Reactive Form

Q.1) What is Reactive Forms? When we use Reactive Form?

- If we have a requirement to create a Complex form that time we use Reactive form. It is Also known as Modern-Driven Form.
- Reactive forms are more Robust (Strong).
- Reactive form does not support Two way data binding.
- Reactive form is Synchronous.
- Synchronous means execution of code line by line.
- When first statement executes successfully after that second statement will be execute.
- Main thing is Most of the code we write in TypeScript side.

Q.2) Difference between Template Driven form and Reactive form.

Template Driven Form	Reactive Form
Easier to create Template Driven Form	Complex to create Reactive Form
Most of the code we write in HTML Side	Most of the code we write in TypeScript side.
Form Structure and Logic is mainly implemented in HTML	Form Structure and Logic is mainly implemented in TypeScript.
Support Two Way DataBinding [(ngModel)]	Does Not Support Two Way DataBinding formControlName
Need to Implement FormsModule in imports array → app.module.ts	Need to Implement ReactiveFormsModule in imports array →app.module.ts
It is Asynchronous	It is Synchronous.
Apply Validations in HTML side.	Apply validations in TypeScript Side.
Complex for Unit Testing.	Easier for Unit Testing
Not Suitable for creation of forms with dynamic structure at run time	Suitable for creation of forms with dynamic structure at run time

Reactive Form

Q.3) What are the Reactive Forms?

- Reactive forms are forms where we define the structure of the form in the component class.
- i.e. we create the form model with Form Groups, Form Controls, and FormArrays.
- We also define the validation rules in the component class.
- Then, we bind it to the HTML form in the template.
- This is different from the template-driven forms, where we define the logic and controls in the HTML template.

Q.4) How to use Reactive Forms?

- Import *ReactiveFormsModule*
- Create Form Model in component class using FormGroup, FormControl & FormArrays
- Create the HTML Form resembling the Form Model.
- Bind the HTML Form to the Form Model

Q.5) Reactive Forms Example Application?

- Create one component and Run ng serve.
- Then imports Reactive Forms Module.
- To work with Reactive forms, we must import the ReactiveFormsModule.
- The ReactiveFormsModule contains all the form directives and constructs for working with angular reactive forms.

```
1
2  import { BrowserModule } from '@angular/platform-browser';
3  import { NgModule } from '@angular/core';
4  import { ReactiveFormsModule } from '@angular/forms';
5
6  import { AppRoutingModule } from './app-routing.module';
7  import { AppComponent } from './app.component';
8
9  @NgModule({
10   declarations: [
11     AppComponent
12   ],
13   imports: [
14     BrowserModule,
15     AppRoutingModule,
16     ReactiveFormsModule
17   ],
18   providers: [],
19   bootstrap: [AppComponent]
20 })
21 export class AppModule { }
```

Reactive Form

- In the template-driven approach, we used ngModel & ngModelGroup directive on the HTML elements.
- The FormsModule automatically creates the FormGroup & FormControl instances from the HTML template.
- This happens behind the scene.

Creating the Form Module

- The FormGroup is created with the following syntax
- contactForm = new FormGroup({})
- In the above, we have created an instance of a FormGroup and named it as contactForm.
- The FormGroup takes 3 arguments. (formControls, validators, async validators)
- a collection of a child controls (which can be FormControl, FormArray, or another FormGroup), **a validator, and an asynchronous validator**.
- The validators are optional.

Adding the Child Controls

- The first argument to FormGroup is the collection of controls.
- The Controls can be FormControl, FormArray or another FormGroup.

```
1
2  contactForm = new FormGroup({
3    firstname: new FormControl(),
4    lastname: new FormControl(),
5    email: new FormControl(),
6    gender: new FormControl(),
7    isMarried: new FormControl(),
8    country: new FormControl()
9  })
10
```

- In the above example, we have added Six FormControl instances each representing the properties
- firstname, lastname, email, gender, ismarried & country.
- The Other two arguments to FormGroup are Sync Validator & Async Validator. They are optional.
- With this our model is ready.

