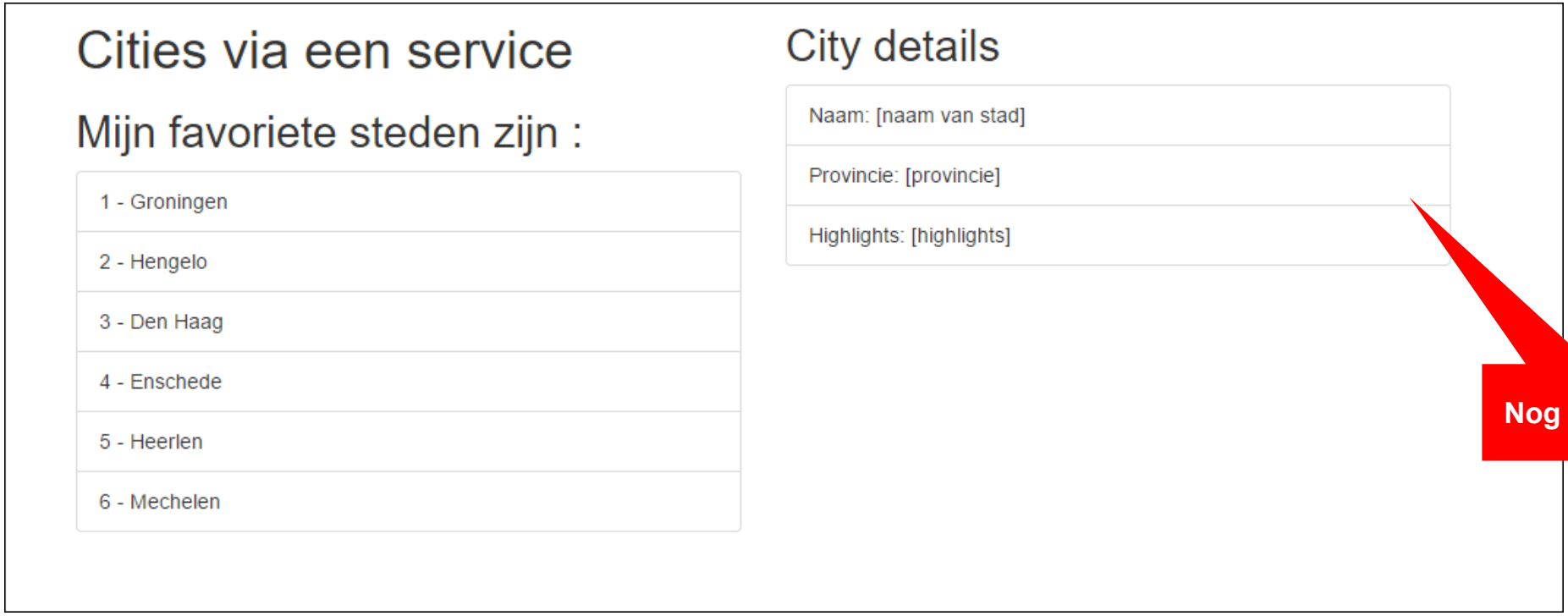
Toevoegen aan declarations:

3. Insluiten in HTML

```
<!-- app.html -->
<div class="row">
    …
   <div class="col-md-6">
      …
      <city-detail></city-detail>
   </div>
</div>
```

