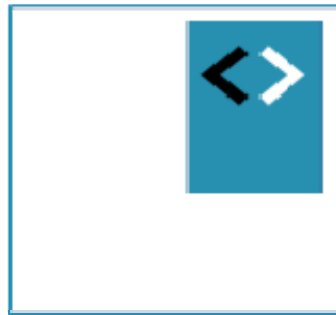




Angular Fundamentals

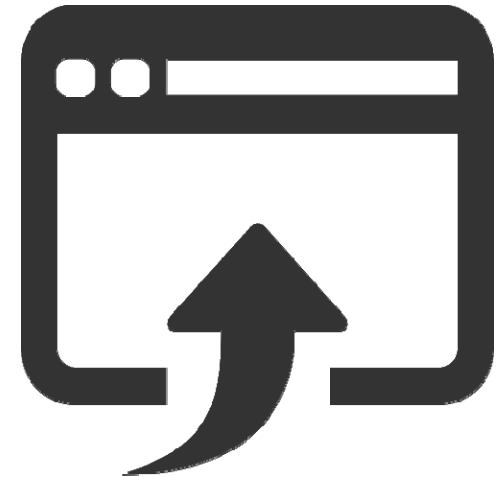
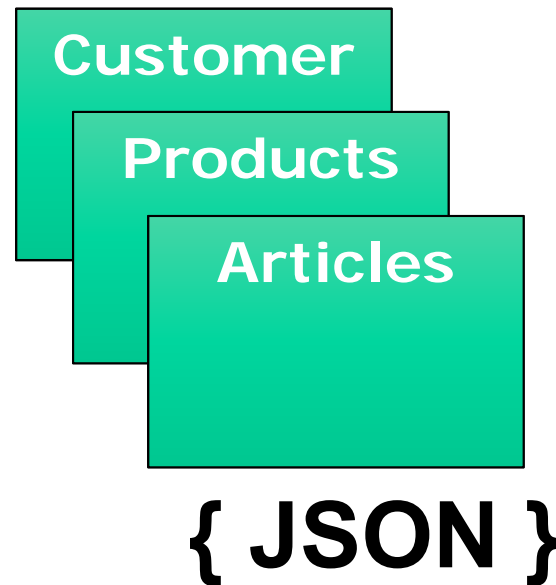
Module - Databinding



Peter Kassenaar –
info@kassenaar.com

Wat is databinding

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



Declaratieve syntaxis

- Nieuwe notatiewijzen in HTML-views/partials.
 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

