Reactive Form Basic

initializing a basic form with controls:

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

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

syncing html and our form created in TS:

we use form group directive-it tells angular use my form group and don't set up on your own

we use property binding because we are passing our form controls here.

```
<div class="row">
  <div class="col-xs-12 col-sm-10 col-
    <form [formGroup]="signupForm">
      <div class="form-group">
```

now we tell angular which control should be assigned to what input we use formcontrolname directive

```
  <input
    type="text"
    id="username"
    formControlName = "username"
    class="form-control">
</div>
```

submitting the form:

use Ng Submit

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

```
<div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-
  <form [formGroup]="signupForm" (ngSubmit)="onSubmit()">
    <div class="form-group">
      <label for="username">Username</label>
```

validation:

we can pass one validators or array of validators as seen in the pictures

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

getting access to controls:

here the username inside the get method is the control name in the TS file

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

Grouping controls:

get also takes a path. we might specify the path because we might have nested form.

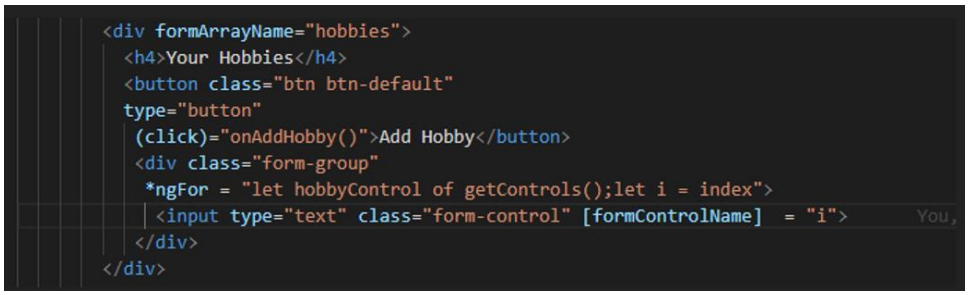
```
ngOnInit(){
    this.signupForm = new FormGroup({
      'userData': new FormGroup({
        'username' : new FormControl(null,Validators.required),
        'email' : new FormControl(null, [Validators.required,Validators.email]),
      }),
/*        'username' : new FormControl(null,Validators.required),
      'email' : new FormControl(null, [Validators.required,Validators.email]), */
      'gender' : new FormControl('male')
    });
  }      You, 2 minutes ago • Uncommitted changes
```

```
<div formGroupName="userData">      You, 2 minutes ago • Uncommitted changes
  <div class="form-group">
    <label for="username">Username</label>
    <input
      type="text"
      id="username"
      formControlName = "username"
      class="form-control">
    <span
      *ngIf="!signupForm.get('userData.username').valid && signupForm.get('userData.username').touched"
      class="help-block">Please enter a valid username
    </span>
  </div>
```

Form Array:

dynamically item add Garna kam lagcha ,suppose yo case ma hamilai euta hobby vanne button banayera add garepaxi

dynamically add hudai jancha .



```
      <div formArrayName="hobbies">
        <h4>Your Hobbies</h4>
        <button class="btn btn-default"
        type="button"
        (click)="onAddHobby()">Add Hobby</button>
        <div class="form-group"
        *ngFor = "let hobbyControl of getControls();let i = index">
          <input type="text" class="form-control" [formControlName]  = "i">      You,
        </div>
      </div>
```

```
  ngOnInit(){
      this.signupForm = new FormGroup({
        'userData': new FormGroup({
          'username' : new FormControl(null,Validators.required),
          'email' : new FormControl(null, [Validators.required,Validators.email]),
        }),
/*          'username' : new FormControl(null,Validators.required),
        'email' : new FormControl(null, [Validators.required,Validators.email]), */
        'gender' : new FormControl('male'),
        'hobbies': new FormArray([])
      });
  }
  onSubmit(){
    console.log(this.signupForm);
  }

  onAddHobby(){
    const control = new FormControl(null, Validators.required);
    (<FormArray>this.signupForm.get('hobbies')).push(control);
   /*  here the code inside small brackets lets TS know that it is a form array */

  }

  getControls(){
    return (<FormArray>this.signupForm.get('hobbies')).controls;
  }
}
```

creating custom validators:

suppose we don't want some username that we will allow the users to use

```
})
export class AppComponent implements OnInit {
  genders = ['male', 'female'];
  signupForm: FormGroup;
  forbiddenUsernames = ['Chris', 'Anna'];
```

```
  forbiddenNames(control: FormControl):{[s:string]:boolean}{
    if(this.forbiddenUsernames.indexOf(control.value) !== -1){
      return{'nameIsForbidden': true};

    }       You, 1 second ago • Uncommitted changes
    return null;
   /*  if validation is successful always pass null or nothing like above we should not pass
    like above with true */
  }
}
```

to implement this method

```
ngOnInit(){
    this.signupForm = new FormGroup({
      'userData': new FormGroup({
        'username' : new FormControl(null,[Validators.required,this.forbiddenNames.bind(this)]),
        'email' : new FormControl(null, [Validators.required,Validators.email]),
      }),
/*        'username' : new FormControl(null,Validators.required),
      'email' : new FormControl(null, [Validators.required,Validators.email]), */
      'gender' : new FormControl('male'),
      'hobbies': new FormArray([])
    });
```

using error codes:

using error codes provided by angular to show invalid username or username is required:

here the property nameIsforbidden and required are seen in the console by searching.

```
<span *ngIf ="signupForm.get('userData.username').errors['nameIsForbidden']">
  This name is invalid
</span>
<span *ngIf ="signupForm.get('userData.username').errors['required']">
  this field is required
</span>
```

creating custom async validators:

for checking the usernames validation, we need to check in the webserver unlike in our own local host,

that is asynchronous operation because response is not coming instantly it needs couples of sec.

so, we use async to return response

setting up async validator:

```
forbiddenEmails(control: FormControl): Promise<any> | Observable<any>{
  const promise = new Promise<any>((resolve,reject)=>{
    setTimeout(() =>{
      if(control.value === 'test@test.com'){
        resolve({'emailIsForbidden': true});
      }else{
        resolve(null);
      }
    },1500);
  });
  return promise;       You, 1 second ago • Uncommitted changes
 }
}
```

implementing it:

```
ngOnInit(){
    this.signupForm = new FormGroup({
      'userData': new FormGroup({
        'username' : new FormControl(null,[Validators.required,this.forbiddenNames.bind(this)]),
        'email' : new FormControl(null, [Validators.required,Validators.email], this.forbiddenEmails),
      }),
/*        'username' : new FormControl(null,Validators.required),
      'email' : new FormControl(null, [Validators.required,Validators.email]), */
```