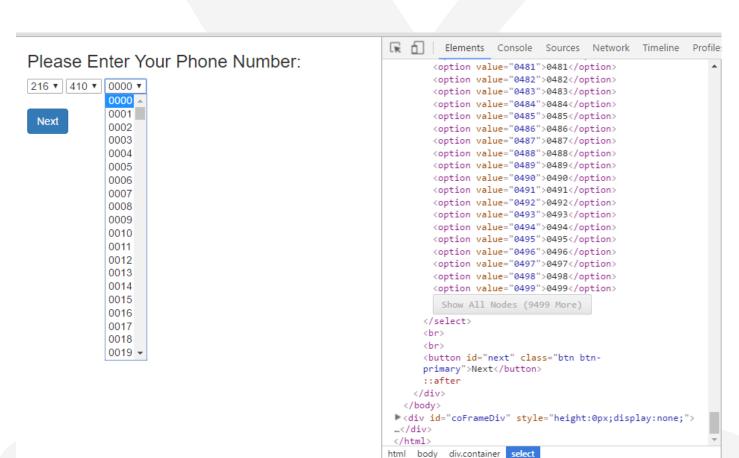
# Forms

# **Forms**

- Process user data
- Validation
- React to input





# Two "styles"

## ReactiveFormsModule

- Defined in TypeScript
- Less template logic
- Validations are functions
- Observables

## **FormsModule**

- Defined in template
- More template logic
- Validation in template / directives
- "Classic" style



# Model driven - ReactiveFormsModule

```
@NgModule({
   imports: [ ReactiveFormsModule ]
})
```

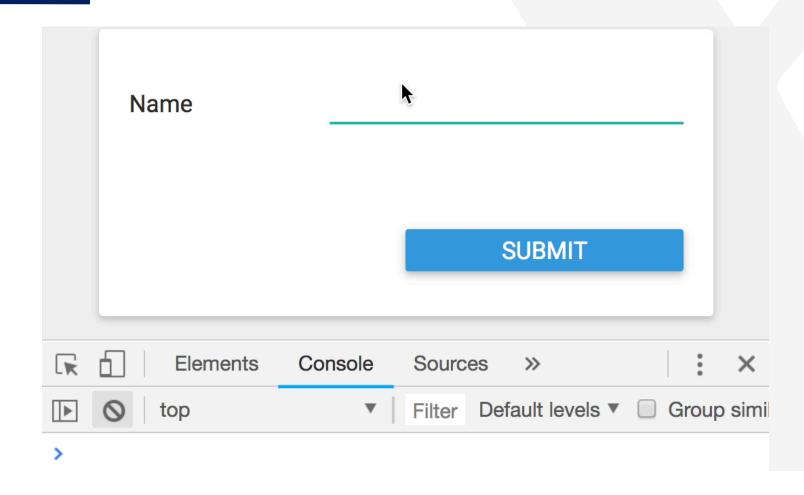


# Model driven - ReactiveFormsModule

- FormControl
- FormGroup
- FormArray
- FormBuilder



# **Setup - Scenario**





# FormGroup & FormControl

```
@Component({ selector: 'contacts' })
export class ContactsComponent {
 contactForm = new FormGroup({
     name: new FormControl()
 });
  submit() {
    console.log(
      this.contactForm.value
```

```
<form [formGroup]="contactForm"</pre>
      (ngSubmit)="submit()">
  <input
    formControlName="name" />
  <button type="submit">
    Submit
  </button>
</form>
```

# FormGroup linked with <form>

```
@Component({ selector: 'contacts' })
export class ContactsComponent {
 contactForm = new FormGroup({
     name: new FormControl()
 });
  submit() {
    console.log(
      this contactForm value
```

```
<form [formGroup]="contactForm"</pre>
      (ngSubmit)="submit()">
  <input
    formControlName="name" />
  <button type="submit">
    Submit
  </button>
</form>
```

# FormControl linked by formControlName to any input

```
@Component({ selector: 'contacts' })
export class ContactsComponent {
 contactForm = new FormGroup({
     name: new FormControl()
 });
  submit() {
    console.log(
      this contactForm value
```

```
<form [formGroup]="contactForm"</pre>
      (ngSubmit)="submit()">
  <input
    formControlName="name" />
  <button type="submit">
    Submit
  </button>
</form>
```

