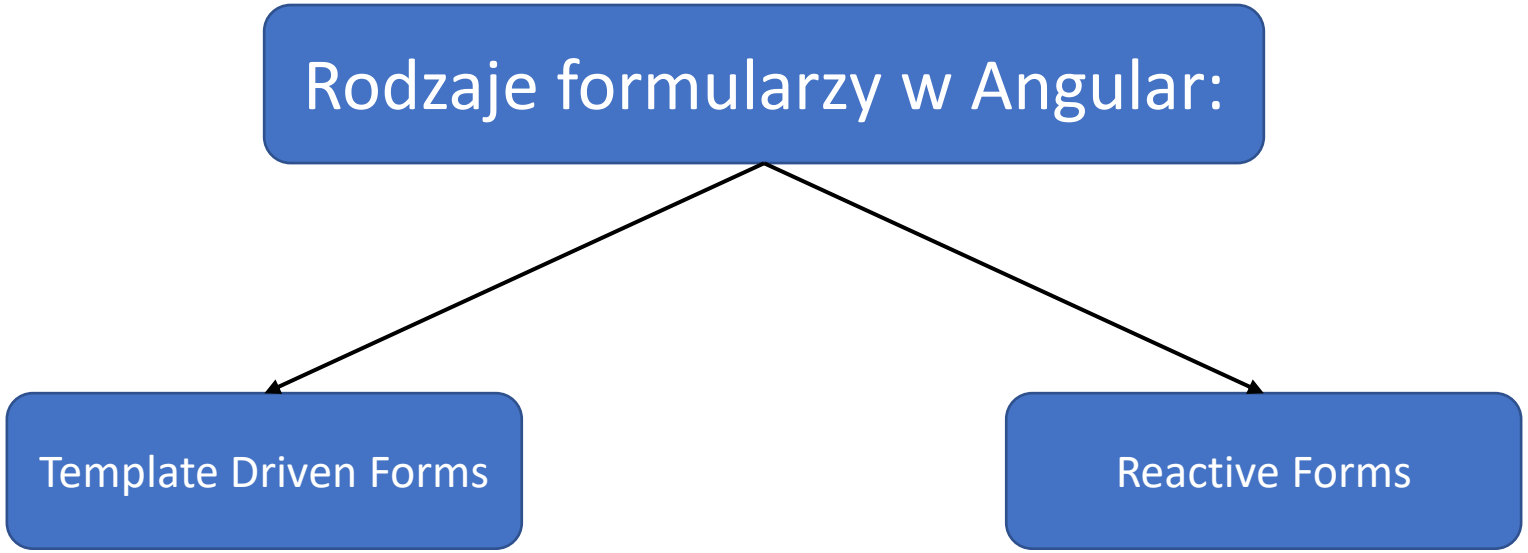


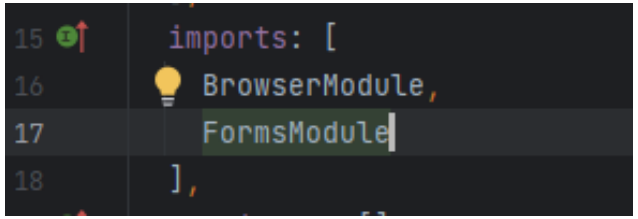
Formularze są tworzone bezpośrednio w szablonie HTML.

Formularze oraz związana z nimi logika są tworzone w pliku TypeScript. Umożliwiają dynamiczne tworzenie pól formularza.

Jeśli chodzi o pracę z formularzami w Angular, to jeśli Nasz formularz ma więcej niż jedną kontrolkę to lepiej wykorzystać Reactive Forms (choć nie jest to wymóg). Reactive Forms są bardziej elastyczne i skalowalne niż Template Driven Forms. Do tego Reacive Forms ułatwiają tworzenie testów jednostkowych dla formularzy i są oparte na Observable (każda kontrolka to obiekt **FormControl**, która pozwala używać metodę **valueChanges()**, która zwraca Observable.

Template Driven Forms

Aby korzystać z Template Driven Forms moduł Angular, w którym pracujemy musi importować wbudowany moduł **FormsModule** (często import FormsModule umieszcza się w tzw. **SharedModule**).



