Global Knowledge®

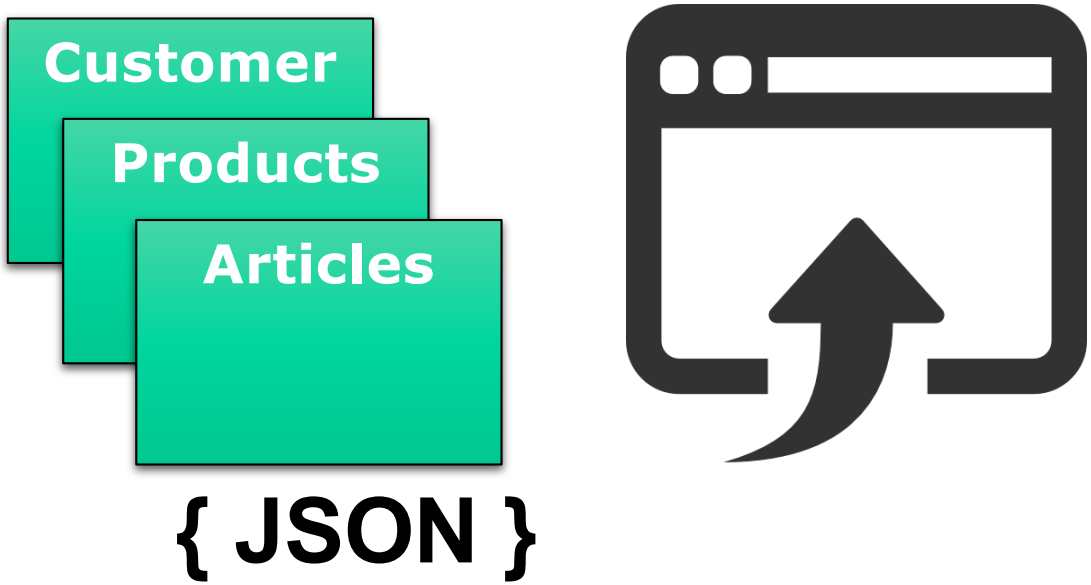
Atos

Angular
Module 2 – Data binding

Peter Kassenaar –
info@kassenaar.com

What is databinding

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



Declarative syntax

- Four (4) kinds of databinding
- Angular specific notation in HTML templates
    1. Simple data binding
    2. Event binding
    3. One-way data binding (Attribute binding)
    4. Two-way data binding

1. Simple data binding syntaxis

Unaltered from AngularJS and other frameworks.

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: `<h1>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()`

