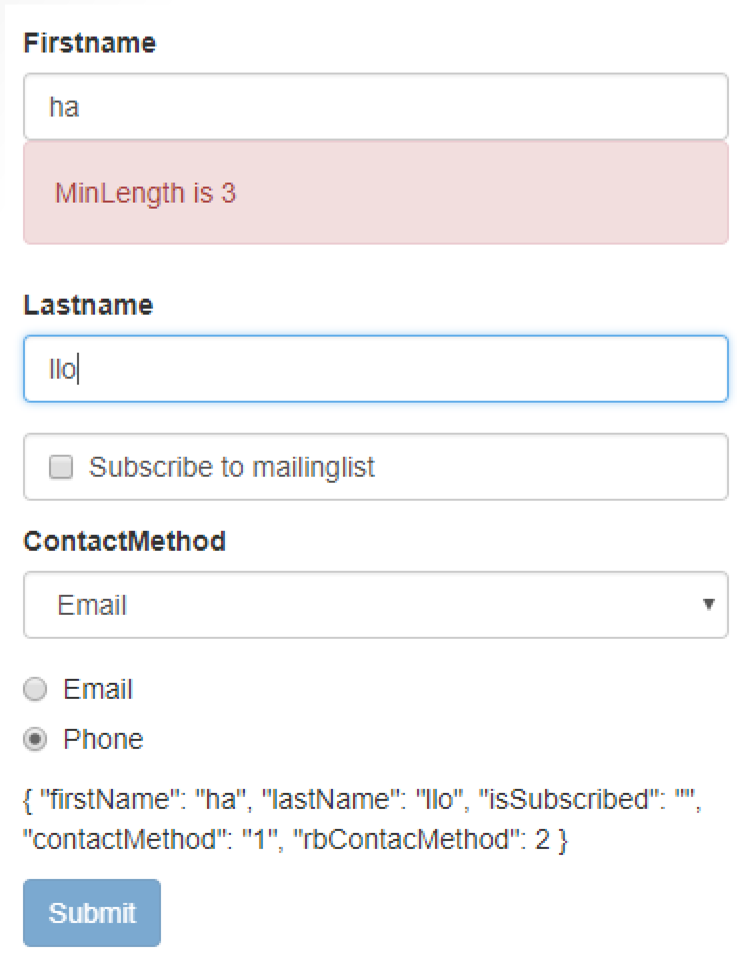
Angular Forms mit Bootstrap



Firstname

ha

MinLength is 3

Lastname

llo

Subscribe to mailinglist

ContactMethod

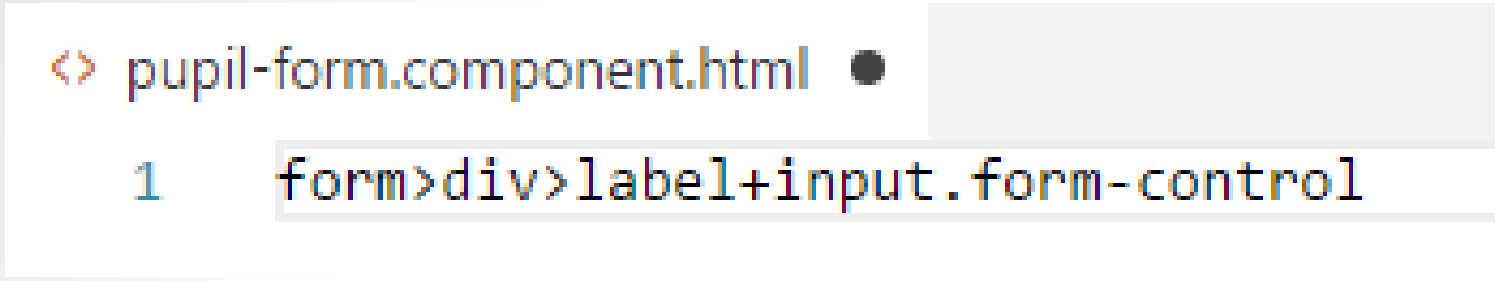
Email

Email

Phone

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

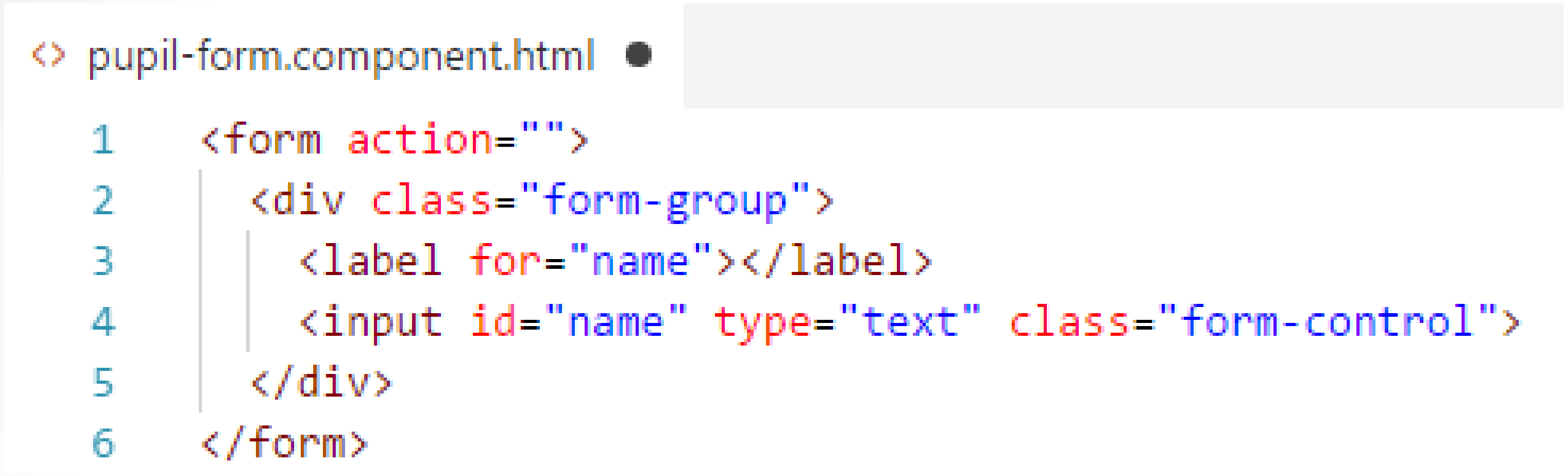
Template mit zen-coding anlegen



pupil-form.component.html ●

```
form>div>label+input.form-control
```

- Elemente entsprechend ergänzen

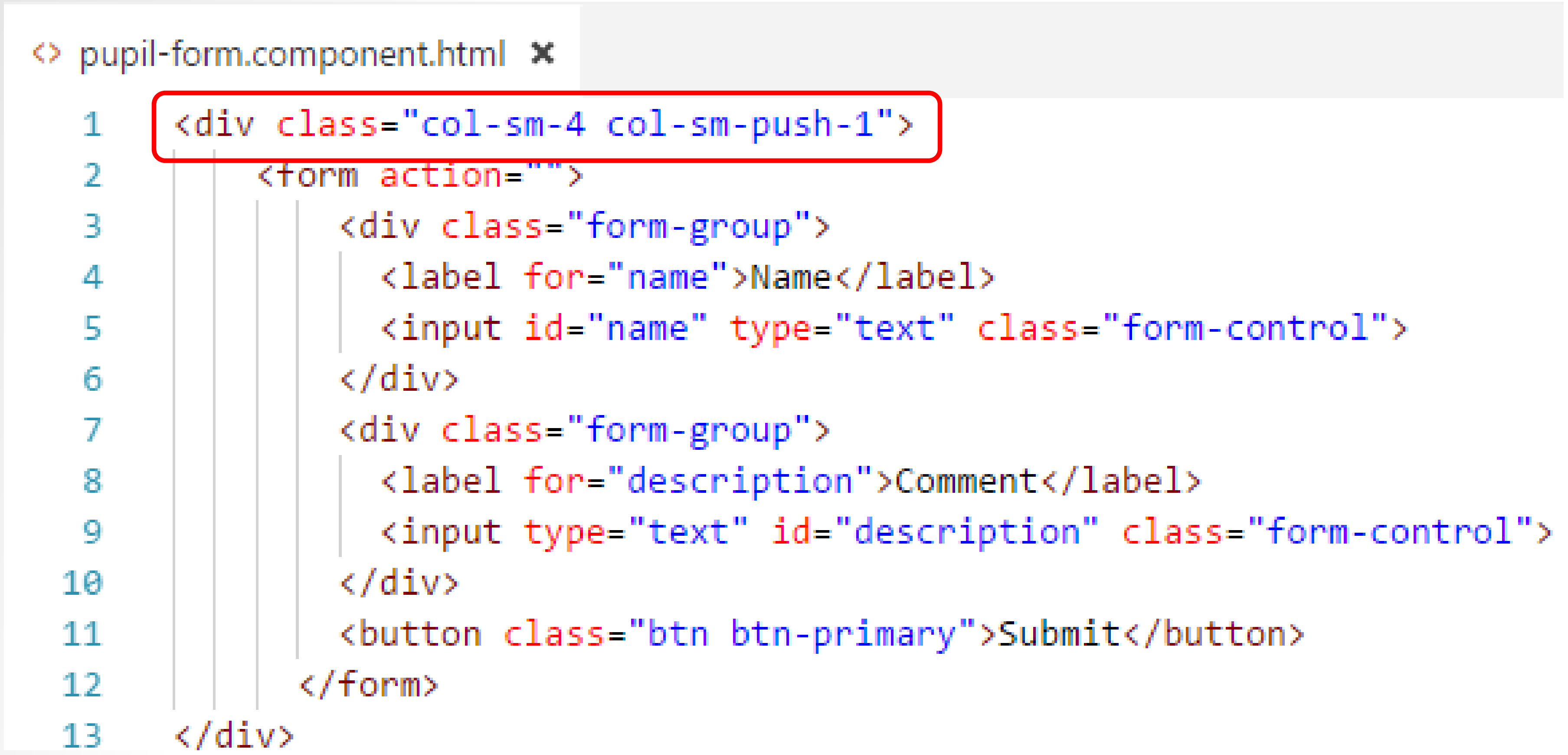


pupil-form.component.html ●

```
<form action="">
  <div class="form-group">
    <label for="name"></label>
    <input id="name" type="text" class="form-control">
  </div>
</form>
```

