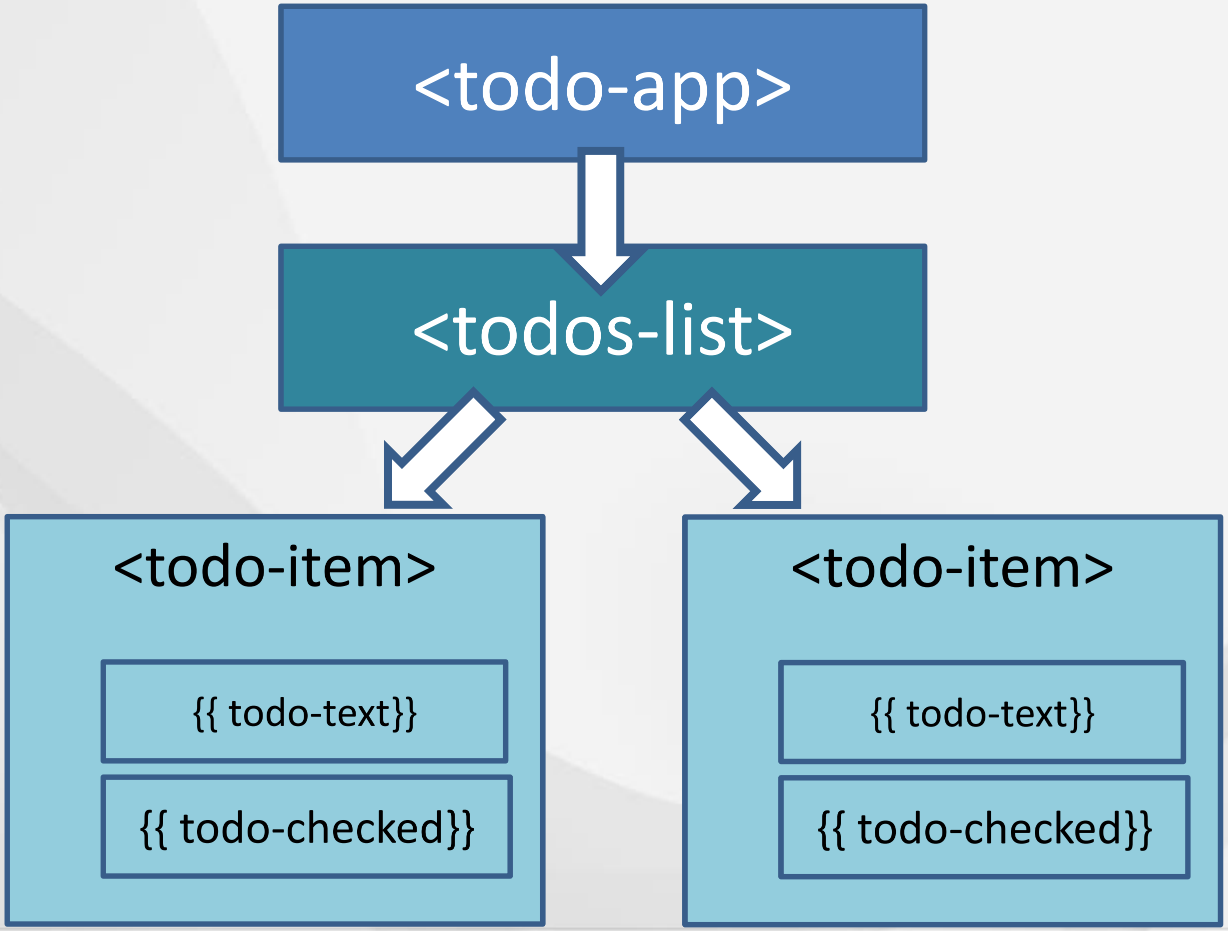
IK WIL

Angular - Module Component Trees

vijfhart IT-OPLEIDINGEN

Angular-app: Tree of components



Application as a tree of components

→ Meerdere components?

1. Zelf toevoegen – of genereren via CLI
2. Via reference invoegen in `@ngModule` (of dit ook door de CLI laten doen)
3. Via HTML de nieuwe `selector` insluiten in de `parent-`component

→ Herhaal deze stappen voor alle benodigde componenten

1. Detailcomponent toevoegen

```
// city.detail.ts
import { Component } from '@angular/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. Importeren in Module

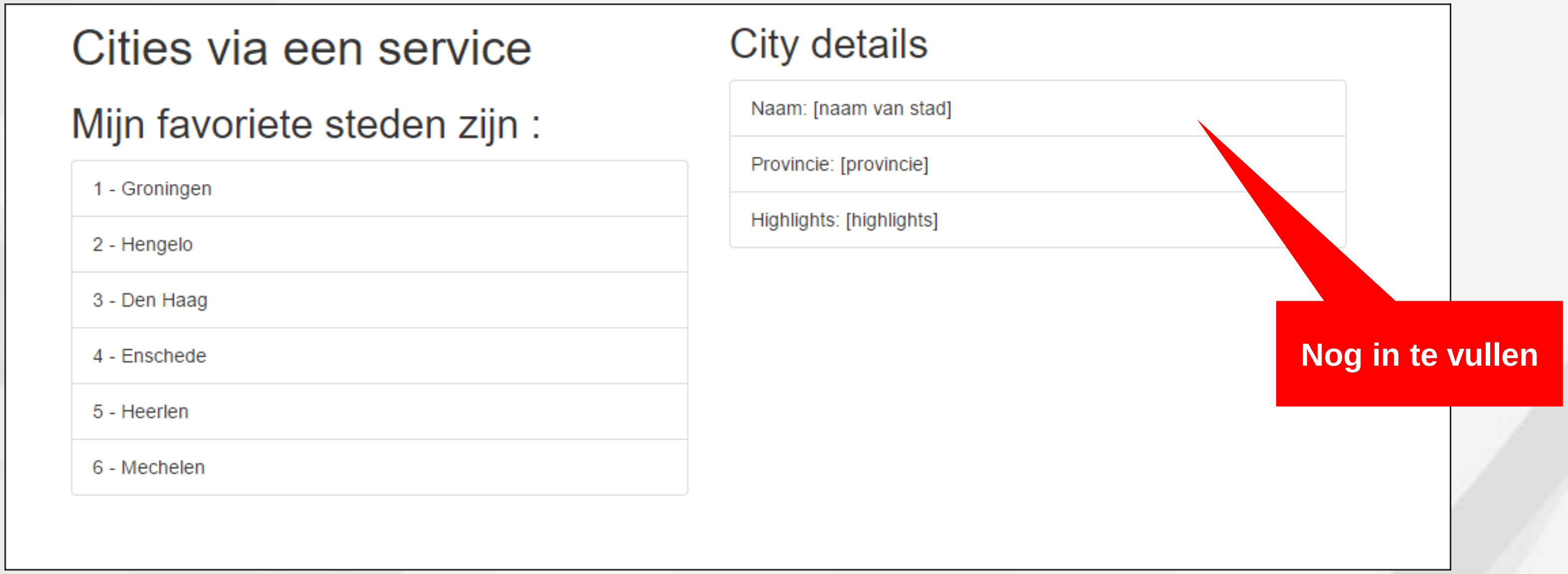
```
// app.module.ts
…
import {CityDetail} from "./city.detail"; // Nieuwe component invoegen