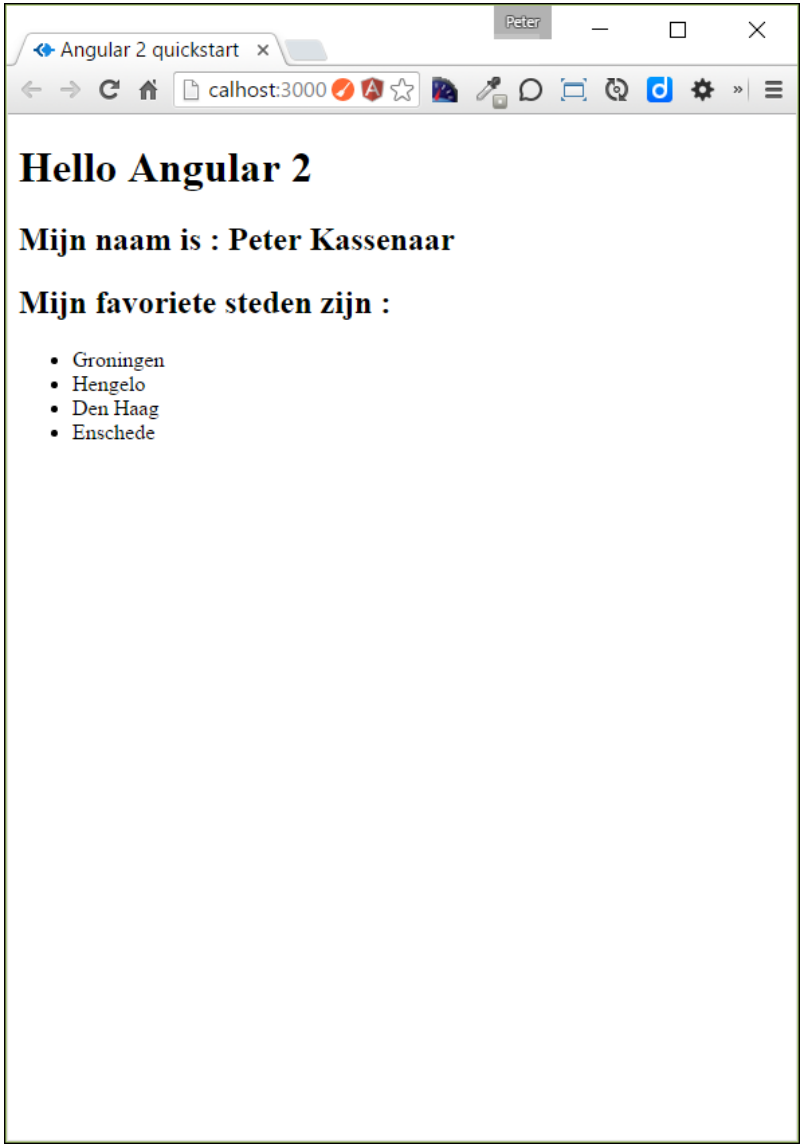
Binding using a loop: `*ngFor`

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:

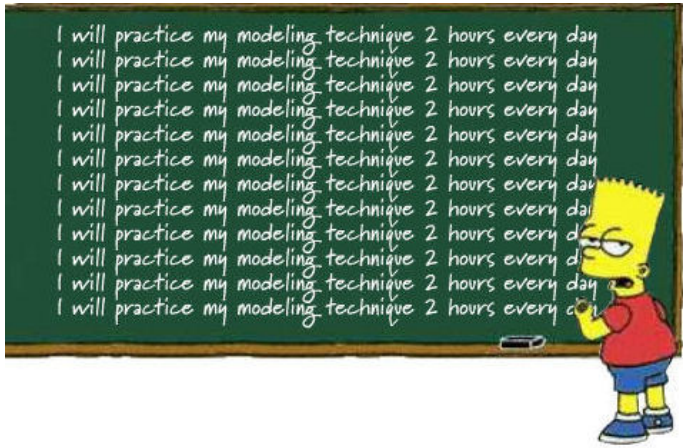
https://angular.io/docs/ts/latest/guide/displaying-data.html

Workshop

- Simple data binding `{{ … }}`
- Properties of the class are bound
- Create some class properties

- Bind them to the template
- Use an array of data to bind to the template
- Exercise 2a, 2b.

Exercise….



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'
})
```

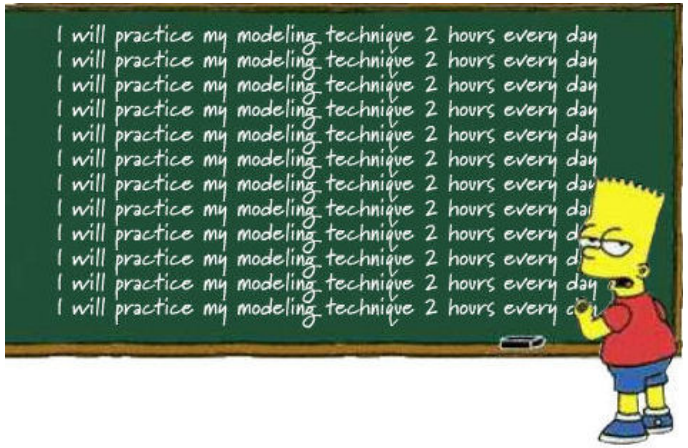
File `app.html`

```
<!-- 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`
- Preferrably – working with a Model
- Optional: external HTML-templates
- Exercise: `2a), 2b), 2c)`

Exercise….



