The Learning Hub

Angular TS

Education and Training Solutions 2023

1. Data Binding
2. Directives

Data Binding

Overview of Data Binding

A data binding technique keeps data synchronized between components and views. Each time the user updates the data in the view, Angular updates the component. Once Angular gets new data for the component, it updates the view.

Overview of Data Binding

There are many uses of data binding. You can show models to the user and change element styles dynamically as well as respond to user events.

There are two basic types of data binding in Angular:

1. One-way binding.
2. Two-way binding.

One-way data-binding

One-way binding allows for data to flow in one direction only. Either from the view to the component or from the component to the view.

Using interpolation and property binding, you can bind data from components to views.

Interpolation Binding

Using interpolation, we can include expressions as part of any string literal in our HTML. The angular evaluates the expression into a string and replaces it with the original string and then updates the view.

Anywhere in the view that uses a string literal, interpolation can be used.
Angular uses double curly braces ({{ }}) in the template to indicate interpolation.

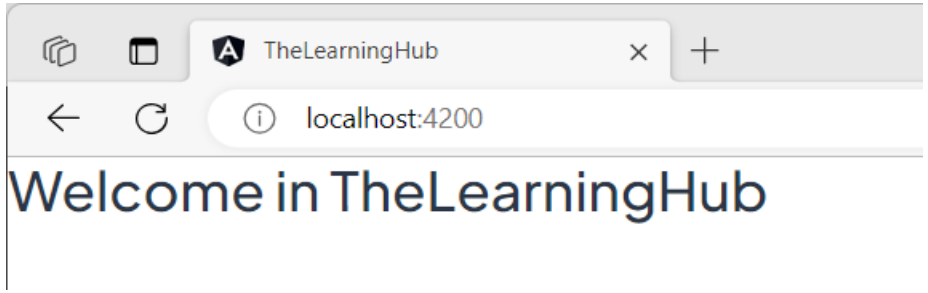
Interpolation Binding Example

In app.component.html

```
<h2>Welcome in {{title}}</h2>
```

Interpolation Binding Example

The Result:



Interpolation Binding

When you launch the app, you will see TheLearningHub in the output.

Angular replaces {{title}} with the title values of the component.

Also, whenever the values of the title change, Angular updates the view. However, not the other way around.

Property Binding

The binding property of an HTML element's property to a property in the component is called property binding.

Angular updates the element's property when the component's value changes.

Property Binding

property binding allows you to set properties such as class, href, src, textContent, etc.

It can also be used to set properties of components or custom directives (properties decorated with Input).

Property Binding Syntax

[binding-target]=”binding-source”

Within the square brackets [] there is the property binding target (or property target). The property name should match the property name of the enclosing element.

Property Binding Syntax

The binding source is enclosed in quotation marks and assigned to the binding target.

The Binding source must be a template expression.

The expression can be a property in the component, a method in the component, a template reference variable, or a combination of all three.

Property Binding Example

In app.component.html.

```
<input type ="text" placeholder="your name" [value]="name" />

<img [src]="imagepath" alt="imagehere">
```

Property Binding Example

The name and image path must be defined in the typescript file.

→ In app.component.ts:

```
name:string ='Dana Kanaan';
imagepath:string='https://miro.medium.com/max/
512/1*FKD2Uy_Q6r6AviZA2VD4RQ.png’;
```

Property Binding Example

→ In app.component.html

```
<input type="text" placeholder="your name" [value]="name“
(change)="handleNameInputChange()" />
```

→ In app.component.ts

```
handleNameInputChange() {
  alert('The value is changed!');  }
```

Handling Events

The most common way to handle events is to pass the event object, $event, to the method that handles it.

A $event object contains information needed by the method, such as the name of the user, or the URL of an image.

Handling Events

Target events determine the shape of $event objects. When the target event is a native DOM element event, $event is a DOM event object that has properties such as target and target. value.

Handling Events Example

→ In app.componemts.html

```
<input type="text" placeholder="your name“
[value]="name"(change)="handleNameInputChange($event)  "/>
```

→ In app.componemts.ts

```
handleNameInputChange = (inputevent:any) =>
{console.log(inputevent.target.value);
 this.name = inputevent.target.value; }
```

Two-way data-binding

By using two-way binding, any changes we make to our model in the component are propagated to the view, and updates to the view are immediately propagated to the underlying component.

App

import
FormsModule

Forms Module

declarations
[ngModel]
export [ngModel]

Two-way data-binding Syntax

The syntax for tow-way data-binding :

<Element [(ngModel)]="property"></Element>.

ngModel

For Angular, the ngModel directive is used to achieve the two-way binding on HTML Form elements. ngModel binds to form elements like input, select, and select an area.

