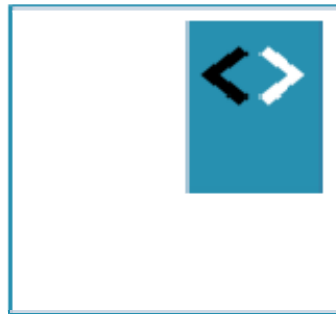


# Angular Fundamentals

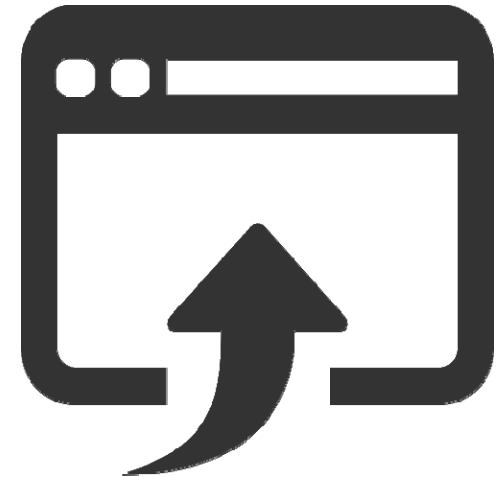
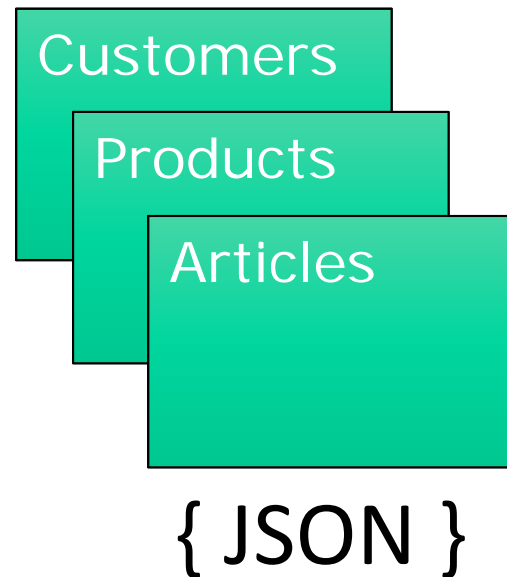
## Module 2 - Databinding



Peter Kassenaar –  
[info@kassenaar.com](mailto:info@kassenaar.com)

# What is databinding

- Show – all kinds of – data in User Interface
- Data can come from:
  - Controller / class
  - Database
  - User input
  - Andere systemen



# Declarative syntax

- New notation in HTML templates (compared to AngularJS).
  1. Simple data binding
  2. Event binding
  3. One-way data binding (Attribute binding)
  4. Two-way data binding

# 1. Simple data binding syntax

Unaltered from AngularJS. Still use double curly braces:

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

```
<div>First Name: {{ person.firstname }}</div>
```

# Always: in conjunction with component/class

```
import {Component} from '@angular/core';  
  
@Component({  
  selector: 'hello-world',  
  template: `

# Hello Angular 2</h1> <h2>My name is : {{ name }}</h2> <h2>My favorite city is : {{ city }}</h2> `, }) export class AppComponent { name = 'Peter Kassenaar'; city = 'Groningen' }


```

# Or: properties via constructor

- `export class AppComponent {`

    name: `string`;

    city: `string`;

`constructor()` {

        // `this.name` = `'...'`;

        // `this.city` = `'...'`;

    }

`ngOnInit()` {

`this.name` = `'Peter Kassenaar'`;