User input and event binding

React to mouse, keyboard, hyperlinks and more

Event binding syntax

Angular: use parentheses for events:

Angular 1:

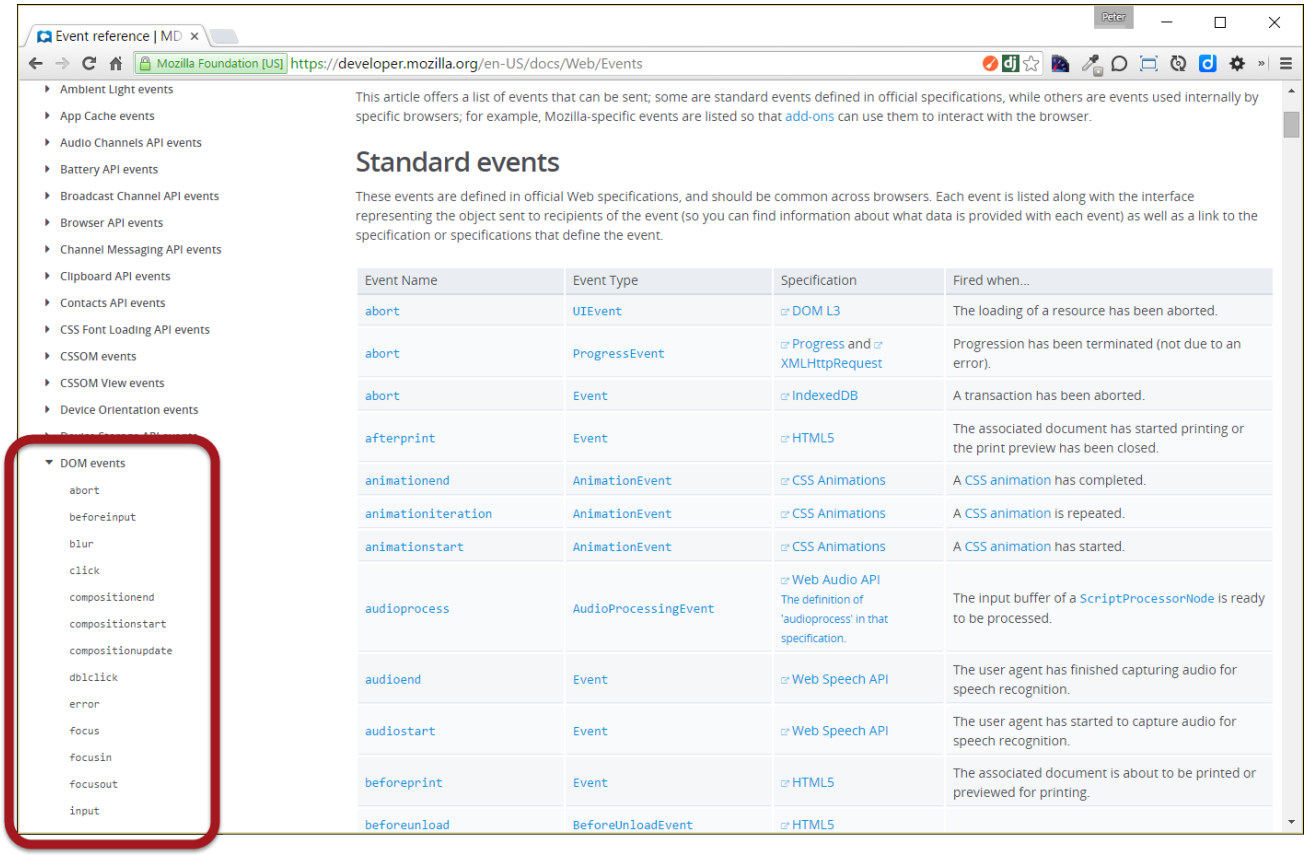
```
<div ng-click="handleClick()">…</div>
```

