



Global Knowledge®

Atos

Angular

Module 2 – Data binding

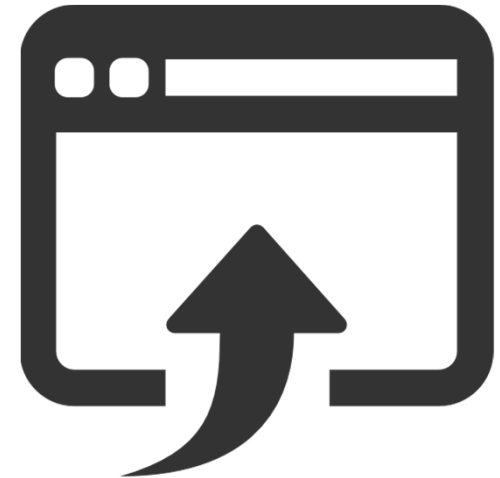
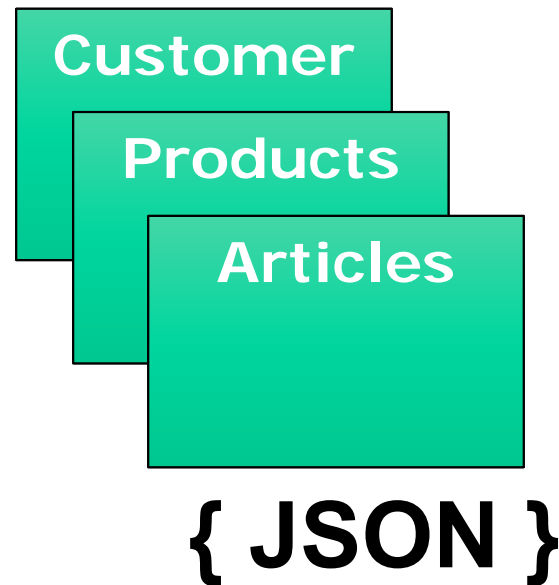
Peter Kassenaar –
info@kassenaar.com

WORLDWIDE LOCATIONS

BELGIUM CANADA COLOMBIA DENMARK EGYPT FRANCE IRELAND JAPAN KOREA MALAYSIA MEXICO NETHERLANDS NORWAY QATAR
SAUDI ARABIA SINGAPORE SPAIN SWEDEN UNITED ARAB EMIRATES UNITED KINGDOM UNITED STATES OF AMERICA

Wat is databinding

- Gegevens (data) tonen in de user interface
- Data afkomstig uit:
 - Controller / class
 - Database
 - User input
 - Andere systemen



Declaratieve syntaxis

- Vier manieren voor databinding in HTML-views/templates.
 1. Simple data binding
 2. Event binding
 3. One-way data binding
 4. Two-way data binding

1. Simple data binding syntax

Ongewijzigd ten opzichte van Angular 1. Dus nog steeds dubbele accolades:

```
<div>Stad: {{ city }}</div>
```

```
<div>Voornaam: {{ person.firstname }}</div>
```

Altijd: samenwerking met component/class

```
import {Component} from '@angular/core';

@Component({
  selector: 'hello-world',
  template: `<h1>Hello Angular 2</h1>
    <h2>Mijn naam is : {{ name }}</h2>
    <h2>Mijn favoriete stad is : {{ city }}</h2>
  `
})

export class AppComponent {
  name = 'Peter Kassenaar';
  city = 'Groningen'
}
```

Of: properties via constructor

- `export class AppComponent {`

`name: string;`

`city: string;`

`constructor() {`

`this.name = '...';`

`this.city = '...';`

`}`

`ngOnInit() {`

`this.name = 'Peter Kassenaar';`

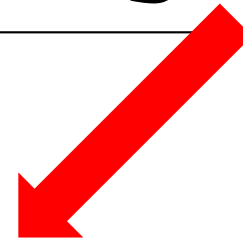
`this.city = 'Groningen';`

`}`

`}`

BEST PRACTICE:

use `ngOnInit()`



Binden via een lus: *ngFor

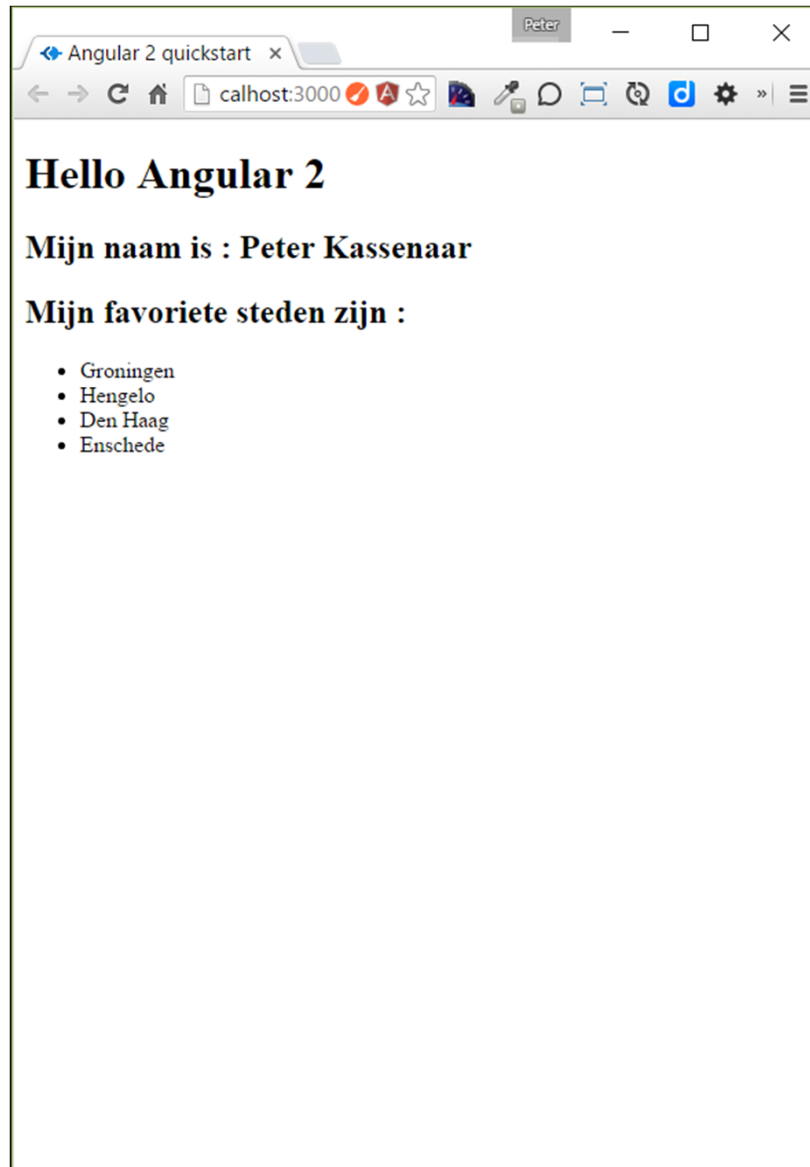
Template:

```
<h2>Mijn favoriete steden zijn :</h2>
<ul>
  <li *ngFor="let city of cities">{{ city }}</li>
</ul>
```

Class:

```
// Class met properties, array met cities
export class AppComponent {
  name:string;
  cities:string[];

  ngOnInit() {
    this.name = 'Peter Kassenaar';
    this.cities = ['Groningen', 'Hengelo', 'Den Haag', 'Enschede']
  }
}
```



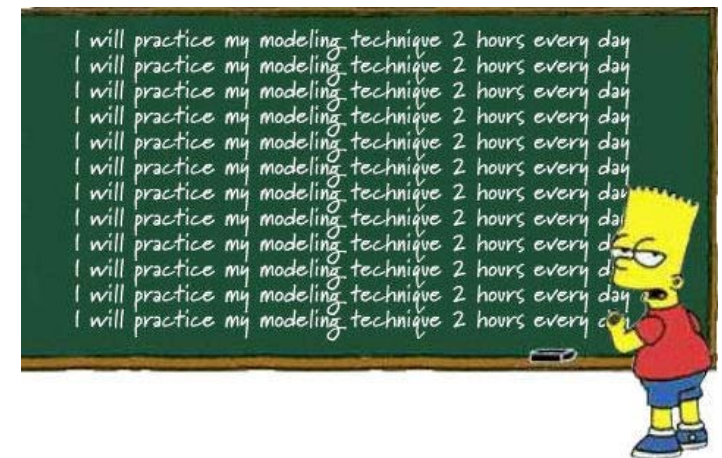
Meer info:

<https://angular.io/docs/ts/latest/guide/displaying-data.html>

Checkpoint

- Simple data binding {{ ... }}
- Properties van de class worden gebonden
- Lussen via *ngFor
- Data staat in een array
- Oefening 2a) en 2b)

Oefening....



Model maken (als in: MVC)

Class met properties die wordt geëxporteerd:

```
export class City{  
  constructor(  
    public id: number,  
    public name: string,  
    public province: string,  
  ){ }  
}
```

Let op de shorthand notatie bij `public id : number :`

1. Maakt lokale parameter
2. Maakt publieke parameter met zelfde naam
3. Initialiseert parameter bij instantiëring van de class met `new`

Model gebruiken

1. Model-class importeren

```
import {City} from './city.model'
```

2. Component aanpassen

```
export class AppComponent {  
  name = 'Peter Kassenaar';  
  cities = [  
    new City(1, 'Groningen', 'Groningen'),  
    new City(2, 'Hengelo', 'Overijssel'),  
    new City(3, 'Den Haag', 'Zuid-Holland'),  
    new City(4, 'Enschede', 'Overijssel'),  
  ]  
}
```

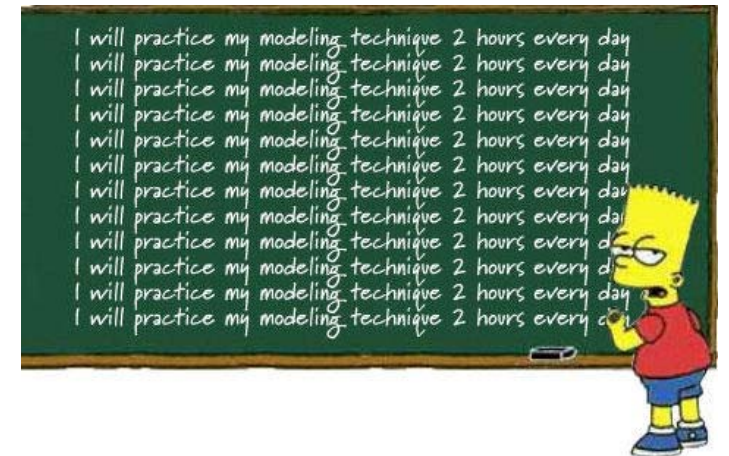
3. View aanpassen

```
<li *ngFor="let city of cities">{{ city.id }} - {{ city.name }}</li>
```