Zweites Textfeld inklusive Attributen

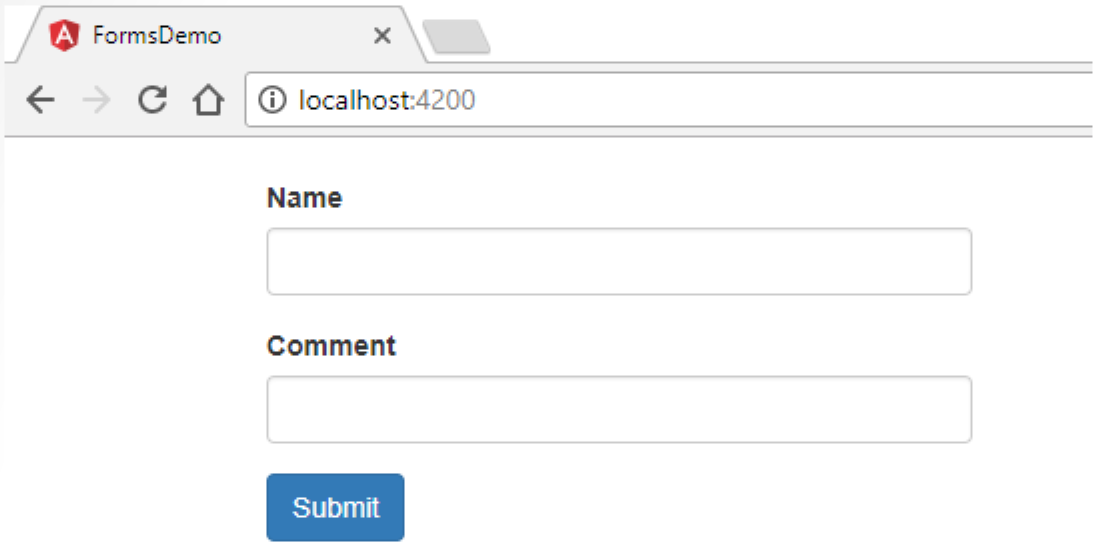
```
div.form-group>label[for='description']+input[id=description].form-control
```



pupil-form.component.html

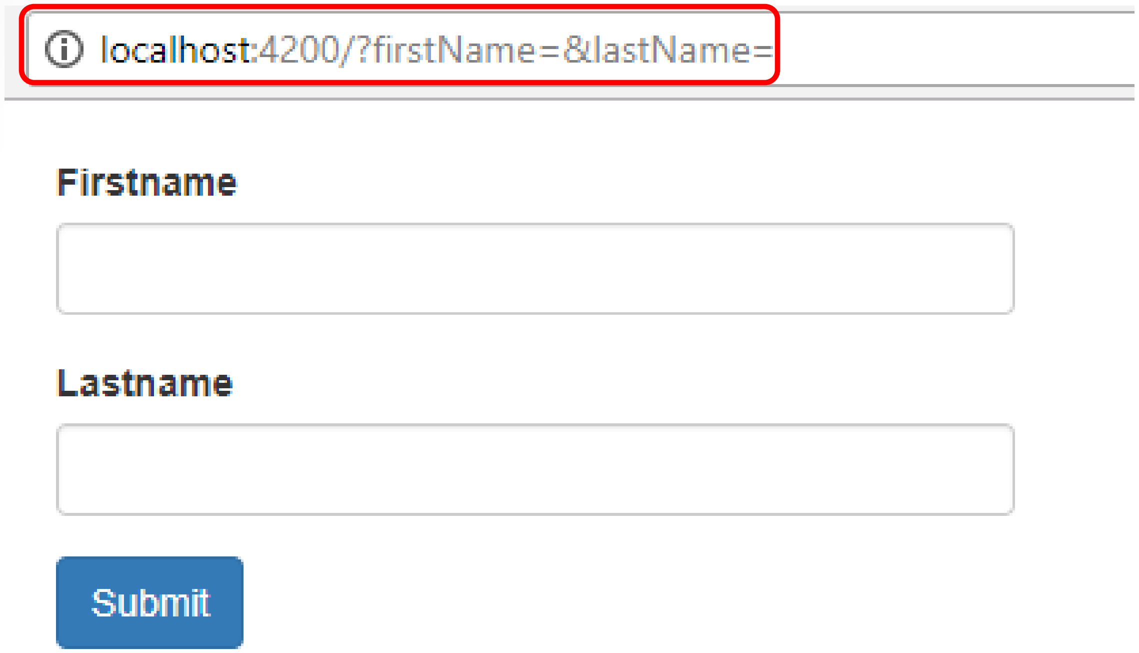
```
<div class="col-sm-4 col-sm-push-1">
    <form action="">
        <div class="form-group">
            <label for="name">Name</label>
            <input id="name" type="text" class="form-control">
        </div>
        <div class="form-group">
            <label for="description">Comment</label>
            <input type="text" id="description" class="form-control">
        </div>
        <button class="btn btn-primary">Submit</button>
    </form>
</div>
```