Angular 2+:

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

<input (blur)="onBlur()">…</div>
```

DOM-events

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



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

3. Binding to non-DOM events

- Binding to non-DOM events: use `@HostListener()`
- Listen to events on the `window` object, decorate an upcoming function
- Passing `$event` is optional
- For instance:

```
// 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 synchronisen';
    console.log('We zijn online!');
}
```

Example event binding

HTML

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

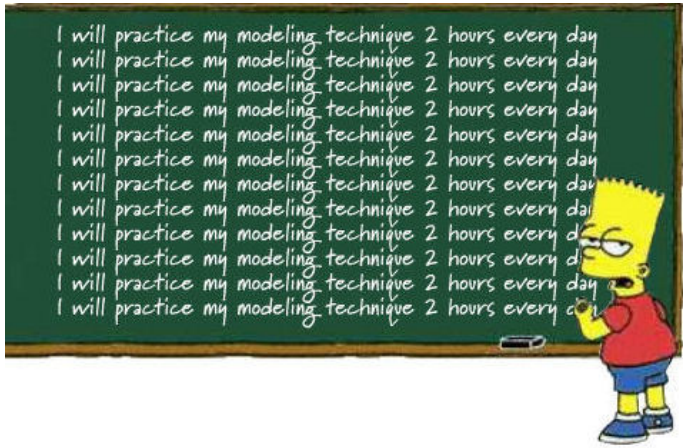
```
export class AppComponent {
   …
   counter: number =0;

   btnClick(){
      alert(‘You clicked '+ ++this.counter +’ times');
   }
 }
```


Hello Angular 2
Mijn favoriete steden zijn :
1 - Groninger
2 - Hengelo
3 - Den Haag
4 - Enschede
De pagina op localhost:3000 meldt het volgende:
Je hebt 1 keer geklikt
Voorkom dat deze pagina extra dialoogvensters weergeeft.
OK
Ik ben een button

Checkpoint

- Event Binding `( … )`
- Bind raw DOM event like `(click)`, `(blur)`, etc. NOT `onclick`, `onblur`, etc.
- Exercise: `3a)`

Exercise….



Event binding with $event

HTML

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