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

    }

}

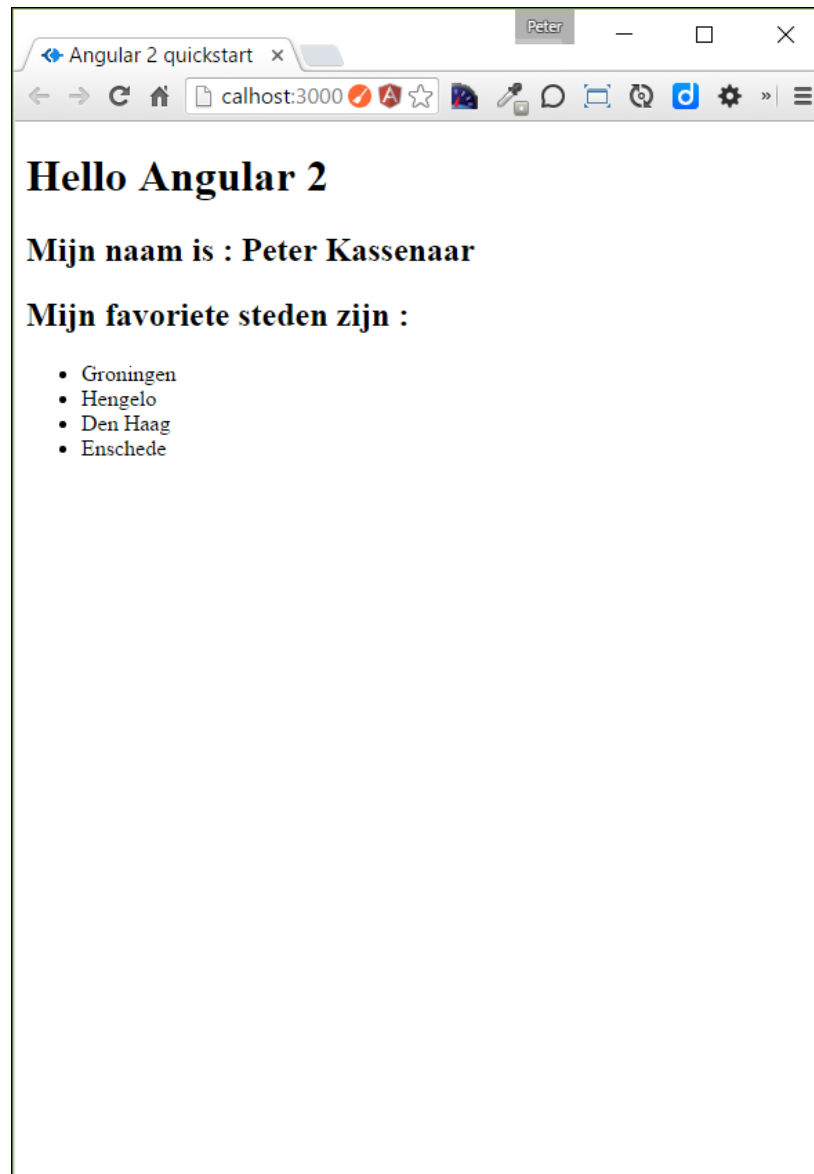
BEST PRACTICE:

use `ngOnInit()`

# Binding using a loop: \*ngFor

Template: `<h2>My favourite cities are:</h2>  
<ul>  
 <li *ngFor="let city of cities">{{ city }}</li>  
</ul>`

Class: *// Class with properties, array with cities*  
`export class AppComponent {  
 name:string;  
 cities:string[];  
  
 constructor() {  
 this.name = 'Peter Kassenaar';  
 this.cities = ['Groningen', 'Hengelo', 'Den Haag', 'Enschede']  
 }  
}`



More info:

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



# Creating a Model (as in: MVC)

A Model as a class with exported public properties:

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

Notice shorthand notation `public id : number :`

1. Defines a private/local parameter
2. Defines a public parameter with the same name
3. Initializes parameter at instantiation of the class with `new`

# Using the Model

1. Import model class

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

2. Alter component

```
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. Alter View

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

# Using `*ngIf` to show conditionally

Use the `*ngIf` directive (pay attention to the asterisk!)

```
<h2 *ngIf="cities.length > 3">There are a lot of favorite cities!</h2>
```



# External templates

If you don't like inline HTML :

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



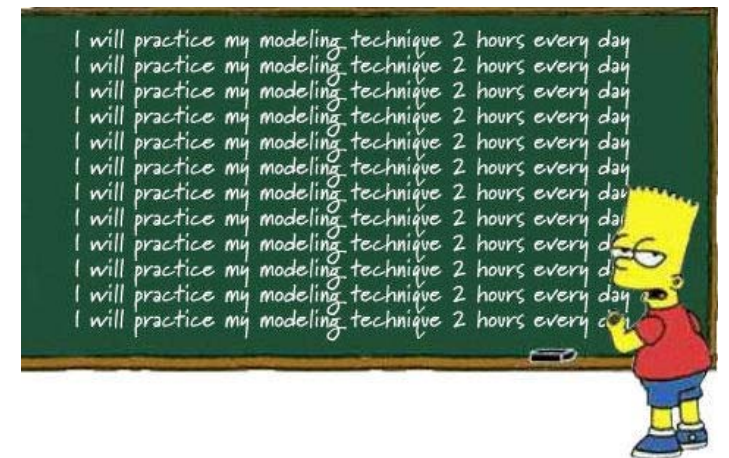
File `app.html`

