Angular Forms

- Form is a container that comprises of set of elements, which allow interaction with our application.
- Form provides an UI from where user can input, edit, delete, view data.
- HTML form comprises of elements like button, textbox, checkbox, radio, listbox etc.
- Angular makes the static HTML form into dynamic.
- HTML presents the form and Angular makes it interactive to handle client-side interactions.
- Based on where Angular is handling interactions the forms in angular are classified into 2 types
    - **Template Driven Forms**
    - **Model Driven Form / Reactive Form**

Template Driven Forms

- A template driven form configures and handles all interactions at View Level (HTML)
- Configuration of a form and its manipulation both handled in HTML template.
- Very optimized controller level interaction. All interactions are at view level.
- It reduces the number of requests to a component.
- It improves the page load time.
- It is good for forms designed in “in-line” technique.
- Template drive form is heavy on page. Slow in handling interactions and rendering.
- Hard to test and extend the form.
- Separation issues. Not loosely coupled.
- You can use template driven forms when you are designing an UI that doesn’t require regular extensions.

- The directives that are used to configure “Form and Form Elements” in template driven approach.
    - NgForm
    - NgModel
- **NgForm:** It provides a set of properties and methods that are used to configure and handle <form> element.
- **NgModel:** It provides a set of properties and methods that are used to configure and handle a form control like button, textbox, checkbox, radio, dropdown list, etc.
- The library for “NgForm and NgModel” is “@angular/forms”
- The module is “FormsModule”

Configuring Form:

```
<form #formName=”ngForm”>

</form>
```

- NgForm provides set of attributes
    - Value
    - pristine
    - dirty
    - valid
    - invalid
    - etc..

Syntax:

```
formName.value

formName.pristine

formName.dirty
```

Configuring a Form Element

- “NgModel” is used to make a static form control into dynamic.

```
<input type=”text” ngModel   #txtName=”ngModel” name=”txtName”>
```

txtName.value

txtName.valid

Key Note:

- A Form Reference must implement “NgForm” to handle the form behaviour.
  <form #frmRegister=”ngForm”> [**ngForm is of type NgForm**]
- NgForm makes the form dynamic.
- NgForm is a member of “**FormsModule**” in “@angular/forms”.
- Form can’t access and submit data of any control without a **Name**. Every control defined in a form must have a **“name”** defined.
  <input type=”text” name=”txtName”>
- Angular can’t recognize any form element dynamically. The static form element must transform into dynamic form element.
- “**NgModel”** is a directive that makes the Static Form element into Dynamic.
  <input type=”text” name=”txtName” ngModel>
- Every dynamic element must have a reference name, which is used as a model name to store its value dynamically.
- Every form element must implement “NgModel”.
- Every control is configured “ngModel”. So that it can have a reference name dynamically.

<input type=”text” name=”txtName” ngModel #txtName=”ngModel”>

Form related properties are derived from “NgForm”
Control related properties are derived from “NgModel”

Accessing the values from Template Form:

Accessing all Values

- You can use “value” property of “NgForm”.
- The “value” property returns an object that contains collection of key value pairs.
- The key/value pairs refer to Control Name and the Control Value.
- All form control and their values can be accessed by using “formName.value” property.

Syntax:

frmRegister.value → Returns collection of all form elements and their values as an object

Accessing any specific value

- You can access the value of any control by using “value” property of “NgModel”

Syntax:

txtName.value → Returns the value of any specific element.