```
// 2. Bind to keyUp-event in the 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
}
```

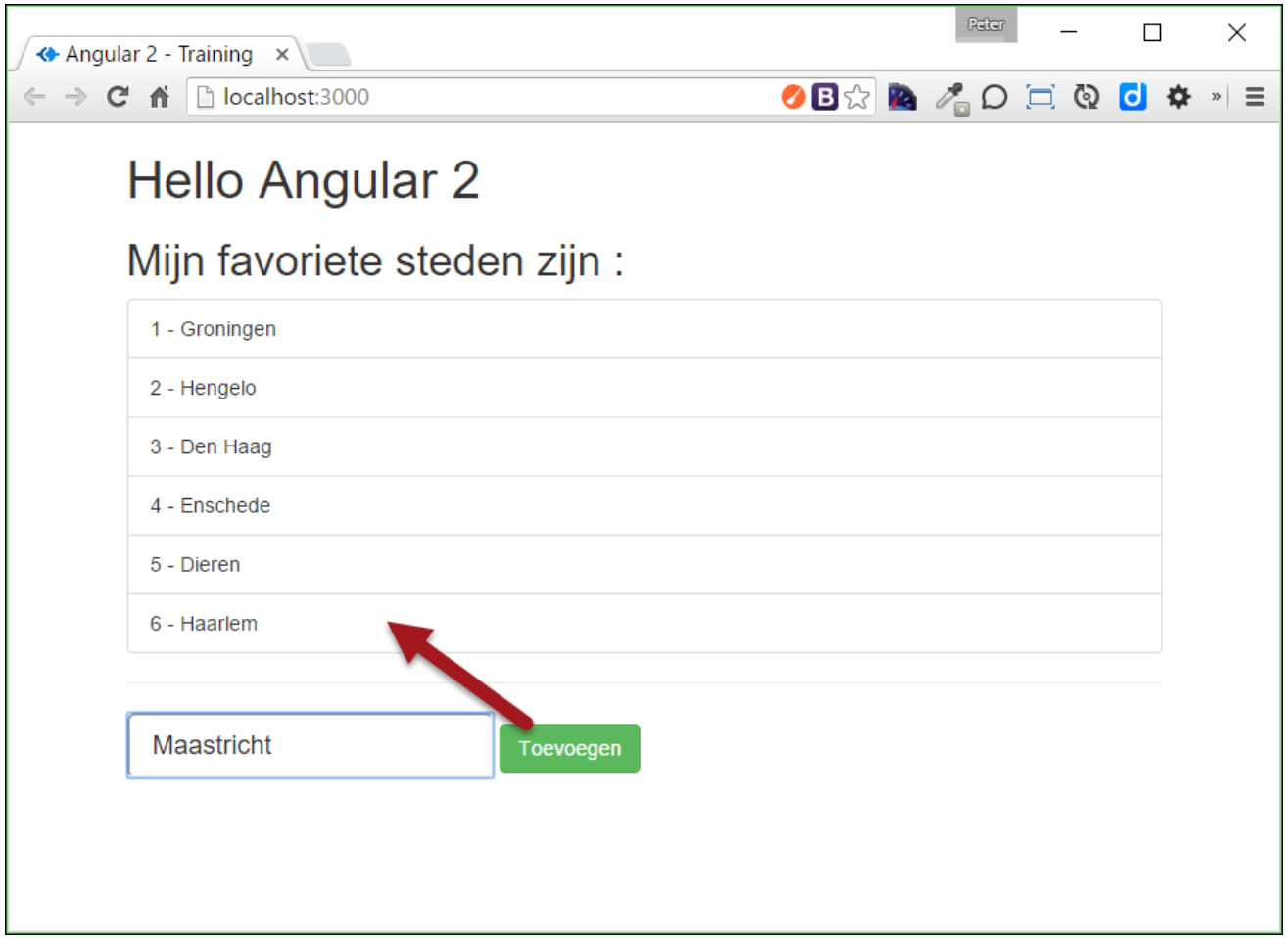
Putting it all together…

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 = '';
   }
}
```

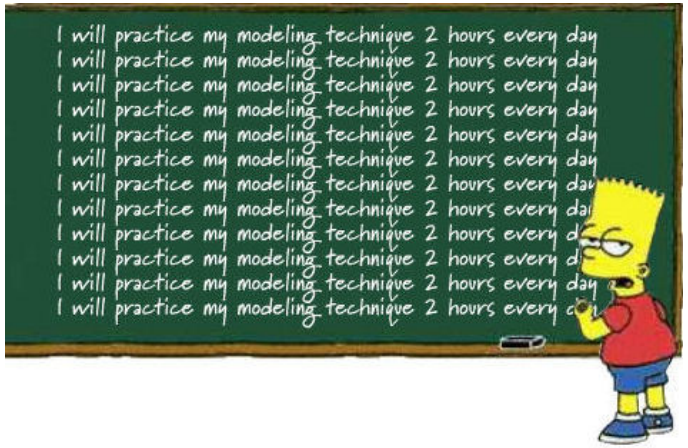


Further reading : https://angular.io/docs/ts/latest/guide/user-input.html

Checkpoint

- Event binding is addressed with `(eventName)="…"`
- Events are being handled by a function inside the component
- Options: use `$event` to pass data to the class
- Or: use a local template variable `#` to pass value to the class
- You can create simple, client sided CRUD-operations this way.
- Exercise: `3d)` and `3e)`

Exercise….



