

Contents

Task 01. Template-Driven Form	2
Task 02. ReactiveComponent Class	6
Task 03. formGroup, formControlName directives	7
Task 04. setValue(), patchValue().....	11
Task 05. FormBuilder.....	12
Task 06. Setting Built-in Validators	13
Task 07. Adjusting Validation Rules in Runtime.....	15
Task 08. Custom Validator	17
Task 09. Custom Validator w/ Parameters.....	19
Task 10. Custom Validator Directive	20
Task 11. Custom Validator Directive w/ Parameters	21
Task 12. Async Custom Validator Directive.....	22
Task 13. Cross-Field Validation.....	25
Task 14. Adjusting Validation Rules	29
Task 15. Displaying Validation Messages	31
Task 16. Reactive Transformations	33
Task 17. Define the input element(s) to duplicate.....	34
Task 18. Define a FormGroup.....	35
Task 19. Refactor to Make Copies.....	36
Task 20. Create a FormArray	37
Task 21. Loop through the FormArray	38
Task 22. Duplicate the Input Elements.....	39

Task 01. Template-Driven Form

1. Создайте файл **app/models/user.ts**. Добавьте в него следующий фрагмент кода

```
export class User {  
  
  constructor(public firstName = '',  
               public lastName = '',  
               public email = '',  
               public sendProducts = false,  
               public addressType = 'home',  
               public street1?: string,  
               public street2?: string,  
               public country = '',  
               public city?: string,  
               public zip?: string) { }  
}
```

2. Внесите изменения в класс **SignupFormComponent**

```
import { Component, OnInit } from '@angular/core';  
import { NgForm } from '@angular/forms';  
import { User } from '../models/user';
```

3. Добавьте свойства

```
countries: Array<string> = ['Ukraine', 'Armenia', 'Belarus', 'Hungary', 'Kazakhstan',  
                             'Poland', 'Russia'];  
user: User = new User();
```

4. Добавьте метод

```
save(signupForm: NgForm) {  
  // Form model  
  console.log(signupForm.form);  
  // Form value  
  console.log(`Saved: ${JSON.stringify(signupForm.value)}`);  
}
```

5. Внесите изменения в шаблон компонента для формы

```
<form class="form-horizontal"  
      novalidate  
      #signupForm="ngForm"  
      (ngSubmit)="save(signupForm)">
```

6. Внесите изменения для поля ввода «First Name»

```
<div class="form-group"  
      [ngClass]="{'has-error': (firstNameVar.touched ||  
firstNameVar.dirty) && !firstNameVar.valid }">  
  
<input class="form-control"  
        id="firstNameId"  
        type="text"  
        placeholder="First Name (required)"  
        required  
        minlength="3"
```

```

        [(ngModel)]=user.firstName
        name="firstName"
        #firstNameVar="ngModel" />
<span class="help-block" *ngIf="(firstNameVar.touched || firstNameVar.dirty) &&
firstNameVar.errors">
    <span *ngIf="firstNameVar.errors.required">
        Please enter your first name.
    </span>
    <span *ngIf="firstNameVar.errors.minlength">
        The first name must be longer than 3 characters.
    </span>
</span>

```

7. Внесите изменения для поля ввода «Last Name»

```

<div class="form-group"
    [ngClass]='{\'has-error\': (lastNameVar.touched || lastNameVar.dirty)
    && !lastNameVar.valid }">

    <input class="form-control"
        id="lastNameId"
        type="text"
        placeholder="Last Name (required)"
        required
        maxlength="50"
        [(ngModel)]=user.lastName
        name="lastName"
        #lastNameVar="ngModel" />

    <span class="help-block" *ngIf="(lastNameVar.touched || lastNameVar.dirty) &&
lastNameVar.errors">
        <span *ngIf="lastNameVar.errors.required">
            Please enter your last name.
        </span>
    </span>
</div>

```

8. Внесите изменения для поля ввода «Email»

```

<div class="form-group"
    [ngClass]="{'has-error': (emailVar.touched || emailVar.dirty) && !emailVar.valid }">

    <input class="form-control"
        id="emailId"
        type="email"
        placeholder="Email (required)"
        required
        pattern="[a-z0-9._%+-]+@[a-z0-9.-]+"
        [(ngModel)]=user.email
        name="email"
        #emailVar="ngModel" />

    <span class="help-block" *ngIf="(emailVar.touched || emailVar.dirty) &&
emailVar.errors">
        <span *ngIf="emailVar.errors.required">
            Please enter your email address.
        </span>
    </span>
</div>

```

```

        <span *ngIf="emailVar.errors.pattern">
            Please enter a valid email address.
        </span>

        <!--
            This one does not work, because Angular doesn't support
            built-in email validator
        -->
        <span *ngIf="emailVar.errors.email">
            Please enter a valid email address.
        </span>
    </span>

```

9. Внесите изменения для чекбокса «Send me your products»

```

<input id="sendProductsId"
        type="checkbox"
        [(ngModel)]="user.sendProducts"
        name="sendProducts" >

```

10. Внесите изменения в элемент div

```

<div *ngIf="user.sendProducts">
    <div class="form-group" >
        <label class="col-md-2 control-label">Address Type</label>

```

11. Внесите изменения в элементы «Home», «Work», «Other»

```

<input type="radio" id="addressType1Id" value="home"
        [(ngModel)]="user.addressType"
        name="addressType">Home

<input type="radio" id="addressType1Id" value="work"
        [(ngModel)]="user.addressType"
        name="addressType">Work

<input type="radio" id="addressType1Id" value="other"
        [(ngModel)]="user.addressType"
        name="addressType">Other

```

12. Внесите изменения в элемент select

```

<select class="form-control"
        id="countryId"
        [(ngModel)]="user.country"
        name="country">
    <option value="">Select a Country...</option>
    <option *ngFor="let country of countries"
        value="{{country}}">{{country}}</option>
</select>

