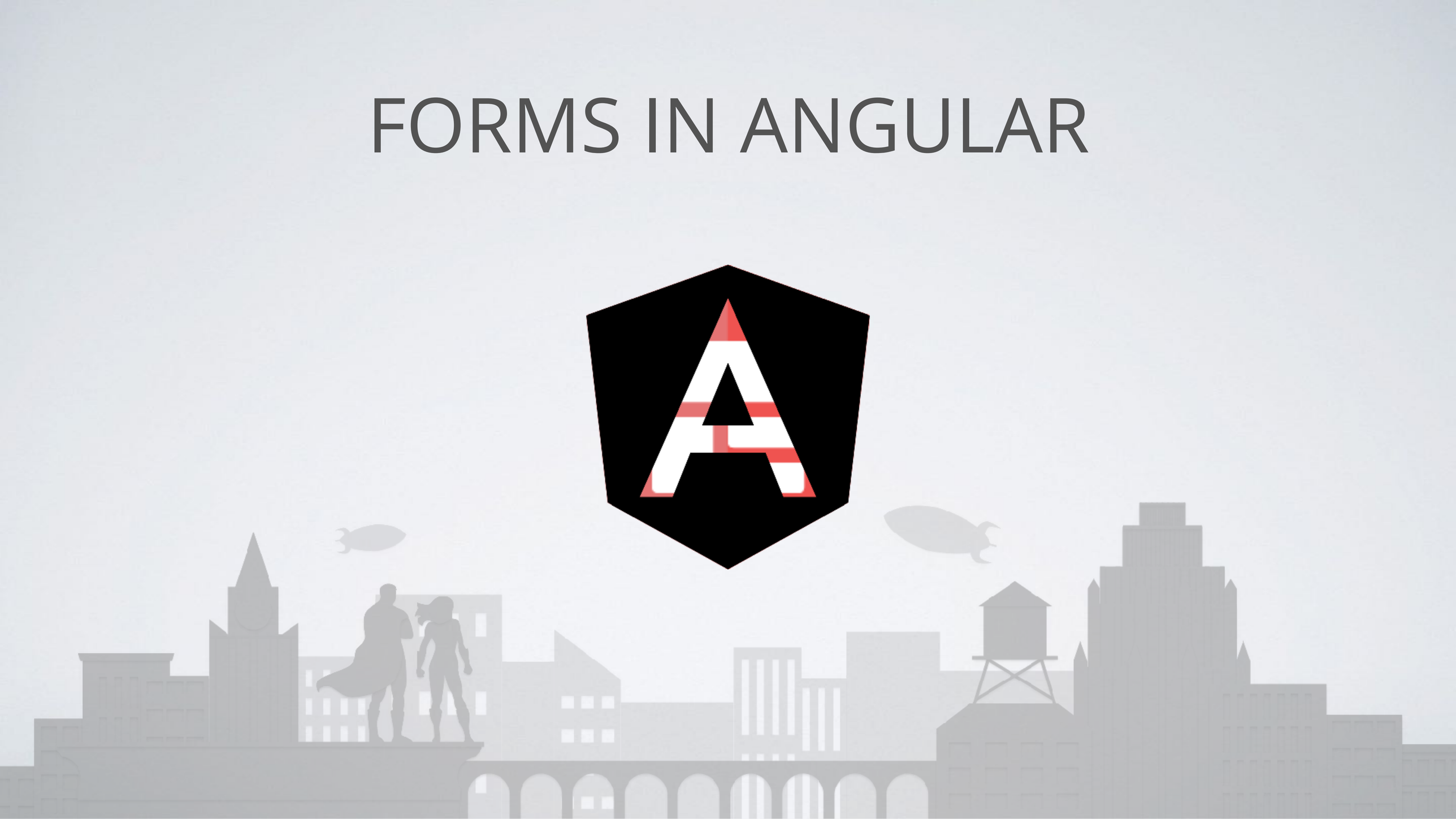
FORMS IN ANGULAR



FORMS IN ANGULAR

- Template Forms
  - import { FormsModule } from ‘@angular/forms';
- Reactive Forms
  - import { ReactiveFormsModule } from '@angular/forms'