Combineren met overige HTML

4. Resultaat



Doel: details van geselecteerde city tonen in child-component

Data flow tussen componenten

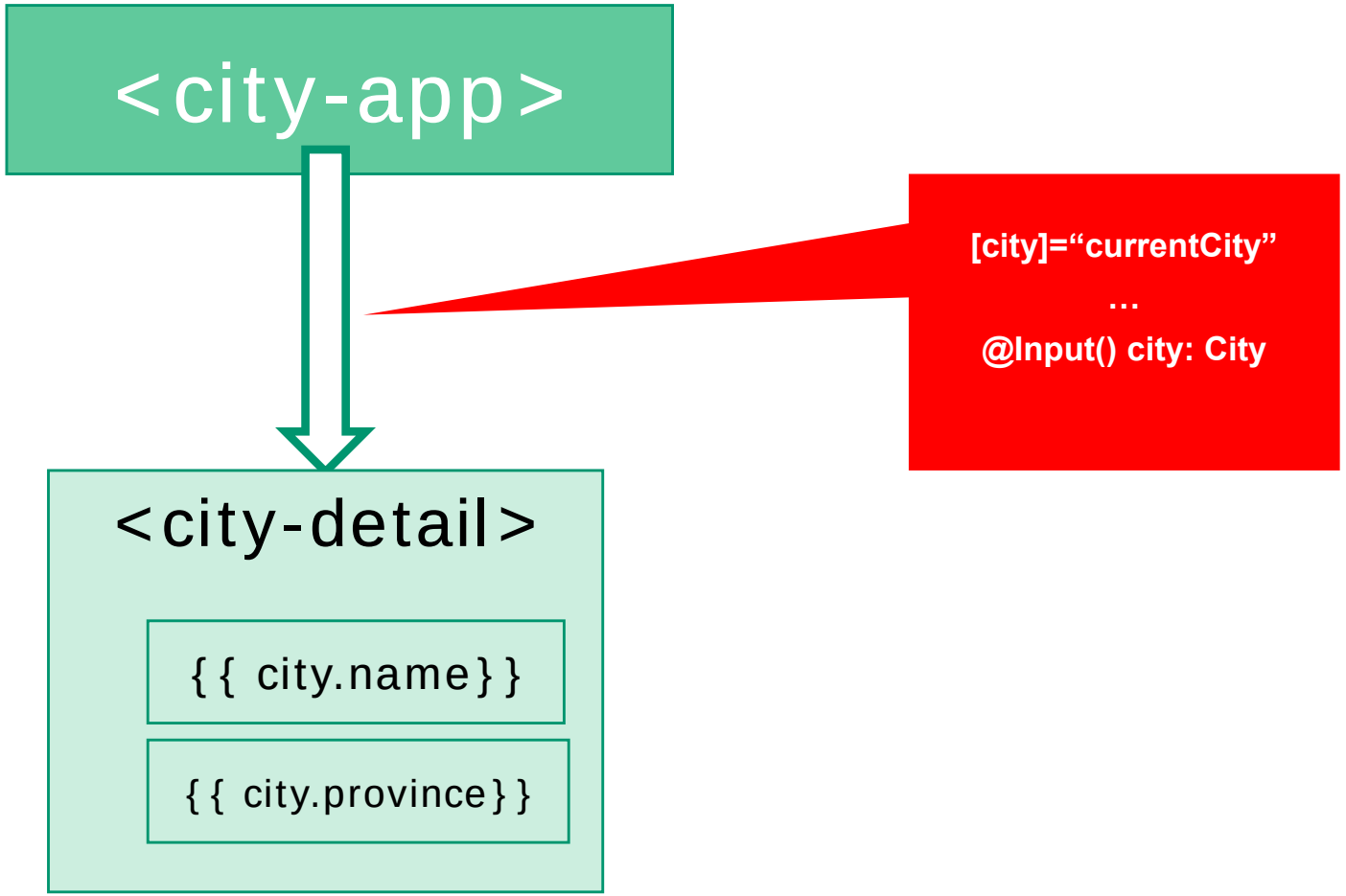
Werken met inputs en outputs

Data flow tussen components

“Data flows in to a component via @Input() ‘s”

Data flows out of a component via @Output()’s”

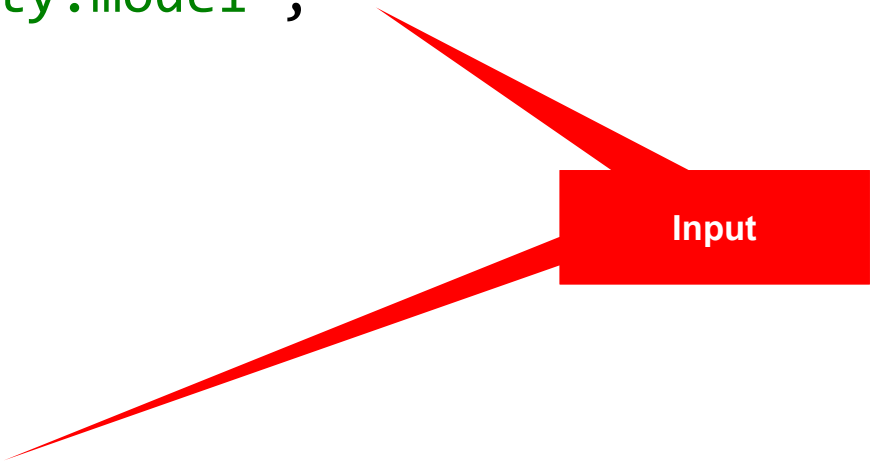
Parent-Child flow: de annotatie @Input()



Werken met @Input()

1. Service `Input` importeren in de betreffende component
2. Annotatie `@Input()` gebruiken in de class definition

```
// city.detail.ts
import { Component, Input } from '@angular/core';
import { City } from "./city.model";

@Component({
    …
})

export class CityDetail {
    @Input() city: City;
}
```



Parent Component aanpassen voor @Input

```
<!-- app.html -->
<div class="row">
   <div class="col-md-6">
      …
      <ul class="list-group">
         <li *ngFor="#city of cities" class="list-group-item"
                (click)="getCity(city)">
            {{ city.id}} - {{ city.name }}
         </li>
      </ul>
      <button *ngIf="currentCity" class="btn btn-primary"
                (click)="clearCity()">Clear</button>
   </div>
   <div class="col-md-6">
   <div *ngIf="currentCity">
         <city-detail [city]="currentCity"></city-detail>
      </div>
   </div>
</div>
```

Aanpassing

Aanpassing!

Parent Component Class uitbreiden

```
export class AppComponent {
    // Properties voor de component/class
    public cities:City[];
    public currentCity:City;

    …

    getCity(city) {
        this.currentCity = city;
    }

    clearCity() {
        this.currentCity = null;
    }

    …
}
```

Resultaat

Cities via een service

Mijn favoriete steden zijn :