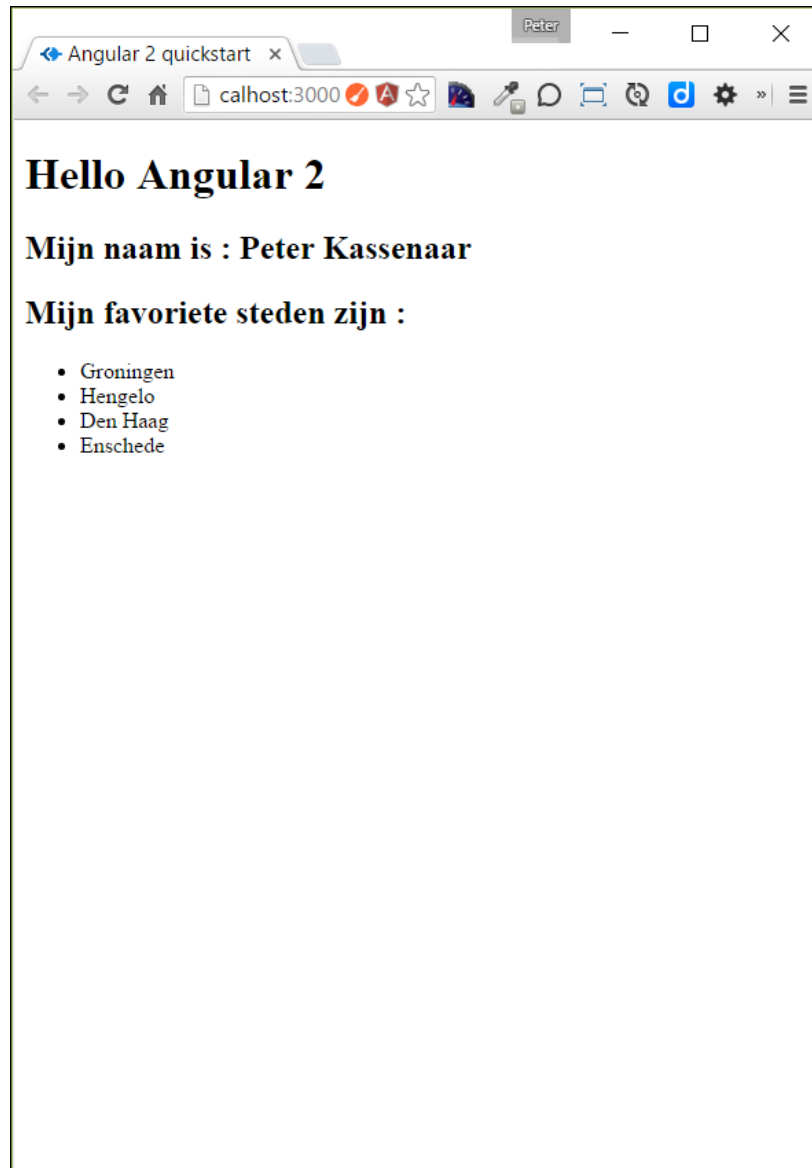
Template:

```
<h2>Mijn favoriete steden zijn :</h2>
<ul>
  <li *ngFor="let city of cities">{{ city }}</li> // vanaf .rc1
</ul>
```

Class:

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

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



Meer info:

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

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>
```

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'  
})
```



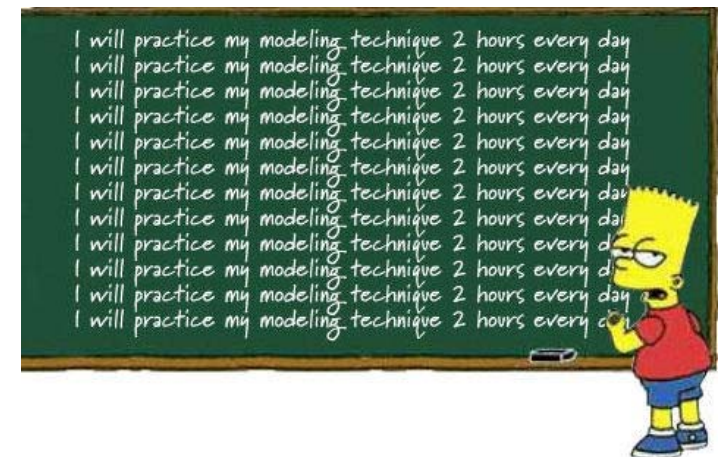
Bestand app.html

```
<!-- 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 `{{ ... }}`
- Properties van de class worden gebonden
- 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>
```

