# Angular Reactive Form Demo

## 1. Setup Project

### 1.1 Install Dependencies

1. Change directory to calab:

* cd calab

1. Install dependencies by running the following command:

* npm install

## 2. Create a Form Component

### 2.1 Create a New Component

1. Create a new component using CLI and name it OrderForm:

* npx -p @angular/cli ng generate component components/order-form

1. Import OrderForm into src/app/app.component.ts:

* import { MyComponentComponent } from './my-component/my-component.component';

1. Inside src/app/app.component.ts update imports to include OrderForm:

* imports: [RouterOutlet, OrderFormComponent],

1. Open src/app/app.component.html template and add the following element after the <div class="divider"...:

* <app-order-form></app-order-form>

## 3. Setup Reactive Form in OrderFormComponent

### 3.1 Update Component

1. Open src/app/components/order-form/order-form.component.ts file and do the following:
   * Update imports to include ReactiveFormsModule:
   * @Component({  
      ...  
      imports: [ReactiveFormsModule],  
      ...  
     })
   * Create FormControl instancess inside your OrderFormComponent.
   * product = new FormControl('');  
     quantity = new FormControl('');
2. Open src/app/components/order-form/order-form.component.html file and do the following:
   * Register controls in the template.
   * <label for="product">Product: </label>  
     <input id="product" type="text" [formControl]="product">  
       
     <label for="quantity">Quantity: </label>  
     <input id="quantity" type="text" [formControl]="quantity">
3. Open src/app/components/order-form/order-form.component.css file and do the following:
   * Add some style.
   * /\* Style inputs \*/  
     input, select {  
      width: 100%;  
      padding: 12px 20px;  
      margin: 8px 0;  
      display: inline-block;  
      border: 1px solid #ccc;  
      border-radius: 4px;  
      box-sizing: border-box;  
     }  
       
     /\* Style the submit button \*/  
     button[type=submit] {  
      width: 100%;  
      background-color: #04AA6D;  
      color: white;  
      padding: 14px 20px;  
      margin: 8px 0;  
      border: none;  
      border-radius: 4px;  
      cursor: pointer;  
     }  
       
     /\* Add a background color to the submit button on mouse-over \*/  
     button[type=submit]:hover {  
      background-color: #45a049;  
     }  
       
     .ng-valid[required], .ng-valid.required {  
      border-left: 5px solid #42A948; /\* green \*/  
     }  
     .ng-invalid:not(form) {  
      border-left: 5px solid #a94442; /\* red \*/  
     }

## 4. Displaying a Form Control Value

### 4.1 Update Component

1. Open src/app/components/order-form/order-form.component.ts file and do the following:
   * Subscribe to the valueChanges observable:
   * ngOnInit() {   
      this.product.valueChanges.subscribe(data => console.log(`Product updated to ${data}`))  
      this.quantity.valueChanges.subscribe(data => console.log(`Quantity updated to ${data}`))  
     }
2. Open src/app/components/order-form/order-form.component.html file and do the following:
   * Access the current value directly through the value property.
   * <p>Current Product: {{product.value}}</p>  
     <p>Current Quantity: {{quantity.value}}</p>

## 5. Grouping Form Controls

### 5.1 Create A FormGroup Instance

1. Open src/app/components/order-form/order-form.component.ts file and do the following:
   * Create a FormGroup instance. Replace current code in OrderFormComponent class with the following:
   * orderForm = new FormGroup({  
      product: new FormControl(''),  
      quantity: new FormControl(''),  
     });
2. Open src/app/components/order-form/order-form.component.html file and do the following:
   * Associate The FormGroup Model And View. Replace current HTML code with the following:
   * <form [formGroup]="orderForm">  
      <label for="product">Product: </label>  
      <input id="product" type="text" formControlName="product">  
      <label for="quantity">Quantity: </label>  
      <input id="quantity" type="text" formControlName="quantity">  
     </form>

### 5.2 Save form Data

1. Open src/app/components/order-form/order-form.component.ts file and do the following:
   * create an onSubmit() callback method, allowing you to process the captured form data as needed:
   * onSubmit() {  
      // TODO: Use EventEmitter with form value  
      console.warn(this.orderForm.value);  
     }
2. Open src/app/components/order-form/order-form.component.html file and do the following:
   * Add an ngSubmit event listener to the form tag with the onSubmit() callback method:
   * <form [formGroup]="orderForm" (submit)="onSubmit()">  
      ...  
     </form>
   * Use button within
   * element to trigger Submit event:
   * <form [formGroup]="orderForm" (submit)="onSubmit()">  
      ... // form elements  
      <button type="submit">Submit</button>  
     </form>

### 5.3 Use FormBuilder Service

1. Open src/app/components/order-form/order-form.component.ts file and do the following:
   * Inject the FormBuilder service into your component using dependency injection:
   * constructor(private formBuilder: FormBuilder){}
   * Replace FormGroup with FormBuilder:
   * orderForm = this.formBuilder.group({  
      product: [''],  
      quantity: [''],  
     });

### 5.4 Define A FormArray Control

1. Open src/app/components/order-form/order-form.component.ts file and do the following:
   * Add new property to your existing FormGroup:
   * instructions: this.formBuilder.array([this.formBuilder.control('')])
   * Create a getter that provides an efficient way to access the values in your form array instance:
   * get instructions() {  
      return this.orderForm.get('instructions') as FormArray;  
     }
   * Define a method to dynamically add an alias control to your form array:
   * addInstruction(){  
      this.instructions.push(this.formBuilder.control(''));  
     }
2. Open src/app/components/order-form/order-form.component.html file and do the following:
   * Integrate the form model’s interests into your template:

* <form [formGroup]="orderForm" (submit)="onSubmit()">  
   ... //form controls  
   <div formArrayName="instructions">  
   <h2>Instructions</h2>  
   <button type="button" (click)="addInstruction()">+ Add Instruction</button>  
   @for (instruction of instructions.controls; track instruction; let i = $index) {  
   <div>  
   <label for="instruction-{{ i }}">Instruction:</label>  
   <input id="instruction-{{ i }}" type="text" [formControlName]="i">  
   </div>  
   }  
   </div>  
   ... //submit button  
  </form>

## 6. Reactive Form Validation

### 6.1 Built-in Validators

1. Open src/app/components/order-form/order-form.component.ts file and do the following:
   * Add required validators to product and quantity form controls. Update orderForm with the following:
   * orderForm = this.formBuilder.group({  
      product: ['', Validators.required],  
      quantity: ['', Validators.required],  
      instructions: this.formBuilder.array([this.formBuilder.control('')])  
     });

### 6.2 Custom Validators

1. Open src/app/components/order-form/order-form.component.ts file and do the following:
   * Create a new method that validates agains list of forbidden product names:
   * forbiddenNameValidator(): ValidatorFn {  
      return (control: AbstractControl): ValidationErrors | null => {  
      const list = ['Lightsaber', 'Millennium Falcon'];  
      const forbidden = list.includes(control.value);  
      return forbidden ? {forbiddenName: {value: control.value}} : null;  
      };  
     }
   * Add forbiddenNameValidator to product form controls.
   * product: ['', [Validators.required, this.forbiddenNameValidator()]],

### 6.3 Review Changes

1. Start Angular Development Server if not yet started:

* npx -p @angular/cli ng serve --host 0.0.0.0
* *Otherwise refresh the browser tab to see updated view.*