Erstes Ergebnis



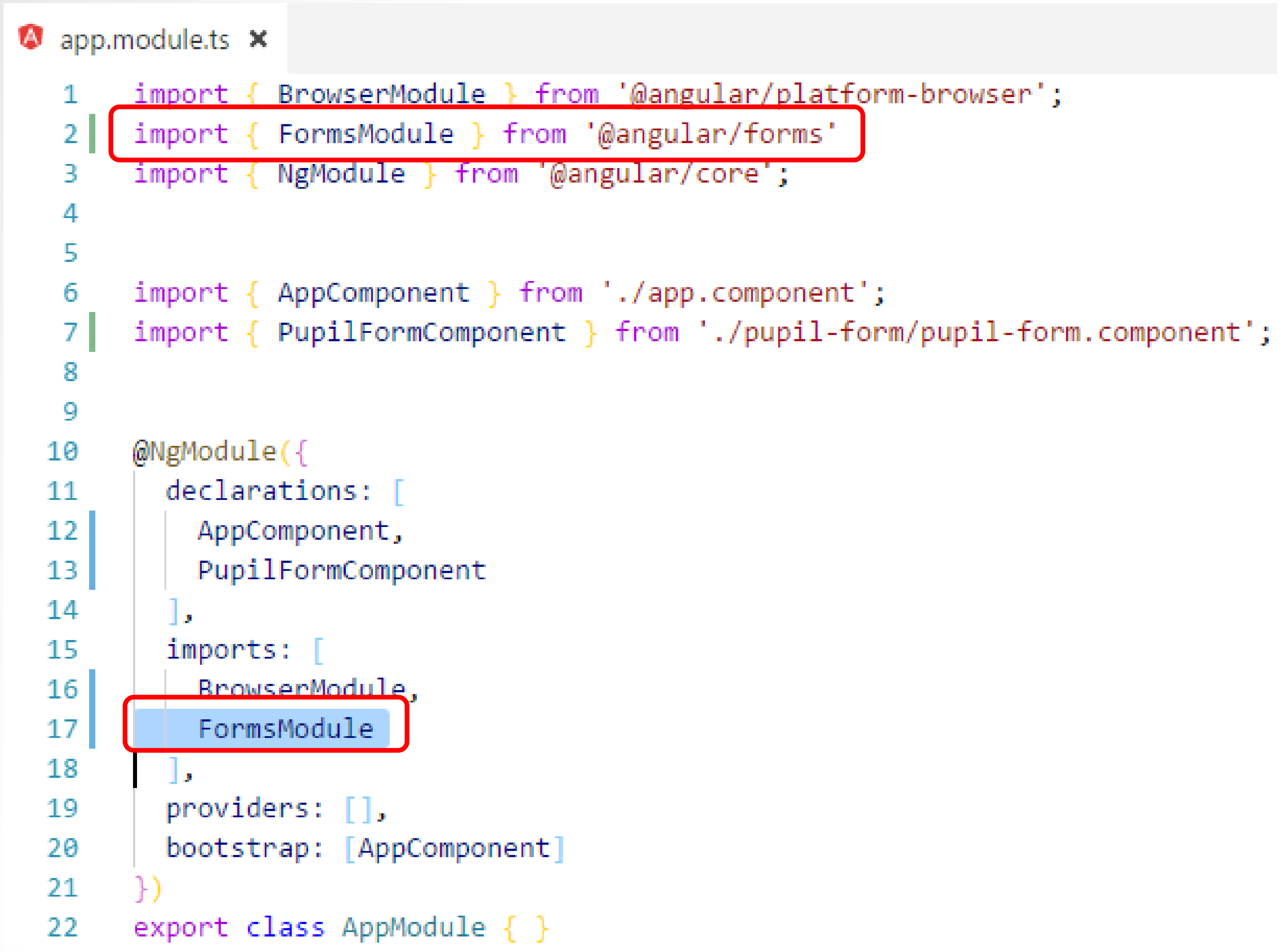
Submit ➔ Postback zum Server

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


localhost:4200/?firstName=&lastName=

HTL LEONDING

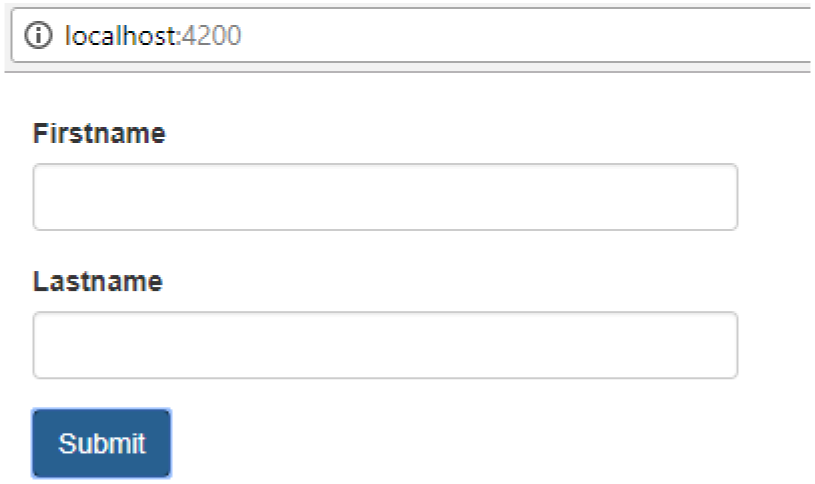
FormsModule registrieren


app.module.ts

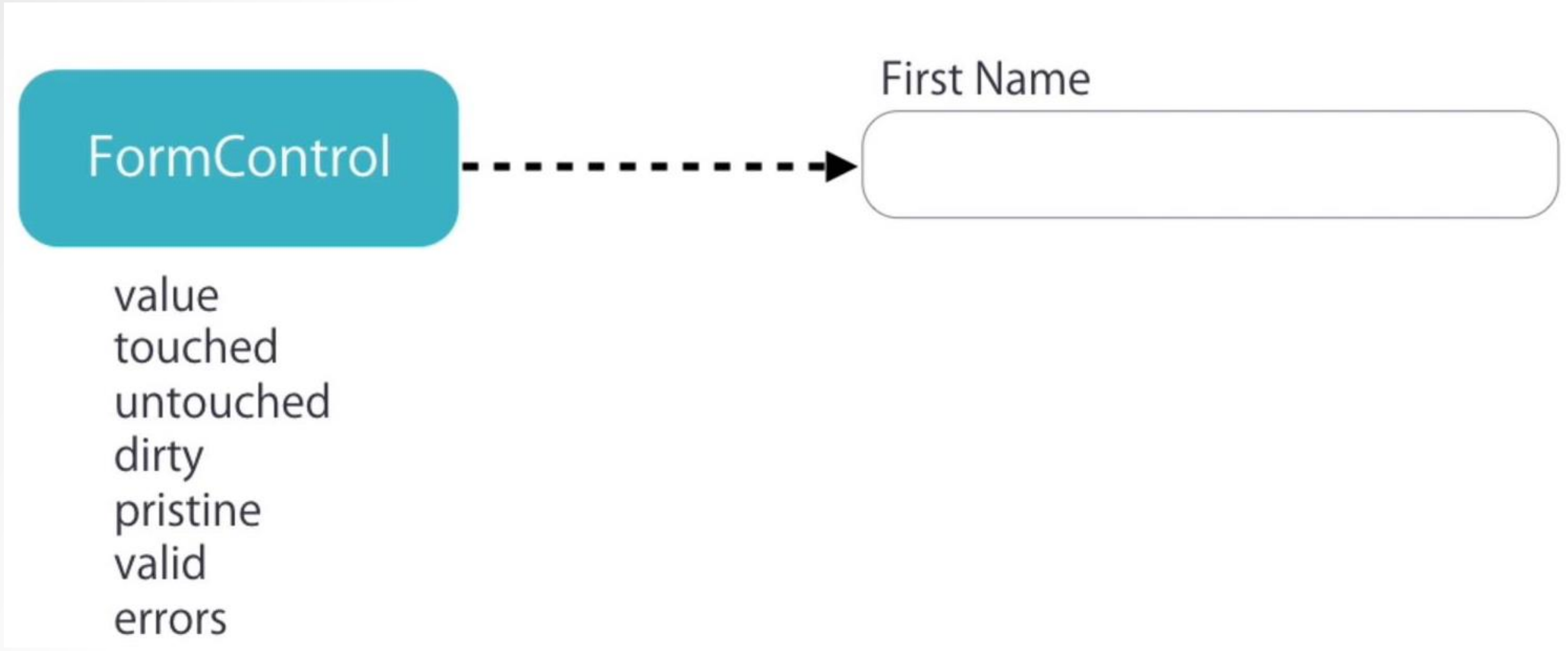
```
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms'
import { NgModule } from '@angular/core';


import { AppComponent } from './app.component';
import { PupilFormComponent } from './pupil-form/pupil-form.component';