Attribute & property binding

Bind values dynamically to HTML attributes and DOM-properties

Attribute binding syntaxis

- 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 :

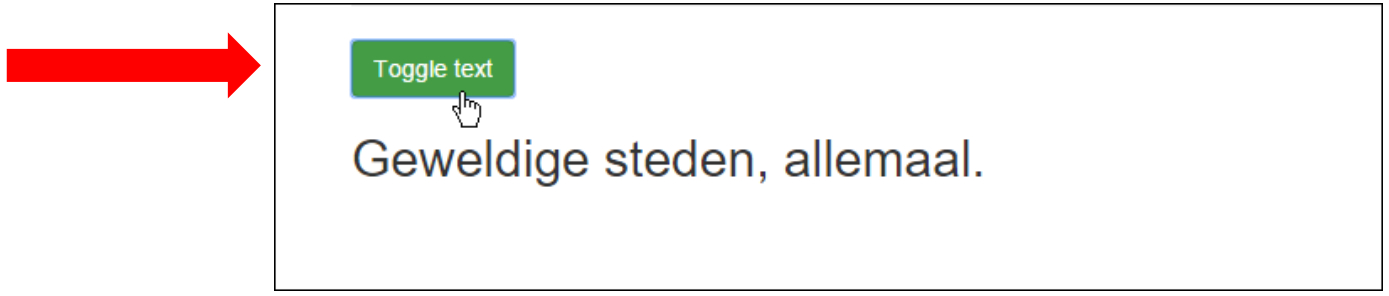
```
<div [hidden]="person.hasEmail">…</div>
<div [style.background-color]="'yellow'">…</div>
```

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:

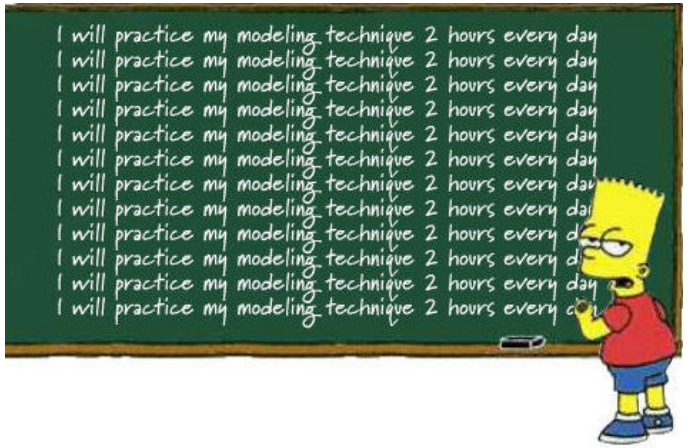
`..\103-attributebinding\src\app\app.component.ts`