@ngModule({
   …
   declarations: […,CityDetail]  // Niet vergeten: invoegen bij declarat
})

export class AppModule {
   …
}
```

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


Cities via een service

Mijn favoriete steden zijn :

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

City details

Naam: [naam van stad]
Provincie: [provincie]
Highlights: [highlights]

Doel: details van geselecteerde city tonen in child-component

Data flow tussen componenten

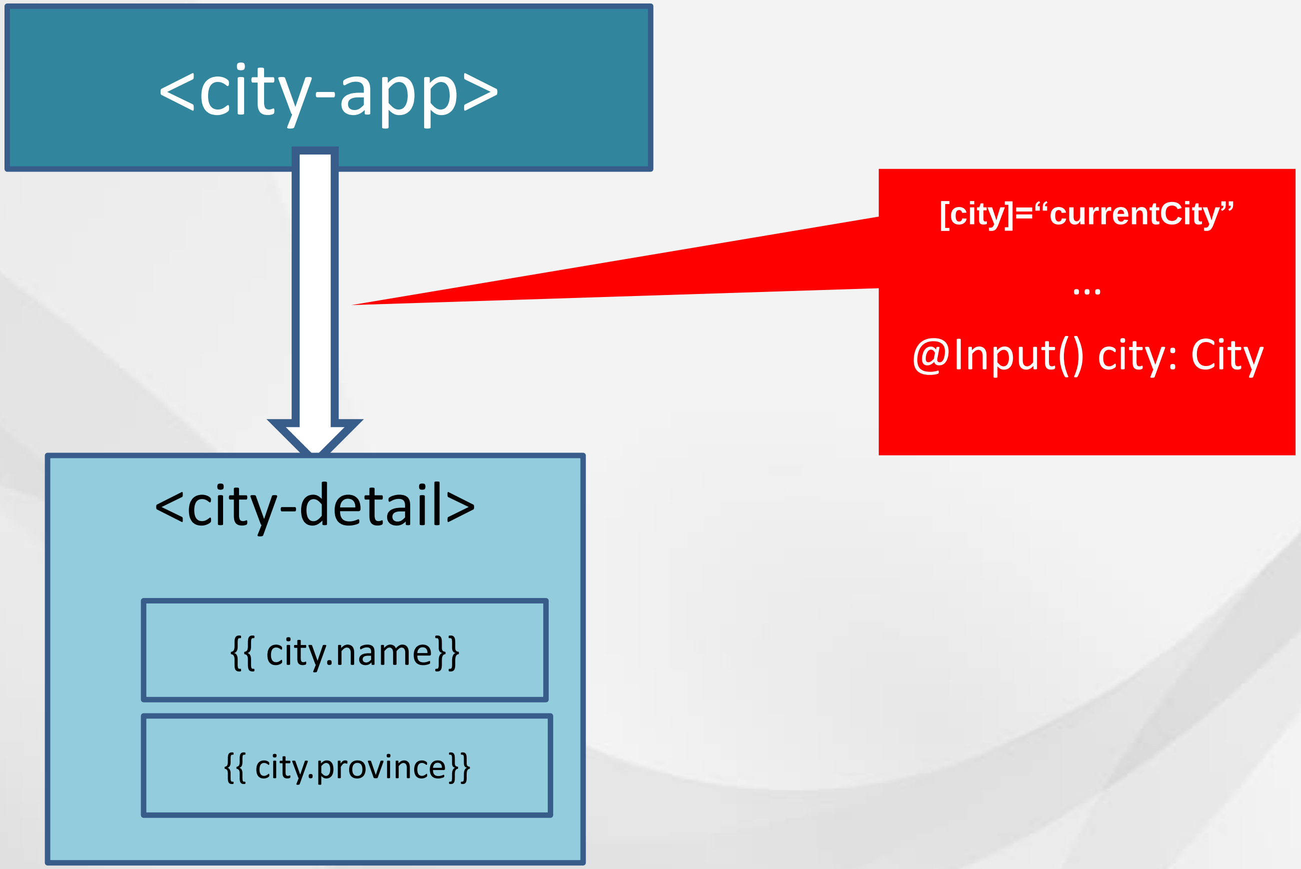
Werken met inputs en outputs

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: decorator `@Input()`



Werken met @Input()

1. Decorator `Input` importeren in 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;
}
```

