- In the module, we have to import ReactiveformsModule from forms. We don't need formsModule.

```
import {  ReactiveFormsModule } from '@angular/forms';
imports: [
   BrowserModule,
 ReactiveFormsModule,
   HttpModule
 ],
```

- In the TS file, we have to implement this in ngOnInit()

```
export class AppComponent implements OnInit {
 genders = ['male', 'female'];
 signupForm: FormGroup;

 ngOnInit(){
   this.signupForm = new FormGroup({});
 }
}
```

- **So far there are no controls to the forms, if we want to add any controls those should be added in FormGroup method and every input value must be declared as a FormControl object which should be imported from forms.**

```
ngOnInit(){
   this.signupForm = new FormGroup({
     'username': new FormControl(null),
     'email': new FormControl(null),
     'gender': new FormControl('male')
   });
 }
```

- **NOW HOW DO WE CONNECT BOTH HTML AND TS WRT to FORMS??**
  - Step1: In HTML, if you previously had a form tag,
    - <form [formGroup]="signupForm"]> - This tells, Hey! Use my formgroup not yours.
  - Step2: Letting know which element is connected to what by adding **FormControlName** to the input field.

```
<input
type="text"
id="username"
```

formControlName="username" - Username here is the property from the TS file

```
class="form-control">
```

- Now how do we submit the form?

```
<form [formGroup]="signupForm" (ngSubmit)="onSubmit()">
```

- Write the definition of the function and you can see the values.

Adding Validation

- In the template driven approach we had required field which should be added in the tag to make it as a required field. But, here, we should add something in the TS
- How?

```
this.signupForm = new FormGroup({
'username': new FormControl(null, Validators.required),
  'email': new FormControl(null, [Validators.required, Validators.email]),
  'gender': new FormControl('male')
  });
```

- If we add something called as Validators (Which is a object of Validators. Should be imported from the angular/core) and make it as required.

Adding The error Messages:

- In the template driven forms we used ngModel and created a local reference object and used it for the error validation
- But, here, we will use **ourformname.get('inputvariable name')** in the ***ngIf** to validate the input and also use the touched functionality to see if user actualy touched it or not.

```
<span *ngIf="!signupForm.get('username').valid &&
signupForm.get('username').touched"
class ="help-block">Please enter a valid User name</span>
```

Grouping the controls :

- Just group few objects into single object in TS and in HTML add those objects under this div with formGroupName tag and give the name you gave to the group.

```
'userData': new FormGroup({
'username': new FormControl(null, Validators.required),
'email': new FormControl(null, [Validators.required, Validators.email])
}),
```

IN HTML:

```
<div formGroupName="userData">
```

And change the path to

```
*ngIf="!signupForm.get('userData.username').valid &&
signupForm.get('userData.username').touched"
```

Using Arrays of Form Controls:

- If we want to add a set of values to single attribute then arrays of form controls is the best bet. How to do that?
- For that in the mainform inside ngOnInit(), we have to create an Object using the **FormArray** and it should be imported.
- `'hobbies' : new FormArray([])`
- In the method.

```
onAddHobby() {
const control = new FormControl(null, Validators.required);
(<FormArray>this.signupForm.get('hobbies')).push(control);
}
```