1 - Groningen

2 - Hengelo

3 - Den Haag

4 - Enschede

5 - Heerlen

6 - Mechelen

Clear

City details

Naam: Enschede

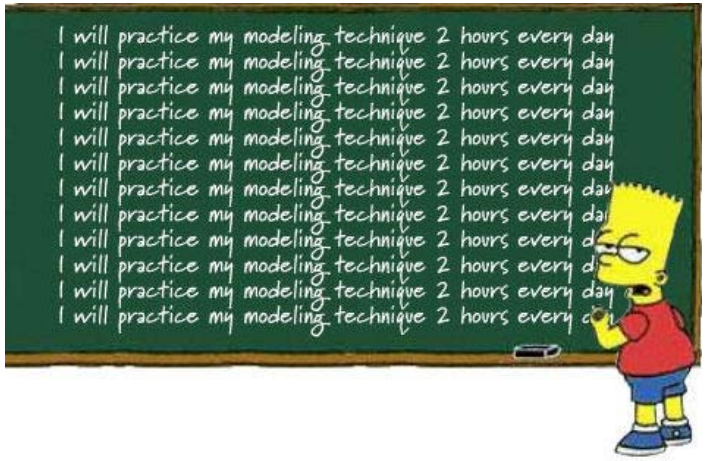
Provincie: Grote Markt

Highlights: Twentse Welle museum

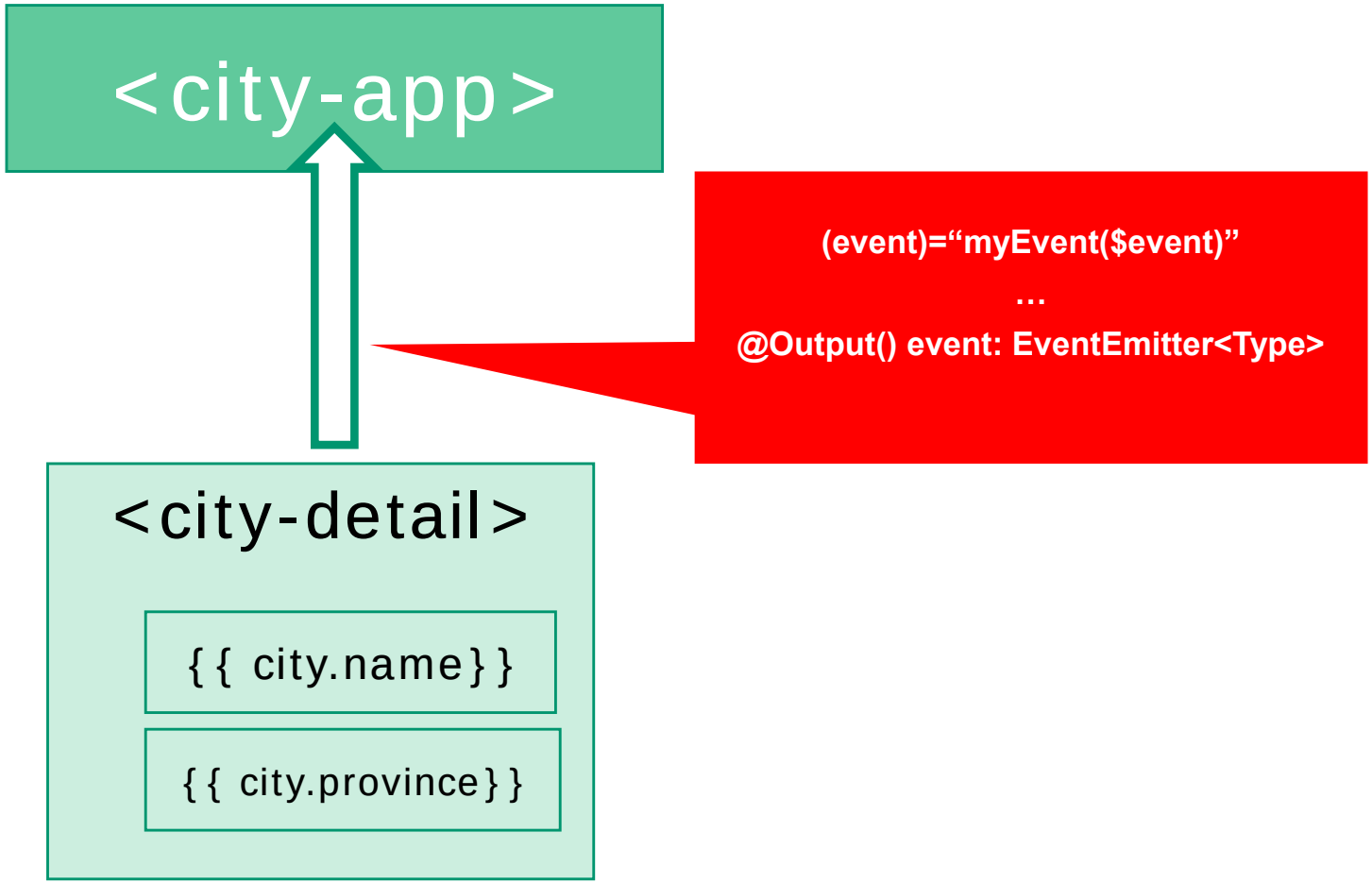
Checkpoint

- Componenten kunnen binnen componenten worden opgenomen
- Breidt de HTML van de Parent Component uit met declaratie van de Child Component
- Denk er aan Child Component te importeren
- Data flow naar Child Component : werken met `@Input()` en `[propName]="data"`

Oefening….



Child-Parent flow: de annotatie @Output()



Werkwijze – idem, maar dan andersom

1. Service `Output` importeren in de betreffende component
2. Annotatie `@Output()` gebruiken in de class definition
3. `EventEmitter` definiëren en optioneel Type Annotation

“With @Output,

data flows up the Component Chain”

Een rating geven aan Cities

```
// city.detail.ts
import { Component, Input, Output, EventEmitter} from ‘@angular/core';

@Component({
    …
    template: `
    <h2>City details
        <button (click)="rate(1)">+1</button>
        <button (click)="rate(-1)">-1</button>
    </h2>
    …
    `
})

export class CityDetail {
    @Input() city:City;
    @Output() rating: EventEmitter<number> = new EventEmitter<number>();

    rate(num) {
        console.log('rating voor ', this.city.name, ': ', num);
        this.rating.emit(num);
    }
}
```

Imports

Bind custom events to DOM

Define & handle custom @Output event

Parent Component voorbereiden op ontvangen custom event

```
<!-- app.html -->
<div *ngIf="currentCity">
   <city-detail [city]="currentCity" (rating)="updateRating($event)">
   </city-detail>
</div>
```

```
// app.component.ts
// increase or decrease rating on Event Emitted
updateRating(rating){
   this.currentCity.rating += rating;
}
```

Rating tonen in HTML

```
<li *ngFor="#city of cities"
    class="list-group-item" (click)="getCity(city)">
   {{ city.id}} - {{ city.name }} ({{i}})
   <span class="badge">{{city.rating}}</span>
</li>
```

Resultaat

Cities via een service

Mijn favoriete steden zijn :

1 - Groningen	0
2 - Hengelo	0
3 - Den Haag	-3
4 - Enschede	0
5 - Heerlen	2
6 - Mechelen	5

