Angular 2 Databinding

Sang Shin
JPassion.com
"Code with Passion!"

Topics

- What is databinding?
- Interpolation
- Property binding
- Local template references
- Event binding
- Two-way databinding
- @Input (custom property binding)
- Component lifecycle

What is Databinding?

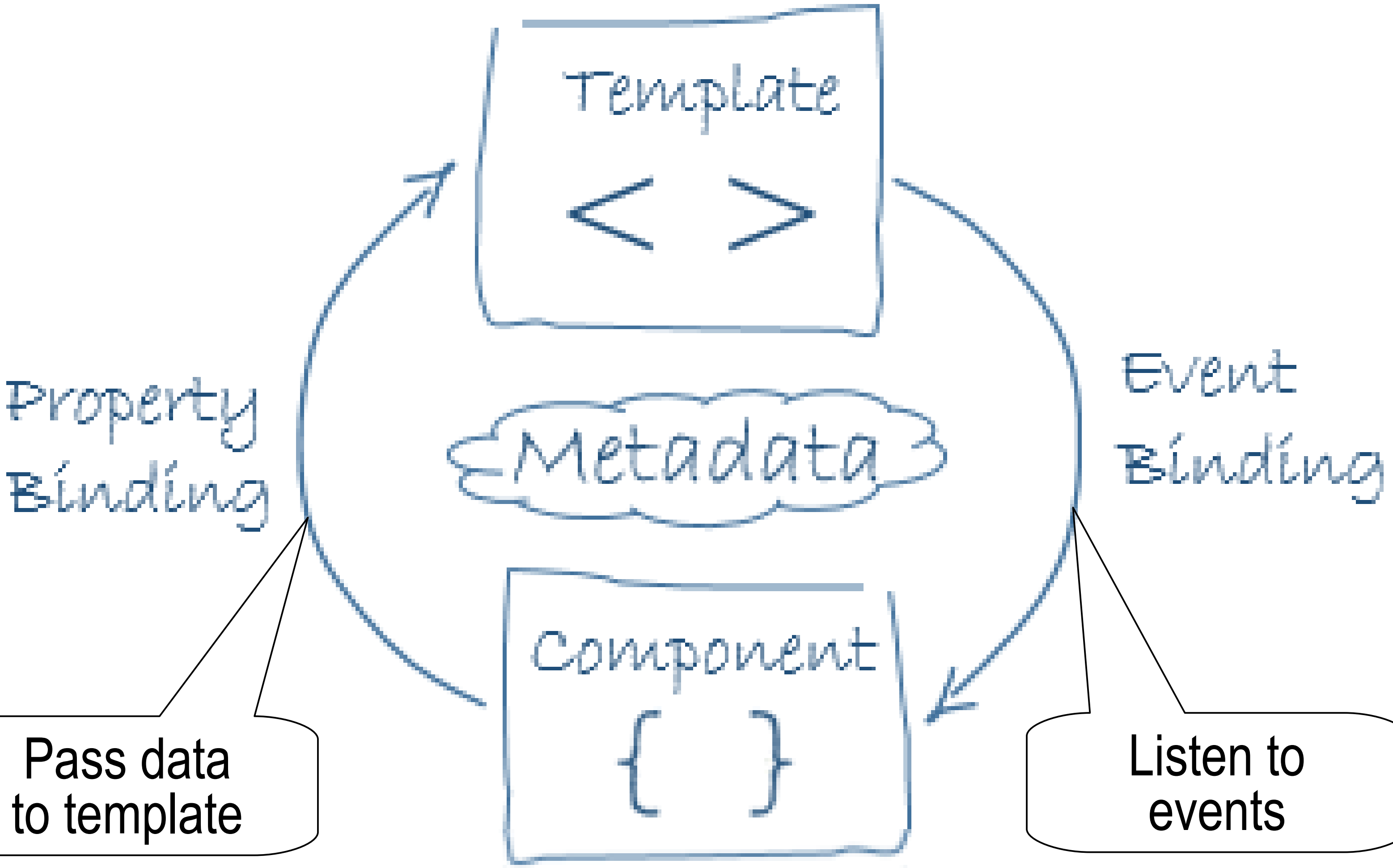
What is and Why Databinding?

