Data-Binding and Event-Handling Lab - Part 1

**Goals**

* Generate new Angular project/components using the Angular CLI tool
* Apply one-way data binding
* Apply two-way data binding

1. Set Up the Project

1.1. Create a New Angular Project

1. Move into the training directory:

cd angular-training

1. Create a new project:

ng new my-data-binding-demo

* + This command may take several minutes to complete.

1.2. Add the Angular Project

1. Start the **Visual Studio Code**  editor.

2. Update Your App

1. Open **src/app/app.component.ts**.
2. Update the **title** field to this:

title = 'Data Binding Demo!';

1. Save the file.

3. Test Your App

1. Move back to the **angular-training** terminal window.
2. Move into the project directory:

cd my-data-binding-demo

1. Run the app:

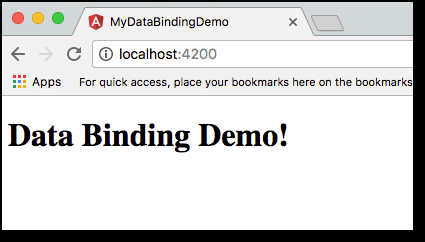
ng serve

1. Wait until you see the following text:
2. ng serve
3. \*\* NG Live Development Server is running on http:*//localhost:4200 \*\**
4. Hash: 719df4eb209182a9a4fc
5. Time: 10090ms
6. chunk {0} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 157 kB {4} [initial] [rendered]
7. chunk {1} main.bundle.js, main.bundle.js.map (main) 4.05 kB {3} [initial] [rendered]
8. chunk {2} styles.bundle.js, styles.bundle.js.map (styles) 9.77 kB {4} [initial] [rendered]
9. chunk {3} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.69 MB [initial] [rendered]
10. chunk {4} inline.bundle.js, inline.bundle.js.map (inline) 0 bytes [entry] [rendered]

webpack: Compiled successfully.

* + Keep this terminal window running. It contains the server that is listening on port 4200.

1. Click **http://localhost:4200** and in your browser, expect to see a screen similar to this:



4. Generate a New Component

In this exercise you add a new component. Instead of manually creating the files, you use the Angular CLI tool to generate the component.

1. Open a new terminal window.
2. Navigate to this project directory:

cd angular-training/my-data-binding-demo

1. Using the Angular CLI tool, create a new **User** component:

ng generate component User

* + This command creates the following files:
  + create src/app/user/user.component.css
  + create src/app/user/user.component.html
  + create src/app/user/user.component.spec.ts
  + create src/app/user/user.component.ts

update src/app/app.module.ts

* + These are the template files related to the **User** component—the CSS, HTML, and TypeScript files.
  + In addition, the **app.module.ts** file is updated to reference the new **User** component.

5. Add New Fields and Display Values

In these exercises, you add new fields to the **User** component. You also update the HTML template to display the field values.

5.1. Update the **User** Component

1. Open **src/app/user/user.component.ts** and review the contents of the file.
2. On the line **above** **constructor**, add the following code:
3. firstName : string = "Mary";

lastName : string = "Doe";

* + This defines fields for the user’s first name and last name.
  + Expect your **user.component.ts** file to look like this:
  + import { Component, OnInit } from '@angular/core';
  + @Component({
  + selector: 'app-user',
  + templateUrl: './user.component.html',
  + styleUrls: ['./user.component.css']
  + })
  + export class UserComponent implements OnInit {
  + firstName : string = "Mary";
  + lastName : string = "Doe";
  + constructor() { }
  + ngOnInit() {
  + }

}

1. Save the file.
2. Open **src/app/user/user.component.html** and replace all of the text in the file with the following code:
3. <h3>Display Data</h3>
4. <p>
5. First Name: {{firstName}}
6. </p>
7. <p>
8. Last Name: {{lastName}}

</p>

* + This code displays the field values for **firstName** and **lastName**. The fields are defined in the **UserComponent** class **user.component.ts**.

1. Save the file.

5.2. Update the Main App Component

Now you add the **User** component to your main app component.

1. Open **src/app/app.component.html**.
2. Add the following code to the **end** of the file:

<app-user></app-user>

* + The main **AppComponent** references **UserComponent** by using the **UserComponent** selector **app-user**. This selector attribute is defined in the **UserComponent** file **user-component.ts**.
  + Expect your final file to look like this:
  + <h1>
  + {{title}}
  + </h1>