Clear

City details +1 -1

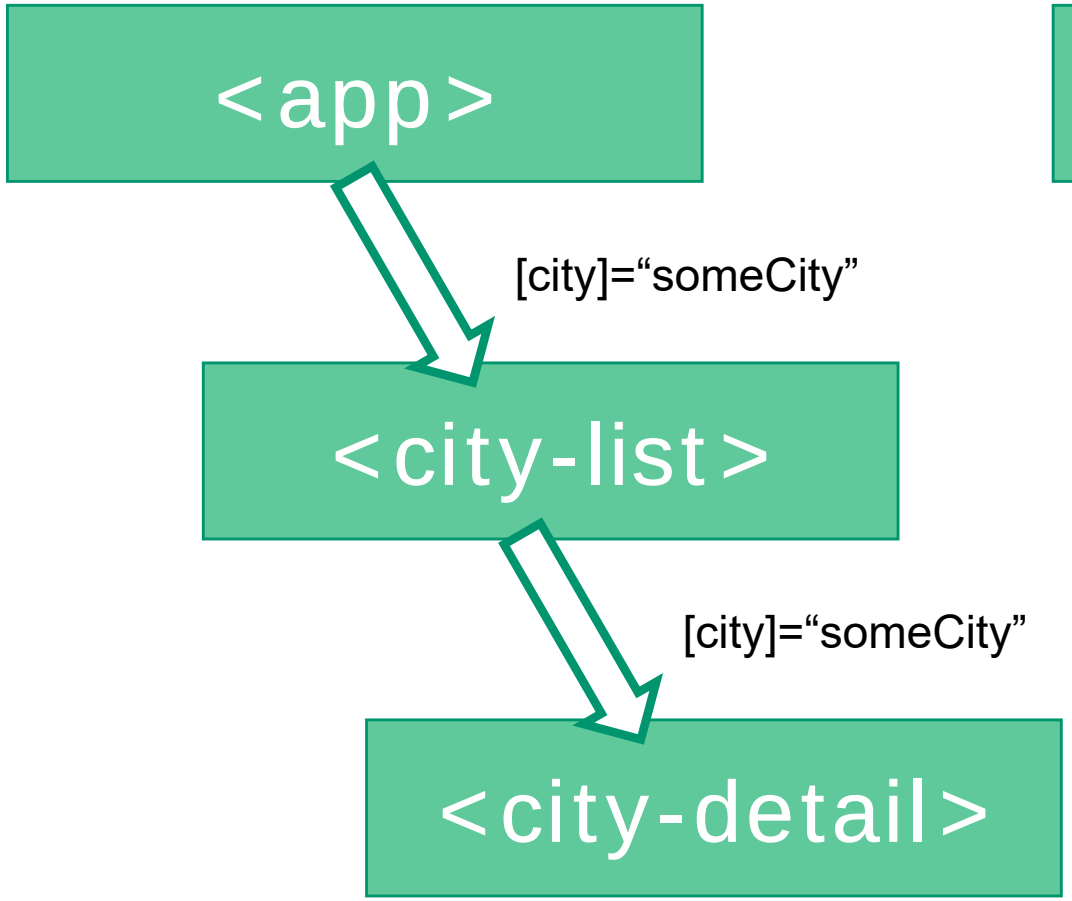
Naam: Den Haag

Provincie: Zuid-Holland

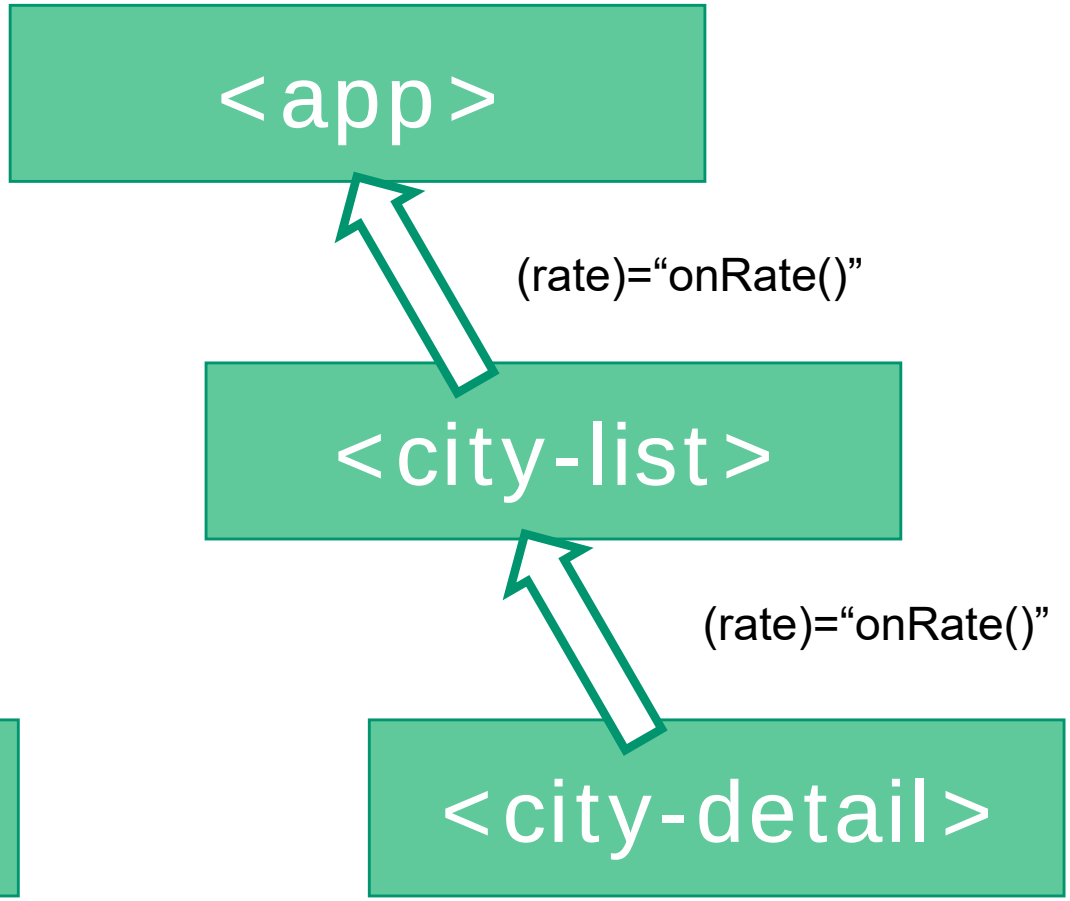
Highlights: Binnenhof

Samenvatting

Parent → Child



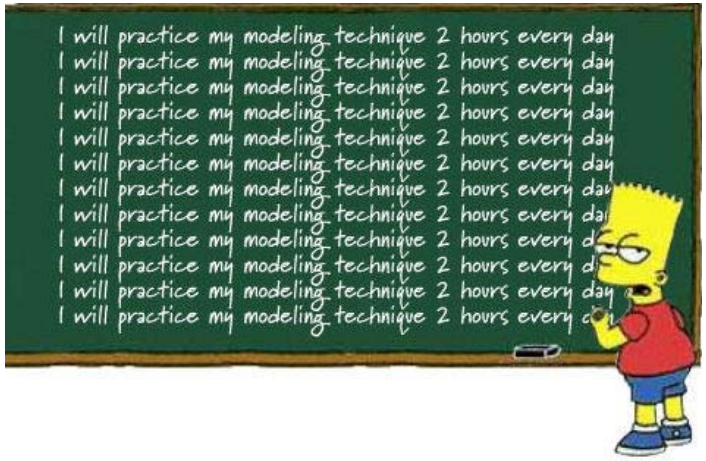
Child → Parent



Checkpoint

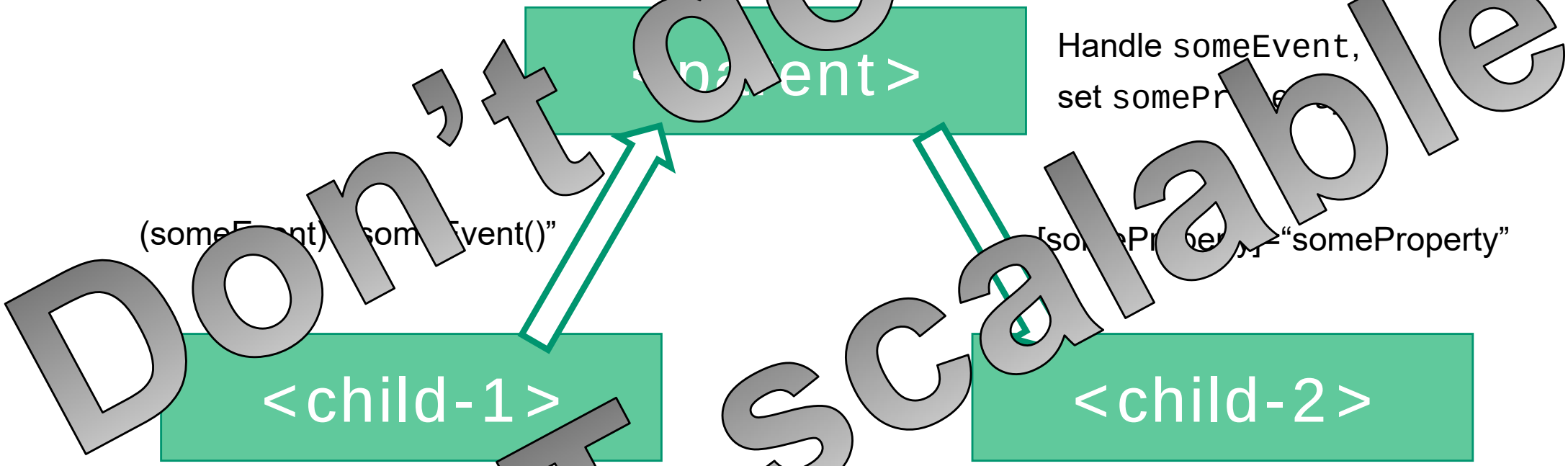
- Data flow naar Parent Component : werken met `@Output()` en `(eventName)="eventHandler($event)"`
- Je kunt allerlei typen Events meegeven