@NgModule({
  declarations: [
    AppComponent,
    PupilFormComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Kein Postback mehr zum Server

- Submit liefert kein Ergebnis
  - EventHandler registrieren

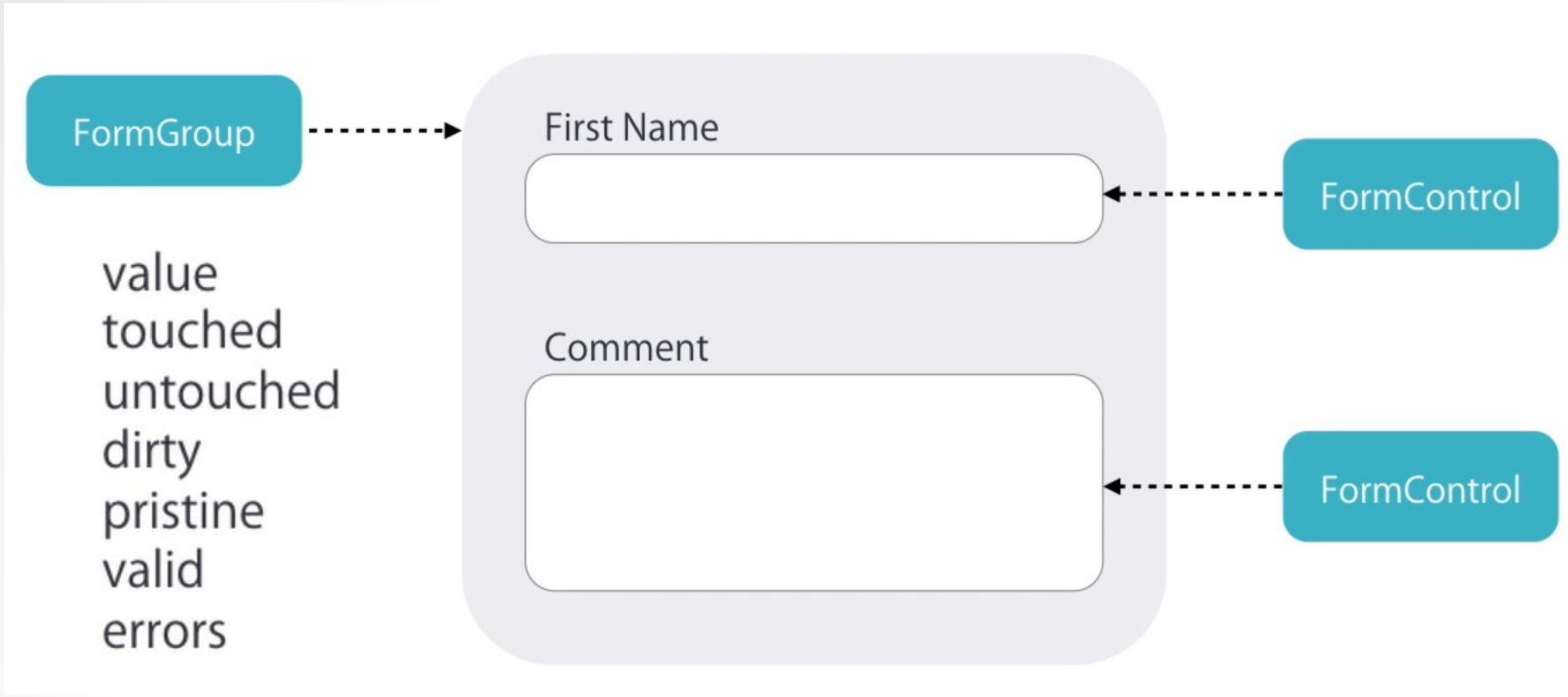


FormControl kapselt Control

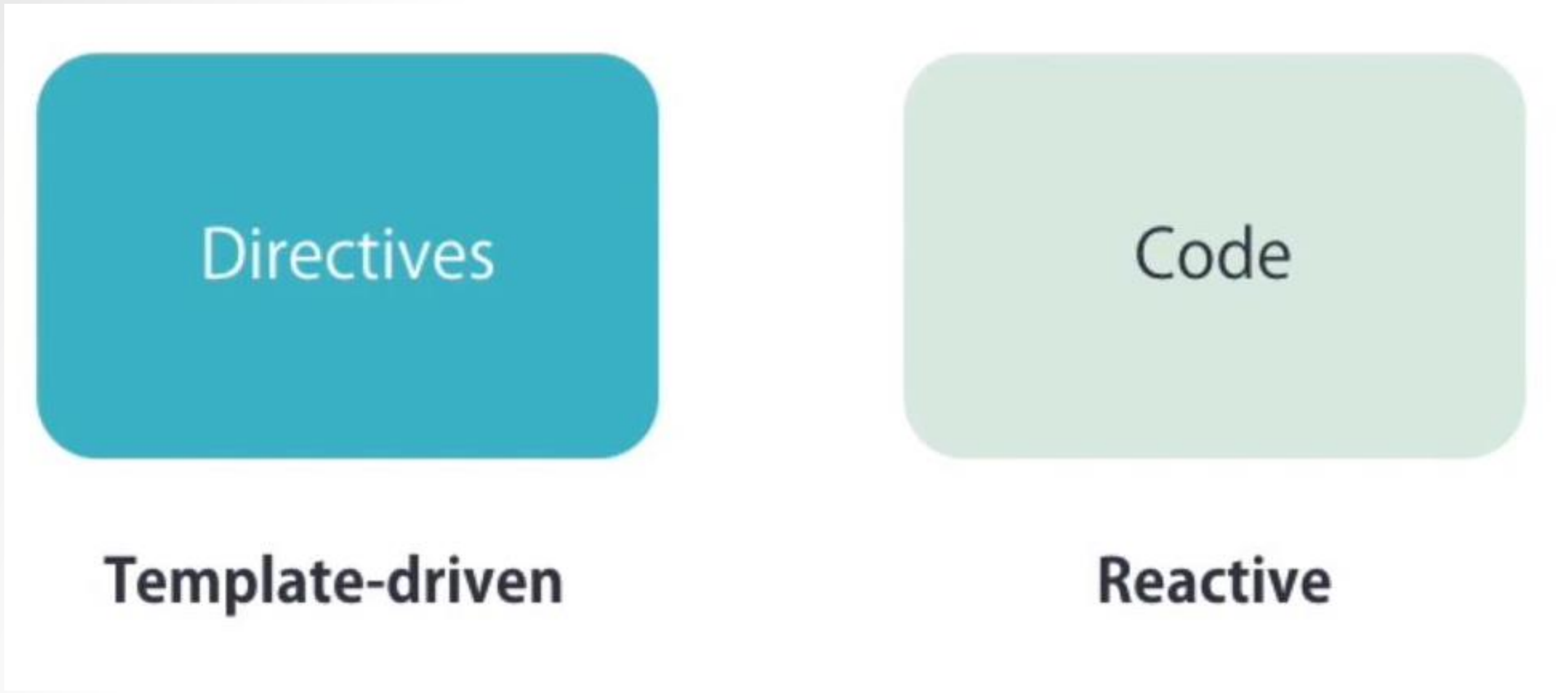


HTL LEONDING

FormGroup kapselt Gruppe von Controls

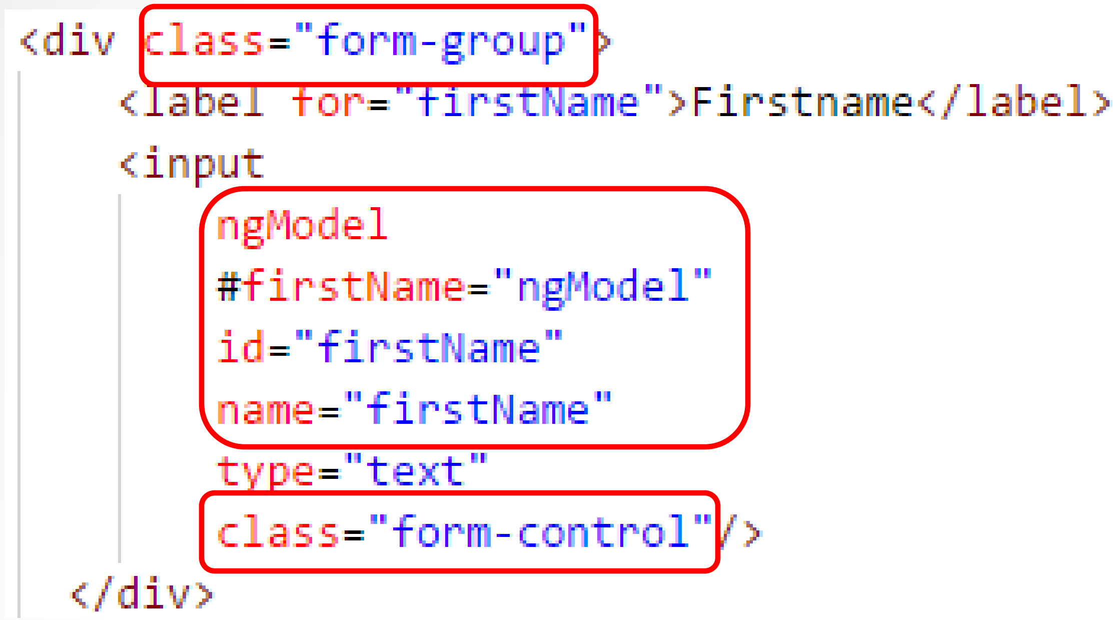


Forms in Angular implementieren



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>
```