The ngModel directive is not part of the Angular Core library; instead, it is part of @angular/forms.

The FormsModule package should be imported into your Angular module.

How to use ngModel

→ In app.module.ts

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

→ In the import section.

```
imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule],
```

Two-way data-binding Example

Creates a simple form using two-way data binding which contains:

- Name
- Email
- Salary
- And then Calculate the annual salary.

Example:

→ In app.component.ts

```
export class AppComponent {
title = ‘TheLearningHub';
name: string = '';
email: string = '';
salary: number = 0;
}
```

Example:

→ In app.component.html

```
<input type="text" placeholder="your name“
 [(ngModel)]="name" />

<input type="text" placeholder="your email"
[(ngModel)]="email" />

<input type="number"  placeholder="your Salary"
[(ngModel)]="salary" />
```

Example:

→ And this code reads the value from the typescript file.

```
<h1>Current name is : {{name}}</h1>
<h1>Current email is : {{email}}</h1>
<h1>Current salary is : {{salary}}</h1>
<h1>Current annual salary is : {{salary*12}}</h1>
```

Example:

→ In app.component.css

```
input {
        display:  block;
        width:  300px;
        padding:  10px;
        font-size:  1em;
        margin-top:  10px;
    }
```

Example:

→ In app.component.html

```
<input type="text"placeholder="your name" [(ngModel)]="name"
(ngModelChange)="handlechange($event)"/>
```

Exercise

Add to the form a button called clear to clear all data

on the HTML page.

Directives

Overview of Directives

Angular directives allow us to manipulate the DOM.

Directives can be used to change the appearance, behaviour, or layout of DOM elements.

The directives in Angular are divided into three types :

1. Component Directive.
2. Structural directives.
3. Attribute directives.

Component Directives

Component directives are used for specifying the template/HTML for the Dom Layout.

Component directives are simple classes decorated with the @component decorator.

Structural Directives

A structural directive can change the layout of the DOM by adding or removing elements.

There are three common structural directives:

1. ngFor
2. ngIf
3. ngSwitch

ngFor Structural Directives

ngFor is an Angular directive that repeats a portion of the HTML template once per iteration of an IEnumerableList (collection).

Syntax:

```
ngFor="let obj of collection"
```

ngFor Example:

→ In app.component.ts

```
customers:any=[{
    customerNo:1,
    name:'Ahmad',
    address:'Jordan',
    city:'Irbid',    },
  {  customerNo:2,
    name:'Amal',
    address:'Jordan',
    city:'Amman'} ,]
```

ngFor Example:

→ In app.component.html

```
<tr *ngFor="let customer of customers;">
        <td>{{customer.customerNo}}</td>
        <td>{{customer.name}}</td>
        <td>{{customer.address}}</td>
        <td>{{customer.city}}</td>
        <td>{{customer.state}}</td>
 </tr>
```

ngIf

By using the ngIf Directive, HTML elements can be added or removed based on an expression.

Boolean values must be returned from the expression.

The element is removed if the expression is false, otherwise, the element is inserted.

ngIf Example

In app.component.html

```
<div *ngIf="toggle">
        This is shown if condition is true
 </div>
```

In app.component.ts

```
toggle:boolean=true;
```

ngSwitch

By using the ngSwitch directive, you can add or remove HTML elements based on match expressions.

ngSwitch directive used with ngSwitchCase and ngSwitchDefault.

Attribute Directives

By using a style directive or attribute, we can change how an element appears or behaves.

There are three common Attribute directives

1. ngModel
2. ngClass
3. ngStyle

Attribute Directives

1. ngModel
In order to achieve the two-way data binding, the ngModel directive is used.

2. ngClass
CSS classes are added or removed from HTML elements using the ngClass property.

ngClass Example

In app.component.css

```
.red { color: red;
background-color: gray; }
.size20 { font-size: 20px; }
```

ngClass Example

In app.component.html

```
<div [ngClass]="'red size20'"> Red Text with Size 20px </div>
```

The Result

Red Text with Size 20px

Exercise

Find out what the NgSwitch directive does,

and in what situations it might be useful.

References

[1] Angular, “Angular,” *Angular.io*, 2019. https://angular.io/

[2] “Complete Angular Tutorial For Beginners,” *TekTutorialsHub*. https://www.tektutorialshub.com/angular-tutorial/

[3]“npm | build amazing things,” *Npmjs.com*, 2019. https://www.npmjs.com/

[4]“Angular Tutorial for Beginners | Simplilearn,” *Simplilearn.com*. https://www.simplilearn.com/tutorials/angular-tutorial (accessed Aug. 19, 2022).

The Learning Hub

Thank You

Break

9:00 - 10:00