```
<!-- HTML in external template -->  
<h1>Hello Angular</h1>  
<p>This is an external template</p>  
<h2>My name is : {{ name }}</h2>  
<h2>My favorite cities :</h2>  
...
```

# Checkpoint

- Simple data binding `{{ ... }}`
- Properties of the class are bound
- Loops and conditional statements with `*ngFor` and `*ngIf`
- Preferably – working with a Model
- Optional: external HTML-templates

## Exercise....





# User input and event binding

React to mouse, keyboard,  
hyperlinks and more

# Event binding syntax

Angular: use parentheses for events:

Angular 1:

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

Angular 2+:

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

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

# DOM-events

- Angular can listen to *any* DOM-event without needing different directives:

This article offers a list of events that can be sent; some are standard events defined in official specifications, while others are events used internally by specific browsers; for example, Mozilla-specific events are listed so that [add-ons](#) can use them to interact with the browser.

## Standard events

These events are defined in official Web specifications, and should be common across browsers. Each event is listed along with the interface representing the object sent to recipients of the event (so you can find information about what data is provided with each event) as well as a link to the specification or specifications that define the event.

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

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



# Example event binding

## HTML

```
<!-- Event binding on button -->  
<button class="btn btn-success"  
    (click)="btnClick()">I am a button</button>
```

```
export class AppComponent {  
    ...  
    counter: number = 0;  
  
    btnClick(){  
        alert('You clicked ' + ++this.counter + ' times');  
    }  
}
```

# Hello Angular 2

Mijn favoriete steden zijn :

- 1 - Groninger
- 2 - Hengelo
- 3 - Den Haag
- 4 - Enschede

Ik ben een button

De pagina op localhost:3000 meldt het volgende: x

Je hebt 1 keer geklikt

☐ Voorkom dat deze pagina extra dialogvensters weergeeft.

OK

# Event binding with \$event

HTML

```
<input type="text" class="input-lg" placeholder="City..."
      (keyup)="onKeyUp($event)"><br>
<p>{{ txtKeyUp }}</p>
```

*// 2. Binden aan keyUp-event in de textbox*

```
onKeyUp(event:any){
    this.txtKeyUp = event.target.value + ' - ';
}
```

# Binding with local template variable

Declare *local template variable* with # → The complete element is passed to the component

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

Class:

