To create a reactive form:

1. Import the following in app.module.ts

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

1. Import the following in app.component.ts

|  |
| --- |
| import { AbstractControl, FormArray, FormControl, FormGroup, ValidationErrors, Validators } from '@angular/forms'; |

1. in app.component.ts  
   signupform:FormGroup;
2. the idea is to initialize this formgroup with new members of formcontrol type similar to the HTML template . each member can hold its initial values and its validators and can be grouped into their sub groups . initialization is in ngoninit

|  |
| --- |
| this.signupform=new FormGroup({        'username':new FormControl(null),        'email':new FormControl(null),        gender:new FormControl('male')      }) |

1. Next step is to sync the code with the HTML template . this is done on 2 levels . the form tag and each control tag
   1. Form tag  
      [formGroup]="signupform"
   2. HTML Control  
      formControlName="username"  
      at this point you should be able to see the angular CSS class like ng-valid implemented
2. Next step is to prepare the submit method by adding onsubmit to the form tag . we don’t need to pass a form refrence like the template approach as we already created it in the type script model

|  |
| --- |
| <form [formGroup]="signupform" (ngSubmit)="onsubmit()"> |
| onsubmit()     {      console.log(this.signupform);     } |

1. Next step is to add validators . they add added in the form control definition as the second argument as an array  
   useremail':new FormControl(null,[Validators.required,Validators.email])
2. Next is adding the error divs in the template by using the get method

|  |
| --- |
| <div class="col-2">        <span class="badge bg-danger" \*ngIf="!signupform.get('username')?.valid && signupform.touched">Invalid UserName</span>      </div> |

1. CSS classes can be still used to add a red border like the following

|  |
| --- |
| input.ng-invalid.ng-dirty{      border: 1px solid red;  } |

1. How to add a dynamic array of controls in the form
   1. Control is called formarray . inside it will be form controls
   2. We change the initialization in ngoninit

|  |
| --- |
| this.signupform=new FormGroup({        'userData':new FormGroup({       'username':new FormControl(null,[Validators.required,this.forbidden.bind(this)]),        'useremail':new FormControl(null,[Validators.required,Validators.email],[this.forbiddenmailasync]),        'usergender':new FormControl('Female')      }),      'hobbies':new FormArray([])      }) |

* 1. We will create a function responsible for adding the new form control to the form array

|  |
| --- |
| onhobbies()     {       const control=new FormControl(null,Validators.required);       (<FormArray>  this.signupform.get('hobbies')).push(control);     } |

* 1. On the template leve we add a div that can add to the form array and loop on the array displaying the added control

|  |
| --- |
| <div class="row" formArrayName="hobbies">    <button class="btn btn-primary" type="button" (click)="onhobbies()">Add Hobby</button>    <hr>  <div \*ngFor="let c of get\_array().controls;let i=index">  <input type="text" class="form-control" [formControlName]="i">  <br>  </div>  </div> |

* 1. Note that I outsourced the get array as a function in the typescript

|  |
| --- |
| get\_array()     {      return this.signupform.get('hobbies') as FormArray     } |

* 1. Controls will be returned in the form submit and can be validated

1. Custom validators
   1. Custom validator is a function to be implemented for validation
   2. Its input is formcontrol
   3. Its output is an object with name value pair like {nameisforbidden:true} or null in case of valid
   4. Signature is : forbidden(C:FormControl):{[s:string]:boolean}|null
   5. Note -> {[s:string]:boolean} is the way to say we expect a name value pair

|  |
| --- |
| forbidden(C:FormControl):{[s:string]:boolean}|null     {        if(this.forbiddennames.includes(C.value) )        {          console.log(this.forbiddennames.indexOf(C.value)+' true');          return {'nameisforbidden':true}        }        else        {          console.log('false');          return null;        }       } |

* 1. To inject to a control go to the form definition and use it like

|  |
| --- |
| 'username':new FormControl(null,[Validators.required,this.forbidden.bind(this)]), |

* 1. Bind(this) because it will not be called from the class . it will be called from the template so this.forbiddennames inside the function will fail .
  2. To catch it and show a div or a span , check on  
     signupform.get('userData.username')!.errors!['nameisforbidden'] ===true
  3. Asynce validators like checking from a web service
     1. Return type is a promise or an observable
     2. The promise or observeable resolve with the name value pair object or none
     3. In the control added in the third parameter

|  |
| --- |
| forbiddenmailasync(C:AbstractControl):Promise<ValidationErrors|any>     {       const mypromise=new Promise((resolve,reject)=>{         setTimeout(() => {           if(C.value=='test@test.com')           {             console.log('inside true')             resolve({forbiddenmail:true});           }           else           {            console.log(C.value+' inside false')            resolve(null) ;           }         }, 5000);       });       return mypromise;     } |



|  |
| --- |
| 'useremail':new FormControl(null,[Validators.required,Validators.email],[this.forbiddenmailasync]) |

* 1. How to track status changes
     1. There is an observable to subscribe to for Value or Status change

|  |
| --- |
| this.signupform.valueChanges.subscribe(        (val=>{          console.log(val);        })      ) |



|  |
| --- |
| this.signupform.statusChanges.subscribe(        (status=>{          console.log(status);        })      ) |

* 1. You can use setvalue to set the whle form or patch value to update a part

|  |
| --- |
| setform(){       let myobj={         userData:{username:'Ahmed Sharaf',useremail:'a7med.sharaf@gmail.com',usergender:'Male'}         }       this.signupform.patchValue(myobj);     } |

* 1. Reset form 🡪 this.signupform.reset();