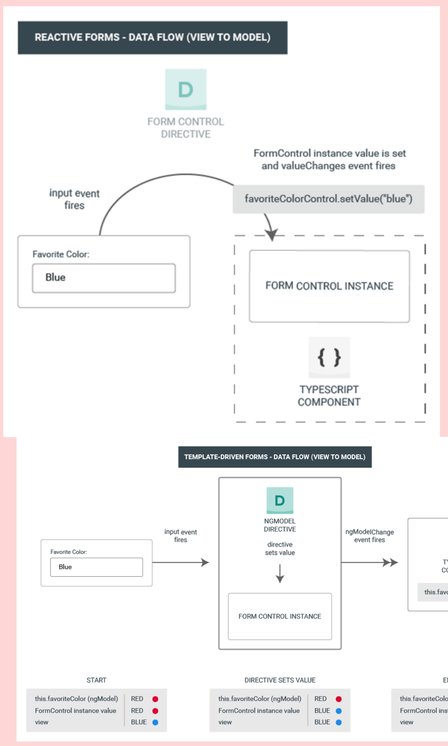
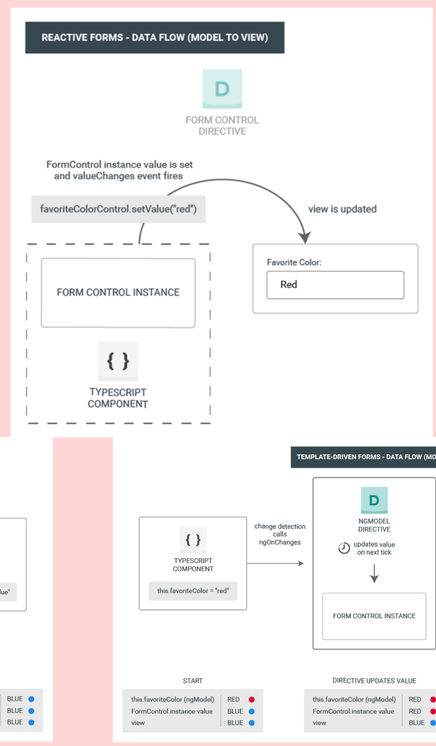
1. Angular Reactive Forms

→ Model-driven forms
→ Offers an easy way to use reactive patterns and validations.
→ They follow the reactive programming style.

→ When coding reactive forms, we will avoid directives like required, ngModel, NgForm and such.
→ Allows us to write expressive code instead of dividing it over different form templates.
→ With reactive forms, you will be able to create and manipulate form control objects directly in the Component.
→ The component is able to observe changes in the form control state and react to them.





2. Angular Template driven forms

→ You can place HTML form controls (such as <input> or <select>) in the component template and bind them to data model properties, using directives like ngModel.
→ A few examples of these directives are: ngModel, required, maxlength.
→ In template-driven forms, we specify directives to bind our models, values, validations, and more.

So, we are actually letting the template do all the work in the background.

* Angular forms building blocks: FormControl, FormGroup, FormArray

i) FormControl: it tracks the value and validity status of an angular form control. It matches to a HTML form control like an input.

```
this.username = new FormControl('akash', Validators.required);

<input type="text" formControlName="username"></input>
```

ii) FormGroup: it tracks the value and validity state of a FormBuilder instance group. It aggregates the values of each child FormControl into one object, using the name of each form control as the key.

If one of the controls inside a group is invalid, the entire group becomes invalid.

```
this.user_data = new FormGroup({
    username: new FormControl('akash', Validators.required),
    city: new FormControl('Bangalore', Validators.required)
});
```

iii) FormArray: is a variation of FormGroup.

The main difference is that its data gets serialized as an array, as opposed to being serialized as an object in case of FormGroup.

This is useful in case of dynamic forms.

```
this.user.data = new FormArray({
    new FormControl('akash', Validators.required),
    new FormControl('Bangalore', Validators.required)
});
```

iv) FormBuilder: is a helper class that creates FormGroup, FormControl, and FormArray instances for us.

It basically reduces the repitition and clutter by handling details of form control creation for you.

```
this.validations_form = this.formBuilder.group({
    username: new FormControl('', Validators.required),
    email: new FormControl('', Validators.compose([
            Validators.required,
            Validators.pattern('^[a-zA-Z0-9_.+-]+@[a-zA-Z0-
            9-]+.[a-zA-Z0-9-.]+$')
        ]))
});
```

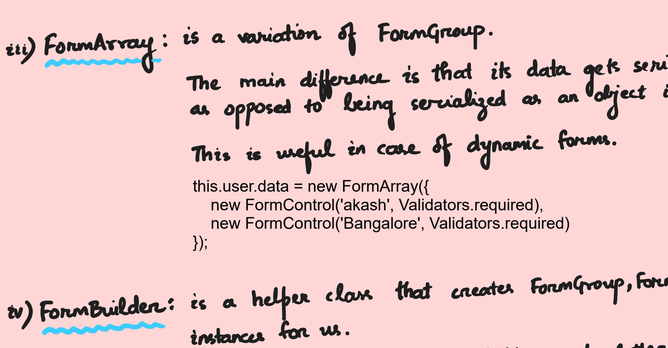
All of them should be imported from the @angular/forms module.

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

Angular is packed with its own validators.

```
this.email = new FormControl(null, {
        validators: Validators.required,
        updateOn: 'blur'
});
```

* Practice



↳ User details form components and constraints:

Input Name	Input Type	Input Validations	Validation Type
Bio	Text area	1. can't be more than 256 characters long	1. Maxlength 256
Birthday	Date picker	1. not empty	1. required
Gender	Select	1. not empty	1. required
Country	Select	1. not empty	1. required
Phone	Telephone	1. not empty 2. valid for the selected country	1. required 2. custom validation
Full Name	Text	1. not empty	1. required

↳ Account details form components and constraints:

Input Name	Input Type	Input Validations	Validation Type
User Name	Text	1. not empty 2. at least 5 characters long 3. can't be more than 25 characters long 4. must contain only numbers and letters 5. unique in our system	1. required 2. minlength 5 3. maxlength 25 4. pattern validation 5. custom validation
Email	email	1. not empty 2. valid email	1. required 2. pattern validation
Password	password	1. not empty 2. at least 5 characters long 3. must contain at least one uppercase, one lowercase, and one number	1. required 2. minlength 5 3. pattern validation
Confirm password	password	1. not empty 2. Equal to password	1. required 2. custom validation
Terms	checkbox	1. accepted	1. pattern validation