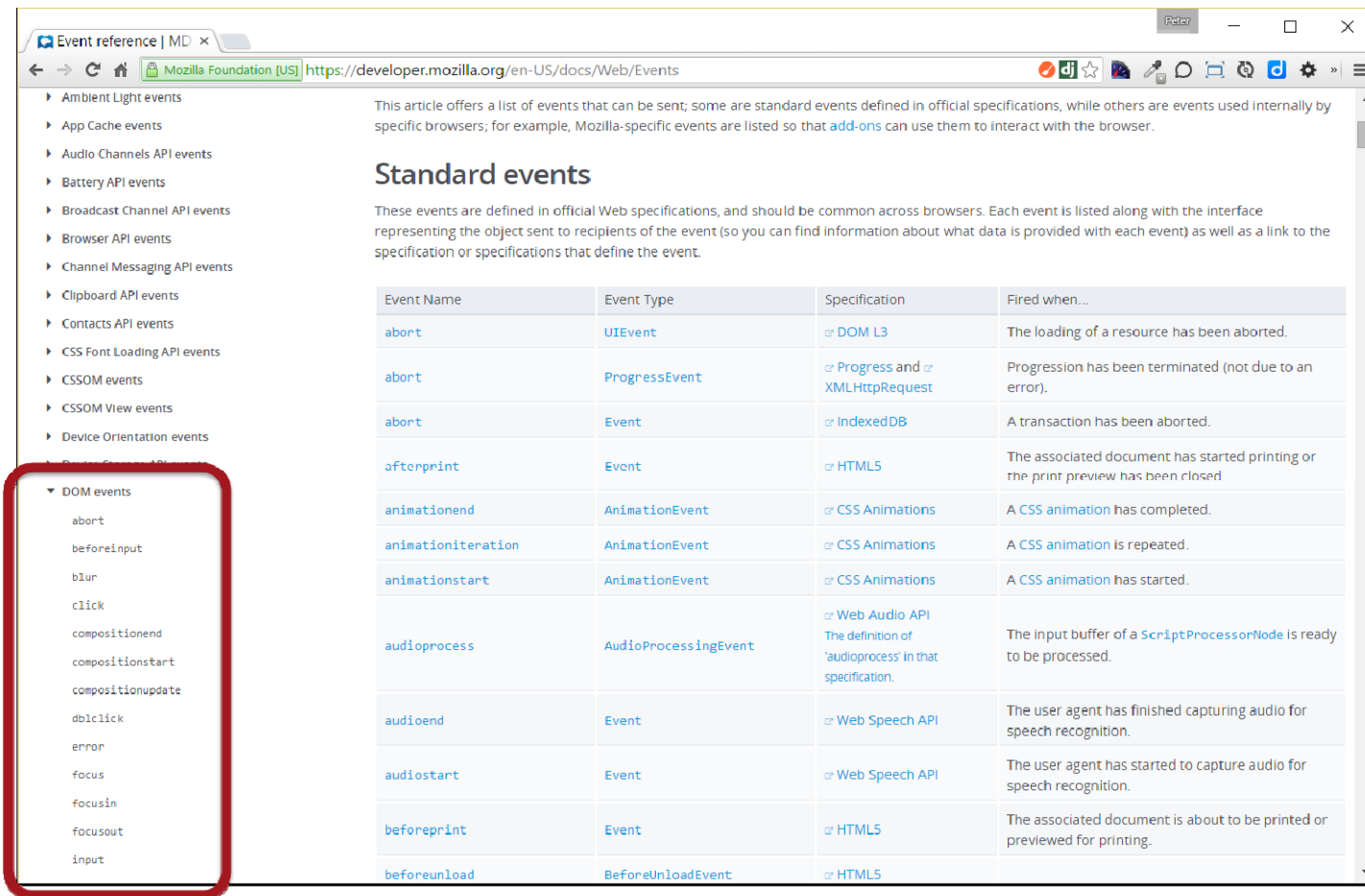
Angular 2:

```
<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 sidebar on the left lists various event categories, with 'DOM events' highlighted by a red box. The main content area displays a table of standard events, including their names, types, specifications, and when they are fired.

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 ScriptProcessorNode 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!');
}
```

}

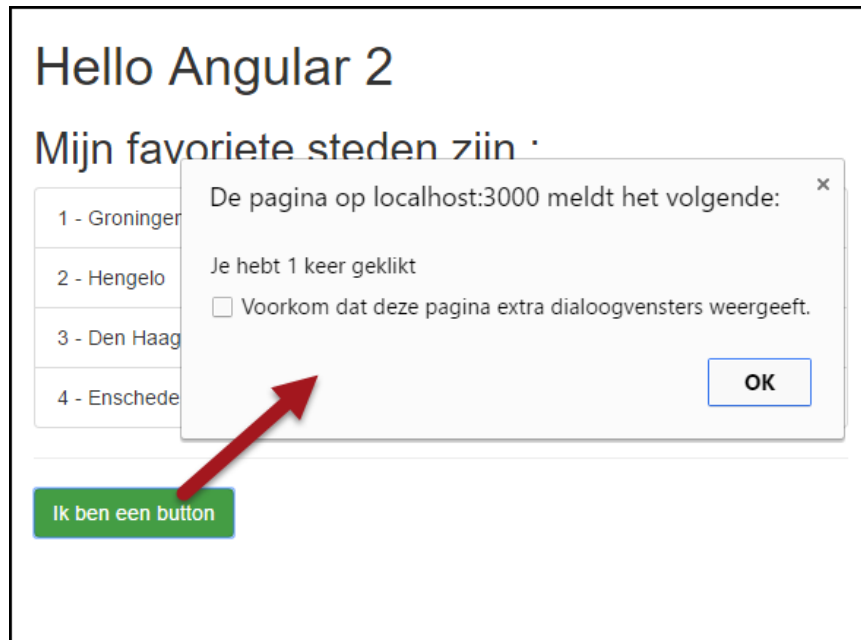
Voorbeeld event binding

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)="betterKeyUp(txtCity)">
<h3>{{ txtCity.value }}</h3>
```

Class:

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

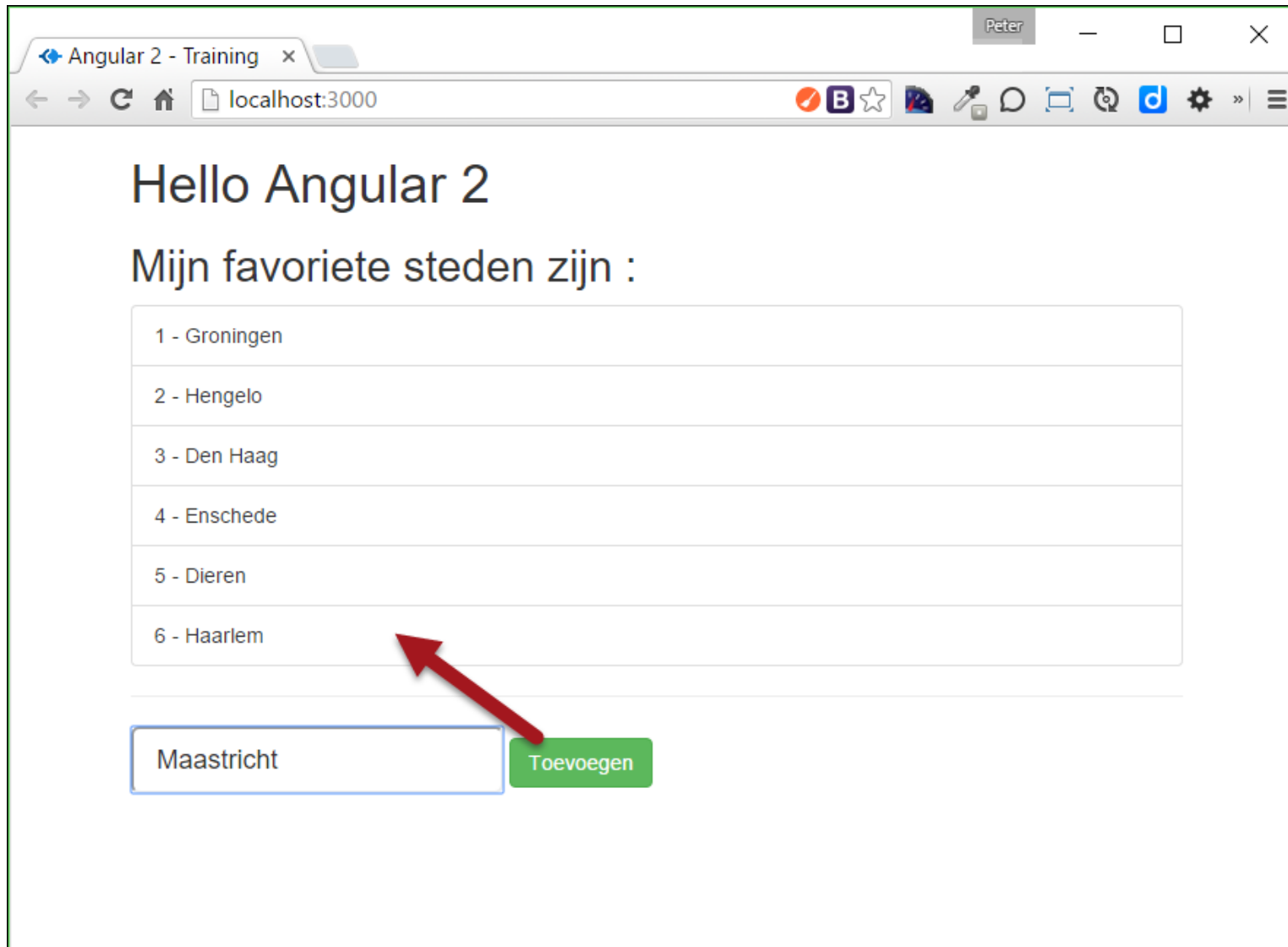
Putting it all together...

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 = '';
    }
}
```