Po imporcie **FormsModule** w komponentach naszego modułu otrzymujemy dostęp do trzech dyrektyw:

- NgModel (używana do Two-way databinding)
- NgForm (umożliwia śledzenie aktualnej wartości kontrolek formularza i statusu walidatorów kontrolek, po imporcie FormsModule domyślnie wszystkie formularze mają aktywną tą dyrektywę)
- NgModelGroup (pozwala grupować kontrolki)

Ważne! Używając dyrektywy **ngModel** musimy także dodać atrybut **name** do tej kontrolki!

Dodanie dyrektywy **ngModel** do kontrolki dodaje także nowe klasy w ramach tej kontrolki formularza. Dzięki temu możemy zmieniać style naszej kontrolki zależnie od tego, czy ma wpisane poprawne dane, czy została zaznaczona, czy jej wartość się zmieniła.

STAN	KLASA JEŚLI PRAWDA	KLASA JEŚLI FAŁSZ
Kontrolka została zaznaczona, a następnie odznaczona (kliknięcie w innej miejsce w aplikacji)	ng-touched	ng-untouched
Wartość w kontrolce się zmieniła (od domyślnej)	ng-dirty	ng-pristine
Wartość kontrolki jest prawidłowa (walidatory nie wykryły błędu)	ng-valid	ng-invalid

Reactive Forms

Aby korzystać z Template Driven Forms moduł Angular, w którym pracujemy musi importować wbudowany moduł **ReactiveFormsModule** (często import ReactiveFormsModule umieszcza się w tzw. **SharedModule**).

```
imports: [CommonModule, ReactiveFormsModule],
```

W ReactiveForms kod naszego formularza tworzymy w klasie komponentu. Dla pojedynczej kontrolki (np. input) używamy klasę (konstruktor) FormControl:

```
username = new FormControl( value: '');
```

Po utworzeniu kontrolki w klasie komponentu, trzeba ją powiązać z kontrolką formularza w szablonie HTML (poprzez **formControl**, które jest dostarczane dzięki dyrektywie **FormControlDirective** (z modułu ReactiveFormsModule)).

```
<input type="text " [formControl]="username">
```

Wartość kontrolki można sprawdzać poprzez **valueChanges** (Observable, któy daje aktualną wartość lub poprzez właściwość **value** (to snapshot wartości). Np. **username.value**

Grupowanie kontrolek (Reactive Forms)

Istnieją dwie możliwości na grupowanie kontrolek w ReactiveForms:

- FormGroup – grupuje kontrolki (grupa, do której nie będą dodawane nowe kontrolki),

```
user = new FormGroup( controls: {
  username: new FormControl( value: ''),
  email: new FormControl( value: ''),
});
```

```
<form [formGroup]="user">
  <input type="text" formControlName="username">
  <input type="text" formControlName="email">
</form>
```

- FormArray – używany do dynamicznych formularzy, można dodawać do niego nowe kontrolki w czasie działania aplikacji.

```
user = new FormGroup( controls: {
  username: new FormControl( value: ''),
  email: new FormControl( value: ''),
  parents: new FormArray( controls: [new FormControl( value: ''), new FormControl( value: '')]),
});

1 usage
get parents() {
  return this.user.get('parents') as FormArray;
}
```

```
<form [formGroup]="user">
  <input type="text" formControlName="username">
  <input type="text" formControlName="email">
  <ng-container formArrayName="parents">
    <div *ngFor="let parent of parents.controls; index as i">
      <input type="text" formControlName="i">
    </div>
  </ng-container>
</form>
```

Korzystając z **this.parents** możemy m.in. dodawać nowe kontrolki (metoda **push()**) oraz usuwać (metoda **removeAt()**)

Form builder (Reactive Forms)

Korzystając z Reactive Forms nie musimy tworzyć formularzy, jak to zostało przedstawione na wcześniejszych slajdach. Można do tego użyć tzw. **Form builder**.

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

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

```
user = this.fb.group( controls: {
  username: [''],
  email: [''],
  parents: this.fb.array( controls: [
    this.fb.control( formState: ''),
    this.fb.control( formState: '')
  ])
})
```