```

13. Внесите изменения в элементы «city», «zip code», «street address1», «street address2»

```

<input type="text"
      class="form-control"
      id="cityId"
      placeholder="City"
      [(ngModel)]="user.city"
      name="city">

<input type="number"
      class="form-control"
      id="zipId"
      placeholder="Zip Code"
      [(ngModel)]="user.zip"
      name="zip">

<input type="text"
      class="form-control"
      id="street1Id"
      placeholder="Street address"
      [(ngModel)]="user.street1"
      name="street1">

<input type="text"
      class="form-control"
      id="street2Id"
      placeholder="Street address (second line)"
      [(ngModel)]="user.street2"
      name="street2">

```

14. Внесите изменения в элемент **button**

```

<button class="btn btn-primary"
        type="submit"
        [disabled]="!signupForm.valid">
  Save
</button>

```

15. Добавьте в конце шаблона следующий фрагмент разметки

```

<br>Dirty: {{ signupForm.dirty }}
<br>Touched: {{ signupForm.touched }}
<br>Valid: {{ signupForm.valid }}
<br>Value: {{ signupForm.value | json }}

```

Task 02. ReactiveComponent Class

1. Создайте компонент **SignupReactiveFormComponent** в папке **app/reactive-forms**. (ng g с signup-reactive-form)
2. Скопируйте в него содержание и темплейт компонента **SignupFormComponent**, исправьте пути к темплейту и стилям, измените селектор на app-signup-reactive-form, измените название класса
3. Внесите изменения в **app.component.html**

```
<div class="container">
  <app-signup-form></app-signup-form>
  <app-signup-reactive-form></app-signup-reactive-form>
</div>
```

4. Внесите изменения в класс компонента **SignupReactiveFormComponent**

```
import { NgForm, FormGroup, FormControl } from '@angular/forms';

userForm: FormGroup;

private createForm() {
  this.userForm = new FormGroup({
    firstName: new FormControl(),
    lastName: new FormControl(),
    email: new FormControl(),
    sendProducts: new FormControl(true)
  });
}

ngOnInit() {
  this.createForm();
}

save() {
  // Form model
  console.log(this.userForm);
  console.log(signupForm.form);
  // Form value
  console.log(`Saved: ${JSON.stringify(this.userForm.value)}`);
  console.log(`Saved: ${JSON.stringify(signupForm.value)}`);
}
```

5. Внесите изменения в **app.module.ts**

```
import { ReactiveFormsModule } from '@angular/forms';

imports: [
  BrowserModule,
  FormsModule,
  ReactiveFormsModule,
  HttpClientModule
],
```

Task 03. formGroup, FormControlName directives

1. Внесите изменения в **signup-reactive-form.component.html**

```
<form class="form-horizontal"
  novalidate
  #signupForm="ngForm"
  (ngSubmit)="save(signupForm)"
  [formGroup]="userForm">

  <button class="btn btn-primary"
    type="submit"
    [disabled]="!signupForm.valid"
    [disabled]="!userForm.valid">
    Save
  </button>

  <br>Dirty: {{ signupForm.dirty }}
  <br>Touched: {{ signupForm.touched }}
  <br>Valid: {{ signupForm.valid }}
  <br>Value: {{ signupForm.value | json }}
  <br>Dirty: {{ userForm.dirty }}
  <br>Touched: {{ userForm.touched }}
  <br>Valid: {{ userForm.valid }}
  <br>Value: {{ userForm.value | json }}

  <input class="form-control"
    id="firstNameId"
    type="text"
    placeholder="First Name (required)"
    required
    minlength="3"
    [(ngModel)]=user.firstName
    name="firstName"
    #firstNameVar="ngModel"
    FormControlName="firstName"/>

  <input class="form-control"
    id="lastNameId"
    type="text"
    placeholder="Last Name (required)"
    required
    maxlength="50"
    [(ngModel)]=user.lastName
    name="lastName"
    #lastNameVar="ngModel"
    FormControlName="lastName"/>

  <input class="form-control"
    id="emailId"
    type="email"
    placeholder="Email (required)"
    required
```

```

        pattern="[a-z0-9._%+-]+@[a-z0-9.-]+"
        [(ngModel)]="user.email"
        name="email"
        #emailVar="ngModel"
        formControlName="email" />

<label>
  <input id="sendProductsId"
    type="checkbox"
    [(ngModel)]="user.sendProducts"
    name="sendProducts"
    formControlName="sendProducts">
    Send me your products
</label>

```

2. Закомментируйте часть разметки, которая начинается с элемента `<div *ngIf="user.sendProducts">` и до закрывающего тега
3. Внесите изменения в атрибут `[ngClass]`

Для firstName

```

[ngClass]="{'has-error': (firstNameVar.touched || firstNameVar.dirty) &&
!firstNameVar.valid }"

[ngClass]="{'has-error': (userForm.get('firstName').touched ||
userForm.get('firstName').dirty) && !userForm.get('firstName').valid }"

```

Для lastName

```

[ngClass]="{'has-error': (lastNameVar.touched || lastNameVar.dirty) &&
!lastNameVar.valid }"

[ngClass]="{'has-error': (userForm.get('lastName').touched ||
userForm.get('lastName').dirty) && !userForm.get('lastName').valid }"

```

Для email

```

[ngClass]="{'has-error': (emailVar.touched || emailVar.dirty) && !emailVar.valid }"

[ngClass]="{'has-error': (userForm.get('email').touched || userForm.get('email').dirty)
&& !userForm.get('email').valid }"

```

4. Внесите изменения в элемент `span` – блок вывода сообщений об ошибках валидации

Для firstName

```

<span class="help-block" *ngIf="(firstNameVar.touched || firstNameVar.dirty) &&
firstNameVar.errors">
  <span *ngIf="firstNameVar.errors.required">
    Please enter your first name.
  </span>
  <span *ngIf="firstNameVar.errors.minlength">
    The first name must be longer than 3 characters.
  </span>
</span>

<span class="help-block" *ngIf="(userForm.get('firstName').touched ||
userForm.get('firstName').dirty) && userForm.get('firstName').errors">