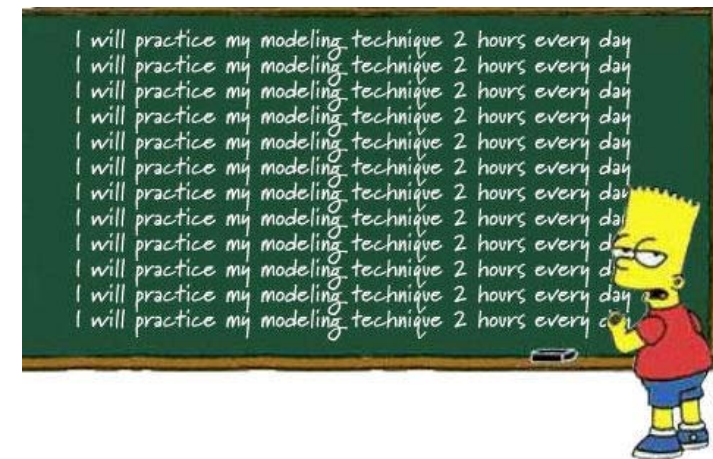


Verder lezen/meer informatie: <https://angular.io/docs/ts/latest/guide/user-input.html>

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)

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>
```

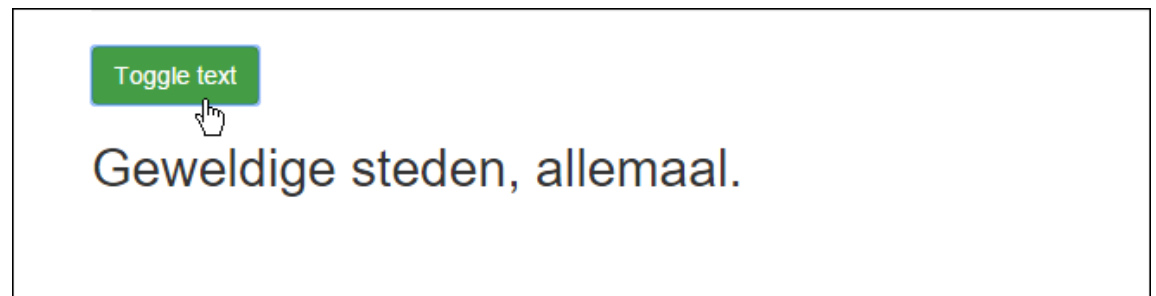
Voorbeeld attribute binding

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

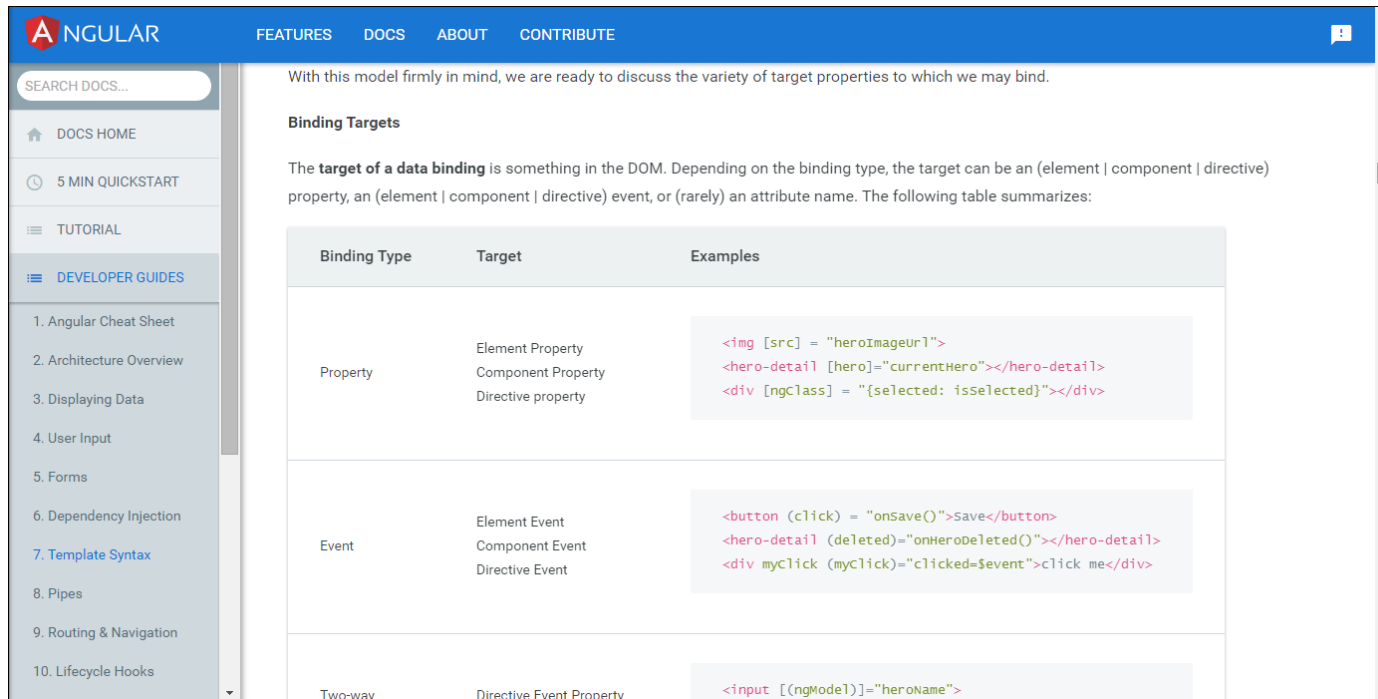


mijn stad: Groningen

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' selected. 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 binding types and 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>
```