Input

Parent Component aanpassen voor `@Input`

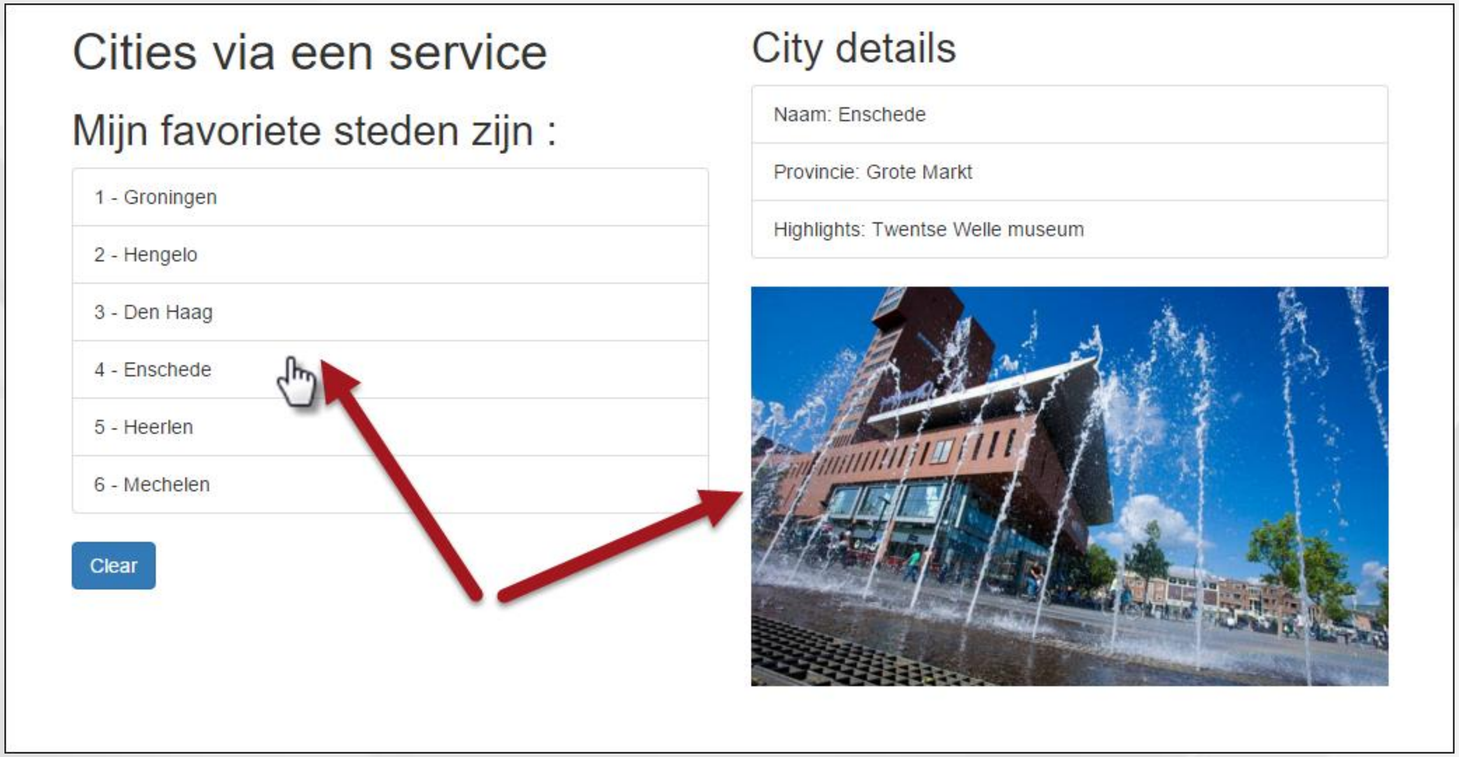
```
<!-- app.html -->
<div class="row">
   <div class="col-md-6">
      …
      <ul class="list-group">
         <li *ngFor="let city of cities" class="list-group-item"
                (click)="getCity(city)">
            {{ city.id}} - {{ city.name }}
         </li>
      </ul>
      <button *ngIf="currentCity"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

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

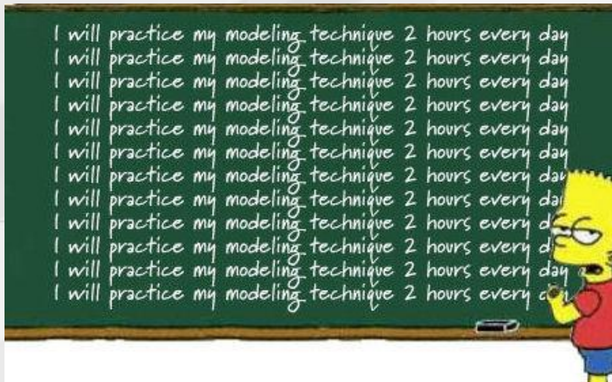
City details

Naam: Enschede
Provincie: Grote Markt
Highlights: Twentse Welle museum

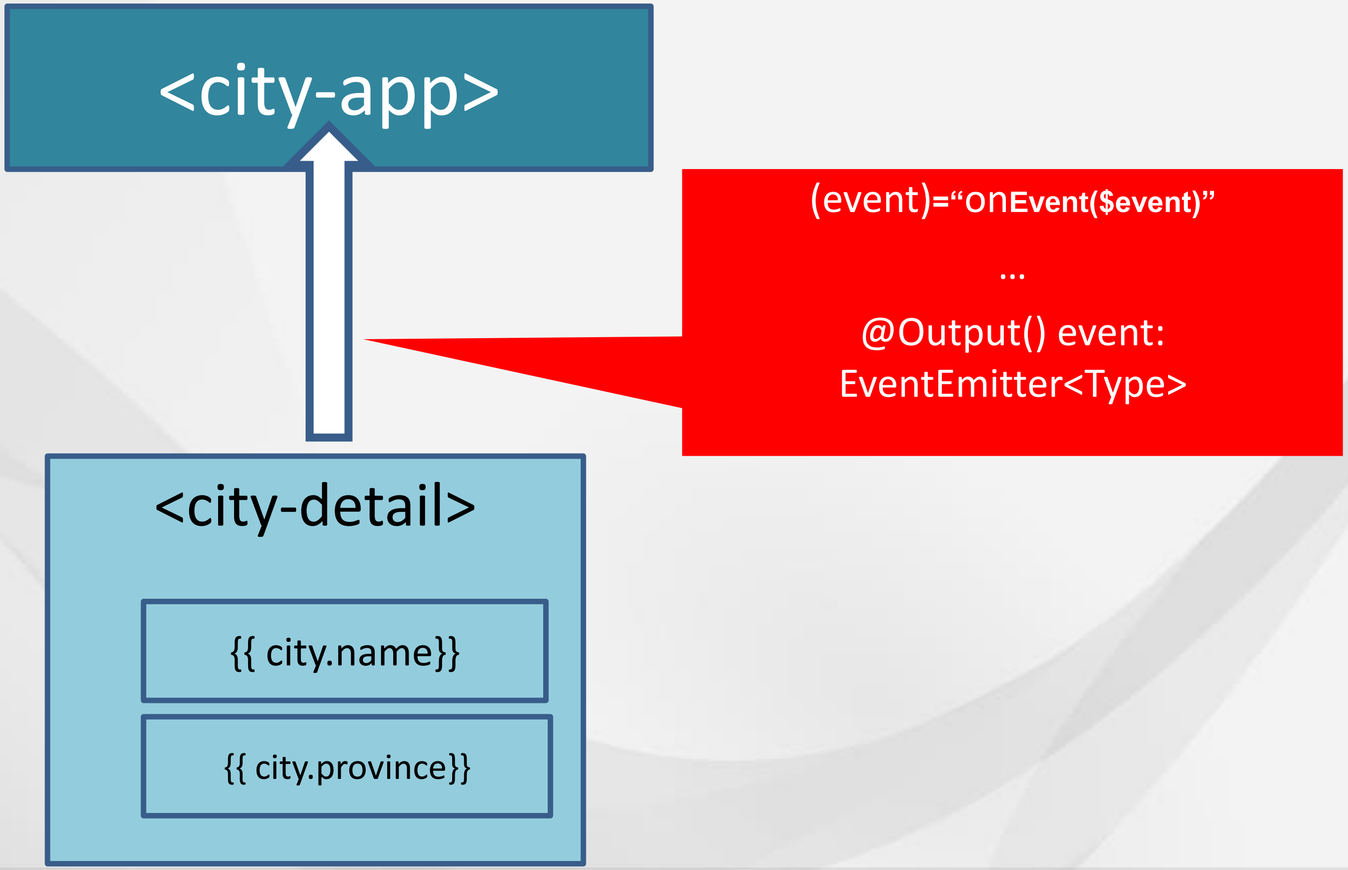
Checkpoint

- 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 in de `@ngModule`
- Data flow naar Child Component : werken met `@Input()` en `[propName]="data"`
- Oefening: `6b) en 6c)`
- Voorbeeld: `/300-components`

Oefening….


I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every

Child-Parent flow: decorator `@Output()`


<city-app>
(event)="onEvent($event)"
…
@Output() event: EventEmitter<Type>
<city-detail>
{{ city.name}}
{{ city.province}}

Werkwijze – idem, maar dan andersom

1. Decorator `Output` importeren in de betreffende component
2. Decorator `@Output()` gebruiken in de class definition
3. `EventEmitter` definiëren en 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>
```