Checkpoint

- Model maken en gebruiken: Class of interface
- Denk aan de juiste `import`-statements
- Best practice; plaats je class (of interface) in de map `/shared`.
- Oefening 2c)

Oefening....



Voorwaardelijk tonen met `*ngIf`

Gebruik de directive `*ngIf` (let op het sterretje!)

```
<h2 *ngIf="cities.length > 3">Jij hebt veel favoriete steden!</h2>
```



Externe templates

Als je niet van inline HTML houdt:

```
@Component({  
  selector    : 'hello-world',  
  templateUrl: './app.html'  
})
```



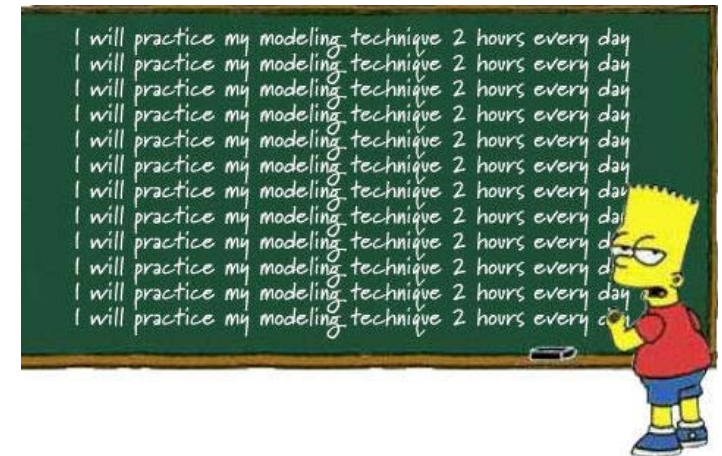
Bestand app.html

```
<!-- HTML in externe template -->  
<h1>Hello Angular 2</h1>  
<p>Dit is een externe template</p>  
<h2>Mijn naam is : {{ name }}</h2>  
<h2>Mijn favoriete steden zijn :</h2>  
...
```

Checkpoint

- Simple data binding `{{ ... }}`
- Gebruik bij voorkeur een Model (class of interface)
- Lussen en voorwaardelijke statement via `*ngFor` en `*ngIf`
- Eventueel : externe HTML-templates
- Oefening 2c) en 2d)

Oefening....





User input en event binding

Reageren op mouse, keyboard, hyperlinks en meer

Event binding syntax

Gebruik ronde haken voor events:

Angular 1:

```
<div ng-click="handleClick()">...</div>
```

Angular 2:

```
<div (click)="handleClick()">...</div>
```

```
<input (blur)="onBlur()">...</div>
```

DOM-events

Angular2+ kan naar *e/k* DOM-event luisteren, zonder dat er een aparte directive voor nodig is:

The screenshot shows the MDN 'Event reference' page. The left sidebar lists various event categories, with 'DOM events' highlighted by a red box. The main content area shows the 'Standard events' section, which includes a table of events.

| Event Name | Event Type | Specification | Fired when... |
|------------------------------------|----------------------|------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|
| abort | UIEvent | DOM L3 | The loading of a resource has been aborted. |
| abort | ProgressEvent | Progress and XMLHttpRequest | Progression has been terminated (not due to an error). |
| abort | Event | IndexedDB | A transaction has been aborted. |
| afterprint | Event | HTML5 | The associated document has started printing or the print preview has been closed. |
| animationend | AnimationEvent | CSS Animations | A CSS animation has completed. |
| animationiteration | AnimationEvent | CSS Animations | A CSS animation is repeated. |
| animationstart | AnimationEvent | CSS Animations | A CSS animation has started. |
| audioprocess | AudioProcessingEvent | Web Audio API The definition of 'audioprocess' in that specification. | The input buffer of a <code>ScriptProcessorNode</code> is ready to be processed. |
| audioend | Event | Web Speech API | The user agent has finished capturing audio for speech recognition. |
| audiostart | Event | Web Speech API | The user agent has started to capture audio for speech recognition. |
| beforeprint | Event | HTML5 | The associated document is about to be printed or previewed for printing. |
| beforeunload | BeforeUnloadEvent | HTML5 | |

