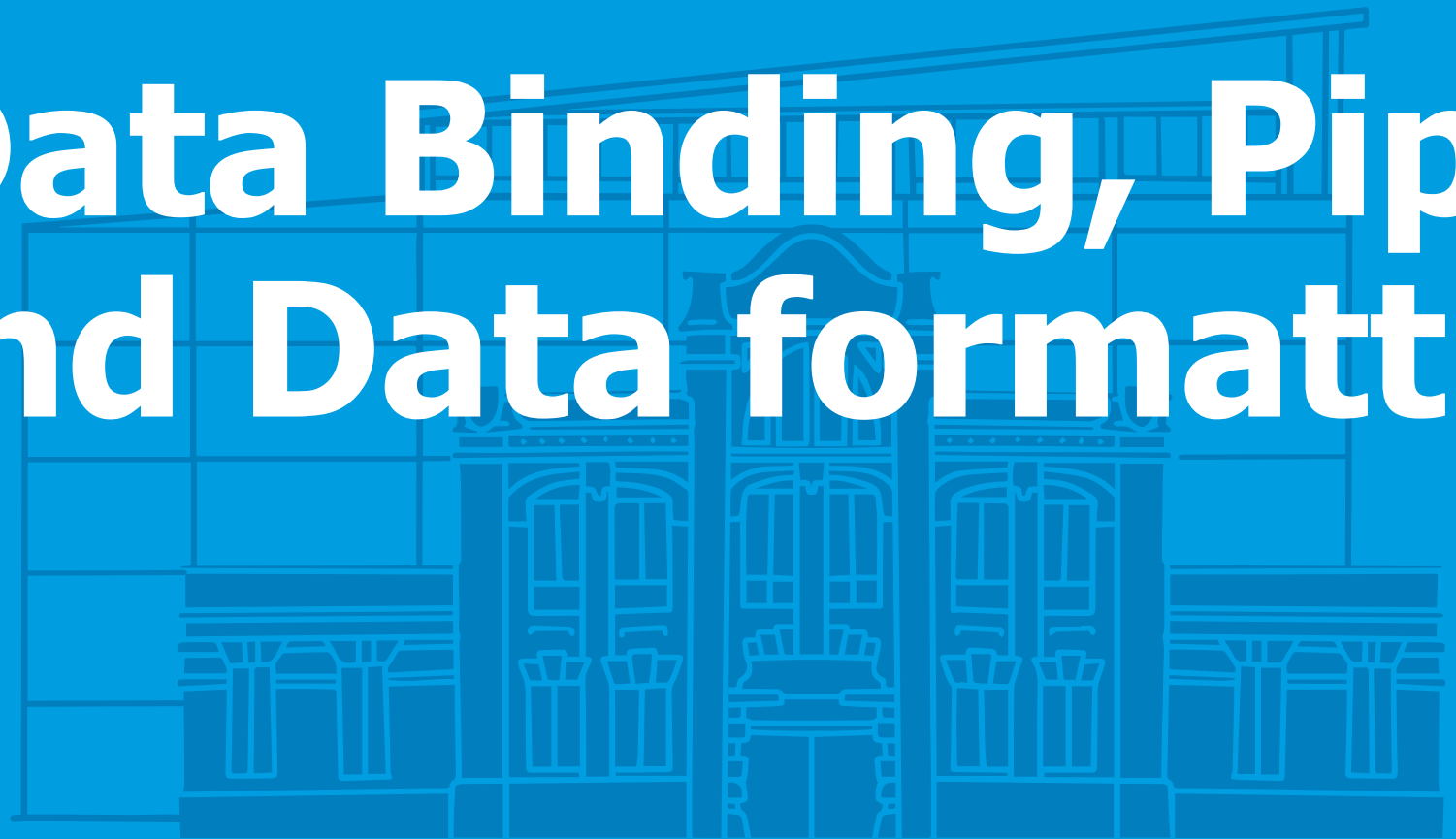
netmind
a bts company

Angular 14 - 07

Data Binding, Pipes and Data formatting



Data binding

Angular provides three categories of data binding according to the direction of data flow:

TYPE	SYNTAX	CATEGORY
Interpolation Property Attribute Class Style	`{{expression}}` `[target]="expression"`	One-way from data source to view target
Event	`(target)="statement"`	One-way from view target to data source
Two-way	`[(target)]="expression"`	Two-way

Data binding

- On the one hand, the possibility of using the mustache syntax as before (interpolation mechanism) is maintained, and on the other hand, a databinding can be expressed by putting the property to bind in parentheses and the destination data in quotes .
- For example, for an image we can continue using interpolation:

```
< img src ="{{ product.imageUrl }}" style ="width:50px;"/>
```

- But we also have an **data binding** alternative, in which the property to bind looks like this:

```
< img [src] ="product.imageUrl" style ="width:50px;" />
```

  - https://angular.io/guide/binding-syntax

- **Databinding is preferred** , but if we have to use a compound expression, interpolation may be an option:

```
< img [src]="'./assets/imgs/'+product.image" style ="width:50px;"/>

< img src ="./assets/imgs/{{ product.imageUrl }}" style ="width:50px;"/>
```

Binding of class properties

```
...

  img_width:number = 100;
  img_height:number = 100;

  products: any[] | null = [
    {
      name: 'Leaf Rake',
      code: 'GUN-0611',
      image: 'LeafRake.png',
      date: 'March 19, 2916',
      price: 19.95,
      stars: 3.2,
    },
    {
      name: 'Garden Cart',
      code: 'GUN-0023',
      image: 'GardenCart.png',
      date: 'March 21, 2916',
      price: 32.99,
      stars: 4.2,
    },
  ];
...
```

```
...
<tr *ngFor="let product of products">
    <td>{{product.name}}</td>
    <td>{{product.code}}</td>
    <td><img
        [src]="'./assets/imgs/'+product.image"
        [title]="product.name"
        [style.maxWidth.px]="img_width"
        [style.maxHeight.px]="img_height"
    /></td>
    <td>{{product.date}}</td>
    <td>{{product.price}}</td>
    <td>{{product.stars}}</td>
</tr>
...
```

Events binding

- In the events part, the syntax is similar, but the corresponding event is enclosed in parentheses (event) .
    - https://angular.io/guide/event-binding-concepts

```
<button (click)="showImages()">Show images</button>
```

- Next, we'll create a property and method on the component class:

```
  show_imgs: boolean = false;
  showImages = ():void => {
    this.show_imgs = true;
  };
```

- Finally we will establish that the image is visible only conditionally:

```
<img *ngIf="show_imgs"
[src]="'./assets/imgs/'+product.image"
[title]="product.name"
[style.maxWidth.px]="img_width"
[style.maxHeight.px]="img_height" />
```

Interpolation expressions

- We still have to make the button text also change when we click on it. This is simple: we change the button text to a tween:

```
<button (click)="showImages()">{{show_imgs?'Hide images':'Show images'}}</button>
```

```
  show_imgs: boolean = false;
  showImages = ():void => {
    this.show_imgs = !this.show_imgs;
  };
```

- **Remember** that interpolation not only reads data, but also evaluates expressions!

Angular events

📎 Full list of Angular Events

```
(click)="myFunction()"
(dblclick)="myFunction()"
(submit)="myFunction()"
(blur)="myFunction()"
(focus)="myFunction()"
(scroll)="myFunction()"
(cut)="myFunction()"
(copy)="myFunction()"
(paste)="myFunction()"
(keyup)="myFunction()"
(keypress)="myFunction()"
(keydown)="myFunction()"
(mouseup)="myFunction()"
(mousedown)="myFunction()"
(mouseenter)="myFunction()"
(drag)="myFunction()"
(drop)="myFunction()"
(dragover)="myFunction()“
```