```
// 3. Bind to keyUp-event via local template variable
betterKeyUp(txtCity){
  //... Handle txtCity as desired
}
```

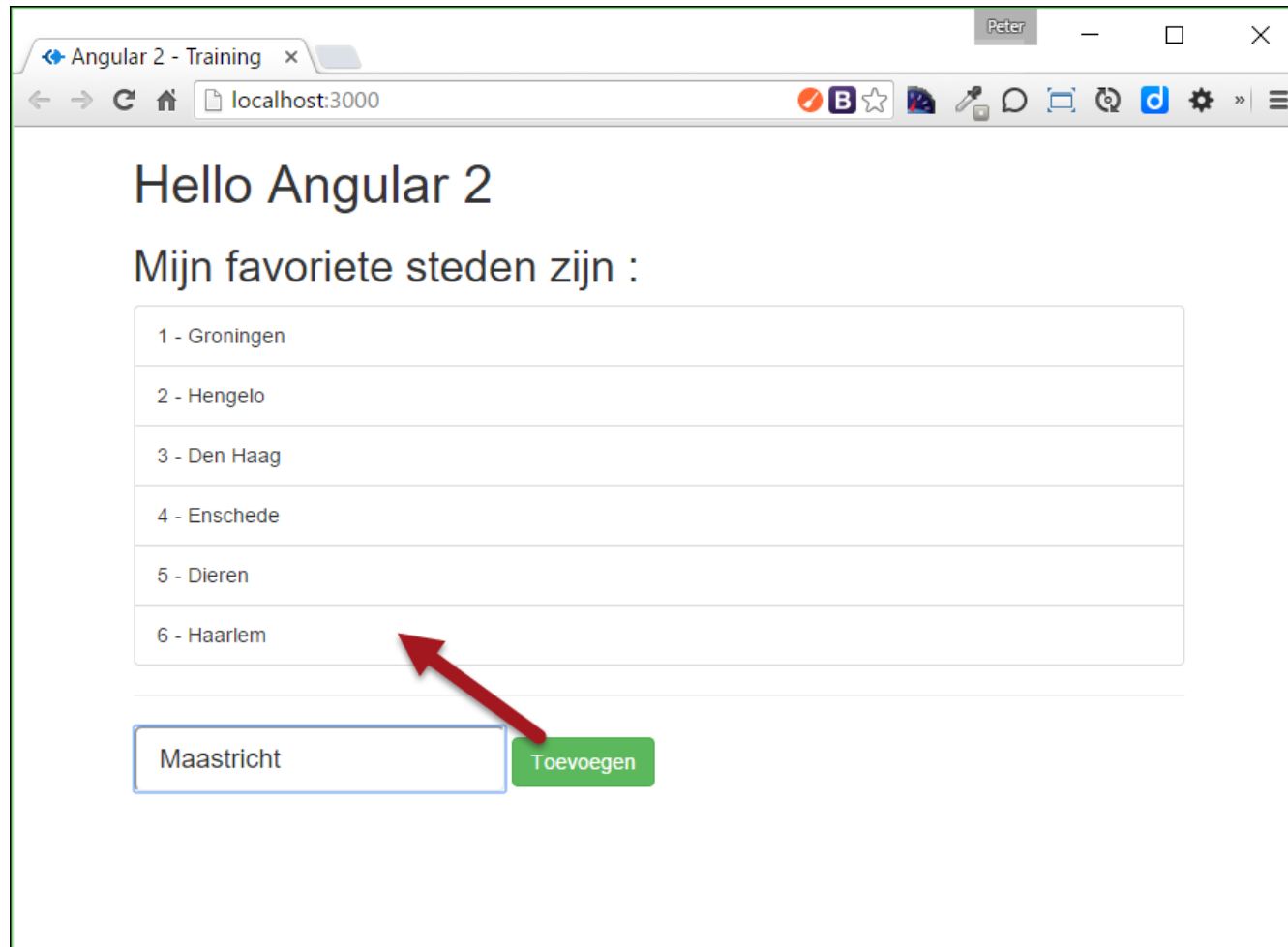
# Putting it all together...

HTML

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

Class

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

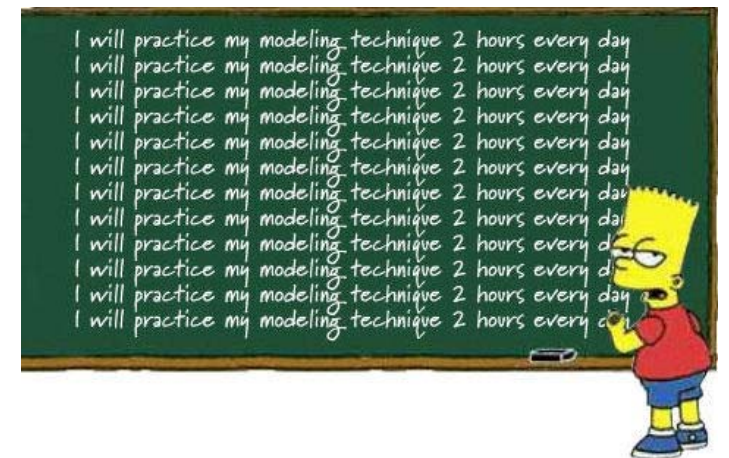


Further reading : <https://angular.io/docs/ts/latest/guide/user-input.html>

# Checkpoint

- Event binding is addressed with `( eventName ) = "..."`
- Events are being handled by a function inside the component
- Use a local template variable `#` to pass elements to the class
- You can create simple, client sided CRUD-operations this way.

## Exercise....





# Attribute & property binding

Bind values dynamically to  
HTML attributes and DOM-  
properties



# Attribute binding syntax

- Bind directly to properties of HTML-elements.
- Also know as *one-way binding*.
- Use square brackets syntax

## Angular 1:

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

## Angular 2:

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

Or :

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