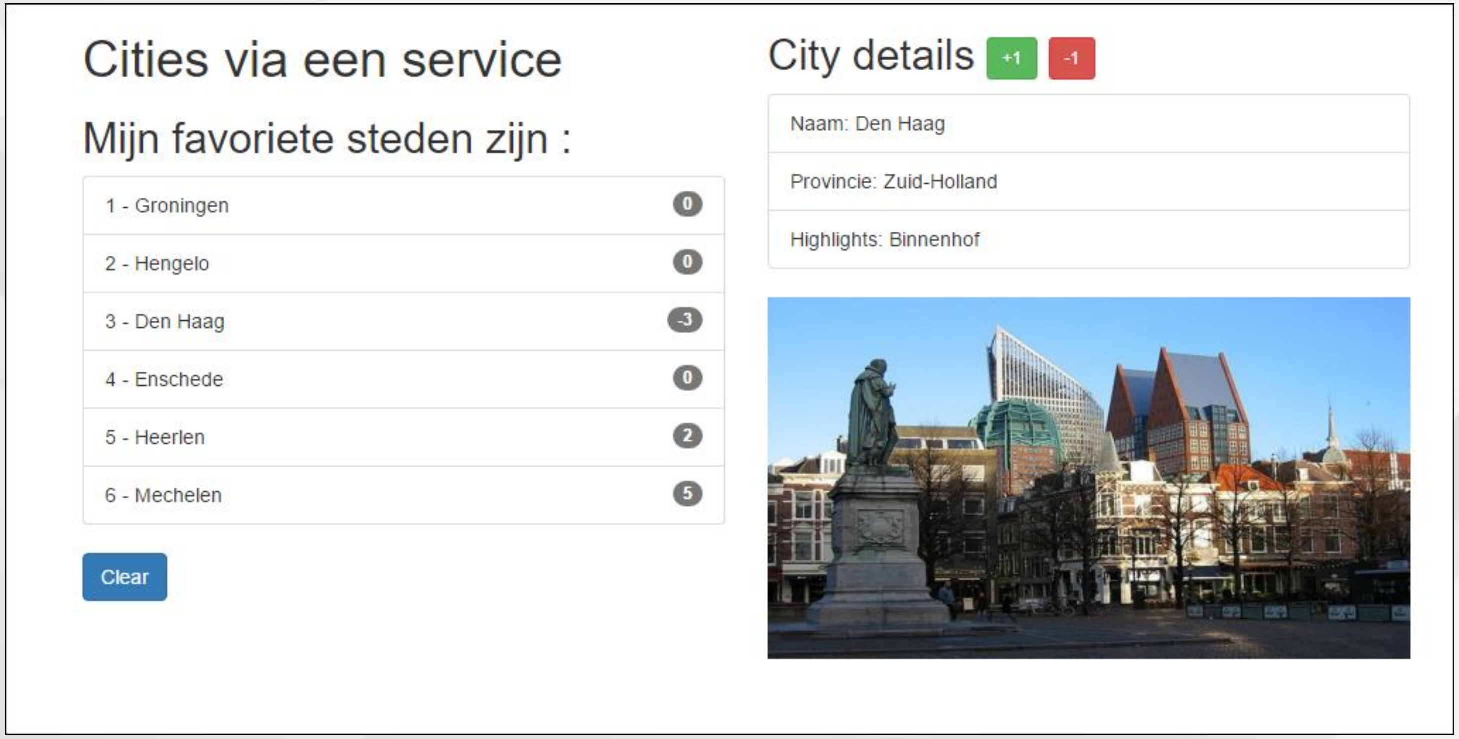
```
// app.component.ts
// increase or decrease rating on Event Emitted
updateRating(rating){
   this.currentCity.rating += rating;
}
```