<https://developer.mozilla.org/en-US/docs/Web/Events>

3. Niet-DOM events binden

- Niet-DOM events binden: `@HostListener()`
- Luister naar events op het `window`-object, decoreer Event Listener functie.
- Doorgeven van `$event` is optioneel
- Bijvoorbeeld:

```
// Decorator voor capture van non-DOM events  
@HostListener('window:offline', ['$event'])  
onOffline(event) {  
    this.msg = 'We zijn offline!';  
    console.log('we zijn nu offline ==>', event);  
}
```

```
// Luisteren naar niet-DOM events: gebruik
// de decorator @HostListener()
@HostListener('window:offline', ['$event']) // $event is optioneel
onOffline(e) {
    console.log(e);
    this.msg = 'We zijn offline!';
    console.log('We zijn offline!');
}

@HostListener('window:online')
onOnline() {
    this.msg = 'We zijn weer online! Ga synchroniseren';
    console.log('We zijn online!');
}
```

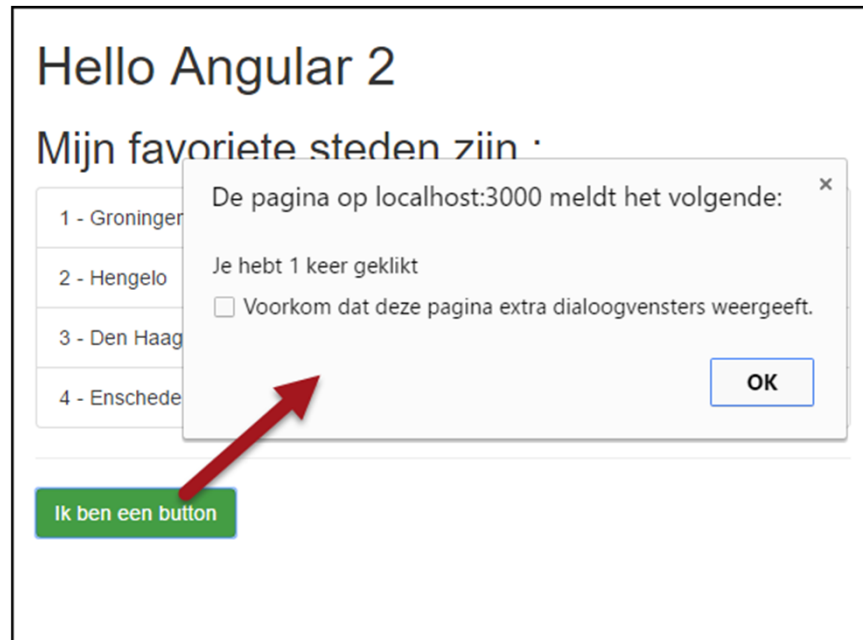
Voorbeeld event binding

HTML

```
<!-- Event binding voor een button -->  
<button class="btn btn-success"  
    (click)="btnClick()">Ik ben een button</button>
```

Class

```
export class AppComponent {  
    ...  
    counter: number = 0;  
  
    btnClick(){  
        alert('Je hebt ' + ++this.counter + ' keer geklikt');  
    }  
}
```



- Veel editors geven intellisense voor de beschikbare events

1. Event binding met \$event

HTML

```
<input type="text" class="input-lg" placeholder="Plaatsnaam..."  
      (keyup)="onKeyUp($event)"><br>
```

```
<p>{{ txtKeyUp }}</p>
```

Class

```
// 2. Binden aan keyUp-event in de textbox  
onKeyUp(event:any){  
    this.txtKeyUp = event.target.value + ' - '  
}
```

2. Binding met local template variable

Declareer *local template variable* met # → Het hele element wordt doorgegeven aan de component

HTML

```
<input type="text" class="input-lg" placeholder="Plaatsnaam..."
      #txtCity (keyup.enter)="betterKeyUp(txtCity)">
<h3>{{ txtCity.value }}</h3>
```

Class:

```
// 3. Binden aan keyUp-event via local template variable
betterKeyUp(txtCity:HTMLInputElement){
  //... Doe iets met txtCity...
}
```