```
<div [style.background-color]=" 'yellow' ">...</div>
```

# Example attribute binding

## HTML

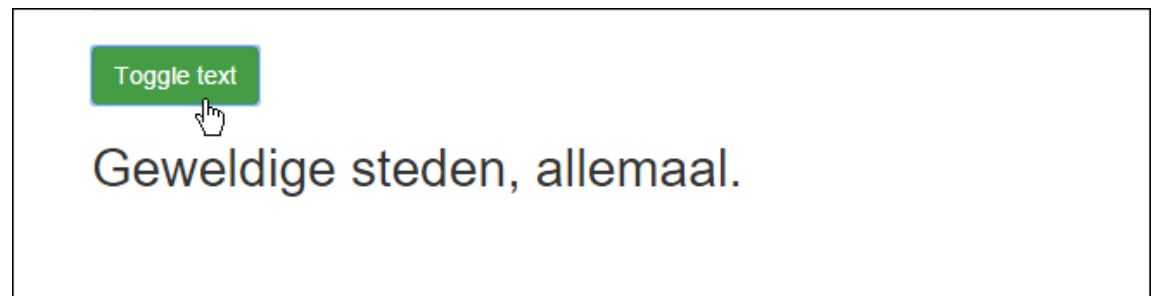
```
<!-- Attribute binding -->
```

```
<button class="btn btn-success" (click)="toggleText()">Toggle text</button>
```

```
<h2 [hidden]="textVisible">I love all these cities!</h2>
```

```
// Toggle attribute: show or hide text.
```

```
toggleText(){  
  this.textVisible = !this.textVisible;  
}
```



# For instance...

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

Class    `export class AppComponent {`  
          `// ...`  
          `currentCity:City     = null;`  
          `cityPhoto:string     = '';`  
  
          `// Update selected city in the UI. New: 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

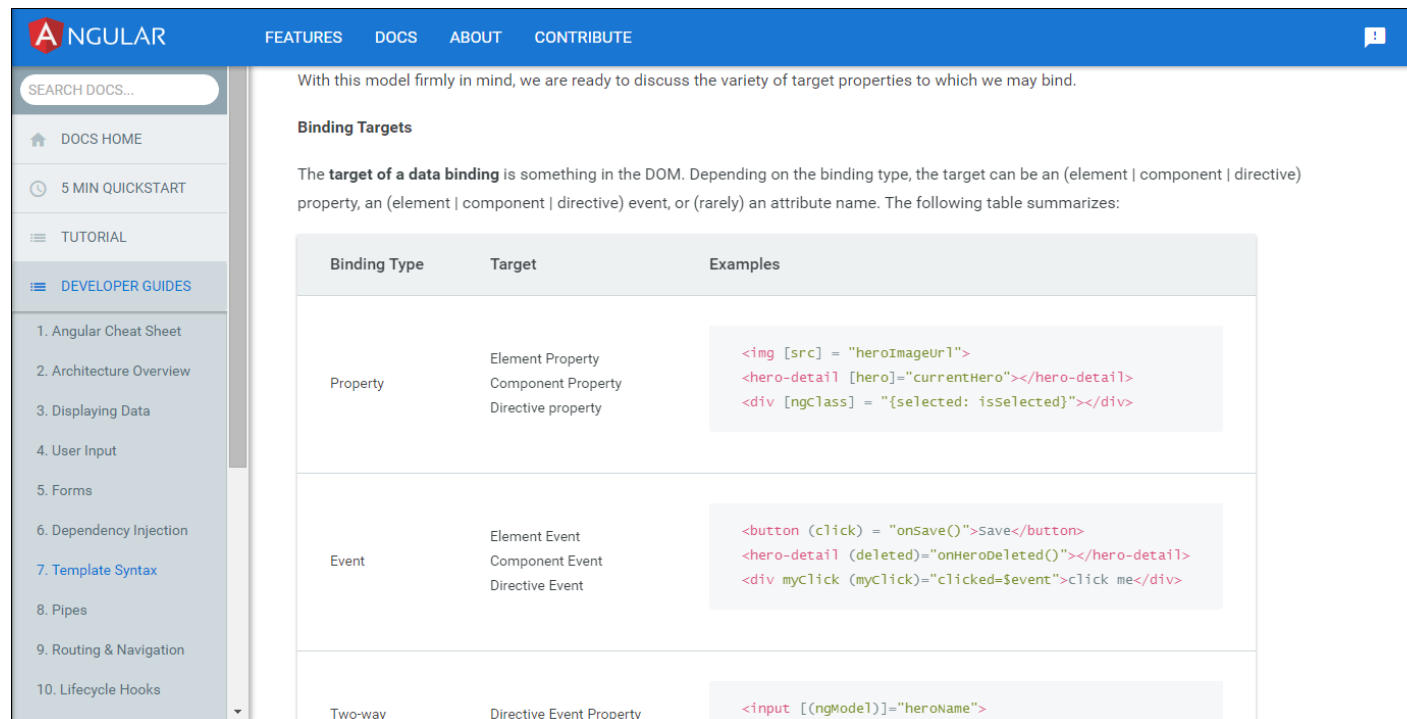


mijn stad: Groningen

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

# More binding-options

- Attribute binding and DOM-property binding: [...]
- Class binding : [ngClass]
- Style binding : [ngStyle]
- <https://angular.io/docs/ts/latest/guide/template-syntax.html>



The screenshot shows the Angular.io website's developer guides section. 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 a target 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>&lt;img [src] = "heroImageUrl"&gt; &lt;hero-detail [hero]="currentHero"&gt;&lt;/hero-detail&gt; &lt;div [ngClass] = "{selected: isSelected}"&gt;&lt;/div&gt;</pre>
Event	Element Event Component Event Directive Event	<pre>&lt;button (click) = "onSave()"&gt;Save&lt;/button&gt; &lt;hero-detail (deleted)="onHeroDeleted()"&gt;&lt;/hero-detail&gt; &lt;div myClick (myClick)="clicked=\$event"&gt;click me&lt;/div&gt;</pre>
Two-way	Directive Event Property	<pre>&lt;input [(ngModel)]="heroname"&gt;</pre>



# Two-way binding

Update user interface and  
class variables at the same  
time

# Two way binding syntax

Was removed from Angular 2 for a while, but returned after complaints from the community:

Angular 1:

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

Angular 2: similar, but notation is a little bizar:

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

# Using [(ngModel)]

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

Which is shorthand-notation for:

