Angular Data Binding Notes

Data Binding: Data Binding is communication between typescript to html(class to html|Model-View) is nothing but Data Binding

->Types of Data Bindings

->one-way Data Binding(unidirectional)

1) Interpolation Binding: Interpolation refers to binding expressions into marked up language.

Example using the interpolation technique where we are binding two values, firstName and the lastName, to the view, enclosed in double curly braces: {{property Name}}.

File Name: app.component.ts

```
import { Component } from "@angular/core";

@Component({

        selector: 'app-example',

        template: `

            <div>

                <strong>{{firstName}}</strong>

                <strong>{{lastName}}</strong>

            </div>  `

              })


export class AppComponent {

        firstName: string = "Yallaling";

        lastName:string = "Goudar";

}
```

Angular Data Binding Notes

2) Property Binding: Property binding is used to set a property of a view element. The binding sets the property to the value of a template expression.

Example using property binding. In this example, we are binding one value, firstName, to the innerHTML property of the span tag. It will bind the value of firstName to the span element.

File Name: app.component.ts

```
import { Component } from "@angular/core";
  @Component({
                selector: 'app-example',
                template: `
                <div>
                        <span [innerHTML]='firstName'></span>
                </div> `
                })
       export class AppComponent {
                 firstName: string = "Yallaling";
       }
```

3)Attribute Binding: Attribute binding is used to set a attribute property of a view element.

example where we are trying to bind a value to the colspan property of the element.

```
<h2>Attribute Binding Example</h2>
<table>
  <tr>
    <td colspan="{{4}}"></td>
  </tr>
</table>
```

Angular Data Binding Notes

This will throw an error "Template parse errors: Can't bind to 'colspan' since it isn't a known native property".

We can only use property binding and interpolation for binding the properties, not attributes. We need separate attribute binding to create and bind to attributes. To learn more about property binding, check out my previous guide Property Binding in Angular(/guides/ property-binding-angular).

Attribute binding syntax is like property binding. In property binding, we only specify the element between brackets. But in the case of attribute binding, it starts with the prefix attar, followed by a dot (.), and the name of the attribute. You then bind the attribute value using an expression that resolves to a string.

Let's consider an example where we are creating a table and setting the colspan attribute of the element. Here, we are setting the colspan to 3 by binding value to attr.colspan attribute property.

File Name: example.component.ts

```
import { Component } from "@angular/core";

        @Component({

          selector: 'app-example',

          template: `

            <div>

              <table>

                <tr><td [attr.colspan]="3">three</td></tr>

               <tr><td>1</td><td>2</td><td>3</td></tr>

            </table>

         </div>  `

        })

        export class ExampleComponent {

        }
```

4) Style Binding: Style binding is used to set a style of a view element.

Example of style binding. In this example, we are binding a color style to the 'h1' element. It will display the text within the h1 tags in a blue color.

File Name: app.component.html

```
<h1 [style.color]="blue">This is a Blue Heading</h1>
```

From View to Component

One-way data binding from view to the component can be achieved by using the event binding technique.

Let's consider an example where within parentheses on the left of the equal sign we have the target event like "click" and on the right side we may have the template statements such as component properties and methods(myFunction() in this case) bind to the event.

1.app.component.html

```
<button (click)="myFunction()">Show alert</button>
```

In the above code, the myFunction() method in the component will be called when user clicks on the button.

Filename app.component.ts

```
import { Component } from "@angular/core";
        @Component({
                selector: 'app-example',
                template: `<button (click)='myFunction()' >Show alert</button>`
               })
        export class AppComponent {
         myFunction(): void {
               alert('Show alert!');
```

```
 }

}
```

Once you run the above code, you will see a button with text "Show alert". When you click that button, it will call the myFunction() method in the component, which will, in turn, execute the alert() method, showing an alert box with the text "Show an alert".

2) Two-way Data Binding in Angular

Two-way data binding in Angular will help users to exchange data from the component to view and from view to the component. It will help users to establish communication bi-directionally.

Two-way data binding can be achieved using a ngModel directive in Angular. The below syntax shows the data binding using (ngModel), which is basically the combination of both the square brackets of property binding and parentheses of the event binding.

1.app.component.html

```
<input type="text" [(ngModel)] = 'val' />
```

Before using ngModel to achieve two-way data binding, it’s very important to import the FormsModule from @angular/forms in app.module.ts file as shown below. FormsModule will contain the ngModule directive.

Filename app.module.ts

```
import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from "@angular/forms";

 import { AppComponent } from './app.component';

import { FormsModule } from "@angular/forms";

        @NgModule({

                imports: [BrowserModule, FormsModule],

                declarations: [ AppComponent],

               bootstrap: [AppComponent]

        })
```

```
export class AppModule { }
```

If you do not import the FormsModule, then you will get Template parse errors and it will result in this error:

"Can't bind to 'ngModel' since it is not a known property of 'input'".

After importing the FormsModule, you can go ahead and bind data using (ngModel) as shown below.

```
import { Component } from '@angular/core';

        @Component({

        selector: 'app-example',

        template: `

        Enter the value  : <input [(ngModel)] ='val'>

        <br>

         Entered value is:  {{val}}

      `

    })

    export class AppComponent {

      val: string = '';

    }
```

Once we run the above code, we will see an input box asking us to enter a value in the view. Any value entered in that input box will be bound with the text below. Let's assume a user entered the text Chandra", then the text will be "Entered value is: Chandra".

5)Class Binding: Class binding is used to set a class property of a view element.

Class binding is used to set a class property of a view element. We can add and remove the CSS class names from an element's class attribute with class binding.

The class binding syntax is also like property binding. In property binding, we only specify the element between brackets. But in the case of class binding, it starts with the prefix class, followed by a dot (.), and the name of the class. You then bind the class value with CSS class name like class.class-name.

Angular Data Binding Notes

The example below shows the standard way of setting class attribute without binding. In this case, we are setting a class attribute with a class name 'myClass' without binding.

1. html file

```
<div class="myClass">Setting class without binding</div>
```

The example below shows setting all the class values with binding. In this case, we are binding class "myClassBinding" with class binding.

1. html

```
<div class="myClass" [class]="myClassBinding">Setting all classes with binding</div>
```

Whenever the template expression evaluates to true, Angular binds that class name to the class binding. It removes the class when the template expression evaluates to false.

Let's see another example where we are binding a specific class name with class binding. In this case, if 'isTrue' value evaluates to true then it will bind the 'myClass' to a class property. If it evaluates to false, then it will not bind the 'myClass' to a class property.

<u>File Name: example.component.ts</u>

```
import { Component } from "@angular/core";

@Component({

        selector: 'app-example',

         template: `

      <div>

      <h1 [class.myClass]="isTrue">This class binding is for true value</h1>

      <h1 [class.myClass]="!isTrue">This class binding is for false value</h1>

      </div>     `

       })

       export class ExampleComponent {

         isTrue: boolean = true;

       }
```