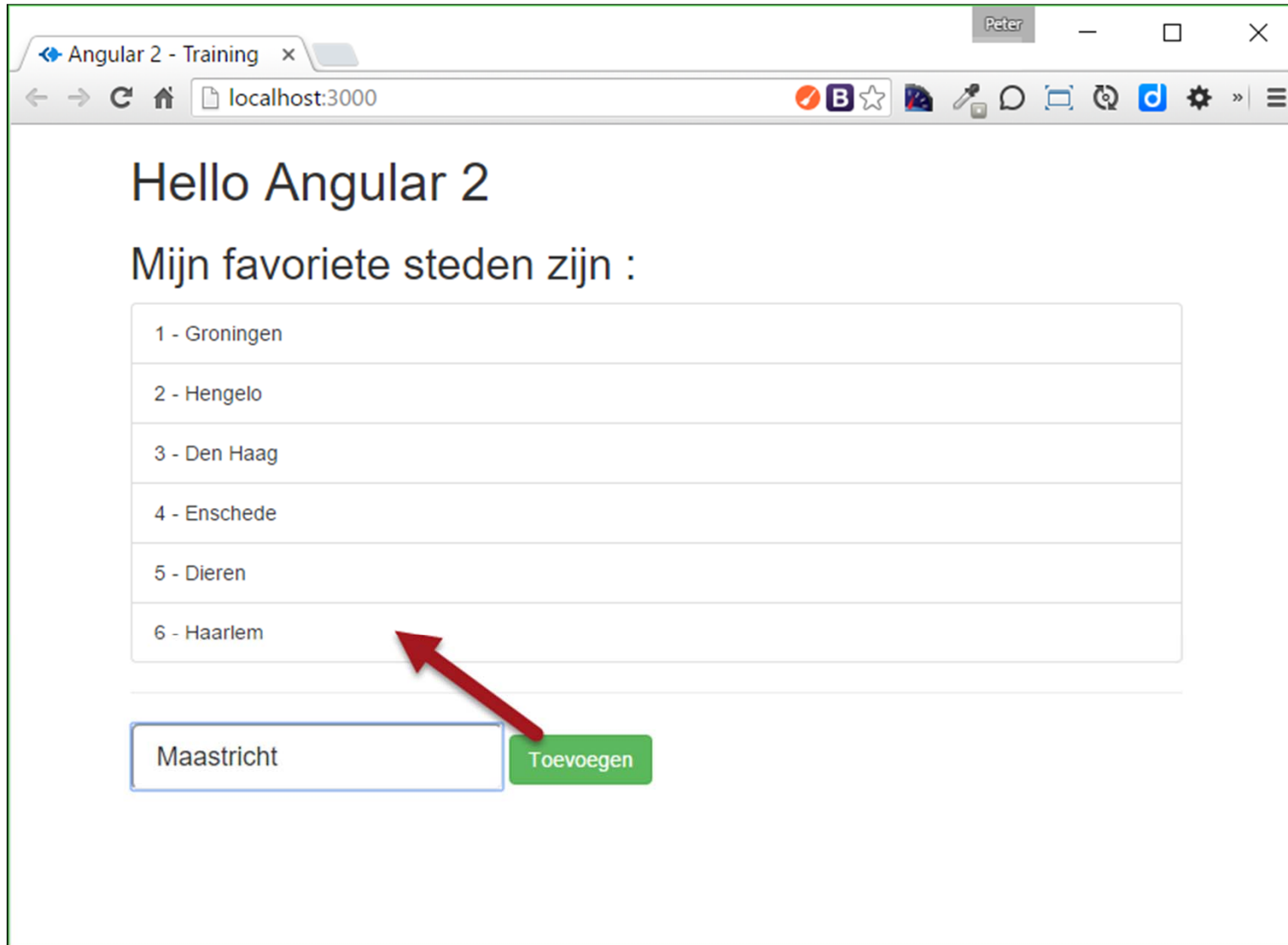

Putting it all together...

HTML

```
<input type="text" class="input-lg" placeholder="Plaatsnaam..." #txtCity>
<button class="btn btn-success"
    (click)="addCity(txtCity)">Toevoegen
</button>
```

Class

```
export class AppComponent {
    // Properties voor de component/class
    ...
    addCity(txtCity:HTMLInputElement) {
        let newID    = this.cities.length + 1;
        let newCity = new City(newID, txtCity.value, 'Onbekend');
        this.cities.push(newCity);
        txtCity.value = '';
    }
}
```

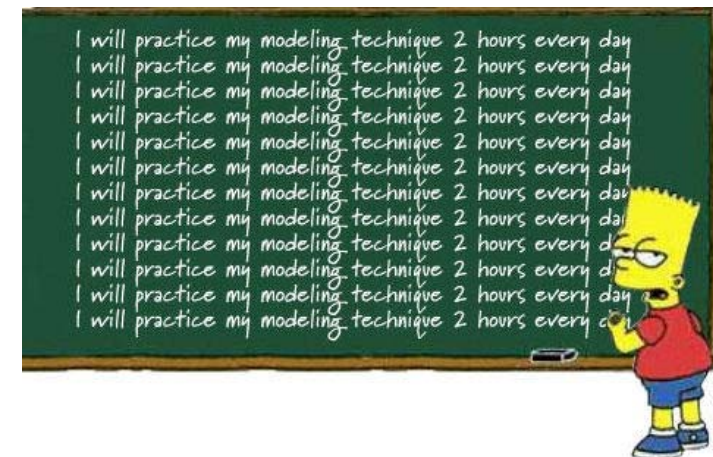


Verder lezen/meer informatie: <https://angular.io/docs/ts/latest/guide/user-input.html>

Checkpoint

- Event binding wordt aangegeven met `(eventName) = "..."`
- Events worden afgehandeld door een event handler-functie in de component
- Gebruik `$event` om het complete, ruwe browserevent door te geven aan de controller
- Gebruik `#` voor *local template variable*
- Op deze manier zijn eenvoudige CRUD-operations te realiseren.
- Oefening 3b) , 3c) , 3d) , 3e)

Oefening....





Attribute & property binding

Eigenschappen binden aan HTML-attributen en DOM-properties

Attribute binding syntax

Rechtstreeks binden aan properties van HTML-elementen.

Ook wel: *one-way binding*.