- Meer info: http://victorsavkin.com/post/118372404541/the-core-concepts-of-angular-2

Oefening….



Communicatie tussen siblings

- via `Output()` van een childcomponent, naar een `@Input()` van andere childcomponent



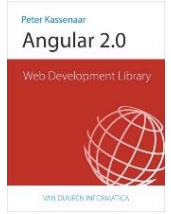
Don't do this!
NOT scalable

Mooiere oplossing – Pub/Sub-systeem met Observables

- http://www.syntaxsuccess.com/viewarticle/pub-sub-in-angular-2.0

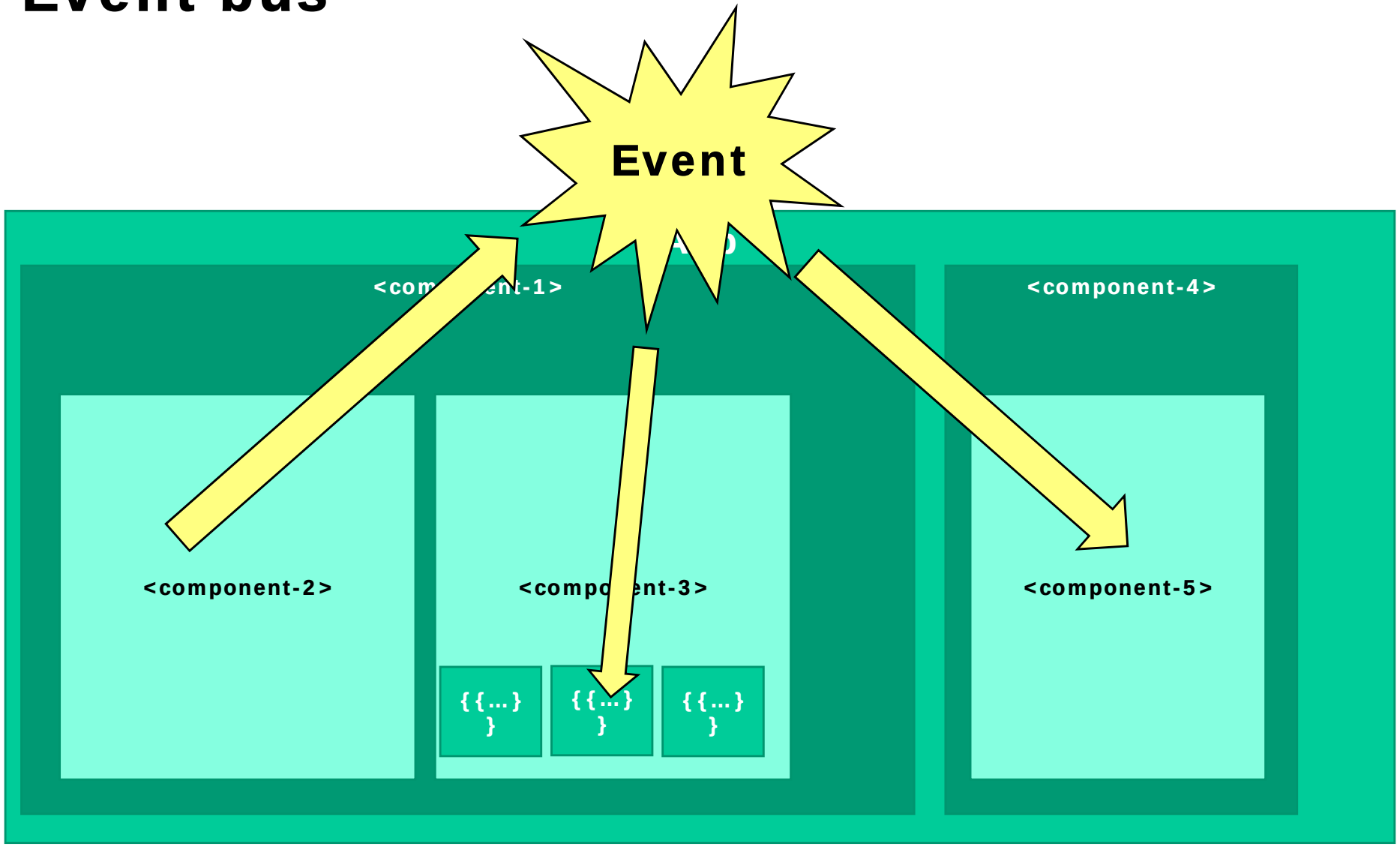
"Custom events,

gebruik een event bus"



p. 173 e.v.