HTL LEONDING

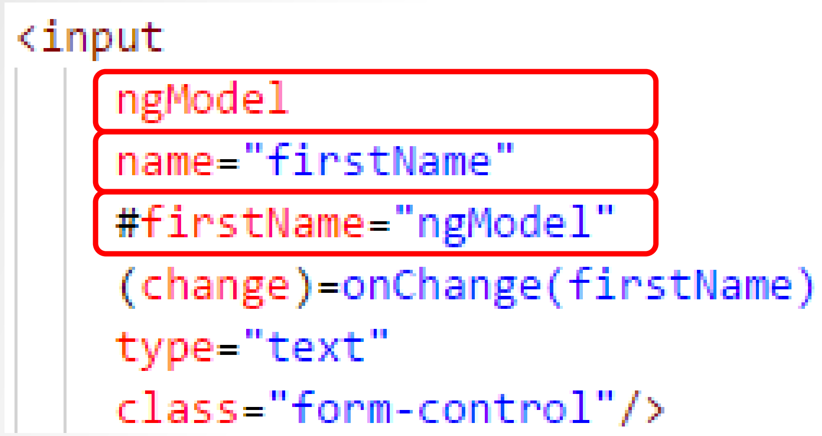
NgModel

- 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

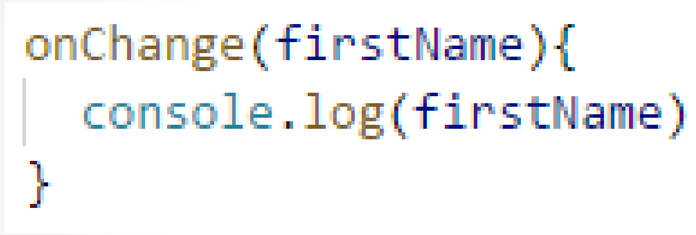
HTL LEONDING

FormControl evaluieren → ngModel

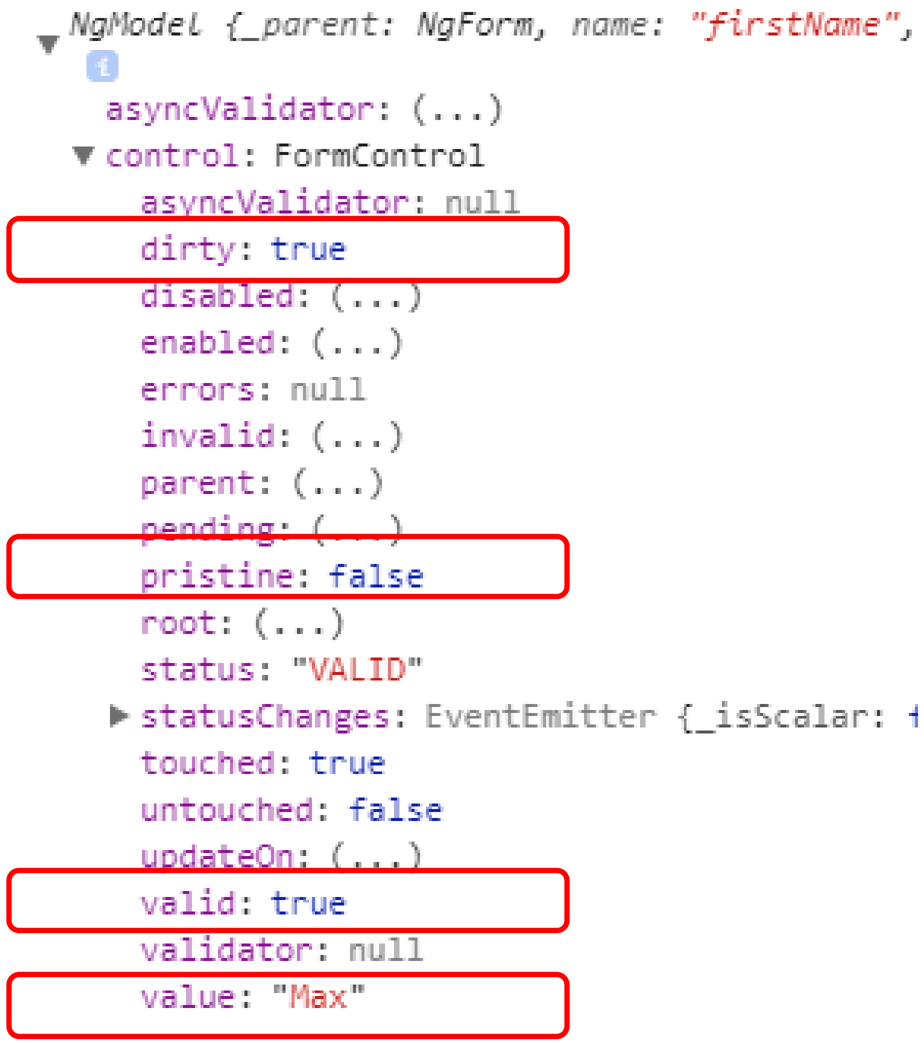
- Temporäre Variable binden
- onChange → Parameter