Rating tonen in HTML

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

Rating

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

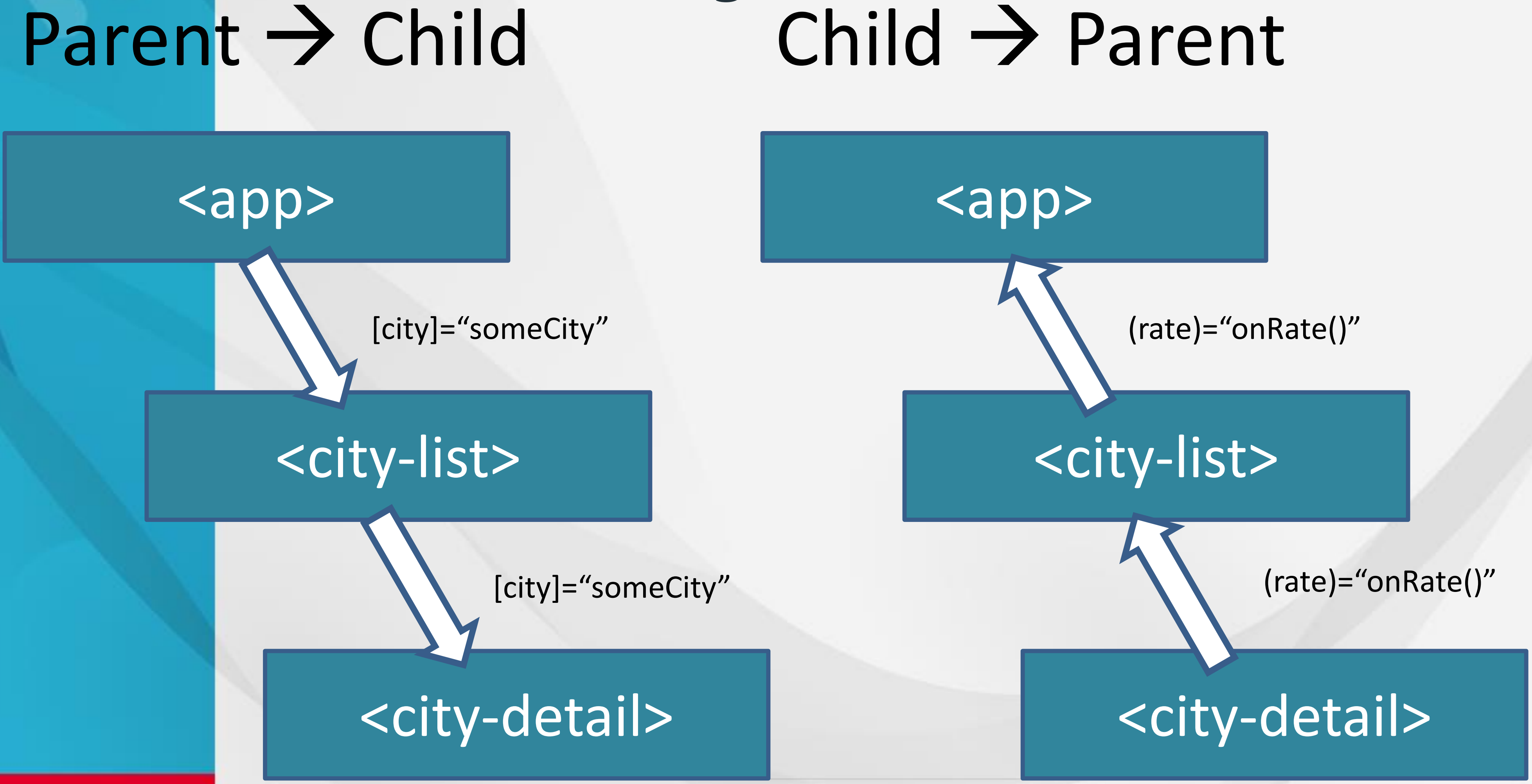
City details

Naam: Den Haag

Provincie: Zuid-Holland

Highlights: Binnenhof

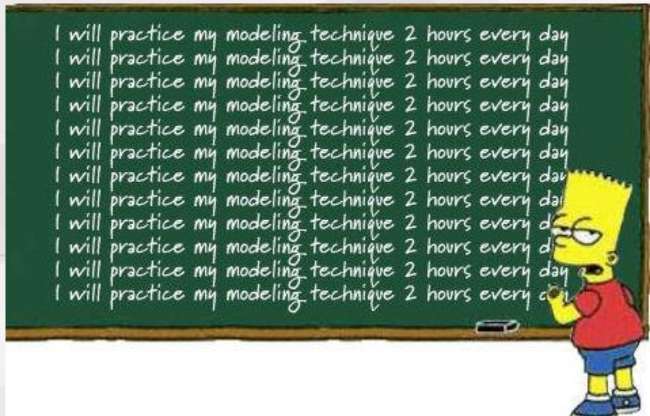
Samenvatting



Checkpoint

- Data flow naar Parent Component : werken met `@Output()` en `(eventName)="eventHandler($event)"`
- Je kunt allerlei typen Events meegeven
- Oefening: `6d)`
- Voorbeeld: `302-components-output`
- Meer info: http://victorsavkin.com/post/118372404541/the-core-concepts-of-angular-2

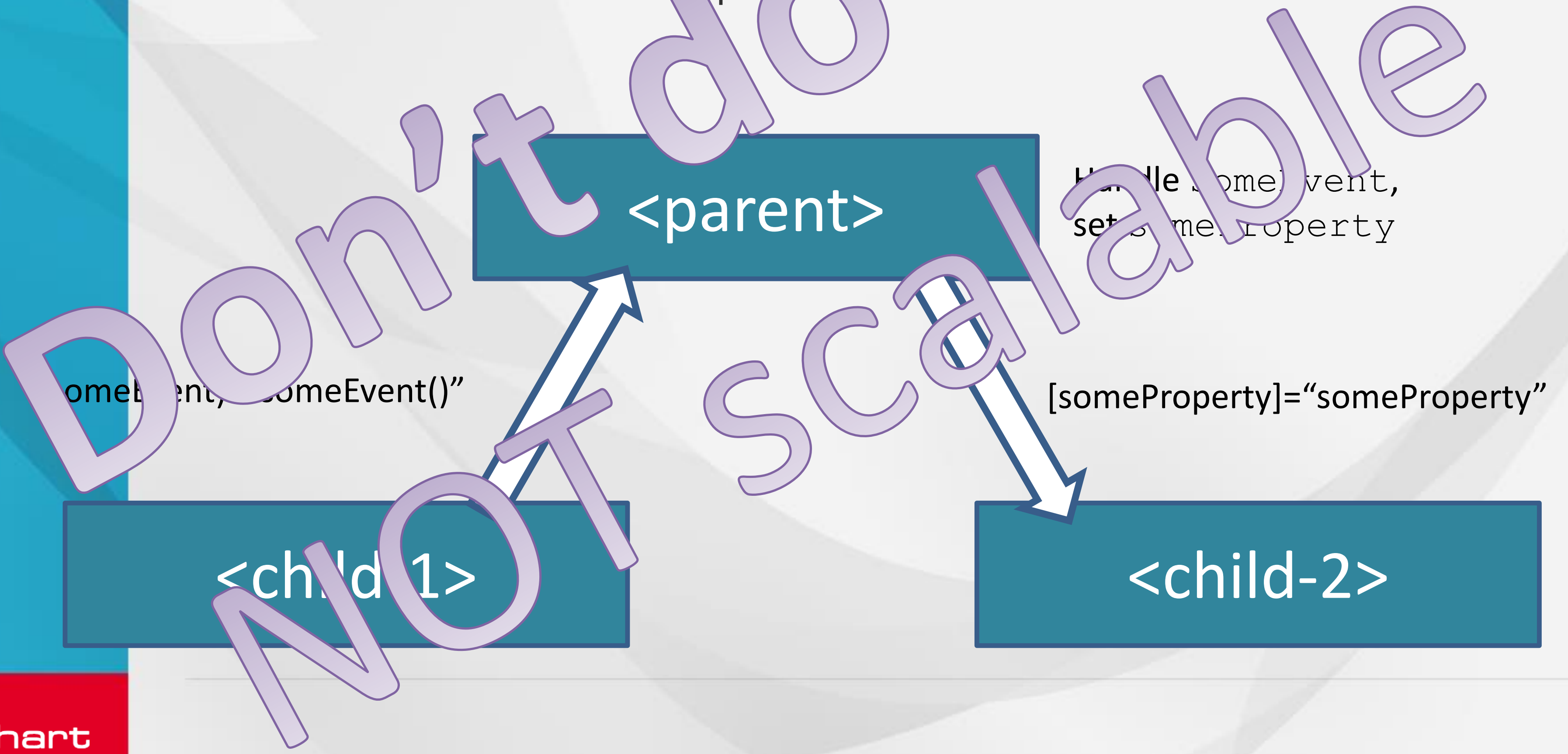
Oefening….