Gebruik blokhaken syntaxis

Angular 1:

```
<div ng-hide="true|false">...</div>
```

Angular 2:

```
<div [hidden]="true">...</div>
```

Of :

```
<div [hidden]="person.hasEmail">...</div>
```

```
<div [style.backgroundColor]=" 'yellow' ">...</div>
```

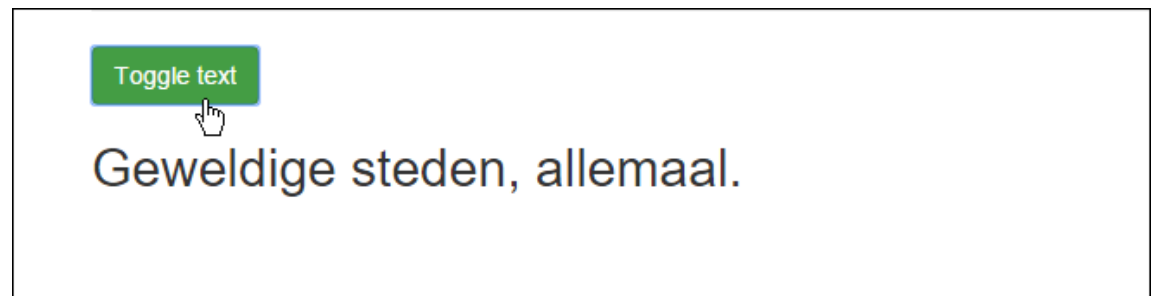
Voorbeeld attribute binding

HTML

```
<!-- Attribute binding -->  
<button class="btn btn-success" (click)="toggleText()">Toggle text</button>  
<h2 [hidden]="textVisible">Geweldige steden, allemaal.</h2>
```

Class

```
// attribuut toggelen: tekst zichtbaar/onzichtbaar maken.  
toggleText(){  
  this.textVisible = !this.textVisible;  
}
```



Bijvoorbeeld...

HTML

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

Class

```
export class AppComponent {
  // ...
  currentCity:City    = null;
  cityPhoto:string    = '';

  // Geselecteerde city updaten in de ui. Nieuw : ES6 String interpolation
  updateCity(city:City) {
    this.currentCity = city;
    this.cityPhoto   = `img/${this.currentCity.name}.jpg`;
  }
}
```

Demo:

..\103-attributebinding\app\app-02.html en

..\app-02.component.ts

Hello Angular 2

Mijn favoriete steden zijn :

1 - Groningen

2 - Hengelo

3 - Den Haag

4 - Enschede

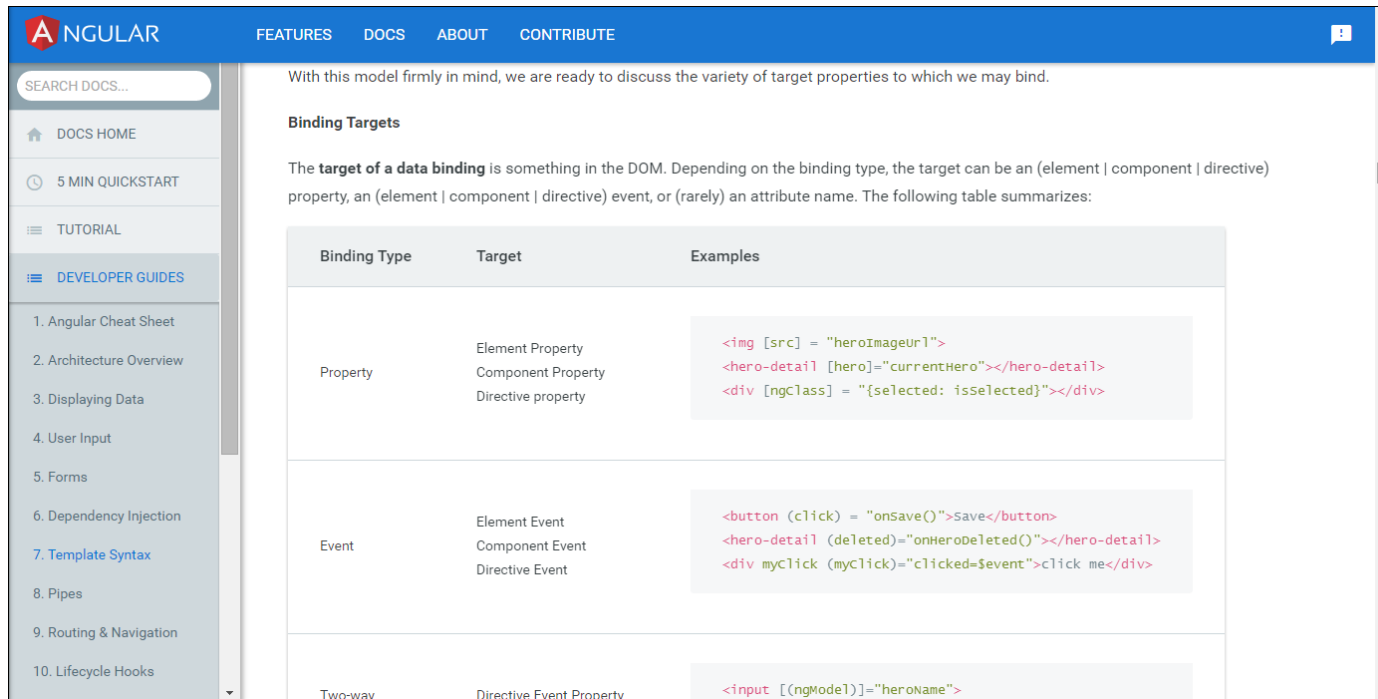


mijn stad: Groningen

Meer informatie: <https://angular.io/docs/ts/latest/guide/template-syntax.html#!#property-binding>

Meer binding-opties

- Attribute binding en DOM-property binding
- Class binding
- Style binding
- <https://angular.io/docs/ts/latest/guide/template-syntax.html>



The screenshot shows the Angular Developer Guides page. The left sidebar contains a search bar and a list of guides, with '7. Template Syntax' highlighted. The main content area is titled 'Binding Targets' and explains that the target of a data binding is something in the DOM. It includes a table summarizing different binding types and their targets.