TEMPLATE FORMS

TEMPLATE FORMS

- Component Class
- HTML-based Template

COMPONENT CLASS

```
export class HeroFormComponent {

  powers = ['Really Smart', 'Super Flexible',
            'Super Hot', 'Weather Changer'];

  model = new Hero(18, 'Dr IQ', this.powers[0], 'Chuck Overstreet');

  submitted = false;

  onSubmit() { this.submitted = true; }
}
```

HTML-BASED TEMPLATE

```
<div class="form-group">
  <label for="name">Name</label>
  <input type="text" class="form-control" id="name"
         required
         [(ngModel)]="model.name" name="name">
</div>

<div class="form-group">
  <label for="alterEgo">Alter Ego</label>
  <input type="text"  class="form-control" id="alterEgo"
         [(ngModel)]="model.alterEgo" name="alterEgo">
</div>
```

HTML-BASED TEMPLATE

```
<div class="form-group">
  <label for="power">Hero Power</label>
  <select class="form-control"  id="power"
          required
          [(ngModel)]="model.power" name="power">
    <option *ngFor="let pow of powers" [value]="pow">{{pow}}</option>
  </select>
</div>
```

CHANGE TRACKING & VALIDATION

State	Class if true	Class if false
The control has been visited.	`ng-touched`	`ng-untouched`
The control's value has changed.	`ng-dirty`	`ng-pristine`
The control's value is valid.	`ng-valid`	`ng-invalid`

CHANGE TRACKING & VALIDATION

```
.ng-valid[required], .ng-valid.required  {
  border-left: 5px solid #BADA55;
}

.ng-invalid:not(form)  {
  border-left: 5px solid #FA113D;
}
```

ERROR HANDLING

```
<form (ngSubmit)="onSubmit()" #heroForm="ngForm">
  <div class="form-group">
    <label for="name">Name</label>
    <input type="text" class="form-control" id="name"
           required
           [(ngModel)]="model.name" name="name"
           #name="ngModel">
    <div [hidden]="name.valid || name.pristine"
         class="alert alert-danger">
      Name is required
    </div>
  </div>
</form>
```

REACTIVE FORMS

REACTIVE FORMS

- AbstractControl
- FormControl
- FormGroup
- FormArray
- FormBuilder

FormControls

```
export class HeroFormComponent {
  heroForm = new FormGroup({
    name: new FormControl(''),
    alterEgo: new FormControl(''),
  });
}
```

FormControls

```
<form [formGroup]="heroForm">

  <label>
    Name:
    <input type="text" formControlName="name">
  </label>

  <label>
    Alter Ego:
    <input type="text" formControlName="alterEgo">
  </label>

</form>
```

FormBuilder

```
export class HeroFormComponent {
  heroForm = this.fb.group({
    name: [''],
    alterEgo: ['']
  });

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

AVAILABLE VALIDATIONS

```
export class HeroFormComponent {
  heroForm = this.fb.group({
    name: ['', [Validators.required,
    Validators.minLength(4),
    forbiddenNameValidator(/bob/i)],
      alterEgo: ['']
  });

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

CUSTOM VALIDATIONS

```
export function forbiddenNameValidator(nameRe: RegExp): ValidatorFn {
  return (control: AbstractControl): {[key: string]: any} | null => {
    const forbidden = nameRe.test(control.value);
    return forbidden ? {'forbiddenName': {value: control.value}} : null;
  };
}
```

CHANGE TRACKING

- FormControl.valueChanges Observer
- ngOnChanges

WHAT'S THE DIFFERENCE

	REACTIVE	TEMPLATE-DRIVEN
Setup (form model)	More explicit, created in component class	Less explicit, created by directives
Data model	Structured	Unstructured
Predictability	Synchronous	Asynchronous
Form validation	Functions	Directives
Mutability	Immutable	Mutable
Scalability	Low-level API access	Abstraction on top of APIs

TERIMA KASIH

https://nexmo.dev/ng-my

@Lakatos88