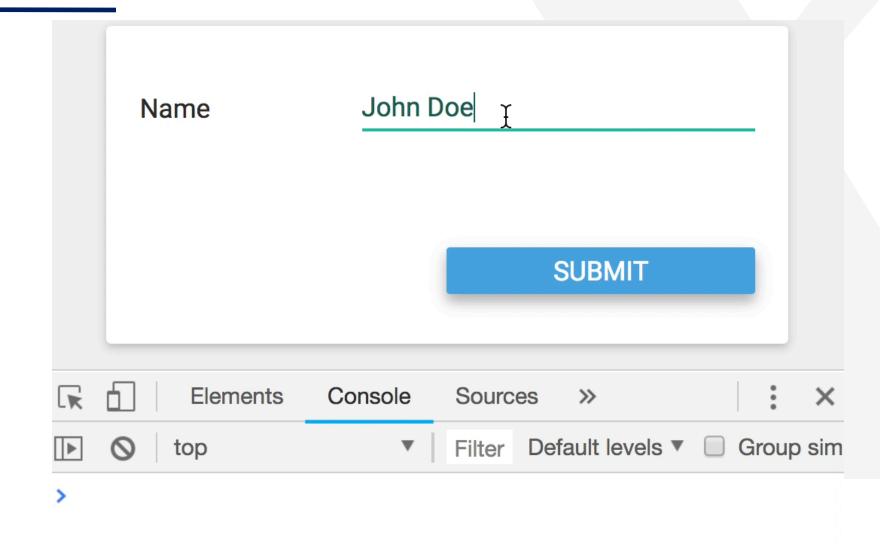


# Catch submit event ( ngSubmit )

```
@Component({ selector: 'contacts' })
export class ContactsComponent {
 contactForm = new FormGroup({
     name: new FormControl()
 });
  submit() {
    console.log(
      this.contactForm.value
```

```
<form [formGroup]="contactForm"</pre>
      (ngSubmit)="submit()">
  <input
    formControlName="name" />
  <button type="submit">
    Submit
  </button>
</form>
```

# **Validation - Scenario**





#### This is what we had...

```
@Component({ selector: 'contacts' })
export class ContactsComponent {
 contactForm = new FormGroup({
     name: new FormControl()
 });
  submit() {
    console.log(
      this contactForm value
```

```
<form [formGroup]="contactForm"</pre>
      (ngSubmit)="submit()">
  <input
    formControlName="name" />
  <button type="submit">
    Send
  </button>
</form>
```

## **Reactive Forms - validation**

#### ... now with validators

```
@Component({ selector: 'contacts' })
export class ContactsComponent {
 contactForm = new FormGroup({
     name: new FormControl('',
            [ Validators.required,
             Validators.minLength(3)]
 });
 get name() { return
   this.contactForm.get('name')
```

```
<form [formGroup]="contactForm"</pre>
      (ngSubmit)="submit()">
  <input
    formControlName="name" />
  <span *ngIf="name.invalid">
    Name is required.
  </span>
</form>
```

## **Reactive Forms - validation**

# Defined on FormControl, linked with formControlName

```
@Component({ selector: 'contacts' })
export class ContactsComponent {
 contactForm = new FormGroup({
     name: new FormControl('',
            [ Validators.required,
             Validators.minLength(3)]
 get name() { return
   this.contactForm.get('name')
```

```
<form [formGroup]="contactForm"</pre>
      (ngSubmit)="submit()">
  <input
    formControlName="name" />
  <span *ngIf="name.invalid">
    Name is required.
  </span>
</form>
```

## **Reactive Forms - validation**

## getters as shortcut for readability

```
@Component({ selector: 'contacts' })
export class ContactsComponent {
 contactForm = new FormGroup({
     name: new FormControl('',
           [ Validators.required,
             Validators.minLength(3)]
 get name() { return
   this.contactForm.get('name')
```

```
<form [formGroup]="contactForm"</pre>
      (ngSubmit)="submit()">
  <input
    formControlName="name" />
  <span *ngIf="name.invalid">
    Name is required.
  </span>
</form>
```