Event bus

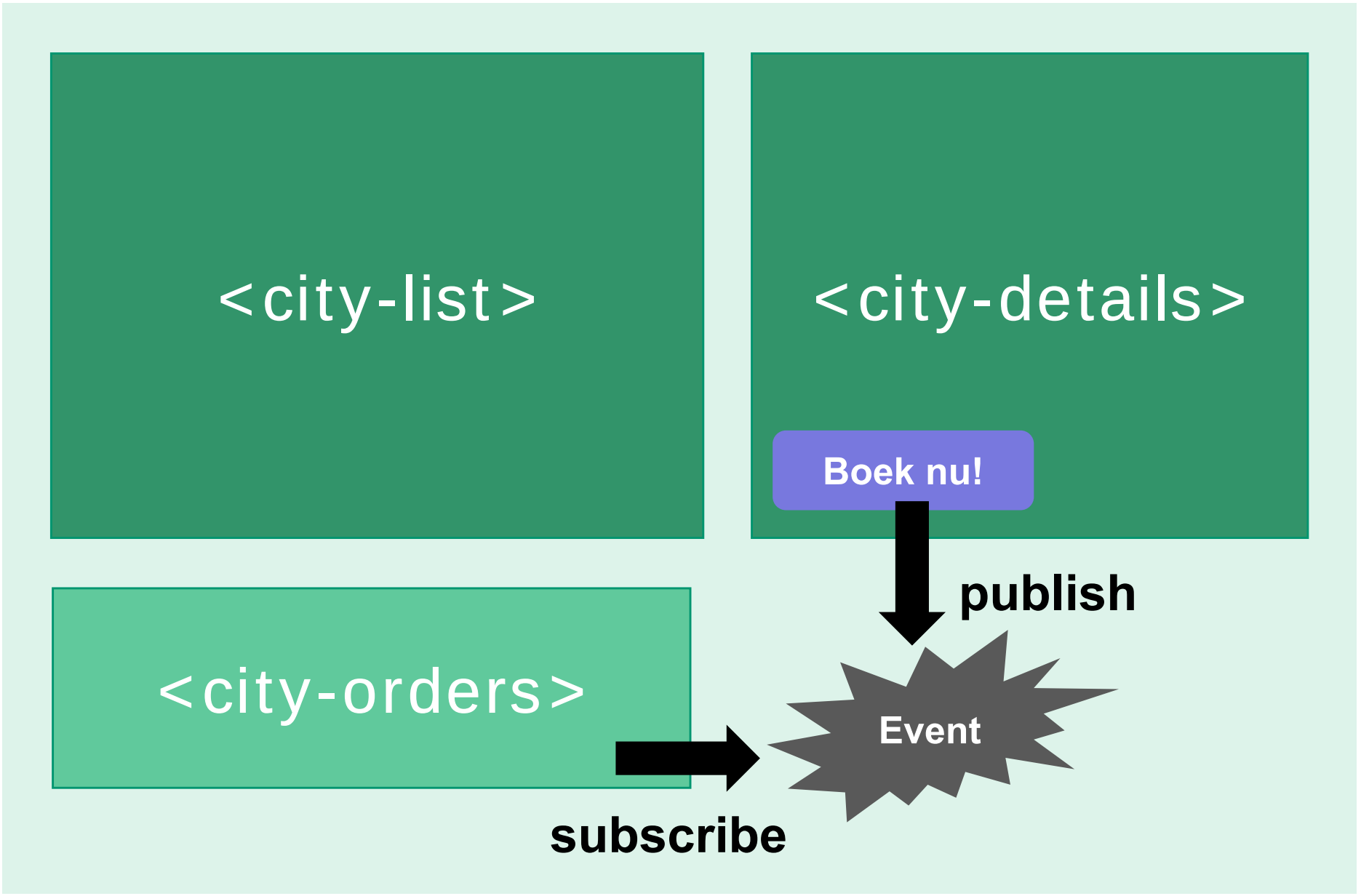

Event
<component-1>
<component-4>
<component-2>
<component-3>
<component-5>
{{...}}
{{...}}
{{...}}

Opties

Uit RxJs-bibliotheek, werken met:

- `EventEmitter()`
- `Observable()`
- `Observer()`
- `Subject()` (zowel `Observable` als `Observer`)

“Publish en Subscribe” – PubSub systeem


<city-list>
<city-details>
Boek nu!
publish
<city-orders>
Event
subscribe

PubSub-service maken

- Stap 1 – Publicatie service maken
- Stap 2 – ‘Producer’, of ‘Publish’ – component maken
- Stap 3 – subscriber-component maken, of toevoegen aan bestaande component.

