We have seen how to validate the data using inbuilt validators. We can write our own validations as well for reactive forms. For example, let us add one more field called email for which we will implement custom validation logic.

We will write a separate function in **registration-form.component.ts** for custom validation as shown below:

1. ...
2. export class RegistrationFormComponent implements OnInit {
3. ...
4. ngOnInit() {
5. this.registerForm = this.formBuilder.group({
6. firstName: ['', Validators.required],
7. lastName: ['', Validators.required],
8. address: this.formBuilder.group({
9. street: [],
10. zip: [],
11. city: []
12. }),
13. email:['',validateEmail]
14. });
15. }
16. }
17. function validateEmail(c: FormControl) {
18. let EMAIL\_REGEXP = /^([a-zA-Z0-9\_\-\.]+)@([a-zA-Z0-9\_\-\.]+)\.([a-zA-Z]{2,5})$/;
19. return EMAIL\_REGEXP.test(c.value) ? null : {
20. emailError: {
21. message: "Email is invalid"
22. }
23. };
24. }

The custom validation function accepts a form control object as a parameter. Through this form control object, we can access the value entered by the user.

**Line 18-26:** If the validation is **successful**, we return **null**. If the validation fails, we have to return an object. The object being returned will have a key and a value. The value can be another object in which can pass some data.

c.value gives us the value entered by the user for that particular form control element.

In the above code, we are returning an object with key emailError. This is the name that will be used in the view HTML to display messaged. The value of emailError is another object. This can also be used in the view to display the error message.

**Line 13:** Binds custom validator called validateEmail to the email field. We can also have multiple validators for the same formControl. For example, we can write

1. email: ['',[Validators.required, validateEmail] ]

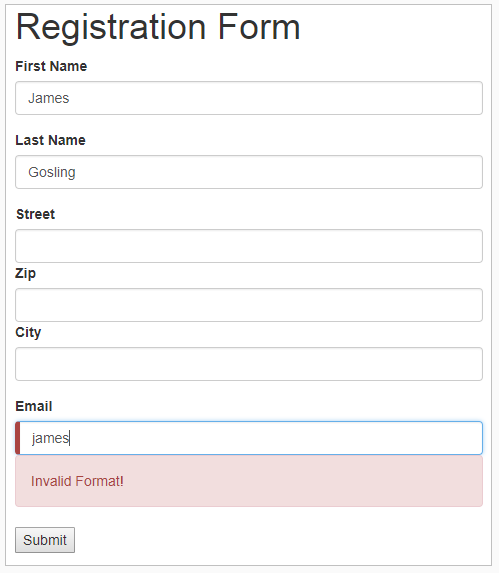
We can display the error message of the field in the HTML as shown below:

1. <p \*ngIf="registerForm.controls['email'].errors!['emailError']" class="alert alert-danger">{{registerForm.controls['email']!.errors!['emailError'].message}}</p> </div>

Now let us learn how to create custom validators with the help of a demo.

**Demo Steps:**

**Problem Statement:**Creating a custom validator for an email field in the employee registration form (reactive form) as shown below:



1. Write a separate function in **registration-form.component.ts** for custom validation as shown below.
2. import { Component, OnInit } from '@angular/core';
3. import { FormBuilder, FormGroup, FormControl, Validators } from '@angular/forms';
4. @Component({
5. selector: 'app-registration-form',
6. templateUrl: './registration-form.component.html',
7. styleUrls: ['./registration-form.component.css']
8. })
9. export class RegistrationFormComponent implements OnInit {
10. registerForm: FormGroup = this.formBuilder.group({
11. firstName: ['', Validators.required],
12. lastName: ['', Validators.required],
13. address: this.formBuilder.group({
14. street: [],
15. zip: [],
16. city: []
17. }),
18. email: ['', validateEmail]
19. });
20. submitted: boolean = false;
21. constructor(private formBuilder: FormBuilder) { }
22. ngOnInit() {
23. *// this.registerForm =*
24. }
25. }
26. function validateEmail(c: FormControl) {
27. let EMAIL\_REGEXP =
28. /^[\w.\_]+@[A-Za-z]+\.(com|co\.in|org)$/;
29. return EMAIL\_REGEXP.test(c.value) ? null : {
30. emailError: {
31. message: "Email is invalid"
32. }
33. };
34. }

1. Add HTML controls for email field in the **registration-form.component.html** file as shown below
2. <div class="container">
3. <h1>Registration Form</h1>
4. <form [formGroup]="registerForm">
5. <div class="form-group">
6. <label>First Name</label>
7. <input type="text" class="form-control" formControlName="firstName">
8. <p \*ngIf="registerForm.controls['firstName'].errors" class="alert alert-danger">This field is required!</p>
9. </div>
10. <div class="form-group">
11. <label>Last Name</label>
12. <input type="text" class="form-control" formControlName="lastName">
13. <p \*ngIf="registerForm.controls['lastName'].errors" class="alert alert-danger">This field is required!</p>
14. </div>
15. <div class="form-group">
16. <fieldset formGroupName="address">
17. <label>Street</label>
18. <input type="text" class="form-control" formControlName="street">
19. <label>Zip</label>
20. <input type="text" class="form-control" formControlName="zip">
21. <label>City</label>
22. <input type="text" class="form-control" formControlName="city">
23. </fieldset>
24. </div>
25. <div class="form-group">
26. <label>Email</label> <input type="text" class="form-control" formControlName="email">
27. <ng-container \*ngIf="registerForm.controls['email'].invalid">
28. <p \*ngIf="registerForm.controls['email'].errors!['emailError']" class="alert alert-danger">
29. {{registerForm.controls['email']!.errors!['emailError'].message}}</p>
30. </ng-container> </div>
31. </div>.
32. <button type="submit" (click)="submitted=true">Submit</button>
33. </form>
34. <br/>
35. <div [hidden]="!submitted">
36. <h3> Employee Details </h3>
37. <p>First Name: {{ registerForm.get('firstName')!.value }} </p>
38. <p> Last Name: {{ registerForm.get('lastName')!.value }} </p>
39. <p> Street: {{ registerForm.get('address.street')!.value }}</p>
40. <p> Zip: {{ registerForm.get('address.zip')!.value }} </p>
41. <p> City: {{ registerForm.get('address.city')!.value }}</p>
42. <p> Email: {{ registerForm.get('email')!.value }} </p>
43. </div>
44. </div>

1. Save the files and check the output in the browser