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



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

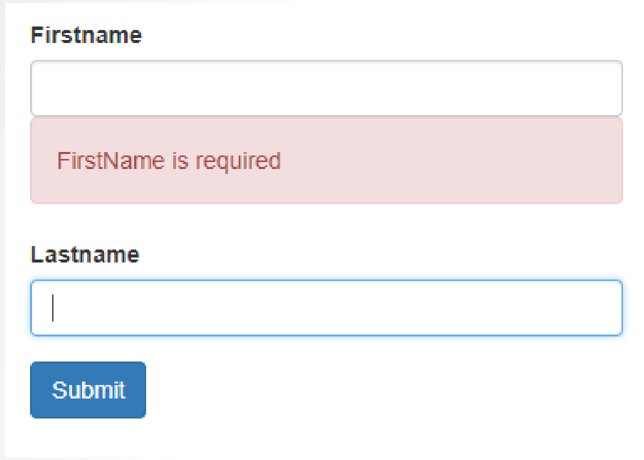


```
▼NgModel {_parent: NgForm, name: "firstName",
   asyncValidator: (...)
  ▼control: FormControl
     asyncValidator: null
     dirty: true
     disabled: (...)
     enabled: (...)
     errors: null
     invalid: (...)
     parent: (...)
     pending: (...)
     pristine: false
     root: (...)
  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

HTL LEONDING

Validierung – Html5-Attribute required

- Validation div
  - Wenn Element geändert und nicht valide ist


Firstname

FirstName is required

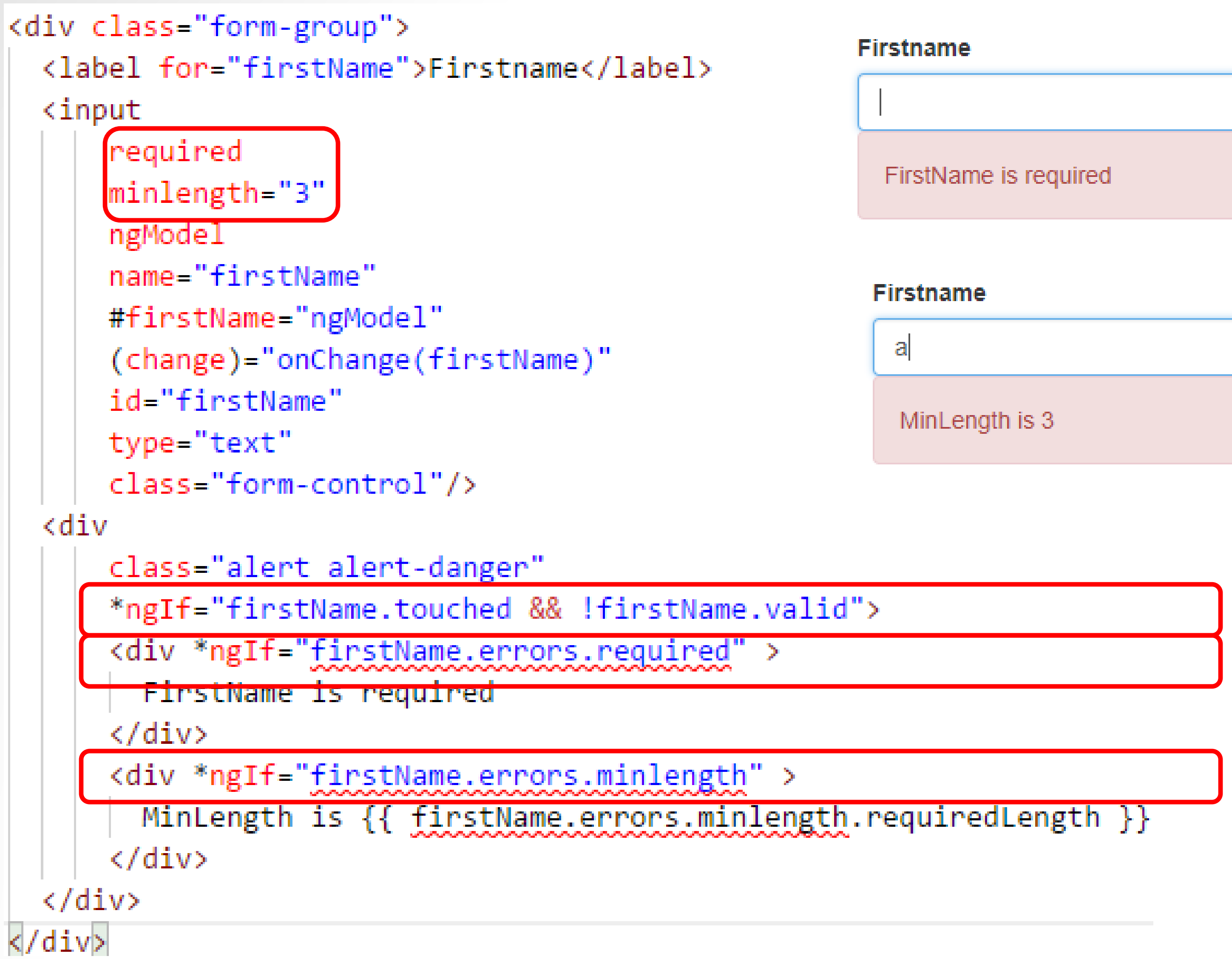
Lastname

Submit

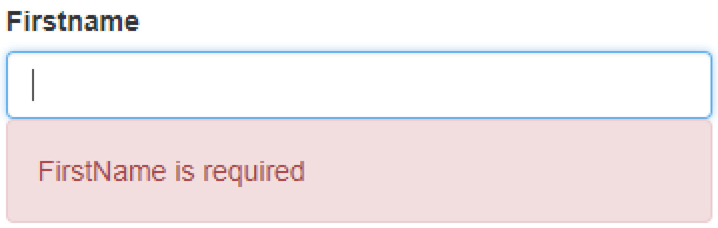
```
<div class="form-group">
  <label for="firstName">Firstname</label>
  <input
    required
    ngModel
    name="firstName"
    #firstName="ngModel"
    (change)=onChange(firstName)
    type="text"
    class="form-control"/>
  <div
    class="alert alert-danger"
    *ngIf="firstName.touched && !firstName.valid">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