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

Checkpoint

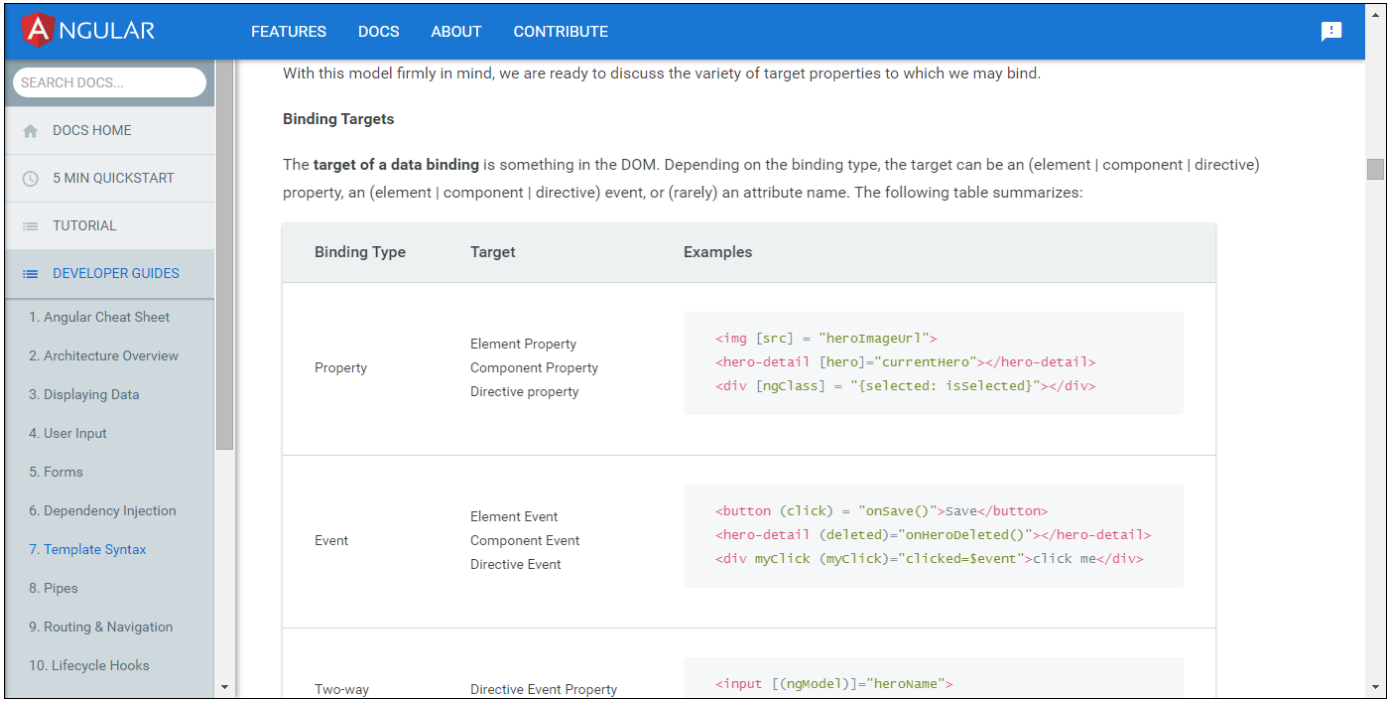
- Attribute binding is addressed with `[attrName]="…"`
- Attributes are bound to a variable on the class.
- You can calculate the variable in the `.ts`-file
- Exercise: `4a)` and `4b)`

Exercise….



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



Two-way binding

Update user interface and class variables at the same time

Two way binding syntaxis

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)]

```
<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,

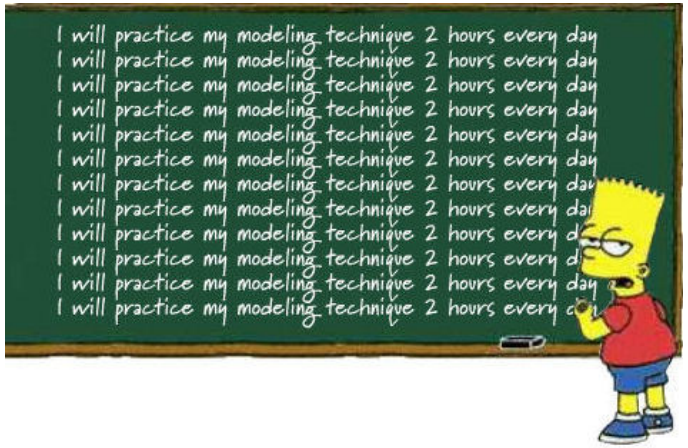
lots of options:

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

Checkpoint

- Two-way binding is addressed with `[(ngModel)]="…"`
- The value of ngModel is updated automagically by Angular.
- It is available in the View/Template and in the TypeScript class.
- Exercise: `4d)`

Exercise….



Binding cheat sheet

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

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`.
`<div [class.extra-sparkle]="isDelightful">`	Binds the presence of the CSS class `extra-sparkle` on

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

Checkpoint

- Databinding in Angular 2 is new
- Learn the new syntax on DOM- and Attribute binding. Also learn event binding en two-way binding.
- Optional: host binding with `@HostListener()`
- 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.