Ex:

- **Templateform.component.ts**

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

```
@Component({
  selector: 'app-templateform',
  templateUrl: './templateform.component.html',
  styleUrls: ['./templateform.component.css']
})
export class TemplateformComponent {
   public product;
   public onFormSubmit(obj){
     this.product = obj;
     alert(this.product.txtName);
   }
}
```

- **Templateform.component.html**

```
<div class="container-fluid">
  <div class="row">
    <div class="col-3">
       <h2>Register Product</h2>
       <form #frmRegister="ngForm"
(submit)="onFormSubmit(frmRegister.value)" >
          <div class="form-group">
             <label>Name</label>
             <div>
                <input ngModel #txtName="ngModel"
name="txtName" type="text" class="form-control">
             </div>
          </div>
          <div class="form-group">
             <label>Price</label>
             <div>
                <input ngModel #txtPrice="ngModel"
name="txtPrice" type="text" class="form-control">
```

```
            </div>
          </div>
          <div class="form-group">
            <label>Quantity</label>
            <div>
              <input ngModel #txtQty="ngModel" name="txtQty"
type="text" class="form-control">
            </div>
          </div>
          <div class="form-group">
            <label>Shipped To</label>
            <div>
              <select ngModel #lstShippedTo="ngModel"
name="lstShippedTo" class="form-control">
                <option>Delhi</option>
                <option>Hyderabad</option>
              </select>
            </div>
          </div>
          <div class="form-group">
            <label>In Stock</label>
            <div>
              <input ngModel #optStock="ngModel"
name="optStock" type="checkbox"> Yes
            </div>
          </div>
          <div class="form-group">
            <button class="btn btn-primary btn-
block">Register</button>
          </div>
        </form>
      </div>
      <div class="col-4">
```

```
        <h2>Product Object</h2>
        <pre>
          {{frmRegister.value | json}}
        </pre>
      </div>
      <div class="col-5">
        <h2>Product Details</h2>
        <dl>
          <dt>Name</dt>
          <dd>{{frmRegister.value.txtName}}</dd>
          <dt>Price</dt>
          <dd>{{txtPrice.value}}</dd>
          <dt>Qty</dt>
          <dd>{{txtQty.value}}</dd>
          <dt>Total</dt>
          <dd>{{txtQty.value * txtPrice.value}}</dd>
          <dt>Shipped To</dt>
          <dd>{{lstShippedTo.value}}</dd>
          <dt>Stock Status</dt>
          <dd>{{(optStock.value==true)?"Available":"Out of
Stock"}}</dd>
        </dl>
      </div>
    </div>
  </div>
```

Configure Forms and Nested Forms with Controls

- You can dynamically create and configure forms.
- It allows to extend the form and make it more asynchronous.
- You can create a form by using “FormGroup” base.

- “FormGroup” is a collection of FormControls.

Syntax:

```
Public parentForm = new FormGroup({
    controlName : new FromControl(),
    controlName : new FormControl(),
    childForm: new FormGroup() {
        controlName:  new FormControl()
    }
})
```

- To bind a form and nested form you have to use the properties
    - [formGroup] – Parent Form
    - [formGroupName] – child Form

Syntax:

```
<form [formGroup]=”parentForm”>
    <div [formGroupName]=”childForm”>

    </div>
</form>
```

- If you are defining a control in form group the control is bound to element by using the attribute “formControlName”

Syntax:

<input type=”text” formControlName=”controlName”>

- The methods used to set and patch values are
    - setValue()
    - patchValue()

Ex:

Reactivedemo.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup } from '@angular/forms';


@Component({
  selector: 'app-reactivedemo',
  templateUrl: './reactivedemo.component.html',
  styleUrls: ['./reactivedemo.component.css']
})
export class ReactivedemoComponent{
  frmRegister = new FormGroup({
```

```
    Name: new FormControl(''),
    Price: new FormControl(''),
    frmDetails: new FormGroup({
      City: new FormControl(''),
      Instock: new FormControl('')
    })
  });
  UpdatePartial(){
    this.frmRegister.patchValue({
      Name: 'Samsung TV',
      frmDetails: {
        City: 'Delhi',
        Instock: true
      }
    });
  }
}
```

Reactivedemo.component.html