I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day

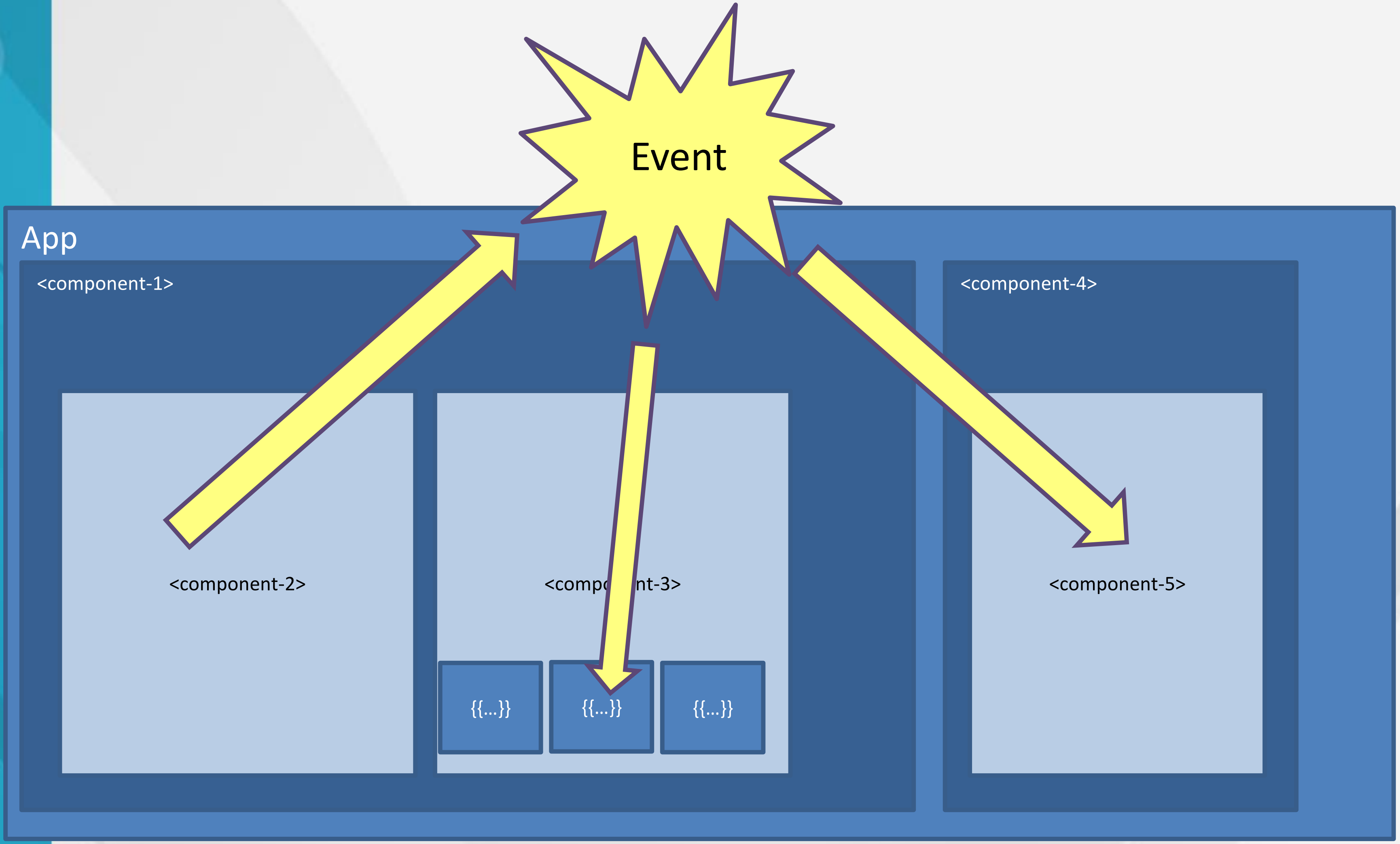
Communicatie tussen Siblings

Communicatie tussen siblings

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


Don't do this!
NOT scalable
<parent>
Handle someEvent, set someProperty
(someEvent)="someEvent()"
[someProperty]="someProperty"
<child-1>
<child-2>

Event bus

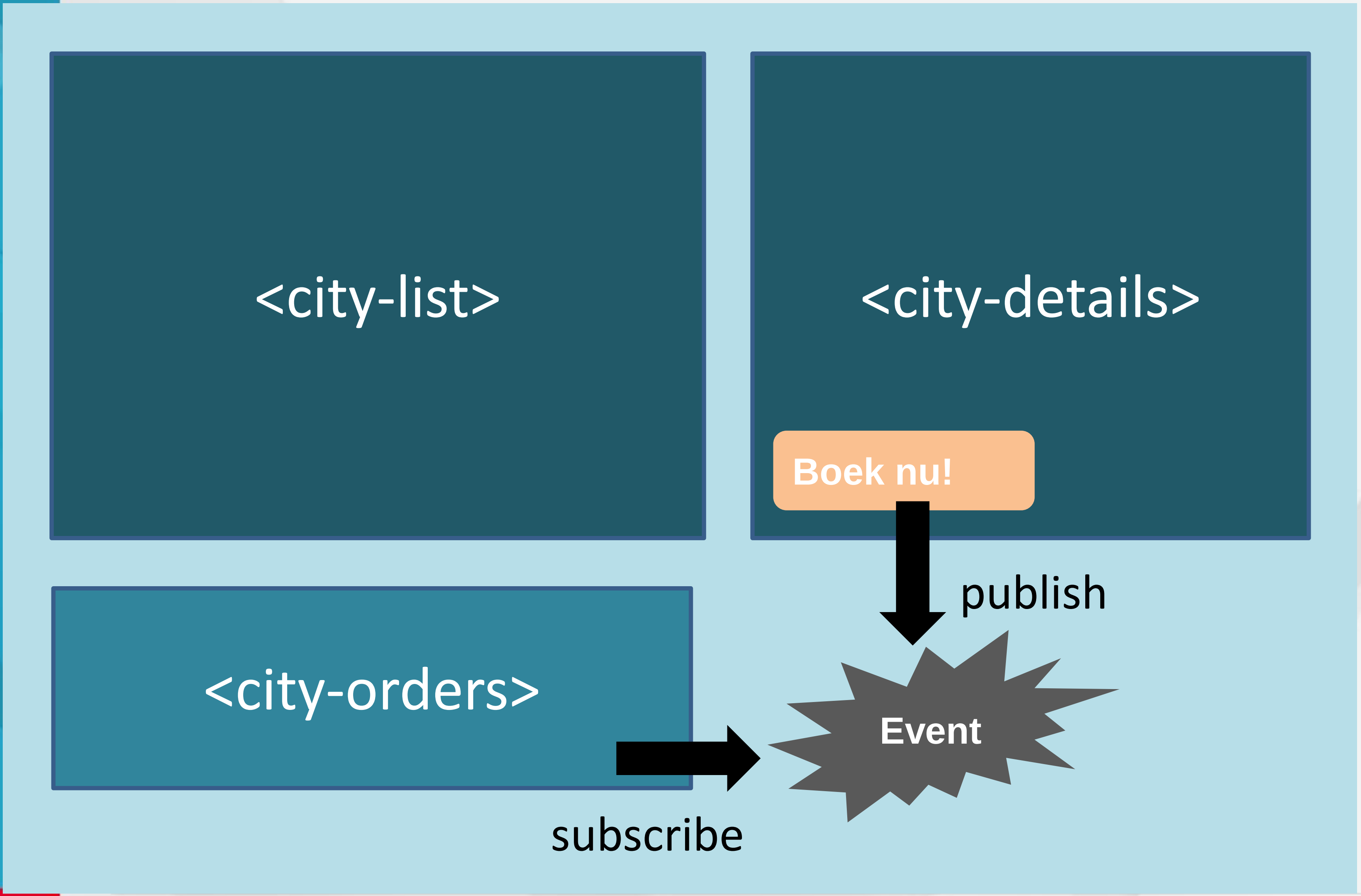


Opties

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
Event
<city-orders>
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)

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>
```

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

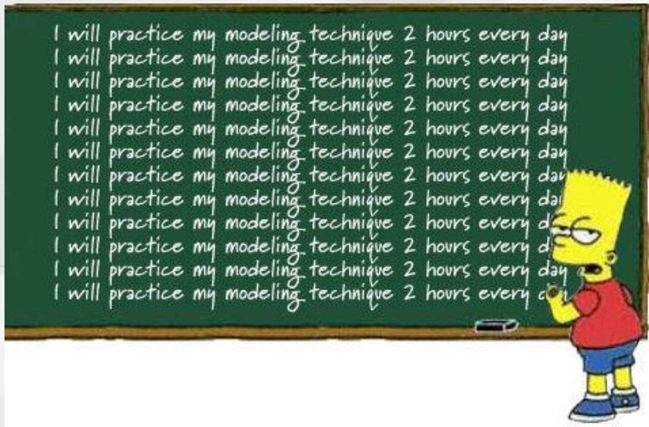
```
//city.orders.ts - Een soort 'winkelmandje',
// bijhouden welke stedentripjes zijn geboekt.
import …