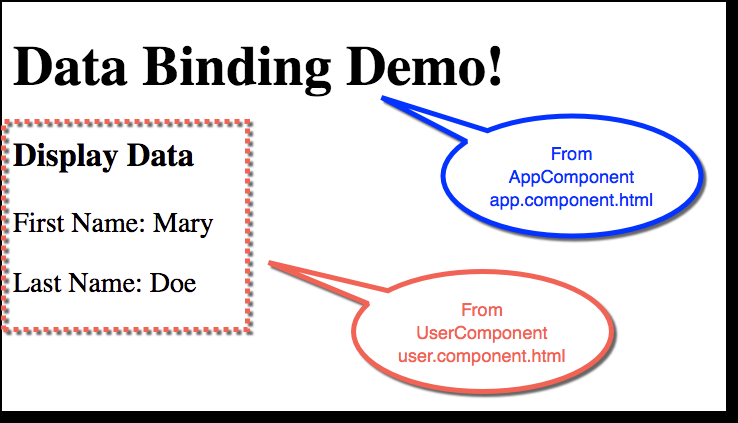
<app-user></app-user>

1. Make sure all of your files are saved.

5.3. Test Your App

Now you test the results in your browser.

|  |  |
| --- | --- |
|  | The **ng serve** command is still running, so it automatically recompiles your code and refreshes the browser. |

1. Move back to your web browser and verify that you see a screen similar to this:
2. 
3. Observe that this screen is a combination of components:
   * **AppComponent** handles displaying the main header **Data Binding Demo**.
   * **AppComponent** also references **UserComponent** with the **app-user** selector.
   * **UserComponent** handles displaying the user’s first and last names. This is defined in the **user.component.html** file.

6. Add Text Fields for One-Way Binding

In these exercises you add text fields for one-way data binding. The initial value of the text field is bound to the value from the **User**component. This is one-way binding.

6.1. Add HTML Text-Input Fields

1. Open **src/app/user/user.component.html**.
2. Add the following code to the **end** of the file:
3. <hr>
4. <h3>Data Binding: One-way (Component -> View)</h3>
5. <p>
6. First Name: <input [value]="firstName" />
7. </p>
8. <p>
9. Last Name: <input [value]="lastName" />

</p>

* + This code adds two new text-input fields.
  + The square brackets around **[value]** bind the **UserComponent** field to the input text. This is an example of one-way binding.
  + The text-input field displays the initial value of the **UserComponent** fields for **firstName** and **lastName**.

|  |  |
| --- | --- |
|  | If you make changes to the input text fields, no updates are made to the fields of **UserComponent**, because this is only one-way binding, from component to view. |

1. Save the file.
   * Expect the final **user.component.html** file to look like this:
   * <h3>Display Data</h3>
   * <p>
   * First Name: {{firstName}}
   * </p>
   * <p>
   * Last Name: {{lastName}}
   * </p>
   * <hr>
   * <h3>Data Binding: One-way (Component -> View)</h3>
   * <p>
   * First Name: <input [value]="firstName" />
   * </p>
   * <p>
   * Last Name: <input [value]="lastName" />

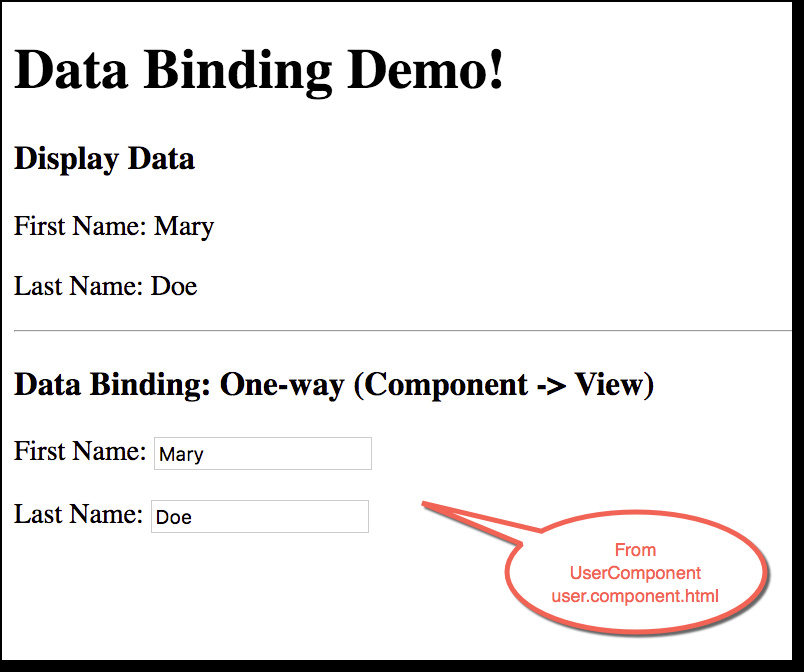
</p>

6.2. Test Your App

Now test the results in your browser.

|  |  |
| --- | --- |
|  | The **ng serve** command is still running, so it automatically recompiles your code and refreshes the browser. |

* Move back to your web browser and verify that you see a screen similar to this:

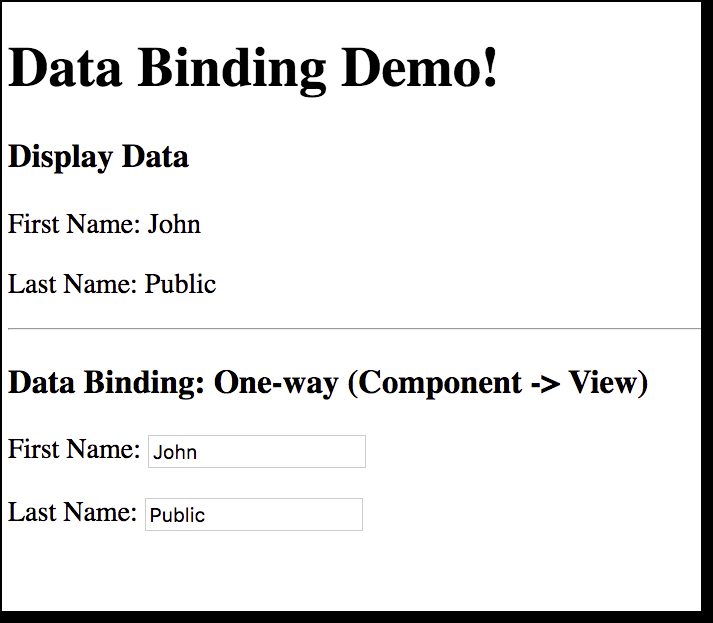


6.3. Test for One-Way Updates

1. Edit **user.component.ts** by changing the field values for **firstName** and **lastName**:
2. firstName : string = "John";

lastName : string = "Public";

1. Save the file.
2. Move back to the web browser and confirm that the new values are displayed:



* + Remember that this is one-way binding, from the component to the view only.

7. Add Support for Two-Way Data Binding

In these exercises, you add and test text fields for two-way binding. The value of the text field is bound to the value from the **User**component. Any updates to the text field also update the values in the component. This is two-way binding.

7.1. Add HTML Text-Input Fields

1. Open **src/app/user/user.component.html**.
2. Add the following code to the **end** of the file:
3. <hr>
4. <h3>Data Binding: Two-way (Component <-> View)</h3>
5. <p>
6. First Name: <input [(ngModel)]="firstName"/>
7. </p>
8. <p>
9. Last Name: <input [(ngModel)]="lastName" />

</p>

* + This code adds two new text-input fields.
  + [**ngModel**] is used for two-way binding.
  + The text-input fields display the initial value of the **UserComponent** fields **firstName** and **lastName**.
  + Any changes made to the text-input field are also made in **UserComponent**. This is an example of two-way binding.

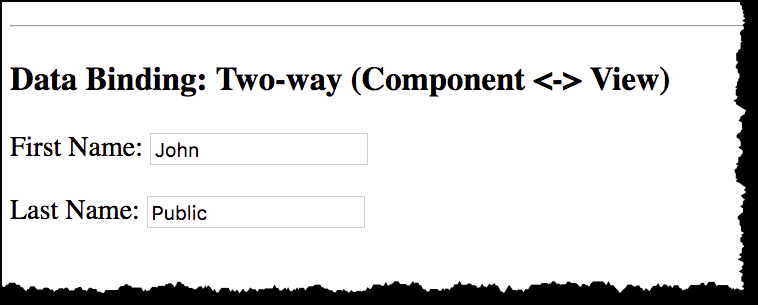
1. Save the file.
   * Expect the final **user.component.html** file to look like this:
   * <h3>Display Data</h3>
   * <p>
   * First Name: {{firstName}}
   * </p>
   * <p>
   * Last Name: {{lastName}}
   * </p>
   * <hr>
   * <h3>Data Binding: One-way (Component -> View)</h3>
   * <p>
   * First Name: <input [value]="firstName" />
   * </p>
   * <p>
   * Last Name: <input [value]="lastName" />
   * </p>
   * <hr>
   * <h3>Data Binding: Two-way (Component <-> View)</h3>
   * <p>
   * First Name: <input [(ngModel)]="firstName"/>
   * </p>
   * <p>
   * Last Name: <input [(ngModel)]="lastName" />

</p>

7.2. Test Your App

Now test your app in the browser.

* Move back to your web browser. Expect to see a screen similar to this:



7.3. Test for Two-Way Updates

1. Move to the new section **Data Binding: Two-way (Component < - > View)**.
2. Change the values in the input text fields and observe how updates are made instantly in other sections of the page.
   * This is two-way data binding.
   * The other sections of the page are bound to the fields **firstName** and **lastName**.
   * Any updates to those fields in the component are reflected in the view.