```
<div class="container-fluid">
  <div class="row">
```

```
<div class="col-3">
  <h2>Register Product</h2>
  <form [formGroup]="frmRegister">
    <fieldset>
      <legend>Basic Info</legend>
      <dl>
        <dt>Name</dt>
        <dd>
          <input type="text"
formControlName="Name" class="form-control">
        </dd>
        <dt>Price</dt>
        <dd>
          <input type="text"
formControlName="Price" class="form-control">
        </dd>
      </dl>
    </fieldset>
    <fieldset>
      <legend>Stock Details</legend>
      <div formGroupName="frmDetails" >
```

```
                <dl>
                    <dt>City</dt>
                    <dd>
                        <select formControlName="City"
class="form-control">
                            <option>Delhi</option>
                            <option>Hyd</option>
                        </select>
                    </dd>
                    <dt>In Stock</dt>
                    <dd>
                        <input formControlName="Instock"
type="checkbox">
                    </dd>
                </dl>
                <button (click)="UpdatePartial()" class="btn
btn-primary  btn-block">Update Details</button>
            </div>
        </fieldset>
    </form>
</div>
```

```
      <div class="col-9">
        <h3>Product Details</h3>
        <dl>
          <dt>Name</dt>
          <dd>{{frmRegister.value.Name}}</dd>
          <dt>Price</dt>
          <dd>{{frmRegister.value.Price}}</dd>
          <dt>City</dt>
          <dd>{{frmRegister.value.frmDetails.City}}</dd>
          <dt>Stock</dt>
          <dd>
{{frmRegister.value.frmDetails.Instock==true?"Availabl
e":"Out of Stock"}}
          </dd>
        </dl>
      </div>
  </div>
</div>
```

Ex: Update All and Partial

Reactivedemo.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup } from
'@angular/forms';


@Component({
  selector: 'app-reactivedemo',
  templateUrl: './reactivedemo.component.html',
  styleUrls: ['./reactivedemo.component.css']
})
export class ReactivedemoComponent{
   frmRegister = new FormGroup({
     Name: new FormControl(''),
     Price: new FormControl(''),
     frmDetails: new FormGroup({
       City: new FormControl(''),
       InStock: new FormControl('')
     })
   });
```

```
  UpdatePartial(): void{
    this.frmRegister.patchValue({
        Price: 45000.56,
        frmDetails: {
          City: 'Hyd'
        }
    });
  }
  UpdateAll(): void {
    this.frmRegister.setValue({
      Name: 'Nike Casuals',
      Price: 4200.33,
      frmDetails: {
        City: 'Chennai',
        InStock: true
      }
    });
  }
}
```

Reactivedemo.component.html

```
<div class="container-fluid">
  <div class="row">
    <div class="col-3">
      <h2>Register</h2>
      <form [formGroup]="frmRegister">
        <fieldset>
          <legend>Basic Info</legend>
          <dl>
            <dt>Name</dt>
            <dd>
              <input formControlName="Name"
type="text" class="form-control">
            </dd>
            <dt>Price</dt>
            <dd>
              <input formControlName="Price" type="text"
class="form-control">
            </dd>
          </dl>
        </fieldset>
```

```
        <fieldset>
          <legend>Stock Details</legend>
          <div formGroupName="frmDetails" >
           <dl>
            <dt>City</dt>
            <dd>
              <select formControlName="City"
class="form-control">
               <option>Delhi</option>
               <option>Chennai</option>
               <option>Hyd</option>
              </select>
            </dd>
            <dt>In Stock</dt>
            <dd>
              <input formControlName="InStock"
type="checkbox"> Yes
            </dd>
           </dl>
          </div>
        </fieldset>
```

```
        <div class="btn-group">
          <button (click)="UpdateAll()" class="btn btn-primary">Update All</button>
          <button (click)="UpdatePartial()" class="btn btn-info">Update Partial</button>
        </div>
      </form>
    </div>
    <div class="col-9">
      <h2>Product Details</h2>
      <dl>
        <dt>Name</dt>
        <dd>{{frmRegister.value.Name}}</dd>
        <dt>Price</dt>
        <dd>{{frmRegister.value.Price}}</dd>
        <dt>City</dt>
        <dd>
          {{frmRegister.value.frmDetails.City}}
        </dd>
        <dt>In Stock</dt>
        <dd>
```

```
{{(frmRegister.value.frmDetails.InStock==true)?"Availa
ble":"Out of Stock"}}
        </dd>
      </dl>
   </div>
  </div>
</div>
```