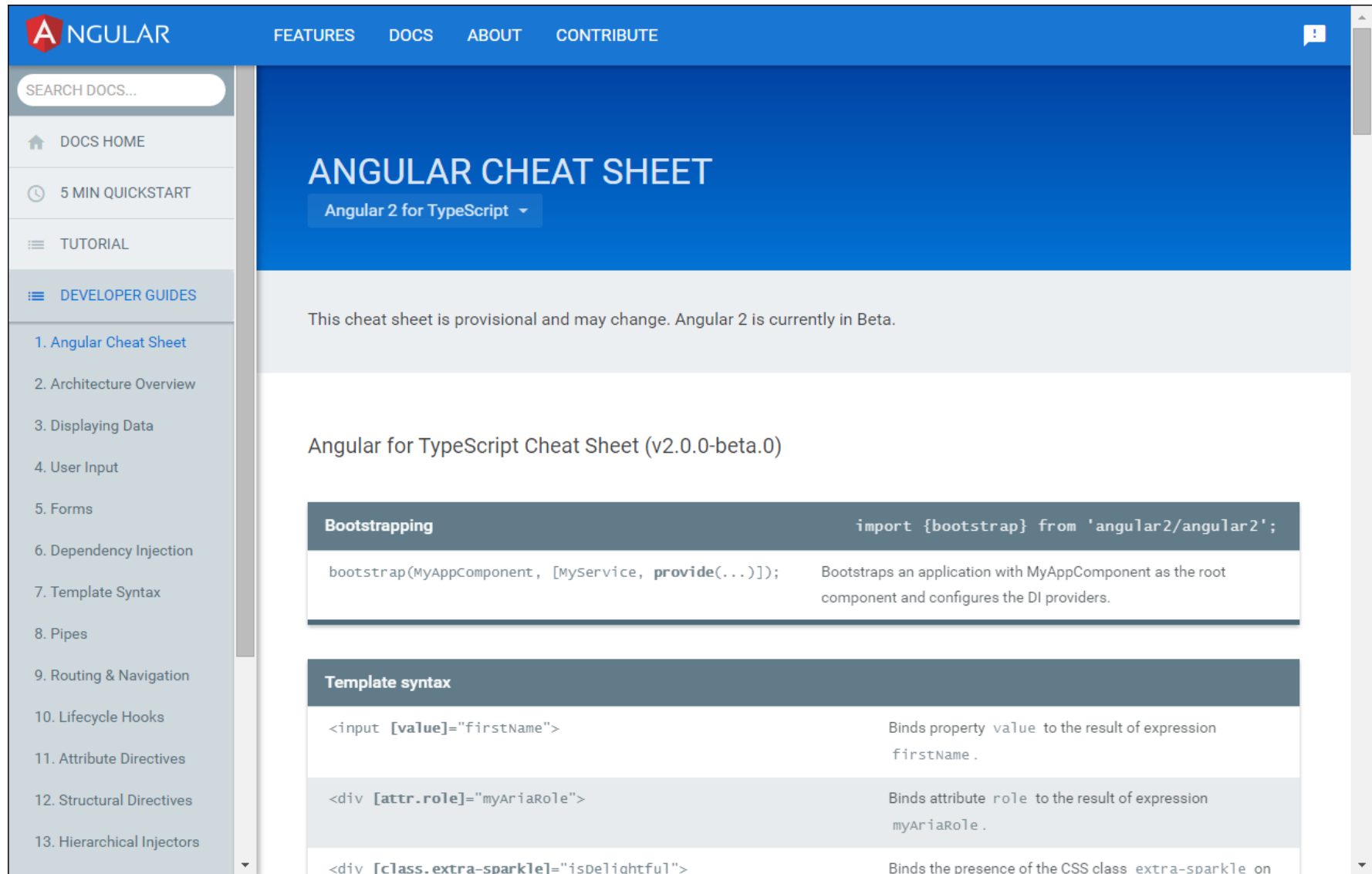
FormsModule importeren

- 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)

Binding cheat sheet



The screenshot shows the Angular website's developer guides section. The left sidebar lists various guides, with '1. Angular Cheat Sheet' selected. The main content area is titled 'ANGULAR CHEAT SHEET' and 'Angular 2 for TypeScript'. A note states: 'This cheat sheet is provisional and may change. Angular 2 is currently in Beta.' Below this, the section 'Angular for TypeScript Cheat Sheet (v2.0.0-beta.0)' is shown. It contains two tables of examples.

Bootstrapping	
<pre>import {bootstrap} from 'angular2/angular2';</pre>	
<pre>bootstrap(MyAppComponent, [MyService, provide(...)]);</pre>	Bootstraps an application with MyAppComponent as the root component and configures the DI providers.

Template syntax	
<pre><input [value]="firstName"></pre>	Binds property <code>value</code> to the result of expression <code>firstName</code> .
<pre><div [attr.role]="myAriaRole"></pre>	Binds attribute <code>role</code> to the result of expression <code>myAriaRole</code> .
<pre><div [class.extra-sparkle]="isDelightful"></pre>	Binds the presence of the CSS class <code>extra-sparkle</code> 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