1. OrderService

```
// order.service.ts
import {Subject} from "rxjs/Subject";
import {Injectable} from "@angular/core";
import {City} from "../model/city.model";

@Injectable()
export class OrderService {
   Stream:Subject<City>;

   constructor() {
      this.Stream = new Subject<City>();
   }
}
```

2. Producer component (‘boek nu’-knop)

In de HTML:

```
<h2>Prijs voor een weekendje weg:
{{ city.price | currency:'EUR':true:'1.2' }}
<button class="btn btn-lg btn-info"
   (click)="order(city)">Boek nu!</button>
</h2>
```

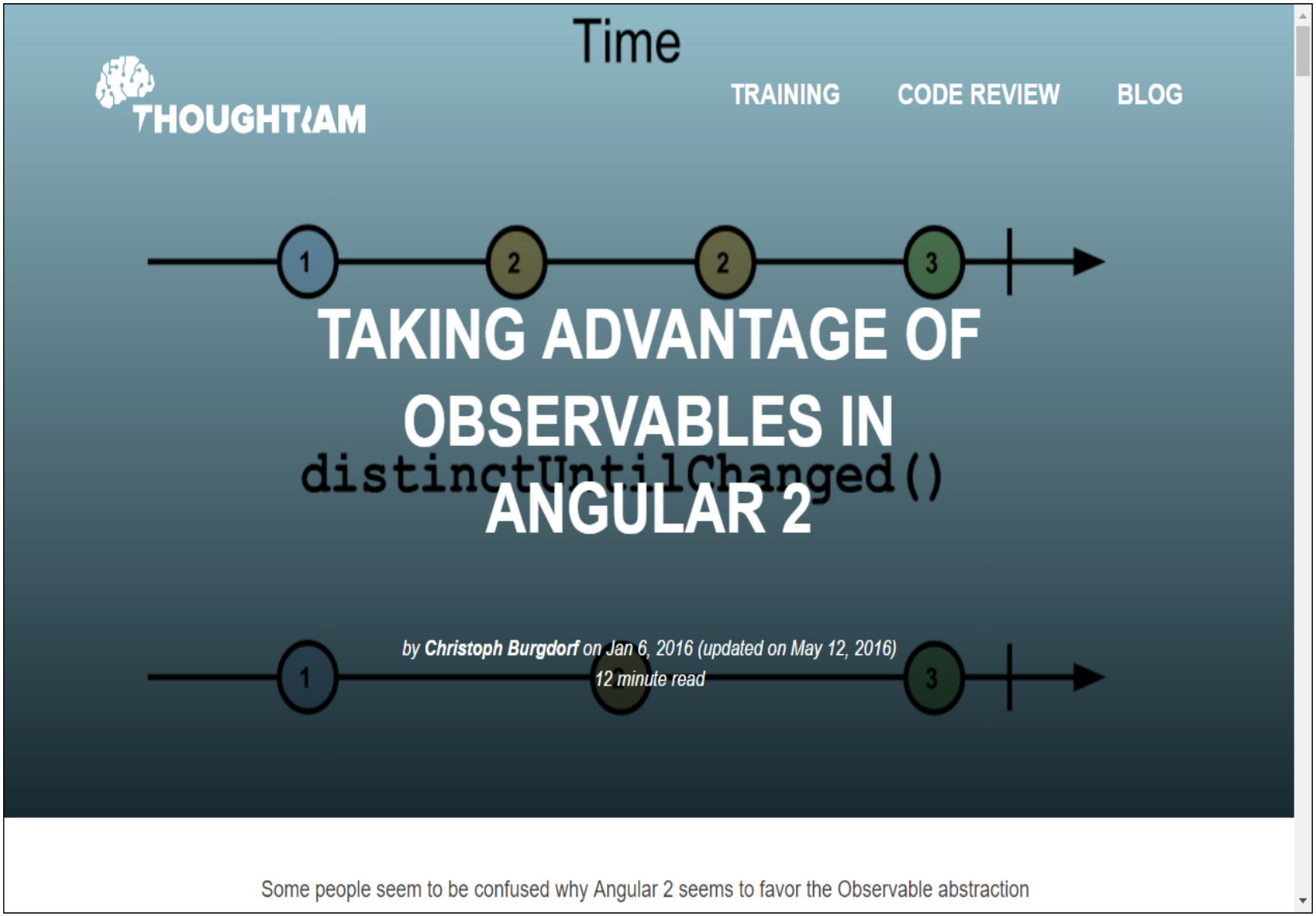
In de class:

```
// Order plaatsen. Event emitten voor deze stad.
// Dit gaan opvangen in city.orders.ts
order(city) {
   console.log(`Stedentripje geboekt voor: ${this.city.name});
   this.orderService.Stream.next(city);
}
```

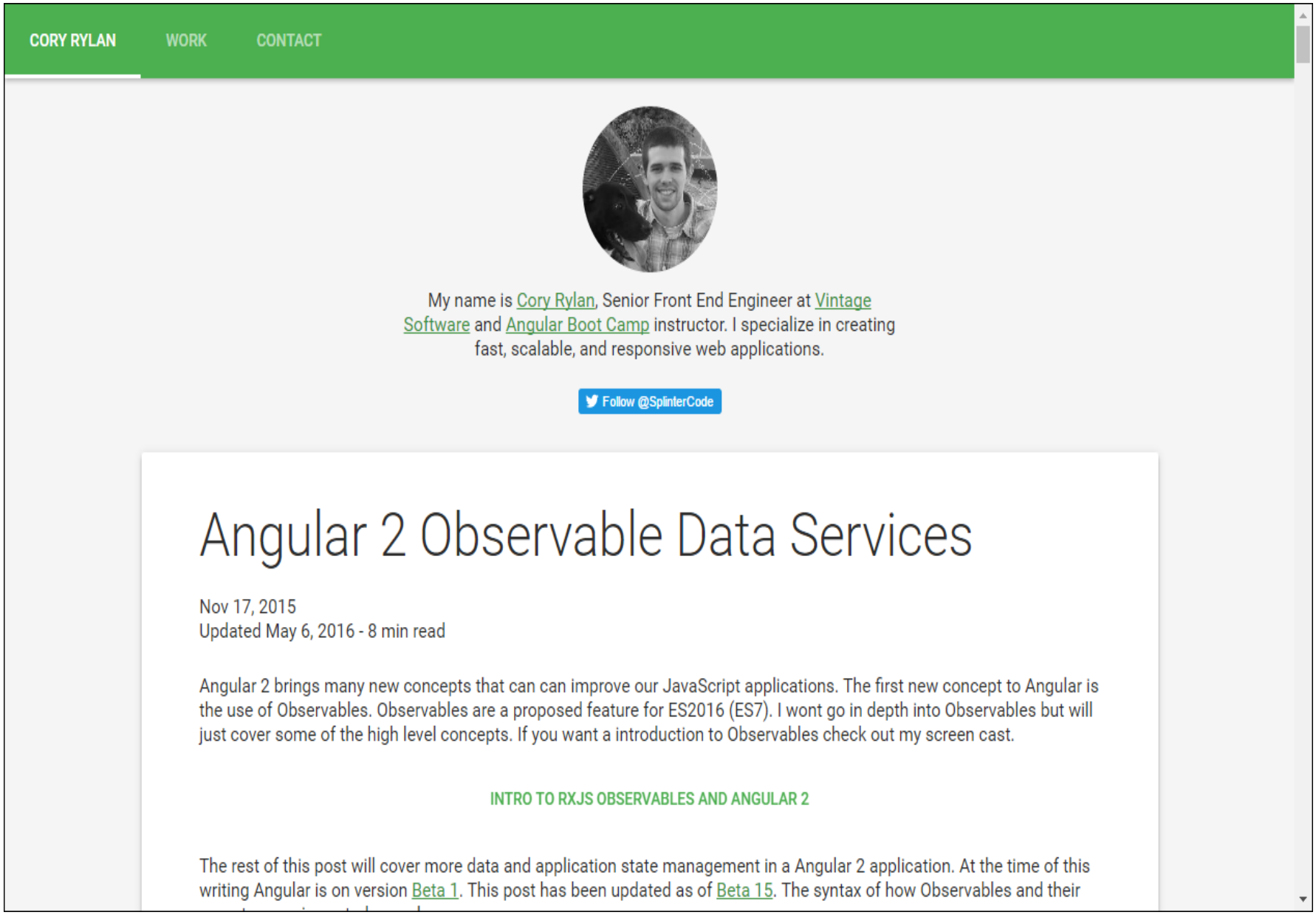
3. Subscriber component

```
//city.orders.ts - Een soort 'winkelmandje',
// bijhouden welke stedentripjes zijn geboekt.
import …

@Component({
   selector: 'city-orders',
   template: `
   <div *ngIf="currentOrders.length > 0">
      …
})

export class CityOrders {
   …
   ngOnInit() {
      this.orderService.Stream
         .subscribe(
            (city:City) => this.processOrder(city),
            (err)=>console.log('Error bij verwerken City-order'),
            ()=>console.log('Complete...')
         )
   }
`    …
}
```