```
<!-- Two-way binding with extended syntax -->
```

```
<input type="text" class="input-lg"
      [value]="newCityExtended"
      (input)="newCityExtended = $event.target.value" />
<h2>{{ newCityExtended }}</h2>
```



# Import FormsModule

- Two-way binding used to be in the Angular Core – now in it's own module
- Import FormsModule in `app.module.ts`!
- `import {FormsModule} from "@angular/forms";`
- ...
- `imports : [BrowserModule, FormsModule],`

# So: passing data from View to Controller,

lots of options:

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

# Binding cheat sheet

The screenshot shows the Angular.io website's 'ANGULAR CHEAT SHEET' page for 'Angular 2 for TypeScript'. 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. A left sidebar contains a 'DEVELOPER GUIDES' section with a list of topics, where '1. Angular Cheat Sheet' is selected. The main content area has a blue header with the title 'ANGULAR CHEAT SHEET' and a dropdown menu set to 'Angular 2 for TypeScript'. Below this, a note 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 features 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).

ANGULAR

FEATURES DOCS ABOUT CONTRIBUTE

SEARCH DOCS...

DOCS HOME

5 MIN QUICKSTART

TUTORIAL

DEVELOPER GUIDES

- 1. Angular Cheat Sheet
- 2. Architecture Overview
- 3. Displaying Data
- 4. User Input
- 5. Forms
- 6. Dependency Injection
- 7. Template Syntax
- 8. Pipes
- 9. Routing & Navigation
- 10. Lifecycle Hooks
- 11. Attribute Directives
- 12. Structural Directives
- 13. Hierarchical Injectors

## ANGULAR CHEAT SHEET

Angular 2 for TypeScript

This cheat sheet is provisional and may change. Angular 2 is currently in Beta.

### Angular for TypeScript Cheat Sheet (v2.0.0-beta.0)

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

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

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

# Built-in directives

- By using the new syntax with ( ) an [ ], a lot of the old directives could be removed from the framework
- Directives that manipulate the DOM : recognized by star/asterisk
- `<div *ngFor="person of Persons">...</div>`
- `<div *ngIf="showDiv">...</div>`
- `<div [ngClass]="setClasses()">...</div>`
- `<div [ngStyle]="setStyles()">...</div>`

# Checkpoint

- Databinding in Angular 2 is new
- Learn the new syntax on DOM- and Attribute binding. Also learn event binding en two-way binding.
- Always edit the class and corresponding View
- A lot of concepts are the same, the way to achieve results are completely new in Angular 2, compared to Angular 1.