HTL LEONDING

Selektive Fehlermeldungen

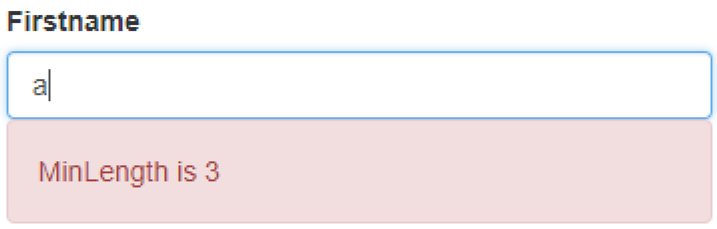


```
<div class="form-group">
  <label for="firstName">Firstname</label>
  <input
      required
      minlength="3"
      ngModel
      name="firstName"
      #firstName="ngModel"
      (change)="onChange(firstName)"
      id="firstName"
      type="text"
      class="form-control"/>
  <div
      class="alert 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



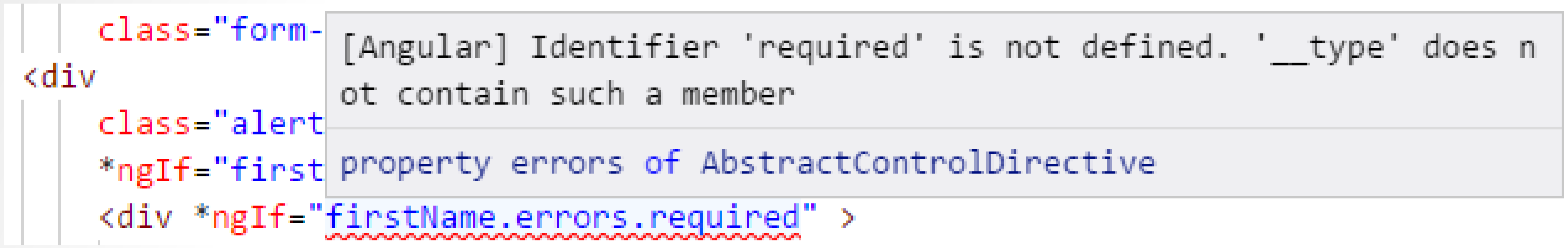
Firstname

a

MinLength is 3

Ohne Typsicherheit

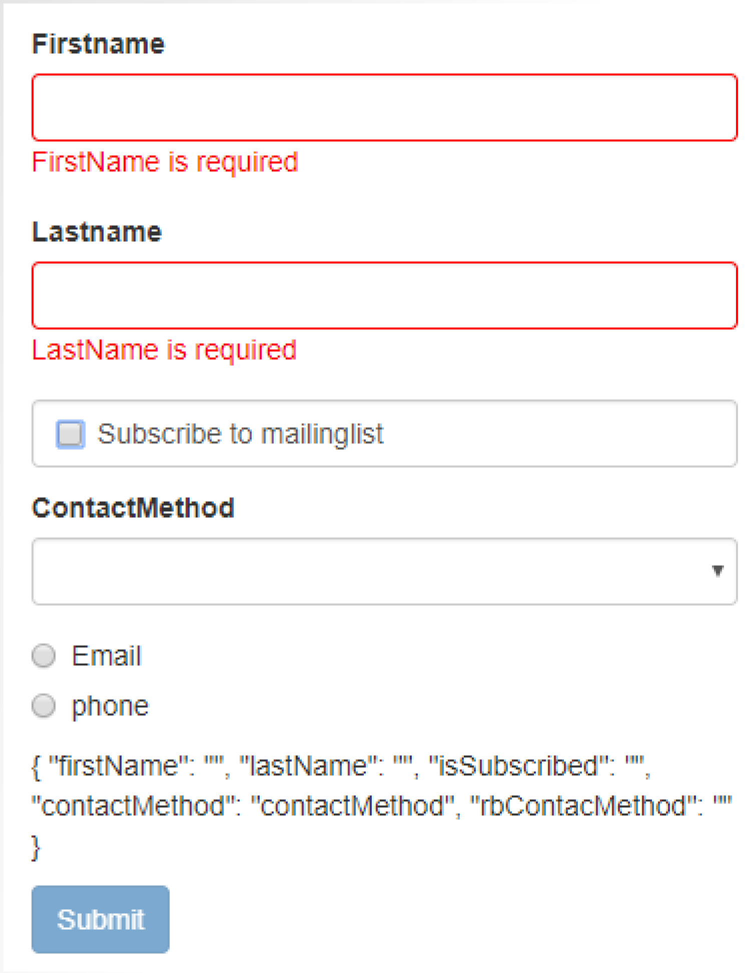
- firstName ist vom Typ object
  - Dank JS trotzdem lauffähig



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

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

property errors of AbstractControlDirective
```

Validierung über css formatieren



Firstname

FirstName is required

Lastname

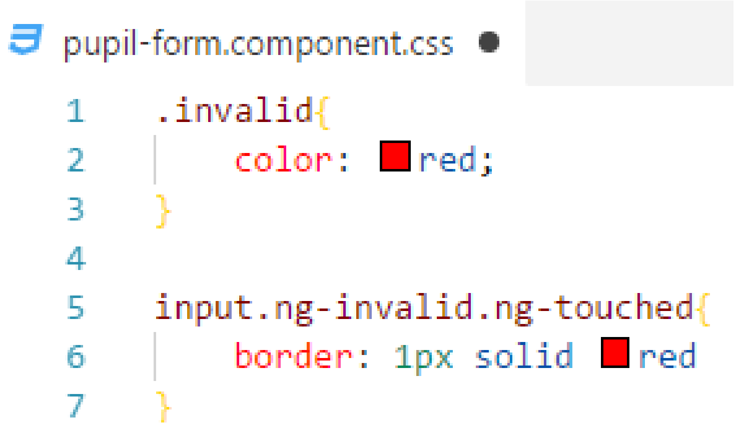
LastName is required

☐ Subscribe to mailinglist

ContactMethod

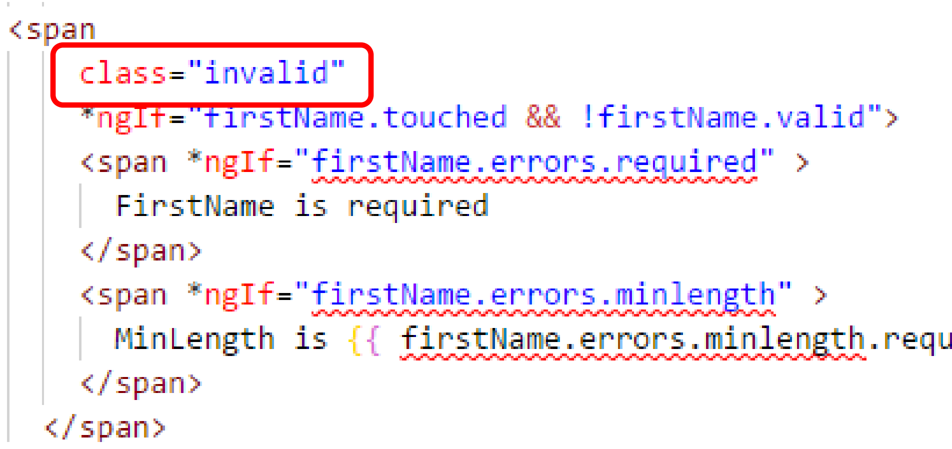
○ Email
○ phone

{ "firstName": "", "lastName": "", "isSubscribed": "", "contactMethod": "contactMethod", "rbContacMethod": "" }