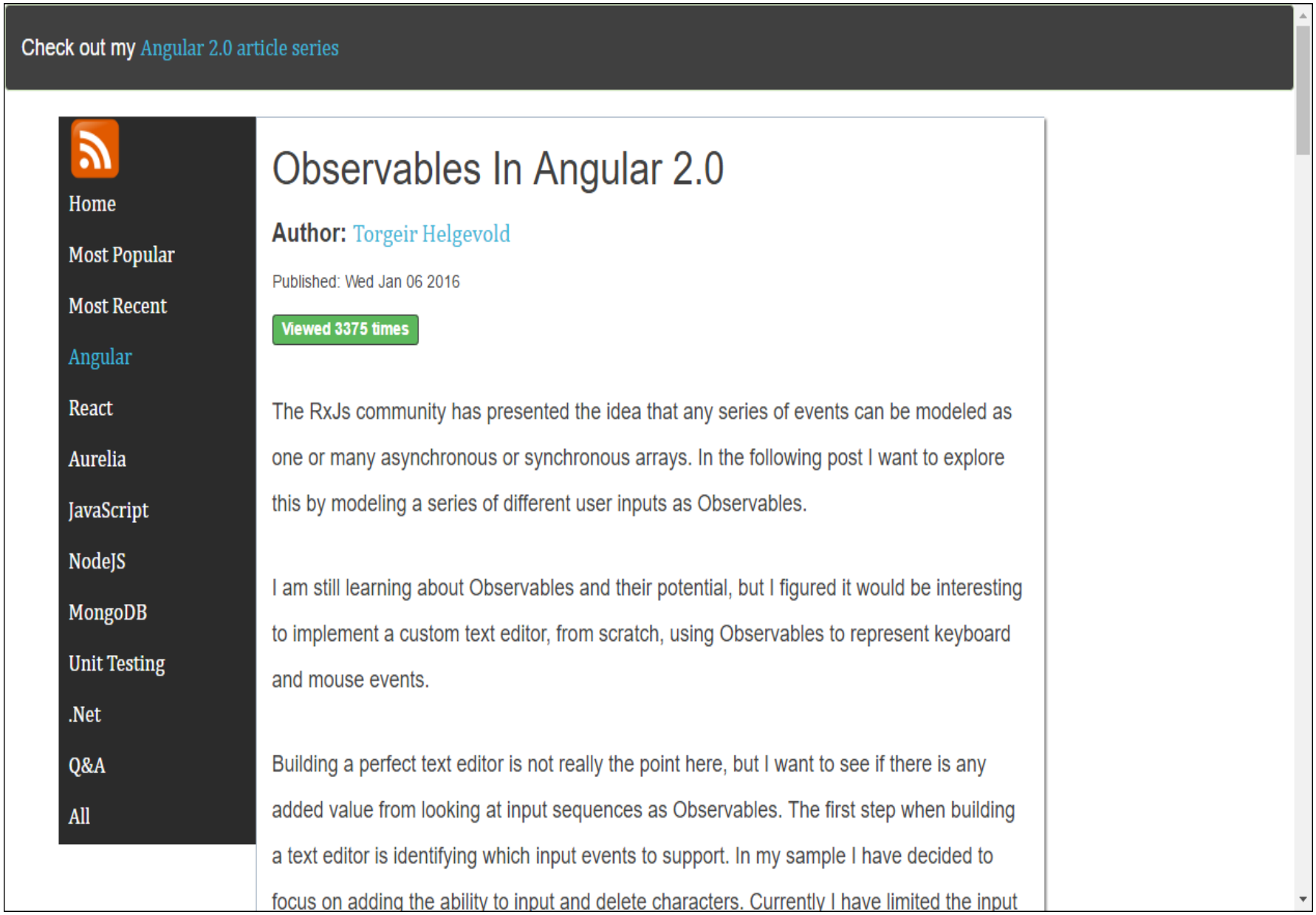
Meer over Observables



http://blog.thoughtram.io/angular/2016/01/06/taking-advantage-of-observables-in-angular2.html



https://coryrylan.com/blog/angular-2-observable-data-services



http://www.syntaxsuccess.com/viewarticle/observables-in-angular-2.0