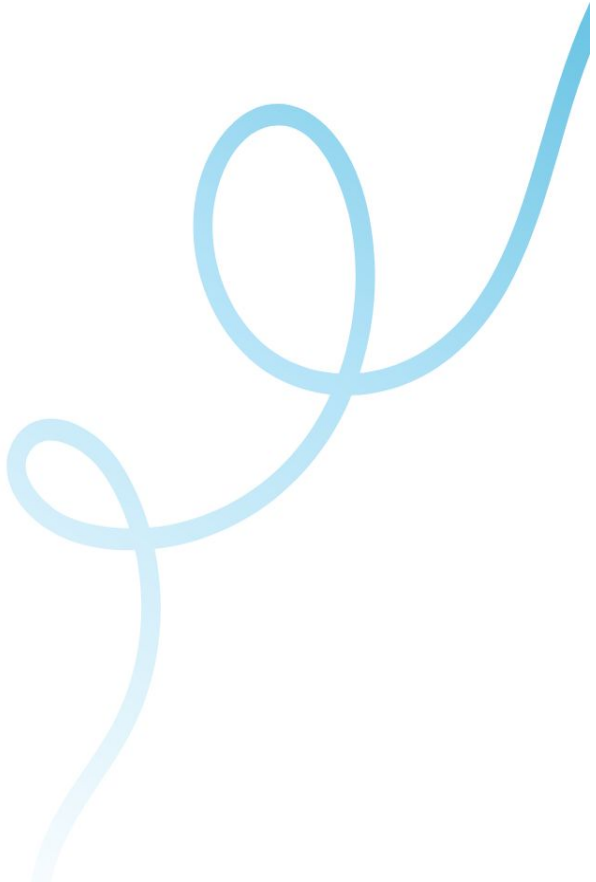
Advanced Forms

Лектор: Петър Маламов

Diadraw
www.diadraw.com

Типове форми в Angular

- Template-driven forms
- Reactive forms



Diadraw
www.diadraw.com

Разлики в двата типа

- **Template-driven forms** - Разчитат на директиви в темплейта, за да създават и манипулират модел на формата. Лесни са за добавяне към приложение, но не са толкова скалируеми, колкото **reactive forms**.

- **Reactive forms** - Осигуряват директен достъп до модел на формата. Те са по-скалируеми, преизползваеми и лесни за тестване от **template-driven forms**.

Diadraw
www.diadraw.com

	Reactive	Template-driven
Setup of form model	Explicit, created in component class	Implicit, created by directives
Data model	Structured and immutable	Unstructured and mutable
Data flow	Synchronous	Asynchronous
Form validation	Functions	Directives

https://angular.dev/guide/forms#key-differences

Diadraw
www.diadraw.com

Важно!

При **template-driven forms** източникът на истина е темплейтът. Директивата NgModel автоматично управлява инстанцията на формата.

При **reactive forms** моделът на формата е единственият източник на истина, той предоставя стойността и състоянието на елемента от формата във всеки момент.

Diadraw
www.diadraw.com

Template-driven forms са подходящи при:

- По-прости форми
- По-малко логика за директна модификация
- Изчистен HTML темплейт

Diadraw
www.diadraw.com

Reactive forms са подходящи при:

- Сложни форми
- По-гъвкаво валидиране и управление на състоянието на формата
- По-голяма модулност

Diadraw
www.diadraw.com

Reactive Forms

Diadraw
www.diadraw.com

Reactive Forms

Използват explicit и immutable подход за управление на данните във формата.

Всяка промяна във формата връща нова инстанция на данните, което помага за следене на промените в нея.

Reactive Forms използват “проследими” потоци от данни, което позволява проследяване на промените в тях.

Diadraw
www.diadraw.com

FormControl

Основен градивен елемент на формата, представляващ едно поле.

Всеки FormControl се използва за създаване и управление на стойността и валидацията на отделни полета.

Осигурява начин за лесно проследяване и актуализиране на стойността на полето, валидността му и свързаните грешки.

Diadraw
www.diadraw.com

Как се използват FormControl?

Diadraw
www.diadraw.com

Ръчна промяна на данните

TS app.component.ts

```ts
export class AppComponent implements OnInit {
  title = '07-advanced-forms';

  name = new FormControl('');

  ngOnInit(): void {
    // get data from server
    const newValue = 'Server Name';
    // update data manually
    this.name.setValue(newValue);
  }
}
```

Позволява ни да променяме данните във формата, без да е необходима интеракция от потребителя.

Diadraw
www.diadraw.com

Основните свойства на FormControl

```
override readonly value: any;
override readonly valid: boolean;
override readonly invalid: boolean;
override readonly pending: boolean;
override readonly disabled: boolean;
override readonly enabled: boolean;
override readonly errors: ValidationErrors;
override readonly pristine: boolean;
override readonly dirty: boolean;
override readonly touched: boolean;
override readonly status: string;
override readonly untouched: boolean;
override readonly statusChanges: Observable<any>;
override readonly valueChanges: Observable<any>;
override readonly validator: ValidatorFn;
override readonly asyncValidator: AsyncValidatorFn;
override reset(value?: any): void;
override hasError(errorCode: string, path?: string | (string | number)[]): boolean;
override getError(errorCode: string, path?: string | (string | number)[]): any;
```

Diadraw
www.diadraw.com

Четене на данни от формата

Има два начина за извличане на данни от формата:

- С параметъра **value** - връща текущата стойност във формата

- С параметъра **valueChanges** - връща наблюдаем поток от данни, към който можем да се абонираме и да следим за промени.

Diadraw
www.diadraw.com

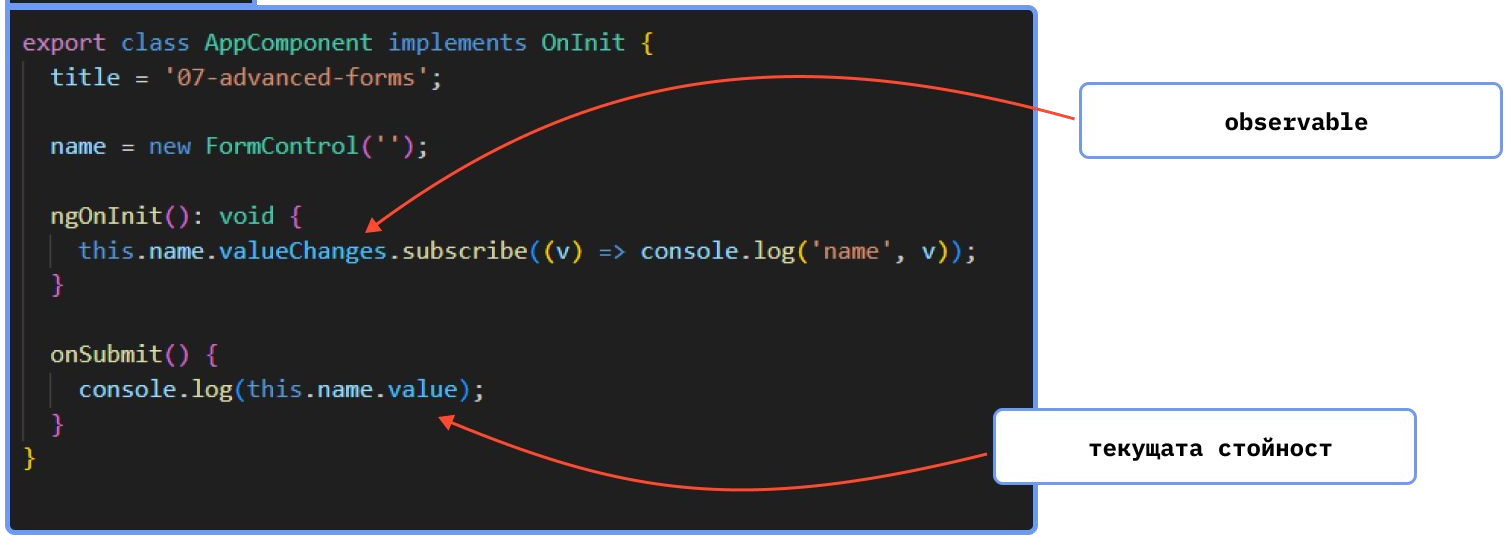
TS app.component.ts

```ts
export class AppComponent implements OnInit {
  title = '07-advanced-forms';

  name = new FormControl('');

  ngOnInit(): void {
    this.name.valueChanges.subscribe((v) => console.log('name', v));
  }

  onSubmit() {
    console.log(this.name.value);
  }
}
```



Diadraw
www.diadraw.com

Групиране на полета

Reactive Forms предоставя два основни начина за групиране на полета:

- Form group
- Form array

Diadraw
www.diadraw.com

Form group

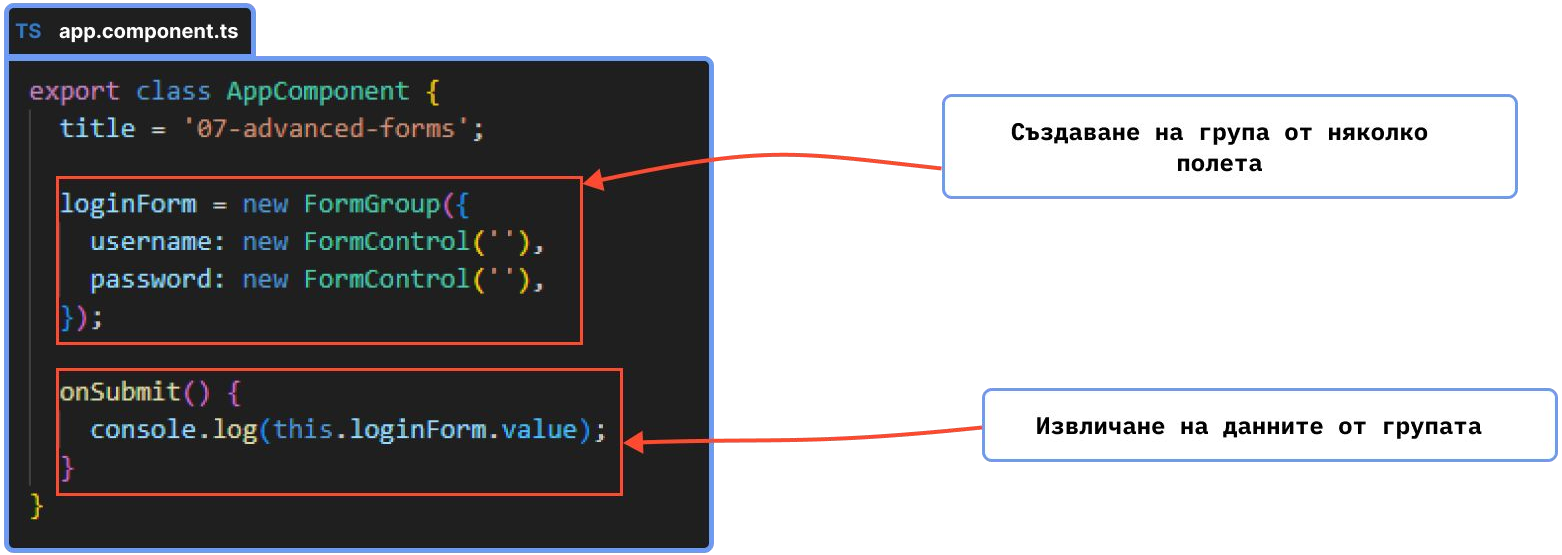
Определя форма с фиксиран набор от контроли, които могат да се управляват заедно.

Инстанцията на **FormGroup** следи за промени в някое от полетата, като всяко поле се проследява според името, с което е дефинирано в групата.

FormGroup има сходни свойства като FormControl, което позволява да се наблюдават промените в цялата група едновременно.

Diadraw
www.diadraw.com

Как се използва?



Diadraw
www.diadraw.com

<> app.component.html

```
<form [formGroup]="loginForm" (ngSubmit)="onSubmit()">
  <div>
    <label for="username">Username: </label>
    <input id="username" type="text" formControlName="username">
  </div>
  <div>
    <label for="password">Password: </label>
    <input id="password" type="password" formControlName="password">
  </div>

  <div>
    <button type="submit" class="border p-2">Submit</button>
  </div>
</form>
```

Създава връзка между модела и полетата във формата.

Създава връзка между полето и конкретния FormControl в групата.

Diadraw
www.diadraw.com

Създаване на вложени групи от форми

Form group могат да приемат както индивидуални инстанции на контроли, така и инстанции на други групи от форми като деца.

Това улеснява създаването на сложни модели на форми и логичното им групиране.

Diadraw
www.diadraw.com

TS app.component.ts

```ts
export class AppComponent {
  title = '07-advanced-forms';

  loginForm = new FormGroup({
    username: new FormControl(''),
    password: new FormControl(''),
    address: new FormGroup({
      street: new FormControl(''),
      city: new FormControl(''),
    }),
  });

  onSubmit() {
    console.log(this.loginForm.value);
  }
}
```

Diadraw
www.diadraw.com

<> app.component.html

```
<form [formGroup]="loginForm" (ngSubmit)="onSubmit()">
  <div>
    <label for="username">Username: </label>
    <input id="username" type="text" formControlName="username">
  </div>
  <div>
    <label for="password">Password: </label>
    <input id="password" type="password" formControlName="password">
  </div>

  <div formGroupName="address">
    <h2>Address</h2>
    <div>
      <label for="street">Street: </label>
      <input id="street" type="text" formControlName="street">
    </div>
    <div>
      <label for="city">City: </label>
      <input id="city" type="text" formControlName="city">
    </div>
  </div>

  <div>
    <button type="submit" class="border p-2">Submit</button>
  </div>
</form>
```

Създава връзка между модела и вложената група.

Създава връзка между полето и конкретния FormControl във вложената група.

Diadraw
www.diadraw.com

Ръчно актуализиране на данните

Има два подхода за промяна на стойностите, когато работим с множество полета във формата.

- **setValue** - Обновява целия модел на данните.
- **patchValue** - Обновява само част от данните.

Diadraw
www.diadraw.com

TS app.component.ts

```ts
export class AppComponent {
  title = '07-advanced-forms';

  signUpForm = new FormGroup({
    username: new FormControl(''),
    password: new FormControl(''),
    address: new FormGroup({
      street: new FormControl(''),
      city: new FormControl(''),
    }),
  });

  updateProfile() {
    this.signUpForm.patchValue({
      address: {
        city: 'Plovdiv',
      },
    });
  }
}
```

Diadraw
www.diadraw.com

Form array

Позволява динамично добавяне, премахване и управление на множество формови контроли в рамките на една група.

Може да управлява колекция от FormControl, FormGroup или други FormArray елементи.

Diadraw
www.diadraw.com

TS app.component.ts

```ts
export class AppComponent {
  title = '07-advanced-forms';

  signUpForm = new FormGroup({
    username: new FormControl(''),
    password: new FormControl(''),
    notes: new FormArray([new FormControl('')]),
  });

  getNotes() {
    return this.signUpForm.get('notes') as FormArray;
  }

  addNote() {
    const notes = this.signUpForm.get('notes') as FormArray;
    notes.push(new FormControl(''));
  }

  onSubmit() {
    console.log(this.signUpForm.value);
  }
}
```

Създаване на динамична група от полета

Динамично добавяне на елементи в групата

Diadraw
www.diadraw.com

TS app.component.ts

```ts
getNotes() {
  return this.signUpForm.get('notes') as FormArray;
}
```

<> app.component.html

```html
<div formArrayName="notes">
  <h2>Notes</h2>
  <button (click)="addNote()">Add note</button>
  @for(note of getNotes().controls; track $index) {
  <div>
    <label for="alias-{{ $index }}">Note:</label>
    <input id="alias-{{ $index }}" type="text" [formControlName]="$index">
  </div>
  }
</div>
```

Създава връзка между динамичните елементи и темплейта.

Създава връзка между конкретния елемент и темплейта

Diadraw
www.diadraw.com

FormBuilder

Angular клас, който улеснява създаването и инициализирането на форми.

Позволява лесното създаване на вложени **FormGroup** и **FormArray** за сложни форми.

FormBuilder позволява също така динамично добавяне или премахване на полета, което улеснява създаването на динамични форми.

Diadraw
www.diadraw.com

Използване на FormBuilder service

TS app.component.ts

```ts
signUpForm = this.fb.group({
  username: [''],
  password: [''],
  notes: this.fb.array(['Note 1']),
});

constructor(private fb: FormBuilder) {}
```

Създаване на форма чрез FormBuilder service

TS app.component.ts

```ts
signUpForm = new FormGroup({
  username: new FormControl(''),
  password: new FormControl(''),
  notes: new FormArray([new FormControl('')]),
});
```

Създаване на форма ръчно

Diadraw
www.diadraw.com

Динамично създаване на полета

```ts
export const serverData = {
  fields: [
    {
      name: 'email',
      label: 'Email',
      value: 'test test',
      type: 'text',
      validators: {
        required: true,
        email: 10,
      },
    },
    {
      name: 'password',
      label: 'Password',
      value: '',
      type: 'password',
      validators: {
        required: true,
        minLength: 10,
      },
    },
  ],
};
```

TS app.component.ts

```ts
export class SignUpComponent {
  jsonData = input.required<ServerForm>();

  signUpForm = computed(() => {
    return this.buildForm(this.jsonData());
  });

  constructor(private fb: FormBuilder) {}

  buildForm(form: ServerForm) {
    const formGroup = this.fb.group({});
    for (const field of form.fields) {
      const validators: ValidatorFn[] = [];
      Object.entries(field.validators).forEach(([key, value]) => {···
      });
      formGroup.addControl(
        field.name,
        this.fb.control(field.value, validators)
      );
    }

    return formGroup;
  }

  onSubmit() {
    console.log(this.signUpForm().value);
  }
}
```

Diadraw
www.diadraw.com

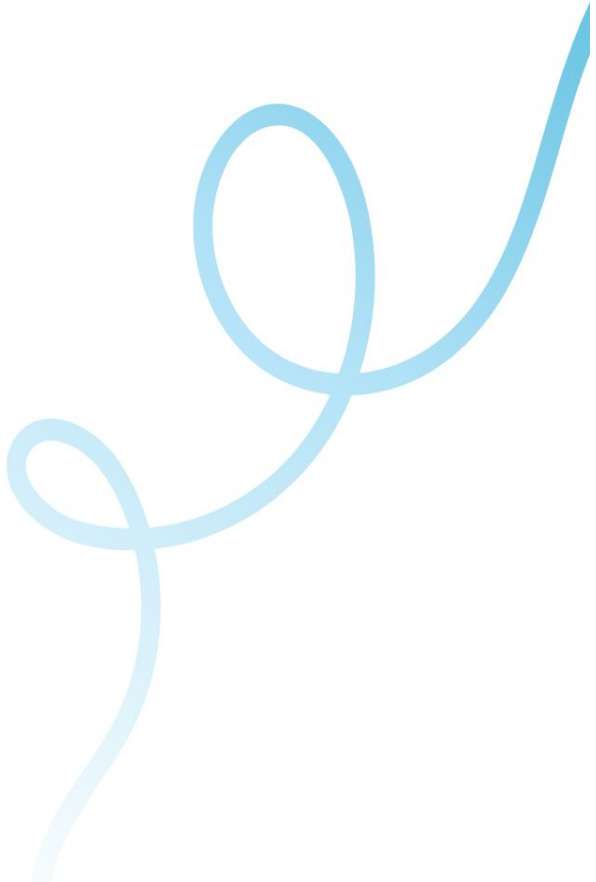
Използване на динамична форма

<> app.component.html

```
<form [formGroup]="signUpForm()" (ngSubmit)="onSubmit()">
    @for(field of jsonData().fields; track $index) {
    <div class="flex flex-col gap-2">
        <label>{{field.label}}</label>
        <input [type]="field.type" class="border p-2" [value]="field.value" [formControlName]='field.name'>
    </div>
    }
    <button type=" submit">Submit</button>
</form>
```

Diadraw
www.diadraw.com

Упражнение



Diadraw
www.diadraw.com

Validators

Diadraw
www.diadraw.com

Forms Validators

Функции, които проверяват стойностите на полетата във формата, за да гарантират, че те отговарят на определени изисквания или правила.

Diadraw
www.diadraw.com

Reactive form validators

В **reactive form** източникът на истина е класът на компонента, поради което валидаторите се добавят директно към модела на формата.

Angular извиква тези функции всеки път, когато стойността на контрола се промени.

Валидационните функции могат да бъдат както **синхронни**, така и **асинхронни**.

Diadraw
www.diadraw.com

Вградени валидатори

```
class Validators {
  static min(min: number): ValidatorFn;
  static max(max: number): ValidatorFn;
  static required(control: AbstractControl<any, any>): ValidationErrors;
  static requiredTrue(control: AbstractControl<any, any>): ValidationErrors;
  static email(control: AbstractControl<any, any>): ValidationErrors;
  static minLength(minLength: number): ValidatorFn;
  static maxLength(maxLength: number): ValidatorFn;
  static pattern(pattern: string | RegExp): ValidatorFn;
  static nullValidator(control: AbstractControl<any, any>): ValidationErrors;

  static compose(validators: null): null;
  static compose(validators: ValidatorFn[]): ValidatorFn;

  static composeAsync(validators: AsyncValidatorFn[]): AsyncValidatorFn;
}
```

Diadraw
www.diadraw.com

Използване на вградените валидатори.

TS app.component.ts

```ts
signUpForm = this.fb.group({
  username: ['', Validators.required],
  password: ['', [Validators.required, Validators.minLength(5)]],
  notes: this.fb.array([], [Validators.required, Validators.minLength(3)]),
});
```

Diadraw
www.diadraw.com

Визуализиране на грешки при валидация

TS app.component.ts

```ts
signUpForm = this.fb.group({
  username: ['', [Validators.required]],
  password: ['', [Validators.required, Validators.minLength(6)]],
  notes: this.fb.array([], [Validators.required]),
});

constructor(private fb: FormBuilder) {}

hasError(field: string) {
  const control = this.signUpForm.get(field);
  return control?.invalid && (control?.touched || control?.dirty);
}

getErrorMessage(field: string) {
  const control = this.signUpForm.get(field);
  if (control?.hasError('required')) {
    return `${field} is required`;
  }
  if (control?.hasError('minlength')) {
    return `${field} must be at least 6 characters`;
  }

  return '';
}
```

<> app.component.html

```html
<div>
  <label for="username">Username: </label>
  <input id="username" type="text" formControlName="username">
  @if(hasError('username')) {
  <p class="text-red-500">{{getErrorMessage('username')}}</p>
  }
</div>
```

Проверка дали конкретното поле има грешка.

Вземане на конкретната грешка спрямо валидатора

Diadraw
www.diadraw.com

Динамично добавяне на валидатори

Reactive Forms позволяват добавянето или премахването на валидатори от формата по време на изпълнението на приложението.

Това е полезно, когато валидаторите зависят от определени условия или от други полета.

Добавяме валидатори чрез метода **setValidators** на FormControl. Премахваме валидатори чрез метода **removeValidators** на FormControl.

Diadraw
www.diadraw.com

TS app.component.ts

```ts
signUpForm = this.fb.group({
  email: ['', [Validators.required, Validators.email]],
  password: ['', [Validators.required, Validators.minLength(6)]],
  age: [0, [Validators.required]],
  parentEmail: ['', [Validators.required, Validators.email]],
});

constructor(private fb: FormBuilder) {}

ngOnInit(): void {
  this.signUpForm.get('age')?.valueChanges.subscribe((age) => {
    if (!age) return;

    if (age < 18) {
      this.signUpForm.controls.parentEmail.setValidators([
        Validators.required,
      ]);
    } else {
      this.signUpForm.controls.parentEmail.removeValidators([
        Validators.required,
      ]);
    }

    this.signUpForm.controls.parentEmail.updateValueAndValidity();
  });
}
```

Динамично добавяне на валидатор

Динамично премахване на валидатор

Diadraw
www.diadraw.com

Важно!

Angular премахва валидатори, като използва техните референции.

Когато използваме валидатори, които приемат параметри, те създават нова инстанция при всяко извикване, което прави премахването им неефективно.

Diadraw
www.diadraw.com

TS app.component.ts

```ts
if (age < 18) {
  this.signUpForm.controls.parentEmail.setValidators([
    Validators.minLength(5),
  ]);
} else {
  this.signUpForm.controls.parentEmail.removeValidators([
    Validators.minLength(5),
  ]);
}
```

`Validators.minLength(5) != Validators.minLength(5)`

Diadraw
www.diadraw.com

app.component.ts

```ts
dynamicValidator = Validators.minLength(5);

ngOnInit(): void {
  this.signUpForm.get('age')?.valueChanges.subscribe((age) => {
    if (!age) return;

    if (age < 18) {
      this.signUpForm.controls.parentEmail.setValidators([
        this.dynamicValidator,
      ]);
    } else {
      this.signUpForm.controls.parentEmail.removeValidators([
        this.dynamicValidator,
      ]);
    }

    this.signUpForm.controls.parentEmail.updateValueAndValidity();
  });
}
```

```ts
Validators.minLength(5) == Validators.minLength(5)
```

Diadraw
www.diadraw.com

Създаване на персонализирани валидатори

Angular ни позволяват да дефинираме специфични правила за валидация, които не са дефинирани от вградените валидатори.

Персонализиран валидатор е функция, която приема контрол и връща обект с информация за грешките или null, ако валидацията е успешна.

Diadraw
www.diadraw.com

TS app.component.ts

```ts
export function customValidator(
  control: AbstractControl
): ValidationErrors | null {
  /* логика за валидация */
  const isValid = control.value === 'test';

  return isValid ? null : { customError: true };
}
```

TS app.component.ts

```ts
export class AppComponent {
  title = '07-advanced-forms';

  signUpForm = this.fb.group({
    username: ['', customValidator],
  });

  constructor(private fb: FormBuilder) {}

  onSubmit() {
    const control = this.signUpForm.get('username');
    const error = control?.errors;
  }
}
```

Създаване на персонализиран валидатор

Използване на персонализиран валидатор

Diadraw
www.diadraw.com

Cross-field валидации

Специални валидатори, които проверяват стойностите на две или повече полета във формата.

Използват се, когато валидацията не може да бъде извършена само на базата на стойността на едно поле, а изисква информация от множество полета.

Cross-field валидаторът се създава като персонализиран валидатор, но се прилага към група от контролери вместо към един единствен.

Diadraw
www.diadraw.com

Примери за Cross-field валидатори

- Проверка за съвпадение на пароли
- Проверка за дати (началната дата е преди крайната дата).

Diadraw
www.diadraw.com

TS app.component.ts

```ts
export function matchingPasswordsValidator(
  control: AbstractControl
): ValidationErrors | null {
  const password = control.get('password')?.value;
  const confirmPassword = control.get('confirmPassword')?.value;

  return password === confirmPassword ? null : { passwordsMismatch: true };
}
```

Създаване на Cross-field валидатор

TS app.component.ts

```ts
export class AppComponent {
  title = '07-advanced-forms';

  signUpForm = this.fb.group(
    {
      username: [''],
      password: [''],
      confirmPassword: ['a'],
    },
    { validators: matchingPasswordsValidator }
  );

  constructor(private fb: FormBuilder) {}

  onSubmit() {}
}
```

Използване на Cross-field валидатор

Diadraw
www.diadraw.com

Асинхронни валидатори

Валидатори, които извършват проверка на стойността на контролера по асинхронен начин.

Те се използват, когато валидацията изисква взаимодействие с външни ресурси, като например API заявки или база данни.

Асинхронните валидатори връщат Observable или Promise, който предоставя резултата от валидацията.

Diadraw
www.diadraw.com

Важно!

Ако асинхронният валидатор връща Observable, той трябва да бъде от **ограничен/краен** тип, тоест в даден момент трябва да завърши с статус complete."

Diadraw
www.diadraw.com

Важно!

Асинхронните валидации се извършват след синхронните и се активират само ако синхронните валидатори са върнали положителен резултат.
Това спестява ненужни заявки, ако данните не са преминали локалните проверки.

Diadraw
www.diadraw.com

Създаване на асинхронен валидатор

TS app.component.ts

```ts
@Injectable({ providedIn: 'root' })
export class UniqueUsernameValidator implements AsyncValidator {
  constructor(private userService: UserService) {}

  validate(control: AbstractControl): Observable<ValidationErrors | null> {
    return this.userService.isUsernameTaken(control.value).pipe(
      delay(1000),
      map((isTaken) => (isTaken ? { usernameTaken: true } : null)),
      tap(() => {
        console.log('tap');
      }),
      catchError(() => of(null))
    );
  }
}
```

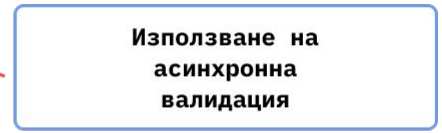
Diadraw
www.diadraw.com

Имплементиране на асинхронен валидатор

TS app.component.ts

```ts
signUpForm = this.fb.group({
  username: [
    '',
    {
      asyncValidators: [
        this.uniqueUsernameValidator.validate.bind(
          this.uniqueUsernameValidator
        ),
      ],
    },
  ],
  password: [''],
  confirmPassword: ['a'],
});

constructor(
  private fb: FormBuilder,
  private uniqueUsernameValidator: UniqueUsernameValidator
) {}
```



Diadraw
www.diadraw.com

Оптимизиране на асинхронната валидация

Функциите за валидация по подразбиране се активират при всяка промяна на полето. При асинхронните валидации това може доведе до performance проблеми.

Angular предлага възможност да определим в кой момент да се задействат функциите за валидация, чрез използването на полето **updateOn** в опциите за валидаторите.

Diadraw
www.diadraw.com

Моменти за активиране на валидацията

- change - при промяна на стойността на полето (по подразбиране)
- submit - преди изпращане на формата
- blur - при загуба на фокус от полето

Diadraw
www.diadraw.com

TS app.component.ts

```ts
signUpForm = new FormGroup({
  username: new FormControl('', {
    asyncValidators: [
      this.uniqueUsernameValidator.validate.bind(
        this.uniqueUsernameValidator
      ),
    ],
    updateOn: 'blur',
  }),
  password: new FormControl(''),
  confirmPassword: new FormControl(''),
});
```

Промяна на момента за валидация

Diadraw
www.diadraw.com

Упражнение

```
singUpForm = this.fb.group({
  username: this.fb.control(''), // задължително
  password: this.fb.control(''), // задължително,
  confirmPassword: this.fb.control(''), // задължително, трябва да е еднаква с паролата
  age: this.fb.control(''), // задължително
  address: this.fb.group({
    city: this.fb.control(''),
    street: this.fb.control(''), // задължително само ако city е попълнен
    postalCode: this.fb.control(''), // задължително само ако city е попълнен
  }),
});
```

Diadraw
www.diadraw.com

Благодаря за вниманието!

Diadraw
www.diadraw.com