```



```

    <span *ngIf="userForm.get('firstName').errors.required">
      Please enter your first name.
    </span>
    <span *ngIf="userForm.get('firstName').errors.minlength">
      The first name must be longer than 3 characters.
    </span>
  </span>

```

Для lastName

```

<span class="help-block" *ngIf="(lastNameVar.touched || lastNameVar.dirty) && lastNameVar.errors">
  <span *ngIf="lastNameVar.errors.required">
    Please enter your last name.
  </span>
</span>
<span class="help-block" *ngIf="(userForm.get('lastName').touched || userForm.get('lastName').dirty) &&
userForm.get('lastName').errors">
  <span *ngIf="userForm.get('lastName').errors.required">
    Please enter your last name.
  </span>
</span>

```

Для email

```

<span class="help-block" *ngIf="(emailVar.touched || emailVar.dirty) &&
emailVar.errors">
  <span *ngIf="emailVar.errors.required">
    Please enter your email address.
  </span>
  <span *ngIf="emailVar.errors.pattern">
    Please enter a valid email address.
  </span>

  <!--
  This one does not work, because Angular doesn't support
  built-in email, phone, date validator
  -->
  <span *ngIf="emailVar.errors.email">
    Please enter a valid email address.
  </span>
</span>
<span class="help-block" *ngIf="(userForm.get('email').touched ||
userForm.get('email').dirty) && userForm.get('email').errors">
  <span *ngIf="userForm.get('email').errors.required">
    Please enter your email address.
  </span>
  <span *ngIf="userForm.get('email').errors.pattern">
    Please enter a valid email address.
  </span>

  <!--
  This one does not work, because Angular doesn't support
  built-in email, phone, date validator
  -->
  <span *ngIf="userForm.get('email').errors.email">

```

Please enter a valid email address.

Task 04. setValue(), patchValue()

1. Внесите изменения в класс компонента **SignupReactiveFormComponent**

```
private setFormValues() {
  this.userForm.setValue({
    firstName: 'Vitaliy',
    lastName: 'Zhyrytskyy',
    email: 'vitaliy_zhyrytskyy@ukr.net',
    sendProducts: false
  });
}

private patchFormValues() {
  this.userForm.patchValue({
    firstName: 'Vitaliy',
    lastName: 'Zhyrytskyy'
  });
}

ngOnInit() {
  this.createForm();
  this.setFormValues();
  // this.patchFormValues();
}
```

2. Посмотрите результат, закомментируйте вызов метода **this.setFormValues();** и раскомментируйте вызов метода **// this.patchFormValues();** Посмотрите результат.

Task 05. FormBuilder

1. Внесите изменения в класс компонента **SignupReactiveFormComponent**

```
import { FormGroup, FormControl, FormBuilder } from '@angular/forms';

constructor(
  private fb: FormBuilder
) { }

private buildForm() {
  this.userForm = this.fb.group({
    firstName: '',
    lastName: {value: 'Zhyrytskyy', disabled: true},
    email: [''],
    sendProducts: true
  });
}

ngOnInit() {
  this.createForm();
  this.buildForm();
  this.setFormValues();
  this.patchFormValues();
}
```

Task 06. Setting Built-in Validators

1. Внесите изменения в компонент **SignupReactiveFormComponent**

```
import { FormGroup, FormControl, FormBuilder, Validators } from '@angular/forms';
```

2. Внесите изменения в метод **buildForm** компонента **SignupReactiveFormComponent**

```
private buildForm() {
  this.userForm = this.fb.group({
    firstName: '',
    firstName: ['', [Validators.required, Validators.minLength(3)]],
    lastName: {value: 'Zhyrytskyi', disabled: true},
    email: ['',],
    sendProducts: true
  });
}
```

3. Внесите изменения в разметку компонента **SignupReactiveFormComponent**

```
<input class="form-control"
  id="firstNameId"
  type="text"
  placeholder="First Name (required)"
  required
  minlength="3"
  formControlName="firstName"/>
```

4. Проверьте работает ли валидация поля First Name
5. Внесите изменения в метод **buildForm** компонента **SignupReactiveFormComponent**

```
private buildForm() {
  this.userForm = this.fb.group({
    firstName: ['', [Validators.required, Validators.minLength(3)]],
    lastName: {value: 'Zhyrytskyi', disabled: true},
    lastName: [
      { value: 'Zhyrytskyi', disabled: false },
      [Validators.required, Validators.maxLength(50)]
    ],
    email: ['',],
    email: [
      '',
      [Validators.required, Validators.pattern('[a-z0-9._%+-]+@[a-z0-9.-]+')]
    ],
    sendProducts: true
  });
}
```

6. Внесите изменения в разметку компонента **SignupReactiveFormComponent**

```
<input class="form-control"
  id="lastNameId"
  type="text"
  placeholder="Last Name (required)"
  required
  maxlength="50"
  formControlName="lastName"/>
```

```
<input class="form-control"
      id="emailId"
      type="email"
      placeholder="Email (required)"
      required
      pattern="[a-z0-9._%+-]+@[a-z0-9.-]+"
      formControlName="email" />
```

7. Проверьте валидацию полей формы: FirtsName, LastName, Email

Task 07. Adjusting Validation Rules in Runtime

1. Внесите изменения в метод **buildForm** компонента **SignupReactiveFormComponent**

```
private buildForm() {
  this.userForm = this.fb.group({
    firstName: ['', [Validators.required, Validators.minLength(3)]],
    lastName: [
      { value: 'Zhyrytskyy', disabled: false },
      [Validators.required, Validators.maxLength(50)]
    ],
    email: [
      '',
      [Validators.required, Validators.pattern('[a-z0-9._%+-]+@[a-z0-9.-]+')]
    ],
    phone: '',
    notification: 'email',
    sendProducts: true
  });
}
```

2. Внесите изменения в разметку компонента **SignupReactiveFormComponent** – после поля email, добавьте еще два элемента управления

```
<div class="form-group"
  [ngClass]="{'has-error': !userForm.get('phone').valid }">
  <label class="col-md-2 control-label"
    for="phoneId">Phone</label>

  <div class="col-md-8">
    <input class="form-control"
      id="phoneId"
      type="tel"
      placeholder="Phone"
      formControlName="phone" />
    <span class="help-block" *ngIf="userForm.get('phone').errors">
      <span *ngIf="userForm.get('phone').errors.required">
        Please enter your phone number.
      </span>
    </span>
  </div>
</div>

<div class="form-group">
  <label class="col-md-2 control-label">Send Notifications</label>
  <div class="col-md-8">
    <label class="radio-online">
      <input type="radio"
        value="email"
        formControlName="notification">Email
    </label>
    <label class="radio-online">
      <input type="radio"
        value="text"
        formControlName="notification">Text
    </label>
  </div>
</div>
```

</div>

3. Добавьте метод в класс компонента **SignupReactiveFormComponent**

```
setNotification(notifyVia: string) {  
  const phoneControl = this.userForm.get('phone');  
  const emailControl = this.userForm.get('email');  
  
  if (notifyVia === 'text') {  
    phoneControl.setValidators(Validators.required);  
    emailControl.clearValidators();  
  }  
  else {  
    emailControl.setValidators( [Validators.required, Validators.pattern('[a-z0-9._%+-]+@[a-z0-9.-]+')]);  
    phoneControl.clearValidators();  
  }  
  phoneControl.updateValueAndValidity();  
  emailControl.updateValueAndValidity();  
}
```

4. Внесите изменения в разметку компонента **SignupReactiveFormComponent**

```
<label class="radio-online">  
  <input type="radio"  
    value="email"  
    formControlName="notification"  
    (click)="setNotification('email')">Email  
</label>  
<label class="radio-online">  
  <input type="radio"  
    value="text"  
    formControlName="notification"  
    (click)="setNotification('text')">Text  
</label>
```


Task 08. Custom Validator

1. Внесите изменения в метод **buildForm** компонента **SignupReactiveFormComponent**

```
private buildForm() {
  this.userForm = this.fb.group({
    firstName: ['', [Validators.required, Validators.minLength(3)]],
    lastName: [
      { value: 'Zhyrytsky', disabled: false },
      [Validators.required, Validators.maxLength(50)]
    ],
    email: [
      '',
      [Validators.required, Validators.pattern('[a-z0-9._%+-]+@[a-z0-9.-]+')]
    ],
    phone: '',
    notification: 'email',
    serviceLevel: '',
    sendProducts: true
  });
}
```

2. Внесите изменения в разметку компонента **SignupReactiveFormComponent**. После блока `div` для элемента «Send Notifications» добавьте следующий фрагмент разметки

```
<div class="form-group"
  [ngClass]="{'has-error': (userForm.get('serviceLevel').touched ||
userForm.get('serviceLevel').dirty) && !userForm.get('serviceLevel').valid }">
  <label class="col-md-2 control-label"
    for="serviceLevelId">Service Level</label>

    <div class="col-md-8">
      <input class="form-control"
        id="serviceLevelId"
        type="number"
        formControlName="serviceLevel" />
      <span class="help-block"
        *ngIf="(userForm.get('serviceLevel').touched || userForm.get('serviceLevel').dirty) &&
userForm.get('serviceLevel').errors">
        <span
          *ngIf="userForm.get('serviceLevel').errors.serviceLevel">
            Please enter correct number from 1 to 5.
          </span>
        </span>
      </div>
    </div>
</div>
```

3. Создайте файл **app/validators/service-level.validator.ts**. Добавьте в него следующий фрагмент кода

```
import { AbstractControl } from '@angular/forms';

export class CustomValidators {
  static serviceLevel(c: AbstractControl): { [key: string]: boolean } | null {
    if (c.value !== undefined && (Number.isNaN(c.value) || c.value < 1 || c.value > 5)) {
```

```

    return {
      'serviceLevel': true
    };
  }
  return null;
}
}

```

4. Создайте файл **app/validators/index.ts**. Добавьте в него следующий фрагмент кода

```
export * from './service-level.validator';
```

5. Внесите изменения в компонент **SignupReactiveFormComponent**

```
import { CustomValidators } from './../../validators';
```

6. Внесите изменения в метод **buildForm** компонента **SignupReactiveFormComponent**

```

phone: '',
notification: 'email',
serviceLevel: '',
serviceLevel: ['', CustomValidators.serviceLevel],
sendProducts: true

```

Task 09. Custom Validator w/ Parameters

1. Внесите изменения в файл **service-level.validator.ts**

```
import { AbstractControl, ValidatorFn } from '@angular/forms';
```

2. Внесите изменения в класс **CustomValidators**, добавьте следующий фрагмент кода

```
static serviceLevelRange(min: number, max: number): ValidatorFn {  
  return (c: AbstractControl): { [key: string]: boolean } | null => {  
    if (c.value !== undefined && (Number.isNaN(c.value) || c.value < min || c.value >  
max)) {  
      return {  
        'serviceLevel': true  
      };  
    }  
    return null;  
  }  
}
```

3. Внесите изменения в метод **buildForm** компонента **SignupReactiveFormComponent**

```
serviceLevel: ['', CustomValidators.serviceLevel],  
serviceLevel: ['', CustomValidators.serviceLevelRange(1,3)],
```

4. Внесите изменения в разметку компонента **SignupReactiveFormComponent**

```
Please enter correct number from 1 to 5.  
Please enter correct number from 1 to 3.
```

Task 10. Custom Validator Directive

1. Создайте файл **app/validators/service-level.directive.ts** используя следующий фрагмент кода

```
import { Directive } from '@angular/core';
import { Validator, AbstractControl, NG_VALIDATORS } from '@angular/forms';

@Directive({
  selector: '[appServiceLevelValidator]',
  providers: [{
    provide: NG_VALIDATORS,
    useExisting: ServiceLevelDirective,
    multi: true
  }]
})
export class ServiceLevelDirective implements Validator {

  validate(c: AbstractControl): { [key: string]: boolean } | null {
    if (c.value !== undefined && (Number.isNaN(c.value) || c.value < 1 || c.value > 3))
    {
      return {
        'serviceLevel': true
      };
    }
    return null;
  }
}
```

2. Внесите изменения в файл **app/validators/index.ts**

```
export * from './service-level.validator';
export * from './service-level.directive';
```

5. Внесите изменения в **app.module.ts** используя следующий фрагмент кода

```
import { ServiceLevelDirective } from './validators/service-level.directive';

declarations: [
  AppComponent,
  SignupFormComponent,
  SignupReactiveFormComponent,
  ServiceLevelDirective
],
```

6. Внесите изменения в **buildForm** компонента **SignupReactiveFormComponent**

```
serviceLevel: ['', CustomValidators.serviceLevelRange(1,3)],
serviceLevel: [''],
```

7. Внесите изменения в темплейт компонента **SignupReactiveFormComponent**

```
<input class="form-control"
  id="serviceLevelId"
  type="number"
  formControlName="serviceLevel"
  appServiceLevelValidator />
```

Task 11. Custom Validator Directive w/ Parameters

1. Внесите изменения в директиву **ServiceLevelDirective**

```
import { Directive, Input } from '@angular/core';

export class ServiceLevelDirective implements Validator {
  @Input() rMin = 1;
  @Input() rMax = 3;

  validate(c: AbstractControl): { [key: string]: boolean } | null {
    console.log(c.value);
    if (c.value !== undefined && (Number.isNaN(c.value) || c.value < 1this.rMin ||
c.value > 3this.rMax)) {
      return {
        'serviceLevel': true
      };
    }
    return null;
  }
}
```

2. Внесите изменения в разметку компонента **SignupReactiveFormComponent**

```
<input class="form-control"
  id="serviceLevelId"
  type="number"
  formControlName="serviceLevel"
  appServiceLevelValidator rMin="1" rMax="4" #serviceLevelVar />
```

```
Please enter correct number from 1 to 3.
Please enter correct number from
{{serviceLevelVar.getAttribute('rMin') || 1}} to
{{serviceLevelVar.getAttribute('rMax') || 3}}.
```

Task 12. Async Custom Validator Directive

1. Создайте директиву в файле **validators/async-email-validator.directive.ts** используя следующий фрагмент кода

```
import { Directive, forwardRef } from '@angular/core';
import { NG_ASYNC_VALIDATORS, Validator, AbstractControl } from '@angular/forms';
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/debounceTime';
import 'rxjs/add/operator/distinctUntilChanged';
import 'rxjs/add/operator/first';

@Directive({
  selector: '[asyncEmailValidator][formControlName], [asyncEmailValidator][ngModel]',
  providers: [
    {
      provide: NG_ASYNC_VALIDATORS,
      useExisting: forwardRef(() => AsyncEmailValidatorDirective), multi: true
    }
  ]
})
export default class AsyncEmailValidatorDirective implements Validator {
  validate(c: AbstractControl): Promise<{ [key: string]: any}>|Observable < {[key: string]: any}> {
    return this.validateEmailPromise(c.value);
    // return
    this.validateEmailObservable(c.value).debounceTime(1000).distinctUntilChanged().first();
  }

  private validateEmailPromise(email: string) {
    return new Promise(resolve => {
      setTimeout(() => {
        if (email === 'existsemail@example.com') {
          resolve({
            asyncEmailInvalid: true
          })
        } else {
          resolve(null);
        }
      }, 2000);
    });
  }

  private validateEmailObservable(email: string) {
    return new Observable(observer => {
      if (email === 'existsemail@example.com') {
        observer.next({asyncEmailInvalid: true});
      } else {
        observer.next(null);
      }
    });
  }
}
```

2. Создайте модуль **validators/validators.module.ts** используя следующий фрагмент кода

```
import { NgModule } from '@angular/core';
```

```

import { CommonModule } from '@angular/common';

import { ServiceLevelDirective } from './service-level.directive';
import AsyncEmailValidatorDirective from './async-email-validator.directive';

@NgModule({
  imports: [
    CommonModule
  ],
  declarations: [
    AsyncEmailValidatorDirective,
    ServiceLevelDirective
  ],
  exports: [
    AsyncEmailValidatorDirective,
    ServiceLevelDirective
  ]
})
export class ValidatorsModule { }

```

3. Внесите изменения в app.module.ts используя следующий фрагмент кода

```

import { ValidatorsModule } from './validators/validators.module';
import { ServiceLevelDirective } from './validators/service-level.directive';

declarations: [
  AppComponent,
  SignupFormComponent,
  SignupReactiveFormComponent,
  ServiceLevelDirective
],

imports: [
  BrowserModule,
  FormsModule,
  ReactiveFormsModule,
  HttpClientModule,
  ValidatorsModule
],

```

4. Внесите изменения в шаблон компонента **SignupReactiveFormComponent** используя следующий фрагмент разметки

```

<input class="form-control"
  id="emailId"
  type="email"
  placeholder="Email (required)"
  FormControlName="email"
  asyncEmailValidator />

<span *ngIf="userForm.get('email').errors.asyncEmailInvalid">
  This email already exists. Please enter other email address.
</span>

```

5. Введите в поле email значение **existsemail@example.com** и убедитесь, что валидация работает.

6. Внесите изменения в директиву валидации в файле **validators/async-email-validator.directive.ts** и убедитесь, что валидация работает.

```
validate(c: AbstractControl): Promise<{ [key: string]: any}>|Observable < {[key:
string]: any}> {
    // return this.validateEmailPromise(c.value);
    // return
    this.validateEmailObservable(c.value).debounceTime(1000).distinctUntilChanged().f
irst();
}
```


Task 13. Cross-Field Validation

1. Внесите изменения в разметку компонента **SignupReactiveFormComponent**. Добавьте следующий фрагмент разметки после блока email.

```
<div class="form-group"
  [ngClass]="{'has-error': (userForm.get('confirmEmail').touched ||
    userForm.get('confirmEmail').dirty) &&
    !userForm.get('confirmEmail').valid }">
  <label class="col-md-2 control-label"
    for="confirmEmailId">Confirm Email</label>

  <div class="col-md-8">
    <input class="form-control"
      id="confirmEmailId"
      type="email"
      placeholder="Confirm Email (required)"
      formControlName="confirmEmail" />
    <span class="help-block" *ngIf="(userForm.get('confirmEmail').touched ||
      userForm.get('confirmEmail').dirty) &&
      userForm.get('confirmEmail').errors">
      <span *ngIf="userForm.get('confirmEmail').errors.required">
        Please confirm your email address.
      </span>
    </span>
  </div>
</div>
```

2. Внесите изменения в **buildForm** компонента **SignupReactiveFormComponent**. Проверьте работает ли валидация добавленная к полю confirmEmail.

```
email: ['',
  [Validators.required, Validators.pattern('[a-z0-9._%+-]+@[a-z0-9.-]+')]
],
confirmEmail: ['', Validators.required],
```

3. Внесите изменения в разметку компонента **SignupReactiveFormComponent**. Перед блоком email добавьте следующий фрагмент разметки и перенесите в него блоки для email и confirmEmail

```
<div formGroupName="emailGroup">
  <!-- Сюда перенести блоки для email и confirmEmail -->
</div>
```

4. Внесите изменения в **buildForm** компонента **SignupReactiveFormComponent**.

```
email: [
  '',
  [Validators.required, Validators.pattern('[a-z0-9._%+-]+@[a-z0-9.-]+')]
],
```

```

],
confirmEmail: ['', Validators.required],
emailGroup: this.fb.group({
  email: ['',
    [Validators.required, Validators.pattern('[a-z0-9._%+-]+@[a-z0-9.-]+')]]
  ],
  confirmEmail: ['', Validators.required],
}),

```

5. Внесите изменения в разметку компонента **SignupReactiveFormComponent**

```

<div class="form-group"
  [ngClass]="{'has-error': (userForm.
get('email').controls.emailGroup.controls.email.touched ||
  userForm.get('emailGroup.email').dirty) &&
  !userForm.get('emailGroup.email').valid }">

  <label class="col-md-2 control-label"
    for="emailId">Email</label>

    <div class="col-md-8">
      <input class="form-control"
        id="emailId"
        type="email"
        placeholder="Email (required)"
        formControlName="email" />
      <span class="help-block"
*ngIf="(userForm.get('emailGroup.email').touched ||
  userForm.get('emailGroup.email').dirty) &&
  userForm.get('emailGroup.email').errors">
        <span
*ngIf="userForm.get('emailGroup.email').errors.required">
          Please enter your email address.
        </span>
        <span
*ngIf="userForm.get('emailGroup.email').errors.pattern">
          Please enter a valid email address.
        </span>

        <!--
          This one does not work, because Angular doesn't support
          built-in email, phone, date validator
        -->
        <!--<span *ngIf="userForm.get('email').errors.email">
          Please enter a valid email address.
        </span>-->
      </span>
    </div>
  </div>
</div>

<div class="form-group"
  [ngClass]="{'has-error':
(userForm.get('emailGroup.confirmEmail').touched ||
  userForm.get('emailGroup.confirmEmail').dirty) &&
  !userForm.get('emailGroup.confirmEmail').valid }">

```

```

        <label class="col-md-2 control-label"
            for="confirmEmailId">Confirm Email</label>

        <div class="col-md-8">
            <input class="form-control"
                id="confirmEmailId"
                type="email"
                placeholder="Confirm Email (required)"
                formControlName="confirmEmail" />
            <span class="help-block"
                *ngIf="(userForm.get('emailGroup.confirmEmail').touched ||
                    userForm.get('emailGroup.confirmEmail').dirty) &&
                    userForm.get('emailGroup.confirmEmail').errors">
                <span
                    *ngIf="userForm.get('emailGroup.confirmEmail').errors.required">
                        Please confirm your email address.
                    </span>
                </span>
            </div>
        </div>
    </div>

```

6. Переименуйте файл **service-level.validator.ts** в **custom.validators.ts**

7. Внесите изменения в файл **validators/index.ts**

```

export * from './service-level.validator';
export * from './custom.validators';

```

8. Внесите изменения в файл **custom.validators.ts**. Добавьте следующий фрагмент кода

```

static emailMatcher(c: AbstractControl): { [key: string]: boolean } | null {
    const emailControl = c.get('email');
    const emailConfirmControl = c.get('confirmEmail');

    if (emailControl.pristine || emailConfirmControl.pristine) {
        return null;
    }

    if (emailControl.value === emailConfirmControl.value) {
        return null;
    }

    return { 'emailMatch': true };
}

```

9. Внесите изменения в метод **buildForm** компонента **SignupReactiveFormComponent**

```

emailGroup: this.fb.group({
    email: ['',
        [Validators.required, Validators.pattern('[a-z0-9._%+-]+@[a-z0-9.-]+')]],
    confirmEmail: ['', Validators.required],
}, {validator: CustomValidators.emailMatcher}),

```

10. Внесите изменения в разметку компонента **SignupReactiveFormComponent**

```

<div formGroupName="emailGroup"
    [ngClass]="{'has-error': userForm.get('emailGroup').errors}">

    <span class="help-block" *ngIf="(userForm.get('emailGroup.confirmEmail').touched ||
        userForm.get('emailGroup.confirmEmail').dirty) &&
        userForm.get('emailGroup.confirmEmail').errors">
        <span *ngIf="userForm.get('emailGroup.confirmEmail').errors.required">
            Please confirm your email address.
        </span>
    </span>
</div>
<div formGroupName="emailGroup" *ngIf="(userForm.get('emailGroup.confirmEmail').touched ||
    userForm.get('emailGroup.confirmEmail').dirty) &&
    (userForm.get('emailGroup.confirmEmail').errors ||
    userForm.get('emailGroup').errors) ">
    <span *ngIf="userForm.get('emailGroup.confirmEmail').errors?.required">
        Please confirm your email address.
    </span>
    <span *ngIf="userForm.get('emailGroup').errors?.emailMatch">
        The confirmation does not match the email address.
    </span>
</div>

```

11. Внесите изменения в метод setNotification

```
const emailControl = this.userForm.get('emailGroup.email');
```

Task 14. Adjusting Validation Rules

1. Внесите изменения в компонент **SignupReactiveFormComponent** используя следующий фрагмент кода

```
// 1
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Subscription } from 'rxjs/Subscription';

// 2
export class SignupReactiveFormComponent implements OnInit, OnDestroy
```

2. Добавьте новое приватное свойство для компонента **SignupReactiveFormComponent** используя следующий фрагмент кода

```
private sub: Subscription[] = [];
```

3. Создайте приватный метод **watchValueChanges** используя следующий фрагмент кода

```
private watchValueChanges() {
  const sub = this.userForm.get('notification').valueChanges
    .subscribe(value => console.log(value));
  this.sub.push(sub);
}
```

4. Внесите изменения в метод **ngOnInit** используя следующий фрагмент кода

```
ngOnInit() {
  this.buildForm();
  this.watchValueChanges();
}
```

5. Добавьте метод **ngOnDestroy**

```
ngOnDestroy() {
  this.sub.forEach(sub => sub.unsubscribe());
}
```

6. Удалите обработчик **setNotification** события click.

```
<div class="form-group">
  <label class="col-md-2 control-label">Send Notifications</label>
  <div class="col-md-8">
    <label class="radio-online">
      <input type="radio"
        value="email"
        formControlName="notification"
        (click)="setNotification('email')">Email
    </label>
    <label class="radio-online">
      <input type="radio"
        value="text"
        formControlName="notification"
        (click)="setNotification('text')">Text
    </label>
  </div>
</div>
```

7. Внесите изменения в метод **watchValueChanges**

```
private watchValueChanges() {  
    const sub = this.userForm.get('notification').valueChanges  
        .subscribe(value => console.log(value));  
        .subscribe(value => this.setNotification(value));  
    this.sub.push(sub);  
}
```

8. Внесите изменения в метод **setNotification**

```
private setNotification(notifyVia: string) {
```

Task 15. Displaying Validation Messages

1. Внесите изменения в класс компонента **SignupReactiveFormComponent** используя следующий фрагмент кода

```
// 1
import { FormGroup, FormControl, FormBuilder, Validators, AbstractControl } from
 '@angular/forms';

// 2
userForm: FormGroup;
emailMessage: string;

// 3
private sub: Subscription[] = [];
private validationMessages = {
  required: 'Please enter your email address.',
  pattern: 'Please enter a valid email address.'
};
```

2. Добавьте метод **setMessage** используя следующий фрагмент кода

```
private setMessage(c: AbstractControl) {
  this.emailMessage = '';
  if ((c.touched || c.dirty) && c.errors) {
    this.emailMessage = Object
      .keys(c.errors)
      .map(key => this.validationMessages[key])
      .join(' ');
  }
}
```

3. Внесите изменения в метод **watchValueChanges**

```
private watchValueChanges() {
  const sub1 = this.userForm.get('notification').valueChanges
    .subscribe(value => this.setNotification(value));
  this.sub.push(sub1);

  const emailControl = this.userForm.get('emailGroup.email');
  const sub2 = emailControl.valueChanges
    .subscribe(value => this.setMessage(emailControl));
  this.sub.push(sub2);
}
```

4. Внесите изменения в темплейт компонента

```
<div class="form-group"
      [ngClass]="{'has-error':
        (userForm.controls.emailGroup.controls.email.touched ||
        userForm.get('emailGroup.email').dirty)
        &&
        !userForm.get('emailGroup.email').valid
      }">
  [ngClass]="{'has-error': emailMessage}">
```

```

<label class="col-md-2 control-label"
  for="emailId">Email</label>

<div class="col-md-8">
  <input class="form-control"
    id="emailId"
    type="email"
    placeholder="Email (required)"
    formControlName="email" />
    <span class="help-block"
*ngIf="(userForm.get('emailGroup.email').touched ||
userForm.get('emailGroup.email').dirty) &&
userForm.get('emailGroup.email').errors">
  <span
*ngIf="userForm.get('emailGroup.email').errors.required">
    Please enter your email address.
  </span>
  <span
*ngIf="userForm.get('emailGroup.email').errors.pattern">
    Please enter a valid email address.
  </span>

  <!--
    This one does not work, because Angular doesn't support
    built-in email, phone, date validator
  -->
  <!--<span *ngIf="userForm.get('email').errors.email">
    Please enter a valid email address.
  </span>-->
  </span>
<span class="help-block" *ngIf="emailMessage">
  {{emailMessage}}
</span>
</div>
</div>

```


Task 16. Reactive Transformations

1. Внесите изменения в компонент **SignupReactiveFormComponent**

```
import { Subscription } from 'rxjs/Subscription';  
import 'rxjs/add/operator/debounceTime';
```

2. Внесите изменения в метод **watchValueChanges**

```
private watchValueChanges() {  
  const sub1 = this.userForm.get('notification').valueChanges  
    .subscribe(value => this.setNotification(value));  
  this.sub.push(sub1);  
  
  const emailControl = this.userForm.get('emailGroup.email');  
  const sub2 = emailControl.valueChanges  
    .debounceTime(1000)  
    .subscribe(value => this.setMessage(emailControl));  
  this.sub.push(sub2);  
}
```

Task 17. Define the input element(s) to duplicate

1. Внесите изменения в метод **buildForm** используя следующий фрагмент разметки

```
sendProducts: true,  
addressType: 'home',  
country: '',  
city: '',  
zip: '',  
street1: '',  
street2: ''
```

2. Раскомментируйте фрагмент разметки компонента **SignupReactiveFormComponent**, который начинается с

```
<!--<div *ngIf="user.sendProducts">
```

3. Внесите изменение в выражение для директивы ***ngIf**

```
<div *ngIf="user.sendProducts">  
<div *ngIf="userForm.get('sendProducts').value">
```

4. Внесите изменения для полей **Home, Work, Other**

```
[(ngModel)]= "user.addressType"  
name="addressType"  
formControlName="addressType"
```

5. Внесите изменения для полей **Country, City, Zip Code, Street Address 1, Street Address 2**

```
// Country  
[(ngModel)]= "user.country"  
name="country"  
formControlName="country"
```

```
// City  
[(ngModel)]= "user.city"  
name="city"  
formControlName="city"
```

```
// Zip Code  
[(ngModel)]= "user.zip"  
name="zip"  
formControlName="zip"
```

```
// Street Address 1  
[(ngModel)]= "user.street1"  
name="street1"  
formControlName="street1"
```

```
// Street Address 2  
[(ngModel)]= "user.street2"  
name="street2"  
formControlName="street2"
```

Task 18. Define a FormGroup

1. Внесите изменения в метод **buildForm**, используя следующий фрагмент кода

```
addressType: 'home',  
country: '',  
city: '',  
zip: '',  
street1: '',  
street2: ''  
addresses: this.fb.group({  
  addressType: 'home',  
  country: '',  
  city: '',  
  zip: '',  
  street1: '',  
  street2: ''  
})
```

2. Обверните 4 div блока, которые находятся в блоке
<div *ngIf="userForm.get('sendProducts').value"> и содержат поля ввода для адреса в
следующую конструкцию

```
<div formGroupName="addresses">...</div>
```

Task 19. Refactor to Make Copies

1. Добавьте приватный метод **buildAddress** в класс компонента `SignupReactiveFormComponent` используя следующий фрагмент кода

```
private buildAddress(): FormGroup {  
    return this.fb.group({  
        addressType: 'home',  
        country: '',  
        city: '',  
        zip: '',  
        street1: '',  
        street2: ''  
    });  
}
```

2. Внесите изменения в метод **buildForm** используя следующий фрагмент кода

```
addresses: this.buildAddress()  
addresses: this.fb.group({  
    addressType: 'home',  
    country: '',  
    city: '',  
    zip: '',  
    street1: '',  
    street2: ''  
}))
```

Task 20. Create a FormArray

1. Внесите изменения в компонент **SignupReactiveFormComponent** используя следующий фрагмент кода

```
import { FormGroup, FormControl, FormArray, FormBuilder, Validators, AbstractControl }  
from '@angular/forms';
```

2. Внесите изменения в метод **buildForm** используя следующий фрагмент кода

```
addresses: this.buildAddress()  
addresses: this.fb.array([this.buildAddress()])
```

3. Добавьте геттер **addresses** для получения FormArray используя следующий фрагмент кода

```
get addresses():FormArray {  
  return <FormArray>this.userForm.get('addresses');  
}
```

4. Внесите изменения в темплейт компонента **SignupReactiveFormComponent**, создайте обертку для блока `<div formGroupName="addresses">` используя следующий фрагмент разметки

```
<div formArrayName="addresses">  
</div>
```

5. Внесите изменения в значение атрибута `formGroupName="addresses"` используя следующий фрагмент разметки

```
<div formGroupName="addresses">  
<div formGroupName="0">
```

Task 21. Loop through the FormArray

1. Внесите изменения в разметку компонента **SignupReactiveFormComponent** используя следующий фрагмент разметки

```
<div formArrayName="addresses">
  <div formGroupName="0">
<div formArrayName="addresses"
  *ngFor="let address of addresses.controls; let i = index">
  <div [formGroupName]="i">
```

2. Внесите изменения в значение атрибута **id** для элементов **Home, Work, Other** используя следующий фрагмент разметки

```
id="addressType1Id"
id="{{ 'addressType1Id' + i }}"
```

3. Внесите изменения в значение атрибута **for** для элементов label для «Country, City, Zip Code», «Street Address 1», «Street Address 2»

```
// Country, City, Zip Code
for="cityId"
attr.for="{{ 'countryId' + i }}"
```

```
// Street Address 1
for="street1Id"
attr.for="{{ 'street1Id' + i }}"
```

```
// Street Address 2
for="street2Id"
attr.for="{{ 'street2Id' + i }}"
```

4. Внесите изменения в значение атрибута **id** для элементов ввода Country, City, Zip Code, Street Address 1, Street Address 2

```
// Country
id="countryId"
id="{{ 'countryId' + i }}"
```

```
// City
id="cityId"
id="{{ 'cityId' + i }}"
```

```
// Zip Code
id="zipId"
id="{{ 'zipId' + i }}"
```

```
// Street Address 1
id="street1Id"
id="{{ 'street1Id' + i }}"
```

```
// Street Address 2
id="street2Id"
id="{{ 'street2Id' + i }}"
```

Task 22. Duplicate the Input Elements

1. Добавьте новый метод для компонента **SignupReactiveFormComponent** используя следующий фрагмент кода

```
addAddress(): void {  
    this.addresses.push(this.buildAddress());  
}
```

2. Внесите изменения в разметку компонента **SignupReactiveFormComponent** используя следующий фрагмент разметки

```
// 1  
<div *ngIf="userForm.get('sendProducts').value">  
    <div class="form-group">  
        <div class="col-md-4 col-md-offset-2">  
            <button class="btn btn-primary"  
                type="button"  
                (click)="addAddress()">  
                Add Another Address  
            </button>  
        </div>  
    </div>  
  
// 2  
<br>userForm.get('emailGroup').errors {{userForm.get('emailGroup').errors | json}}  
<br>Street: {{addresses.get('0.street1')?.value}}
```