Angular

AJ

Template Reference Variables

You declare a reference variable by using the hash symbol (#). The #firstNameInput declares a firstNameInput variable on an <input> element.

```
<input type="text" #firstNameInput>

<input type="text" #lastNameInput>
```

After that, you can access the variable anywhere inside the template. For example, I pass the variable as a parameter on an event.

<button (click)="show(lastNameInput)">Show</button>

Remember that the lastNameInput belongs to HTMLInputElement type.

```
show(lastName: HTMLInputElement){
   console.log(lastName.value);
}
```

Usually, the reference variable can only be accessed inside the template. However, you can use ViewChild decorator to reference it inside your component.

```
import {ViewChild, ElementRef} from '@angular/core';// Reference
firstNameInput variable inside Component
@ViewChild('firstNameInput') nameInputRef: ElementRef;
```

After that, you can use this.nameInputRef anywhere inside your Component.

```
show(lastName: HTMLInputElement){
  this.fullName = this.nameInputRef.nativeElement.value + ' ' +
lastName.value;
}
```

Two-way Data Binding

Angular >= 2.x doesn't come with such a (built-in) two-way data binding anymore. However, this doesn't mean we can't create directives that support two-way data binding. Implements two-way data binding: **ngModel**

In order to understand what that means, let's take a look at this code snippet here:

```
<input [(ngModel)]="username">

<p>Hello {{username}}!</p>
```

```
<input [value]="username" (input)="username = $event.target.value">

<p>Hello {{username}}!</p>
```

Let's take a closer look at what's going on here:

- **[value]="username"** - Binds the expression `username` to the input element's `value` property
- **(input)="expression"** - Is a declarative way of binding an expression to the input element's `input` event (yes there's such event)
- **username = $event.target.value** - The expression that gets executed when the `input` event is fired
- **$event** - Is an expression exposed in event bindings by Angular, which has the value of the event's payload

Structural directives in Angular

- **NgFor:** It is a repeater directive that customizes data display. It can be used to display a list of items.
- **NgIf:** It removes or recreates a part of DOM tree depending on an expression evaluation.

```
public names = [
  { name: "Kamal"},
  { name: "Mitchel"},
  { name: "Yoon"},
  { name: "Johnson"},
  { name: "Jet Li"}
];
```

```
<p class="alert alert-success" *ngIf="names.length >
2">Currently there are more than 2 names!</p>
   <p class="alert alert-danger" *ngIf="names.length <=
2">Currently there are less than 2 names left!</p>
   <ul>
       <li *ngFor="#nam of names"
        (click)="onNameClicked(nam)"
        >{{ nam.name }}</li>
      </ul>
```