@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...')
         )
   }
`     …
}
```

vijfhart IT-OPLEIDINGEN

Checkpoint

- Event Bus : 'onzichtbaar' werken met Streams en `Subject`
- Er zijn opties voor het werken met Observable Streams.
- Voorbeeld: `\303-pubsub-ordercomponent`
- Oefening 6e) e-commerce applicatie maken.

Oefening….

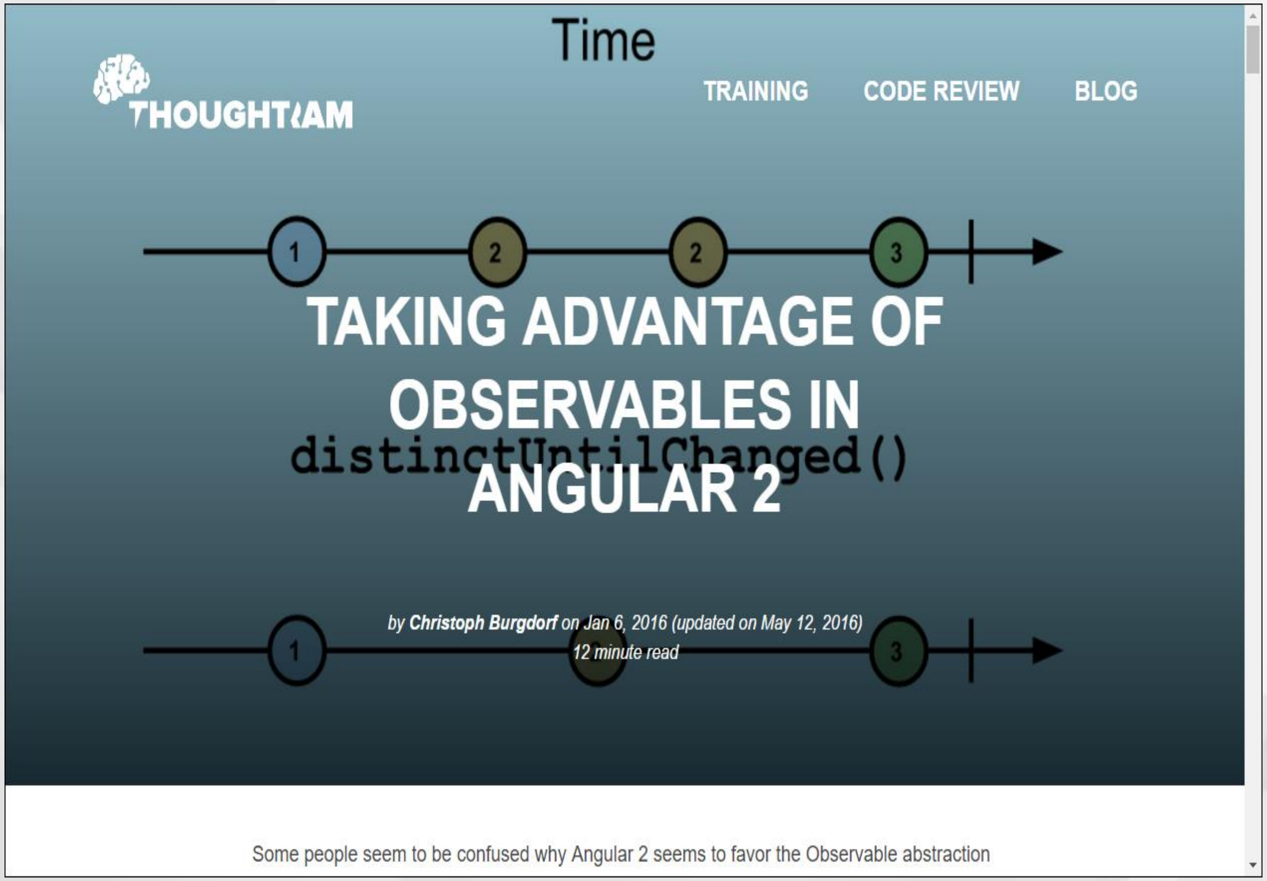


vijfhart IT-OPLEIDINGEN

Bonus sheets: more info

Some pointers to more information on the internet

Meer over Observables

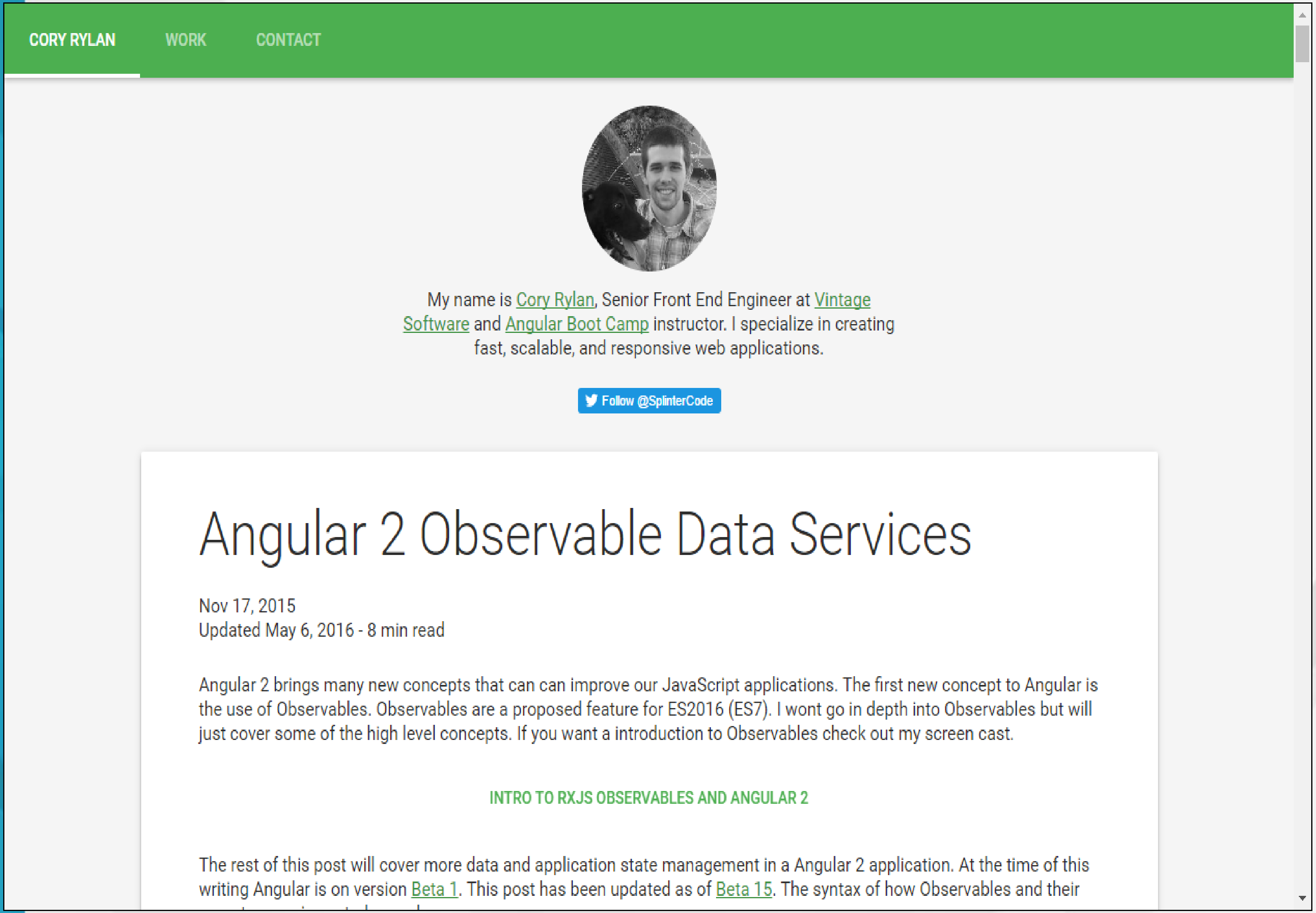


TAKING ADVANTAGE OF OBSERVABLES IN ANGULAR 2

by **Christoph Burgdorf** on Jan 6, 2016 (updated on May 12, 2016)
12 minute read

Some people seem to be confused why Angular 2 seems to favor the Observable abstraction

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



My name is Cory Rylan, Senior Front End Engineer at Vintage Software and Angular Boot Camp instructor. I specialize in creating fast, scalable, and responsive web applications.

Angular 2 Observable Data Services

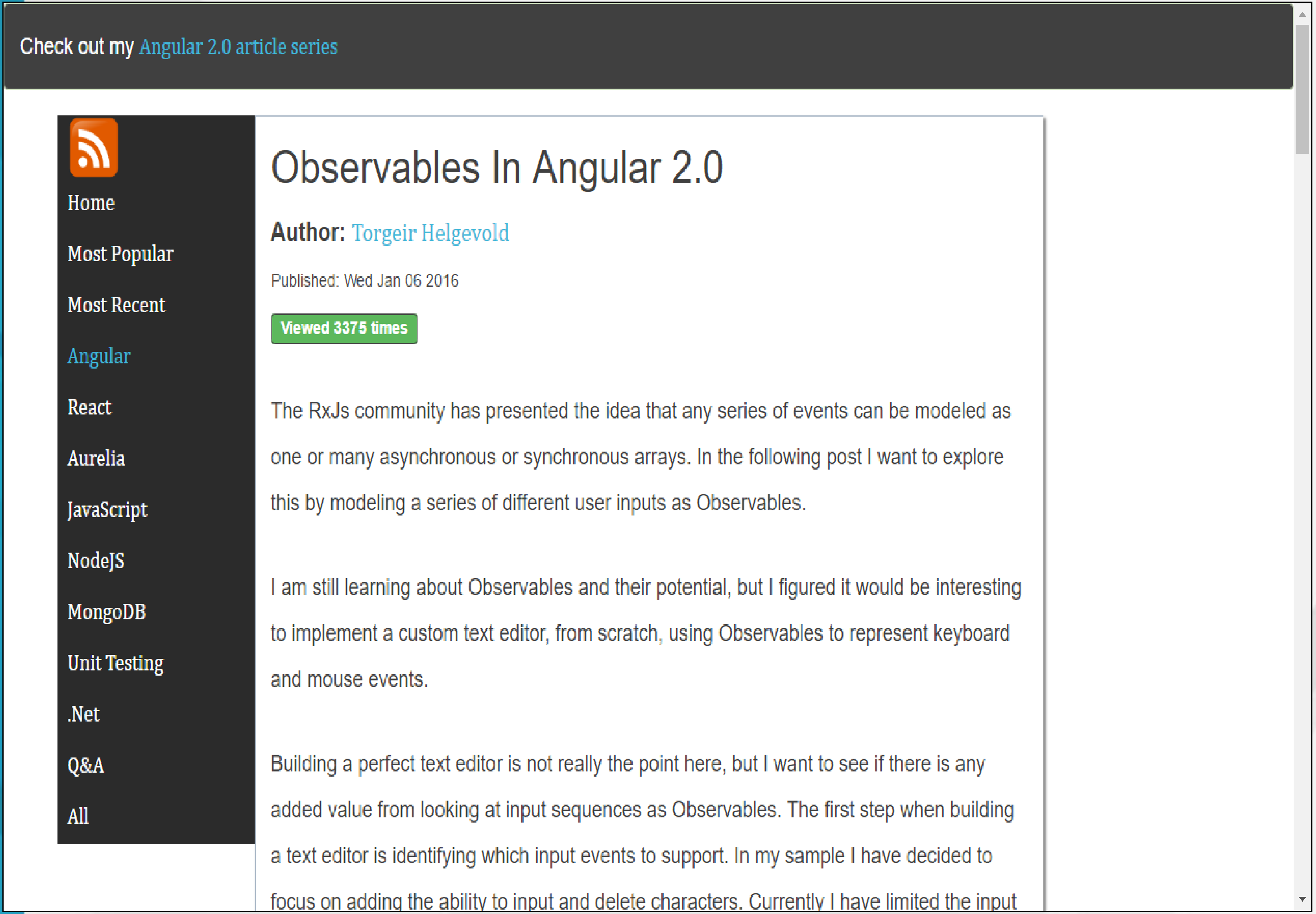
Nov 17, 2015
Updated May 6, 2016 - 8 min read

Angular 2 brings many new concepts that can can improve our JavaScript applications. The first new concept to Angular is the use of Observables. Observables are a proposed feature for ES2016 (ES7). I wont go in depth into Observables but will just cover some of the high level concepts. If you want a introduction to Observables check out my screen cast.

INTRO TO RXJS OBSERVABLES AND ANGULAR 2

The rest of this post will cover more data and application state management in a Angular 2 application. At the time of this writing Angular is on version Beta 1. This post has been updated as of Beta 15. The syntax of how Observables and their

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



Check out my Angular 2.0 article series

Observables In Angular 2.0

Author: Torgeir Helgevold

Published: Wed Jan 06 2016

Viewed 3375 times

The RxJs community has presented the idea that any series of events can be modeled as one or many asynchronous or synchronous arrays. In the following post I want to explore this by modeling a series of different user inputs as Observables.

I am still learning about Observables and their potential, but I figured it would be interesting to implement a custom text editor, from scratch, using Observables to represent keyboard and mouse events.

Building a perfect text editor is not really the point here, but I want to see if there is any added value from looking at input sequences as Observables. The first step when building a text editor is identifying which input events to support. In my sample I have decided to focus on adding the ability to input and delete characters. Currently I have limited the input

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