Form Builder in Reactive Approach

- Form builder is a **service** provided by Angular to configure forms and its elements dynamically.
- FormBuilder uses singleton pattern.
- Memory is allocated for first request to form the same memory is used across multiple requests.
- FormBuilder is the base for configuring forms and its controls, it provides the following methods
    - group()
    - control()
    - array()

Form Builder Method	**Description**

group()	It configures a form group with set of elements. It dynamically creates <form> element. It can be used to create nested forms.
control()	It configures form element like <input>, <select>, <option>, <textarea> etc.
array()	It configures a collection of form controls. It allows to add or remove controls dynamically.

- The properties that are used to bind with UI elements
  - formGroup : Parent Form
  - formGroupName : Child Form
  - formControlName : Form elements
- **FormBuilder** is a member of "@angular/forms"

Syntax:

```
constructor(private fb: FormBuilder) { }

parentForm = this.fb.group({

   controlName: ['value', validators],

   controlName: ['value', validation],
```

```
    childForm: this.fb.group({
        controlName: [‘value’, validation]
      })
  });

  <form [formGroup]=”parentForm”>
    <div formGroupName=”childForm”>
      <input type=”text” formControlName=”controlName”>
    </div>
  </form>
```

Ex:

Builderdemo.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder } from '@angular/forms';

@Component({
  selector: 'app-builderdemo',
  templateUrl: './builderdemo.component.html',
  styleUrls: ['./builderdemo.component.css']
})
```

```
export class BuilderdemoComponent implements
OnInit {

  constructor(private fb: FormBuilder) { }

  frmRegister = this.fb.group({
    Name: [''],
    Price: [''],
    frmDetails: this.fb.group({
      City: [''],
      InStock: ['']
    })
  });

  ngOnInit(): void {
  }

}
```

Builderdemo.component.html

```
<div class="container-fluid">
```

```
 <div class="row">
  <div class="col-3">
   <h2>Register Product</h2>
   <form [formGroup]="frmRegister">
    <fieldset>
      <legend>Basic Info</legend>
      <dl>
        <dt>Name</dt>
        <dd>
          <input type="text"
formControlName="Name" class="form-control">
        </dd>
        <dt>Price</dt>
        <dd>
          <input type="text"
formControlName="Price" class="form-control">
        </dd>
      </dl>
    </fieldset>
    <fieldset>
      <legend>Stock Details</legend>
```

```
          <div formGroupName="frmDetails">
            <dl>
              <dt>City</dt>
              <dd>
                <select formControlName="City"
class="form-control" >
                  <option>Delhi</option>
                  <option>Hyd</option>
                </select>
              </dd>
              <dt>In Stock</dt>
              <dd>
                <input type="checkbox"
formControlName="InStock">  Yes
              </dd>
            </dl>
          </div>
        </fieldset>
      </form>
    </div>
    <div class="col-9">
```

```
        <h2>Product Details</h2>
        <dl>
            <dt>Name</dt>
            <dd>{{frmRegister.value.Name}}</dd>
            <dt>Price</dt>
            <dd>{{frmRegister.value.Price}}</dd>
            <dt>City</dt>
            <dd>{{frmRegister.value.frmDetails.City}}</dd>
            <dt>Stock</dt>
            <dd>
{{(frmRegister.value.frmDetails.InStock==true)?"Availa
ble":"Out  of Stock"}}
            </dd>
        </dl>
    </div>
  </div>
</div>
```