The target event determines the shape of the **$event object**. If the target event is a native DOM element event, then $event is a DOM event object, with properties such as target and target.value.

```
<input [value]="currentItem.name“ (input)="currentItem.name=getValue($event)">
```

Two -way binding

- **ngModel** directive uses the **banana in a box [()]** syntax .
  - https://angular.io/api/forms/NgModel

```
<input [(ngModel)]="filter_text" />
```

```
@Component()
export class ProductsListComponent {
     ...
     filter_text: string = ‘’;
     ...
}
```

- Outer brackets indicate property binding (whichever is indicated after the equals sign), while inner brackets indicate event (any input that changes the data it points to).

FormsModule import

- Now, if we want this type of functionality in input elements, we have to modify our main module, to import this functionality from the "**FormsModule**" module.
- This module contains that and other features of the input mechanisms and their passage to the data model.
- Therefore, we must return to the death of our module, and modify it as follows:

```
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms’;
...

@NgModule({
  declarations: [...
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Association with a view model

- Now, we can change our filter mechanism, which will contain this syntax in the template:

```
<input [(ngModel)]="filter_text" placeholder="input a text to filter"/>
<h3>Products filtered by: {{filter_text}}</h3>
```

- And we define the property in our class:

```
@Component()
export class ProductsListComponent {
    ...
    filter_text: string = '';
}
```

Pipes “|” (output filters)

- Output pipes or filters allow you to modify an input and generate a decorated output.
  - For example, formatting dates or currencies
  - https://angular.io/guide/pipes

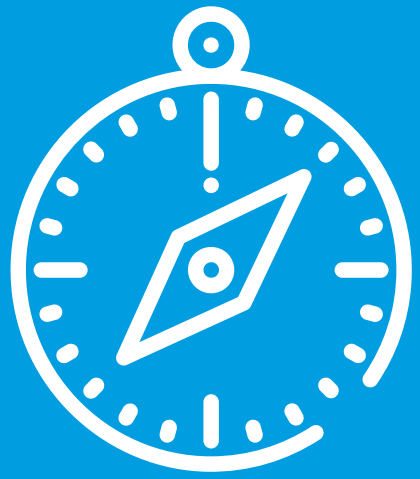
- They are written after an expression and modify the way it is interpreted in the DOM , without modifying the input data at any time.

```
<tr *ngFor="let product of products">
    <td>{{product.name}}</td>
    <td>{{product.code | lowercase}}</td>
    <td>
        <img *ngIf="show_imgs"
         [src]="'./assets/imgs/'+product.image" [title]="product.name | uppercase"
         [style.maxWidth.px]="img_width" [style.maxHeight.px]="img_height" />
    </td>
    <td>{{product.date}}</td>
    <td>{{product.price | currency:'USD':true:'1.2-2'}}</td>
    <td>{{product.stars}}</td>
</tr>
```

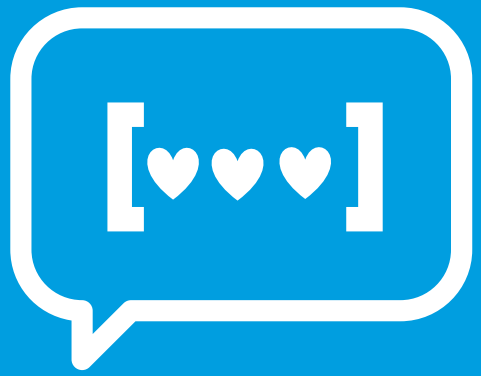
Let's put it into practice: Tasks/Projects App

In the tasks component:

- Add a button that allows you to delete a specific task.
- Add an input that allows showing the tasks whose title matches the text of the field



Next steps



We would like to know your opinion!

Please, let us know what you think about the content.
From Netmind we want to say thank you, we appreciate time and effort you have taking in answering all of that is important in order to improve our training plans so that you will always be satisfied with having chosen us
quality@netmind.es

netmind
a bts company

Thanks!

Follow us: