

Angular Forms mit Bootstrap

Firstname

ha

MinLength is 3

Lastname

llo|

☐ Subscribe to mailinglist

ContactMethod

Email ▼

☐ Email

☒ Phone

```
{ "firstName": "ha", "lastName": "llo", "isSubscribed": "",  
  "contactMethod": "1", "rbContacMethod": 2 }
```

Submit

Template mit zen-coding anlegen

```
<> pupil-form.component.html ●  
1  form>div>label+input.form-control
```

- Elemente entsprechend ergänzen

```
<> pupil-form.component.html ●  
1  <form action="">  
2    <div class="form-group">  
3      <label for="name"></label>  
4      <input id="name" type="text" class="form-control">  
5    </div>  
6  </form>
```

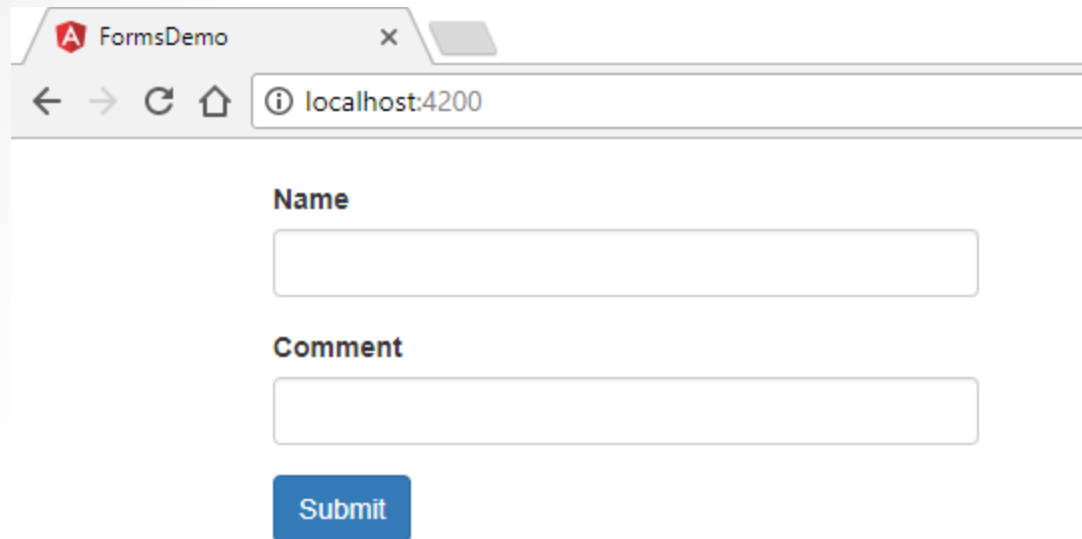
Zweites Textfeld inklusive Attributen

```
div.form-group>label[for='description']+input[id=description].form-control
```

<> pupil-form.component.html ✕

```
1  <div class="col-sm-4 col-sm-push-1">
2      <form action="">
3          <div class="form-group">
4              <label for="name">Name</label>
5              <input id="name" type="text" class="form-control">
6          </div>
7          <div class="form-group">
8              <label for="description">Comment</label>
9              <input type="text" id="description" class="form-control">
10         </div>
11         <button class="btn btn-primary">Submit</button>
12     </form>
13 </div>
```

Erstes Ergebnis



A screenshot of a web browser window. The tab is labeled 'FormsDemo' with a red 'A' icon. The address bar shows 'localhost:4200'. The page content includes a form with two input fields. The first field is labeled 'Name' and is empty. The second field is labeled 'Comment' and is also empty. Below the fields is a blue button labeled 'Submit'.

Name

Comment

Submit

Submit → Postback zum Server

- Angular soll den Submit abfangen und clientseitig verarbeiten
- NgFormModule einbinden → Angular verwaltet per Default alle Formulare
- <https://angular.io/api/forms/NgForm>



The screenshot shows a web browser interface. At the top, the address bar contains the URL `localhost:4200/?firstName=&lastName=`, which is highlighted with a red rectangular border. Below the address bar, there are two text input fields. The first field is labeled "Firstname" and the second field is labeled "Lastname". Both fields are currently empty. Below these fields is a blue button with the text "Submit".

FormsModule registrieren

```
app.module.ts x
1  import { BrowserModule } from '@angular/platform-browser';
2  import { FormsModule } from '@angular/forms';
3  import { NgModule } from '@angular/core';
4
5
6  import { AppComponent } from './app.component';
7  import { PupilFormComponent } from './pupil-form/pupil-form.component';
8
9
10 @NgModule({
11   declarations: [
12     AppComponent,
13     PupilFormComponent
14   ],
15   imports: [
16     BrowserModule,
17     FormsModule
18   ],
19   providers: [],
20   bootstrap: [AppComponent]
21 })
22 export class AppModule { }
```

Kein Postback mehr zum Server

- Submit liefert kein Ergebnis
 - EventHandler registrieren

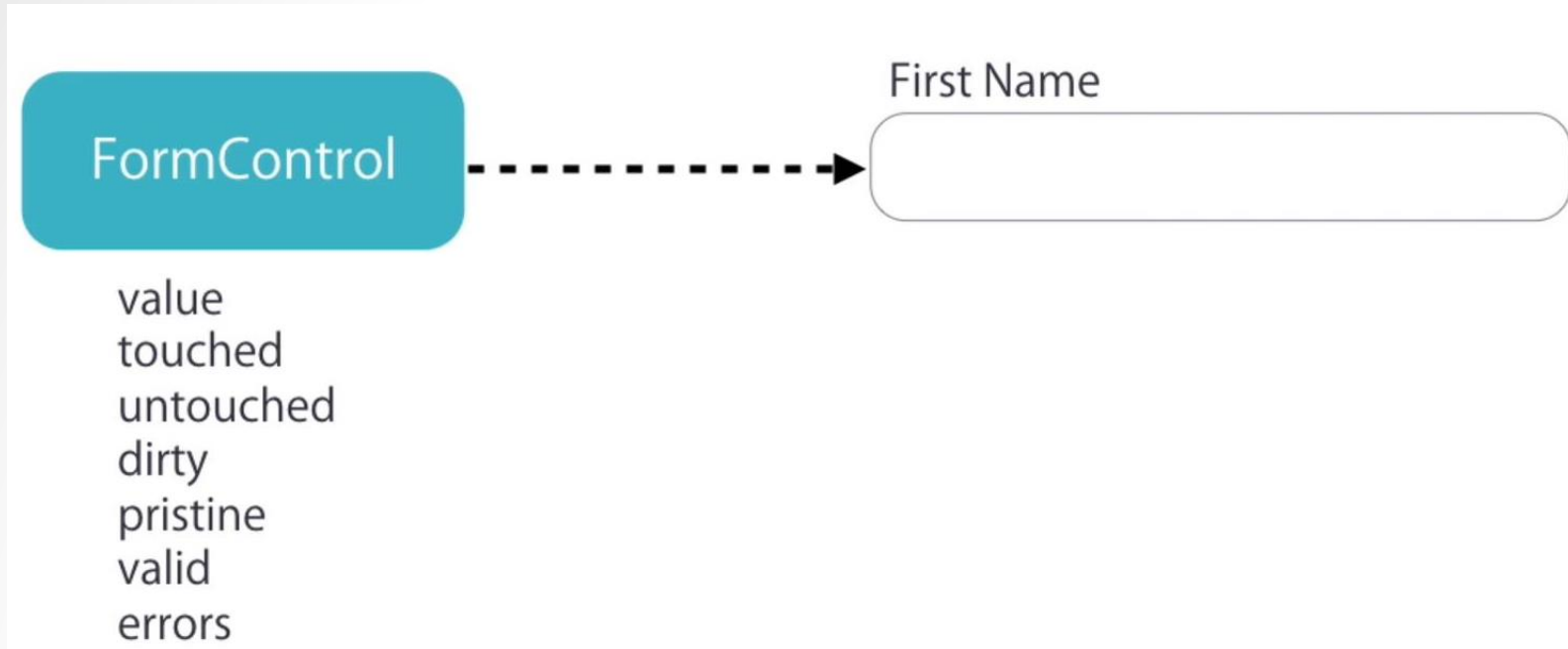
📘 localhost:4200

Firstname

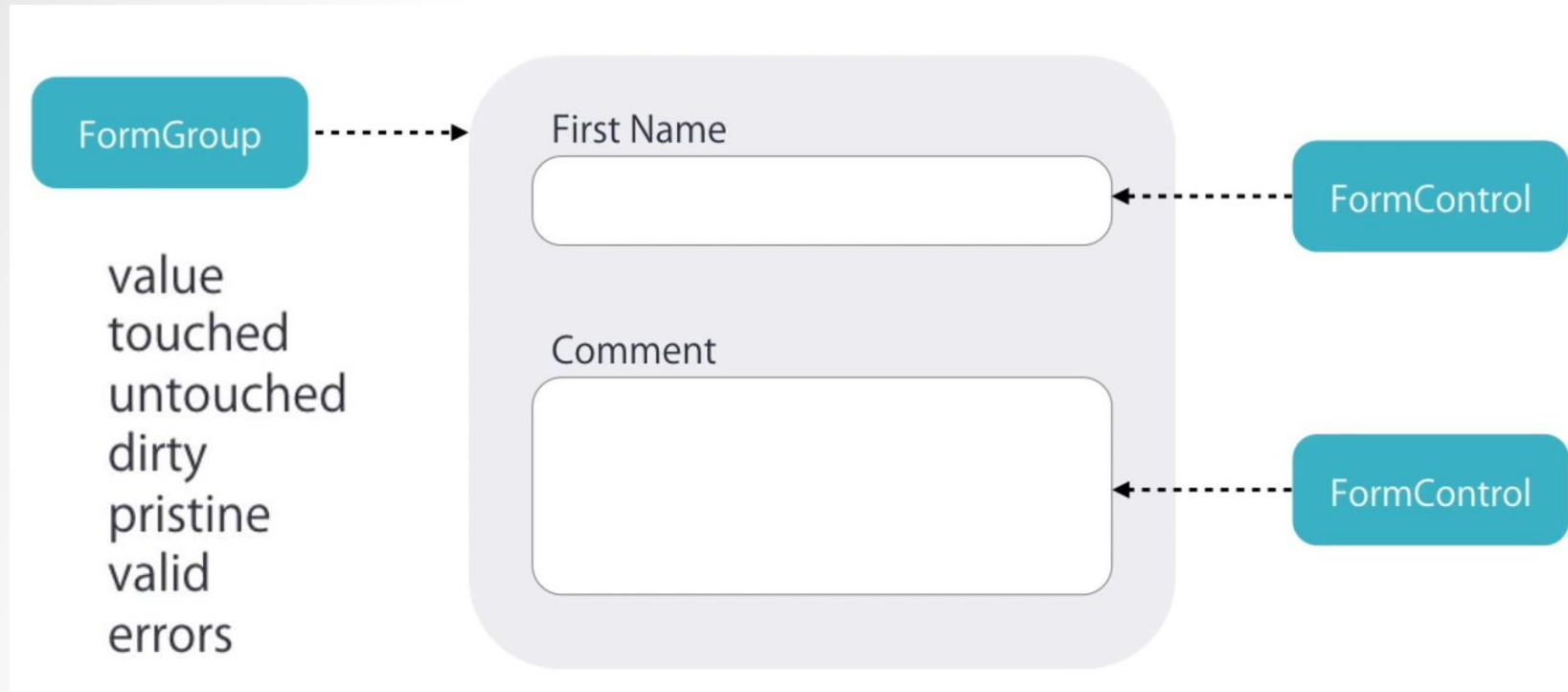
Lastname

Submit

FormControl kapselt Control



FormGroup kapselt Gruppe von Controls



Forms in Angular implementieren



The diagram consists of two rounded rectangular boxes side-by-side. The left box is teal and contains the word 'Directives'. Below it is the text 'Template-driven'. The right box is light green and contains the word 'Code'. Below it is the text 'Reactive'.

Directives

Template-driven

Code

Reactive

Values aus Controls auslesen → ngModel

- Bekannte Directive (two-way-binding)
- DatenControls Property ngModel hinzufügen
 - name zur Identifizierung
- Als Templatevariable zugreifbar machen → Validation

```
<div class="form-group">  
  <label for="firstName">Firstname</label>  
  <input  
    ngModel  
    #firstName="ngModel"  
    id="firstName"  
    name="firstName"  
    type="text"  
    class="form-control"/>  
</div>
```

NgModel

- Definiert das Control als „DatenControl“
- Bindingmöglichkeiten
 - PropertyBinding (oneway) zur Initialisierung
 - TwoWay-Binding zur Synchronisierung
- Speichern in lokale Variable
 - Zugriff auf die Properties von ngModel
 - Validierung
- <https://angular.io/api/forms/NgModel>

FormControl evaluieren → ngModel

- Temporäre Variable binden
- onChange → Parameter

```
<input
  ngModel
  name="firstName"
  #firstName="ngModel"
  (change)=onChange(firstName)
  type="text"
  class="form-control"/>
```

```
onChange(firstName){
  console.log(firstName)
}
```

```
▼ NgModel {_parent: NgForm, name: "firstName",
  i
  asyncValidator: (...),
  ▼ control: FormControl
    asyncValidator: null
    dirty: true
    disabled: (...)
    enabled: (...)
    errors: null
    invalid: (...)
    parent: (...)
    pending: (...)
    pristine: false
    root: (...)
    status: "VALID"
    ► statusChanges: EventEmitter {_isScalar: 1
      touched: true
      untouched: false
      updateOn: (...)
      valid: true
      validator: null
      value: "Max"
```

Validierung – Html5-Attribute required

- Validation div
 - Wenn Element geändert und nicht valide ist

Firstname

Firstname is required

Lastname

Submit

```
<div class="form-group">
  <label for="firstName">Firstname</label>
  <input
    required
    ngModel
    name="firstName"
    #firstName="ngModel"
    (change)=onChange(firstName)
    type="text"
    class="form-control"/>
  <div
    class="alert alert-danger"
    *ngIf="firstName.touched && !firstName.valid">
    Firstname is required
  </div>
</div>
```

Selektive Fehlermeldungen

```
<div class="form-group">
  <label for="firstName">Firstname</label>
  <input
    required
    minlength="3"
    ngModel
    name="firstName"
    #firstName="ngModel"
    (change)="onChange(firstName)"
    id="firstName"
    type="text"
    class="form-control"/>
  <div
    class="alert alert-danger"
    *ngIf="firstName.touched && !firstName.valid">
    <div *ngIf="firstName.errors.required" >
      FirstName is required
    </div>
    <div *ngIf="firstName.errors.minlength" >
      MinLength is {{ firstName.errors.minlength.requiredLength }}
    </div>
  </div>
</div>
```

Firstname

FirstName is required

Firstname

MinLength is 3

Ohne Typsicherheit

- firstName ist vom Typ object
 - Dank JS trotzdem lauffähig

```
class="form-  
<div  
class="alert  
*ngIf="first  
<div *ngIf="firstName.errors.required" >
```

[Angular] Identifier 'required' is not defined. '__type' does not contain such a member

property errors of AbstractControlDirective

Validierung über css formatieren

Firstname

FirstName is required

Lastname

LastName is required

☐ Subscribe to mailinglist

ContactMethod

☐ Email
☐ phone

```
{ "firstName": "", "lastName": "", "isSubscribed": "",  
  "contactMethod": "contactMethod", "rbContactMethod": ""  
}
```

Submit

pupil-form.component.css

```
1  .invalid{  
2    color: red;  
3  }  
4  
5  input.ng-invalid.ng-touched{  
6    border: 1px solid red  
7  }
```

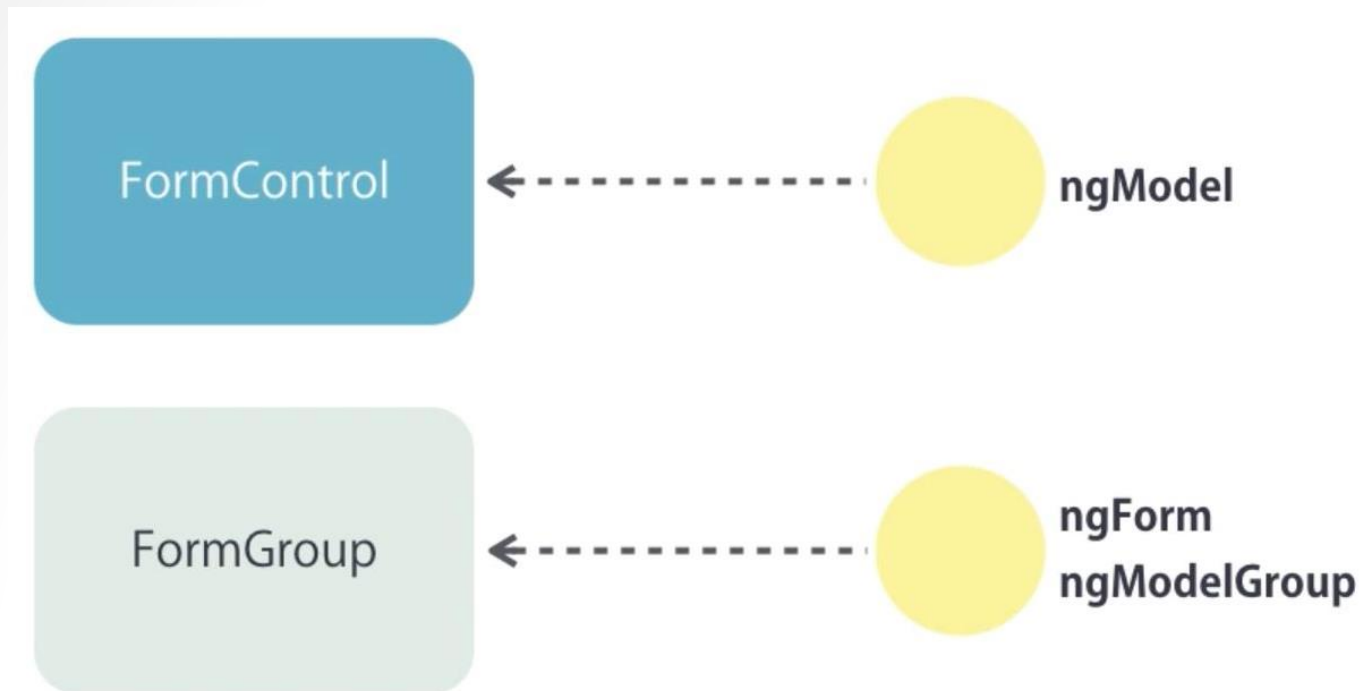
<span

class="invalid"

```
*ngIf="firstName.touched && !firstName.valid">  
<span *ngIf="firstName.errors.required" >  
  FirstName is required  
</span>  
<span *ngIf="firstName.errors.minlength" >  
  MinLength is {{ firstName.errors.minlength.requ  
</span>  
</span>
```

FormGroup

- FormGroup fasst mehrere FormControl zusammen
- Directive ngForm bietet das Outputproperty submit() an



EventBinding auf submit

```
<form (submit)=onSubmit($event)>
```

```
export class PupilFormComponent {
```

```
  onSubmit(event){  
    | console.log(event);  
  }
```

```
▼ Event {isTrusted: true, type: "submit", target: form.  
  bubbles: true  
  cancelBubble: false  
  cancelable: true  
  composed: false  
  currentTarget: null  
  defaultPrevented: true  
  eventPhase: 0  
  isTrusted: true  
  ▶ path: (10) [form.ng-untouched.ng-pristine.ng-valid,  
    returnValue: false  
  ▶ srcElement: form.ng-untouched.ng-pristine.ng-valid  
  ▶ target: form.ng-untouched.ng-pristine.ng-valid  
    timeStamp: 8894.09  
    type: "submit"  
  ▶ __proto__: Event
```

NgForms für gesamtes Formular

- Hat alle wichtigen Statusinfos
 - valid
 - touched
 - ...
- Liefert Objekt als value