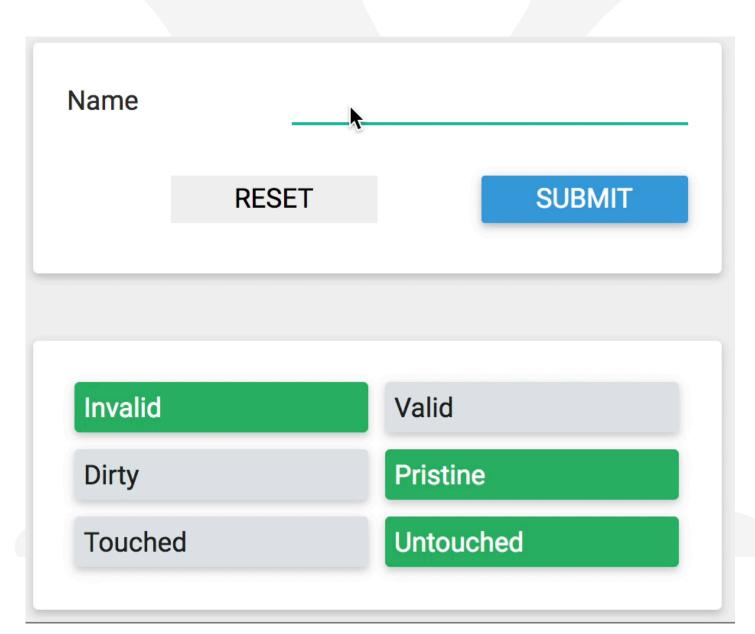
# **Reactive Forms - Validators**

```
• required
• minLength(i)
• pattern(//)
• (c: FormControl): ValidationErrors | null => {
}
```



## Form states

- invalid / valid
- dirty / pristine
- touched / untouched



## Reactive Forms - form builder

# Bigger forms

```
@Component({ selector: 'contacts' })
export class ContactsComponent {
  contactForm: FormGroup;
  ngOnInit() {
      this.contactForm = new FormGroup({
        name: new FormControl('',
             [ Validators.required, Validators.minLength(3)]
         surname: new FormControl('')
             [ Validators.required, Validators.minLength(1)]
         surnameprefix: new FormControl(''),
         address: new FormGroup({
           zipcode: new FormControl('',
             [ Validators.required, Validators.pattern(/.*/)],
              [ validZipCodeBackendValidator ]),
           street: new FormControl('',
              [ Validators.required, Validators.pattern(/.*/)])
         birthdate: new FormControl('',
             [ Validators.required, Validators.minLength(1)]
```



## Reactive Forms - form builder

#### Shorter notation

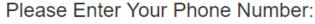
```
@Component({ selector: 'contacts' })
export class ContactsComponent {
  contactForm: FormGroup;
  constructor(private fb: FormBuilder) {}
  ngOnInit() {
    this.contactForm = this.fb.group({
      name: ['', [ Validators.required, Validators.minLength(3) ] ],
       surname: ['', [ Validators.required, Validators.minLength(1)] ],
       surnameprefix: '',
       address: this.fb.group({
         zipcode: ['', [ Validators.required, Validators.pattern(/.*/) ],
                        [ validZipCodeBackendValidator ] ],
         street: ['', [ Validators.required, Validators.pattern(/.*/) ] ]
      birthdate: ['',[ Validators.required, Validators.minLength(1) ] ]
});
```



- FormGroup, FormControl
- Validation
- Form states
- FormBuilder



# Code



```
216 ▼ 410 ▼ 0000 ▼
             0000 -
             0001
 Next
             0002
             0003
             0004
             0005
             0006
             0007
             0008
             0009
             0010
             0011
             0012
             0013
             0014
             0015
             0016
             0017
             0018
             0019 🕶
```

```
Elements Console Sources Network Timeline
        <option value="0481">0481</option>
        <option value="0482">0482</option>
        <option value="0483">0483</option>
        <option value="0484">0484</option>
        <option value="0485">0485</option>
        <option value="0486">0486</option>
        <option value="0487">0487</option>
        <option value="0488">0488</option>
        <option value="0489">0489</option>
        <option value="0490">0490</option>
        <option value="0491">0491</option>
        <option value="0492">0492</option>
        <option value="0493">0493</option>
        <option value="0494">0494</option>
        <option value="0495">0495</option>
        <option value="0496">0496</option>
        <option value="0497">0497</option>
        <option value="0498">0498</option>
        <option value="0499">0499</option>
         Show All Nodes (9499 More)
      </select>
      <br>
      <button id="next" class="btn btn-
      primary">Next</button>
      ::after
    </div>
  </body>
\div id="coFrameDiv" style="height:0px;display:none;">
...</div>
</html>
html body div.container select
```