- We want to generate and display dynamic contents

- Without a framework-level databinding, a developer himself/herself would be responsible for
  - > Pushing data values into the HTML controls
  - > Handling user actions, which triggers value updates

- Angular databinding framework handles all these for you
  - > All you have to do is to use proper databinding markup's in the template

Databinding between Component & Template

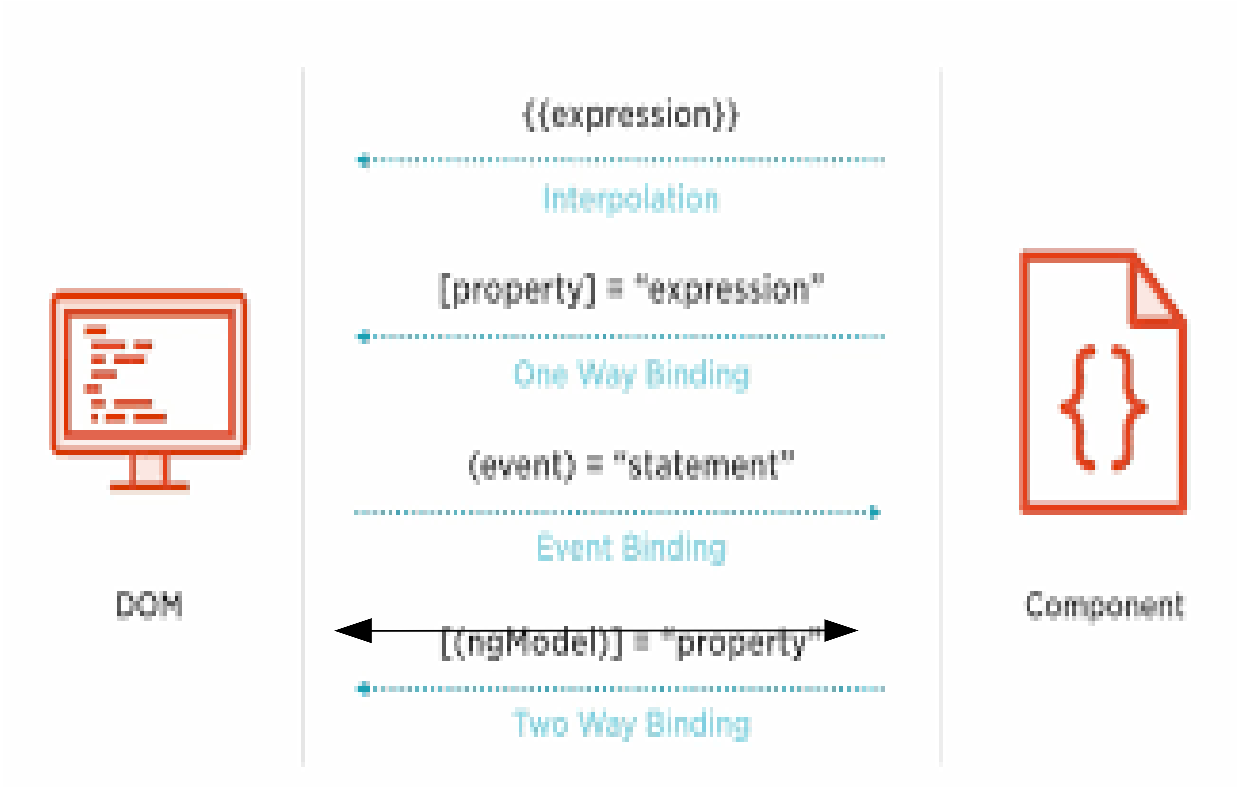


Pass data to template

Listen to events

4 Schemes of Databinding

- Each scheme has a direction — from the DOM to component, to the DOM from component, or in both directions
- Three are one-way databinding and one is two-day databinding



Interpolation

Interpolation

- {{ expression resolves to a string }}

- The expression is typically the name of a component property: Angular replaces that name with the string value of the property

“title” is a component property

```
<h3>
  {{ title }}
  <img src="{{ heroImageUrl }}" style="height:30px">
</h3>
```

- The expression can invoke methods of the host component

```
{{ "This is a message from " + getSomeData() }}
```

component method

Lab: Interpolation

- Add two properties with types to the component and display them using interpolation
  - stringData: string
  - numberData: number
- Add a method called getAllData() to the component and call it – getAllData() method should return combined value of stringData and numberData
  - {{ "This is a message from " + getAllData() }}
- Optional lab
  - Add “name” and “age” properties to the component
  - Add getPersonalData() method to the component
  - Access them using interpolation

Property Binding

Property Binding

- [property] = “expression resolving to a required value type”
  - > “3+5”
  - > “propertyOfComponent”
  - > “methodOfComponent()”