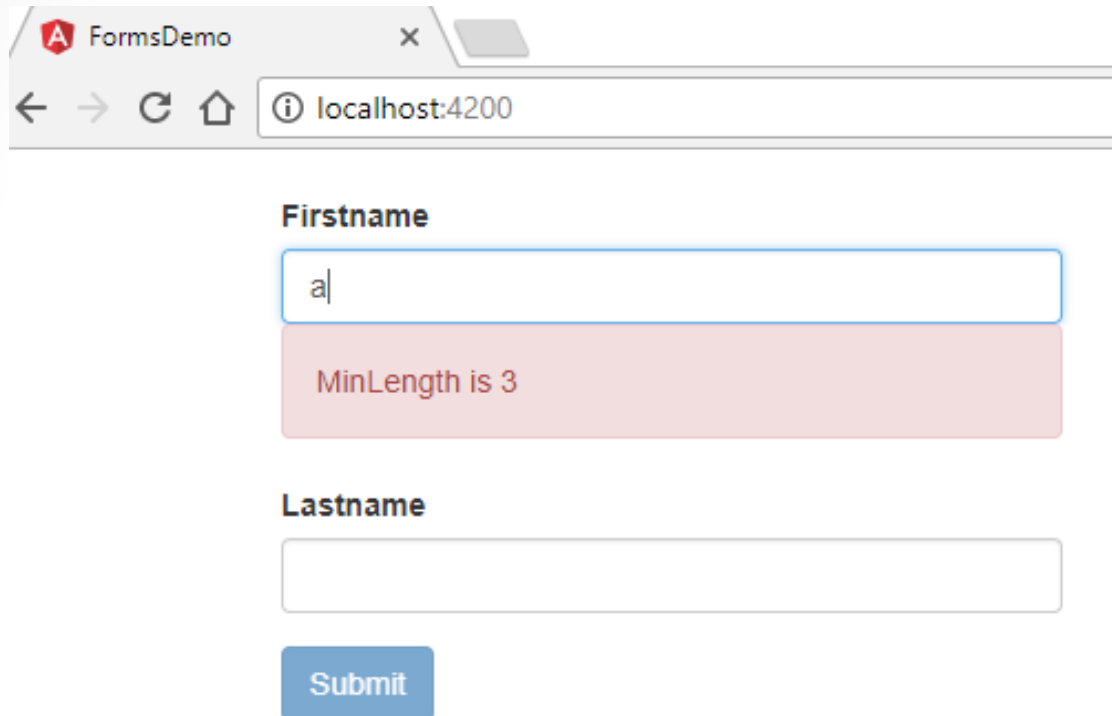
```
<form
  #f="ngForm"
  (submit)=onSubmit(f)>
```

```
▼ {firstName: "dsasad", lastName: "sdsad"} ⓘ
  firstName: "dsasad"
  lastName: "sdsad"
  ► __proto__: Object
```

```
▼ NgForm ⓘ
  control: (...)
  controls: (...)
  dirty: (...)
  disabled: (...)
  enabled: (...)
  errors: (...)
  ► form: FormGroup {validator: nu
    formDirective: (...)
    invalid: (...)
  ► ngSubmit: EventEmitter {_isSca
    path: (...)
    pending: (...)
    pristine: (...)
    status: (...)
    statusChanges: (...)
    submitted: true
    touched: true
    untouched: (...)
    valid: false
    value: (...)
    valueChanges: (...)
  ► _directives: [NgModel]
  ► __proto__: ControlContainer
```