Reactive Form

HTML Template

```
<form>

 <p>
   <label for="firstname">First Name </label>
   <input type="text" id="firstname" name="firstname">
 </p>

 <p>
   <label for="lastname">Last Name </label>
   <input type="text" id="lastname" name="lastname">
 </p>

 <p>
   <label for="email">Email </label>
   <input type="text" id="email" name="email">
 </p>

 <p>
   <label for="gender">Geneder </label>
   <input type="radio" value="male" id="gender" name="gender"> Male
   <input type="radio" value="female" id="gender" name="gender"> Female
 </p>

 <p>
   <label for="isMarried">Married </label>
   <input type="checkbox" id="isMarried" name="isMarried">
 </p>


 <p>
   <label for="country">country </label>
   <select id="country" name="country">
     <option value="1">India</option>
     <option value="2">USA</option>
     <option value="3">England</option>
     <option value="4">Singapore</option>
```

```
   </select>
 </p>

 <p>
   <button type="submit">Submit</button>
 </p>

</form>

```

- Created a simple form using <form> tag having firstname, lastname, email, gender, ismarried, country and submit button.
- Now our next step is to Bind the template to the model(ts file).
- We need to tell angular that we have a model for the form.
- So it is Done using formGroup Directive.

```

<form [formGroup]="contactForm">

```

Reactive Form

- We use the square bracket (one-way binding) around `FormGroup` directive and assign our form model
- Next we need to Bind each form field to an instance of the FormControl models.
- We use FormControlName directive for this to bind formControl instance.
- We add this directive to every form field element in our form.

```
1) <input type="text" id="firstname" name="firstname" formControlName="firstname">
2) <input type="text" id="lastname" name="lastname" formControlName="lastname">
3) <input type="text" id="email" name="email" formControlName="email">
4) <input type="radio" value="male" id="gender" formControlName="gender"> Male
5) <input type="radio" value="female" id="gender" formControlName="gender"> Female
6) <input type="checkbox" id="isMarried" formControlName="isMarried">
7) <select id="country" name="country" formControlName="country">
```

- In above image we bind all our FormControls to Form Fields.

Submit Button

- We have submit button having type submit.
- ngSubmit directive binds itself to the click event of the submit button.
- We are using event binding to the ngSubmit to OnSubmit method.
- When user click on the submit button ngSubmit call the OnSubmit method.

```
1
2 <form [formGroup]="contactForm" (ngSubmit)="onSubmit()">
3
```

Final Template

```
1
2 <form [formGroup]="contactForm" (ngSubmit)="onSubmit()">
3
4   <p>
5     <label for="firstname">First Name </label>
6     <input type="text" id="firstname" name="firstname" formControlName="firstname">
7   </p>
8
9   <p>
10    <label for="lastname">Last Name </label>
11    <input type="text" id="lastname" name="lastname" formControlName="lastname">
12  </p>
13
14  <p>
15    <label for="email">Email </label>
16    <input type="text" id="email" name="email" formControlName="email">
17  </p>
```

Reactive Form

```
  <p>
    <label for="gender">Geneder </label>
    <input type="radio" value="male" id="gender" name="gender" formControlName="gender"> Male
    <input type="radio" value="female" id="gender" name="gender" formControlName="gender"> Female
  </p>

  <p>
    <label for="isMarried">Married </label>
    <input type="checkbox" id="isMarried" name="isMarried" formControlName="isMarried">
  </p>


  <p>
    <label for="country">country </label>
    <select id="country" name="country" formControlName="country">
      <option value="1">India</option>
      <option value="2">USA</option>
      <option value="3">England</option>
      <option value="4">Singapore</option>
    </select>
  </p>


  <p>
    <button type="submit">Submit</button>
  </p>

</form>

```

Now Receive Data into the Component class (ts file)

- The last step is to receive the form data in the component class.
- We need to create OnSubmit method.

```

onSubmit() {
  console.log(this.contactForm.value);
}
```

- We are using this.contactForm.value to send the values of our form data to the console window.

```
export class AppComponent {
  title = 'mdf';

  contactForm = new FormGroup({
    firstname: new FormControl(),
    lastname: new FormControl(),
    email: new FormControl(),
    gender: new FormControl(),
    isMarried: new FormControl(),
    country: new FormControl()
  })


  onSubmit() {
    console.log(this.contactForm.value);
  }
}


```

Reactive Form

Q.6) What is FormControl ?

- A `FormControl` takes 3 arguments.
- A default value, a Sync validator, and an asynchronous validator.
- All of them are optional.

Q.7) Default Value ?

- You can pass a default value either as a string or as an object of key-value pair.

```
1
2 //Setting Default value as string
3 firstname= new FormControl('Sachin');
4
```

Q.8) Sync Validator ?

- The second parameter is an array of sync Validators.
- Angular has some built-in Validators such as `required` and `minLength` etc.

```
1
2 firstname: new FormControl('', [Validators.required,Validators.minLength(10)]),
3
```

Q.9) Asynchronous Validators ?

- The third argument is the Async Validator.
- The syntax of Async Validators is similar to Sync Validators.