- Binding target can be a property of DOM element

  <input [value] = “expression”>

  <button [disabled]=”expression”>

  <img [src] = “expression”>

Lab: Property binding

- Try the string interpolation first
  - <input type=”text” value=”{{ stringData }}”>
- Try property binding (to the DOM properties)
  - <input [value]=”stringData”>
  - <input [value]=”numberData”>
  - <button [disabled]="switch">Click me</button>
  - <img [src]="imageSrc" alt=""> (Use "http://jpassion.com/images/duke.jpg")

Local Template References

Local template reference to DOM element

- You can provide local template references to a DOM element by using #
- It is local to the template and is not available to the component class

```
<p #myParagraph> test </p>
<p>{{myParagraph.textContent}}</p>


<input type=”text” #myInput>
<button (click)=”onClick(myInput.value)”>
```

Event Binding

Event Binding

- (click) = “expression handling the event”

- The (click) event binding typically calls a method in the component

  ```
  <button (click)="onClick()"></button>

  <input type=”text  #myinput>
  <button (click)="onClick(myinput.value)"></button>
  ```

- Or inline expression can be used as well

  ```
  <button (click)=”items.push(myinput.value)”></button>
  ```

Lab: Event binding

- Add a button with event binding, when clicked, call a method in the component
  - Just use console.log(“method is called”) inside the method to verify that the method is called
- Use event handling to switch on and switch off another button's *disabled* property

- Create an <input ..> element with local template reference

```
<input type=”text” #myinput>
```

- Add a button with event binding, when clicked, get a value of an <input> element via local template reference and display it back to the page

```
<button (click)=”onClick(myinput.value)”>Click me</button>
```

Two-way Binding

Two-way databinding

```
<input [(ngModel)]="user.name">
```

- Combines property and event binding in a single notation, using the ngModel directive
- In two-way databinding,
  - Change in the input box changes the corresponding property
  - Change in the property gets reflected in the input box
- Two-way databinding has a convenience but it also has performance implication
  - Use it only when needed

Lab: Two-way databinding

- Create a new component called “two-way-databinding” in the same directory of “databinding” component
  - > ng g c two-way-databinding --flat
- Create person object with name and age properties

  person = { name: 'Sang', age: 99 }
- Add <input> element whose value is two-way bound with the name property
- Add <input> element whose value is one-way bound with the name property
- Study whenever a new value is entered in one <input> element, how the other <input> reflects it
  - > Only the two-way databound input element will change the other

@Input() (Custom Property Binding)

@Input()

- Use it with a property in a child component in order to receive external value set in template of the parent (hosting) component

```
@Input()
result: number = 5;

@Input('result2')
resultxxx = 15;

// For string type to work, the property
// value has to be " 'san francisco' " not “san francisco”
@Input()
city = "boston";
```

Component class of “my-child” (child element)

```
<h4> Custom property binding: </h4>
<my-child [result]='10' [result2]=20
          [city]=" 'san francisco' ">
</my-child>
```

Inside of the template of parent (hosting) component

@Input()

- The external value typically comes from property of a parent component

```
@Input()
result: number;

@Input()
city: string;
```

```
<h4> Custom property binding: </h4>
<my-child [result]=”parentResult”
            [city]=" parentMethod()">
</my-child>
```

Component class
of “my-child” (child element)

Inside of the template of
parent (hosting) component

Lab: @Input

- Add a new property to the child component
  - @Input() result: number
  - @input() someValue: string – make sure the hosting component pass it with single quote within double quote “ 'some String' ”
- Use property binding in the template of the parent (hosting) component to pass value to the result property
- Try @Input('differentName')

Component Lifecycle

Component LifeCycle

- Angular calls lifecycle hook methods on directives and components as it creates, changes, and destroys them
  - ngOnChanges – every time data-bound property gets changed
  - ngOnInit – once when component is initialized
  - ngDoCheck – every time Angular change detection cycle starts
  - ngOnDestroy – once when component is destroyed
- Each interface has a single hook method whose name is the interface name prefixed with ng
  - OnInit interface has a hook method named ngOnInit
  - OnDestroy interface has a hook method named ngOnDestroy

Example

```
export class DatabindingComponent implements OnInit, OnDestroy {

  ...
  constructor() { }

  ngOnInit() {
    console.log("ngOnInit called");
  }

  ngOnDestroy() {
    console.log("ngOnDestroy called");
  }

}
```

Lab: Component Lifecycle

- Log a message to the console whenever lifecyle methods get called
- Optional lab
  - Use *ngIf directive to remove a component and observe *ngOnDestroy()* method gets called

Code with Passion!
JPassion.com