With this model firmly in mind, we are ready to discuss the variety of target properties to which we may bind.

Binding Targets

The **target of a data binding** is something in the DOM. Depending on the binding type, the target can be an (element | component | directive) property, an (element | component | directive) event, or (rarely) an attribute name. The following table summarizes:

| Binding Type | Target | Examples |
|--------------|--------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Property | Element Property Component Property Directive property | <pre> <hero-detail [hero]="currentHero"></hero-detail> <div [ngClass] = "{selected: isSelected}"></div></pre> |
| Event | Element Event Component Event Directive Event | <pre><button (click) = "onSave()">Save</button> <hero-detail (deleted)="onHeroDeleted()"></hero-detail> <div myClick (myClick)="clicked=\$event">click me</div></pre> |
| Two-way | Directive Event Property | <pre><input [(ngModel)]="heroName"></pre> |



Two-way binding

User interface en logica gelijktijdig updaten

Two way binding syntax

Is een tijdje weg geweest uit Angular 2, maar op veler verzoek toch teruggekeerd

Angular 1:

```
<input ng-model="person.firstName" />
```

Angular 2: de notatie is een beetje bizar:

```
<input [(ngModel)]="person.firstName" />
```

[(ngModel)] gebruiken

HTML

```
<input type="text" class="input-lg" [(ngModel)]="newCity" />
<h2>{{ newCity }}</h2>
```

Dat is shorthand-notatie voor:

```
<!-- Two-way binding met uitgebreide syntaxis -->
<input type="text" class="input-lg"
      [value]="newCityExtended"
      (input)="newCityExtended = $event.target.value" />
<h2>{{ newCityExtended }}</h2>
```

FormsModule importeren

- Vroeger maakte de Formulier-functionaliiteit standaard deel uit van Angular.
- Nu niet meer – apart importeren in `app.module.ts`!
- `import {FormsModule} from "@angular/forms";`
- ...
- `imports : [BrowserModule, FormsModule],`

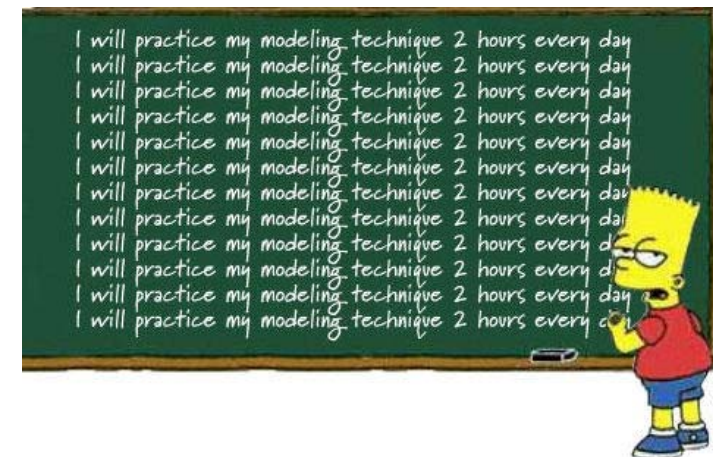
Dus: data doorgeven van View → Controller

1. Using `$event`
2. Using a Local Template Variabele `#NameVar`
3. Using `[(ngModel)]` (to be used in simple situations, mostly not on complex forms)
4. `HostBinding/@HostListener` (via `@`-decorators)
5. Decorator `@ViewChild()`