Submit-Button disablen

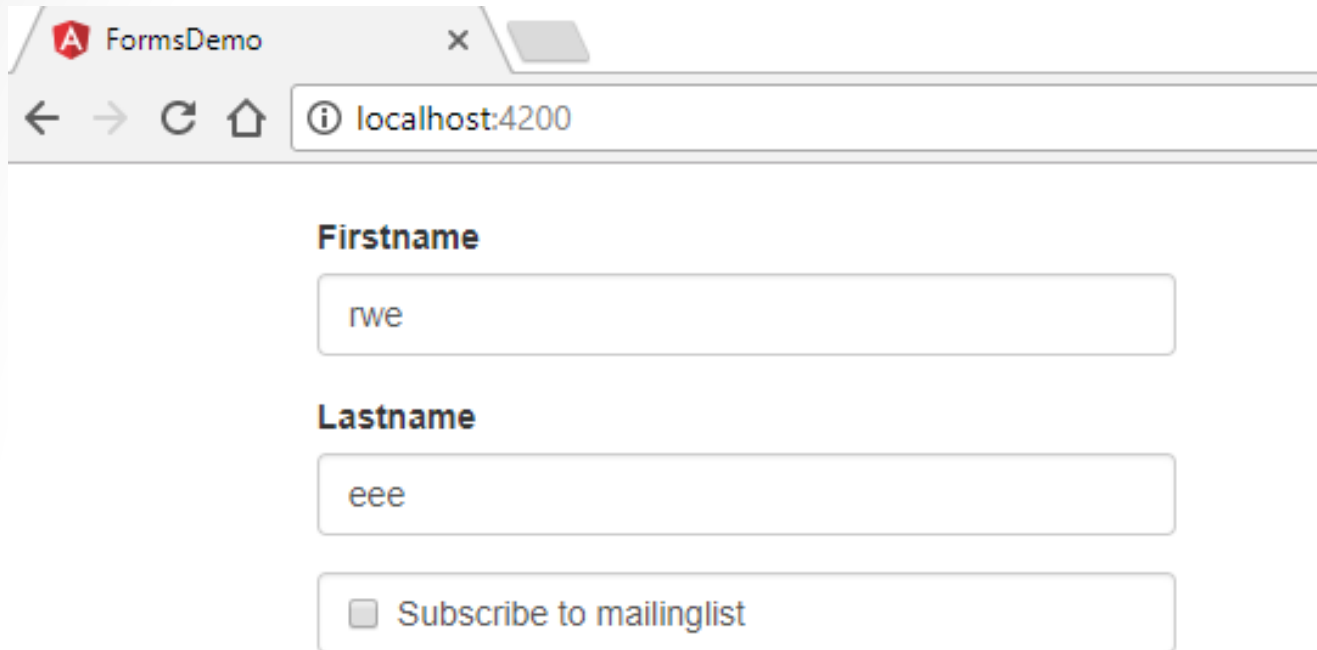
```
<button [disabled]="!f.valid"
```



The screenshot shows a web browser window titled "FormsDemo" with the address bar displaying "localhost:4200". The form contains two input fields: "Firstname" and "Lastname". The "Firstname" field contains the letter "a" and has a red error message "MinLength is 3" below it. The "Lastname" field is empty. Below the fields is a blue "Submit" button, which is disabled due to the validation error.

Beispiel CheckBox

```
<div class="checkbox form-control">
  <label >
    <input type="checkbox" ngModel name="isSubscribed">
    Subscribe to mailinglist
  </label>
</div>
```



FormsDemo

← → ↻ 🏠 ⓘ localhost:4200

Firstname

rwe

Lastname

eee

☐ Subscribe to mailinglist

DropDownListbox

```
<div class="form-group">
  <label for=" ContacMethod">ContactMethod</label>
  <select ngModel="contactMethod"
    name="contactMethod"
    id="contactMethod"
    class="form-control">
    <option value=""></option>
    <option *ngFor="let method of contactMethods"
      | value={{method.id}}>
      {{method.name}}
    </option>
  </select>
</div>
```

Firstname

rwe

Lastname

eee

☐ Subscribe to mailinglist

ContactMethod

Phone

Email

Phone

{ "firstName": "rwe", "lastName": "eee", "isSubscribed": false, "contactMethod": "2", "rbContacMethod": 1 }

Submit

Radiobuttons

```
<div
  class="radio"
  *ngFor="let rbMethod of contactMethods">
  <label>
    <input ngModel
      type="radio"
      name="rbContacMethod"
      [value]="rbMethod.id" />
    {{rbMethod.name}}
  <br/>
  </label>
</div>
```

Firstname

ha

MinLength is 3

Lastname

llo|

☐ Subscribe to mailinglist

ContactMethod

Email ▼

- ☐ Email
☒ Phone

{ "firstName": "ha", "lastName": "llo", "isSubscribed": "",
"contactMethod": "1", "rbContacMethod": 2 }

Submit