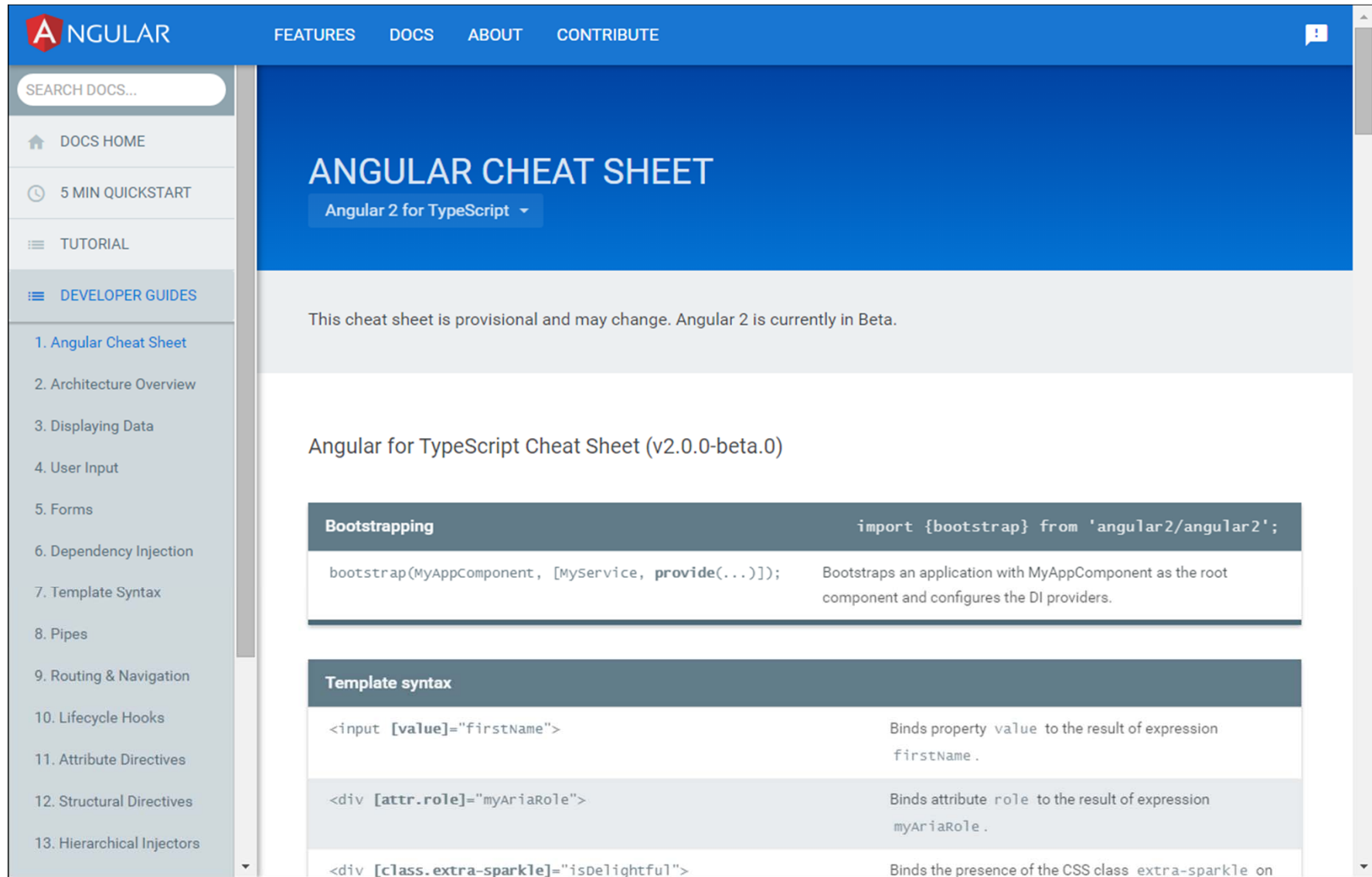
Checkpoint

- Attribute binding wordt aangegeven met `[attributeName]="..."`
- Op deze manier maak je attributen van HTML-tags dynamisch
- Aan de rechterkant plaats je een variabele van de component
- Via `[(ngModel)]` kun je in Angular two-way binding gebruiken
- Oefening 4a), 4b), 4c)

Oefening....



Binding cheat sheet

The image is a screenshot of the Angular website's 'ANGULAR CHEAT SHEET' page. The page has a blue header with the Angular logo and navigation links: FEATURES, DOCS, ABOUT, and CONTRIBUTE. A search bar is located in the top left. On the left side, there is a sidebar with a 'DEVELOPER GUIDES' section expanded, showing a list of topics from '1. Angular Cheat Sheet' to '13. Hierarchical Injectors'. The main content area has a blue background with the title 'ANGULAR CHEAT SHEET' and a sub-header 'Angular 2 for TypeScript'. Below this, a grey box states: 'This cheat sheet is provisional and may change. Angular 2 is currently in Beta.' The main content is titled 'Angular for TypeScript Cheat Sheet (v2.0.0-beta.0)'. It contains two sections: 'Bootstrapping' and 'Template syntax'. The 'Bootstrapping' section shows the code `import {bootstrap} from 'angular2/angular2';` and `bootstrap(MyAppComponent, [MyService, provide(...)]);`, with an explanation that it bootstraps an application with MyAppComponent as the root component and configures the DI providers. The 'Template syntax' section shows three examples of Angular template syntax: `<input [value]="firstName">` (binds property value to the result of expression firstName), `<div [attr.role]="myAriaRole">` (binds attribute role to the result of expression myAriaRole), and `<div [class.extra-sparkle]="isDelightful">` (binds the presence of the CSS class extra-sparkle on).

<https://angular.io/docs/ts/latest/guide/cheatsheet.html>

Ingebouwde directives

Veel directives konden vervallen door de nieuwe syntaxis. Er zijn er nog maar weinig over.

Directives die het DOM manipuleren: herkenbaar aan sterretje/asterisk

```
<div *ngFor="person of Persons">...</div>
```

```
<div *ngIf="showDiv">...</div>
```

```
<div [ngClass]="setClasses()">...</div>
```

```
<div [ngStyle]="setStyles()">...</div>
```

Samenvatting...

- Databinding is in Angular 2 vernieuwd
- Leer werken met de nieuwe notatie voor DOM- en Attribute binding, event binding en two-way binding
- Pas altijd de Component en de bijbehorende View aan.
- Veel concepten komen overeen, de uitwerking is totaal nieuw, in vergelijking met Angular 1