pupil-form.component.css

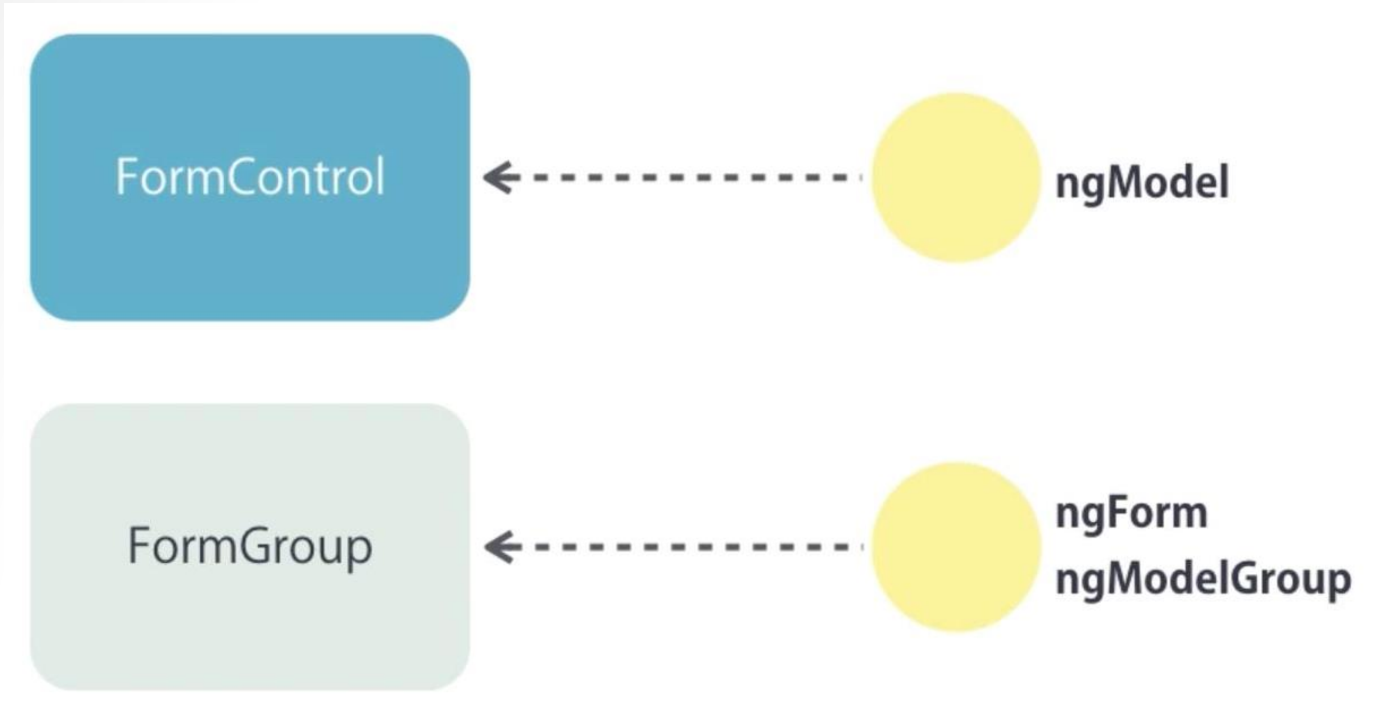
```css
.invalid{
    color: 🟥red;
}

input.ng-invalid.ng-touched{
    border: 1px solid 🟥red
}
```



```html
<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>
```

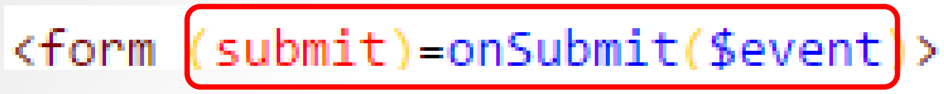
FormGroup

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

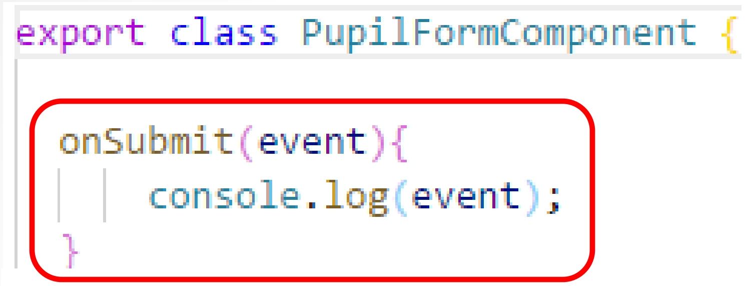


HTL LEONDING

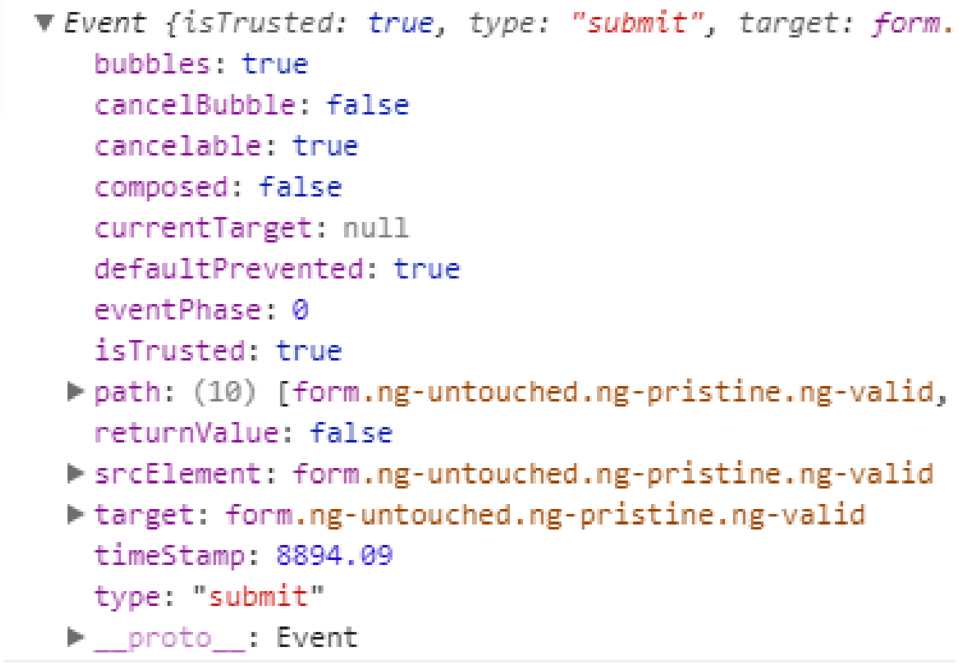
EventBinding auf submit



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



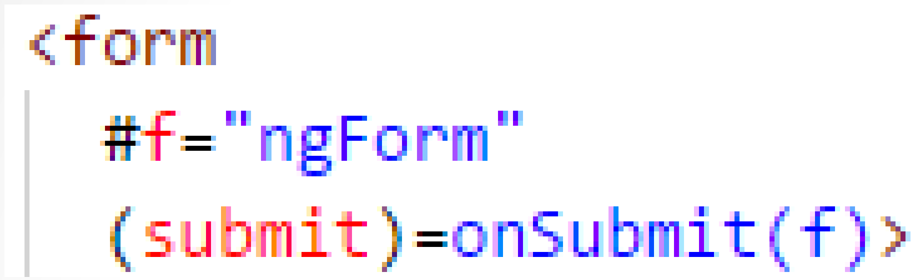
```
export class PupilFormComponent {

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



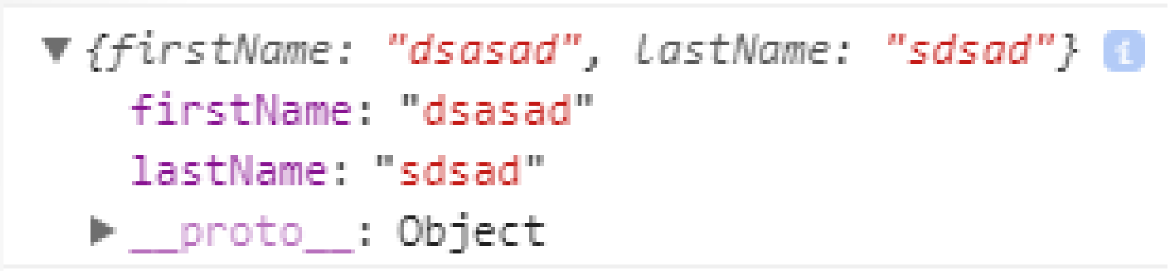
```
▼ Event {isTrusted: true, type: "submit", target: form.
    bubbles: true
    cancelBubble: false
    cancelable: true
    composed: false
    currentTarget: null
    defaultPrevented: true
    eventPhase: 0
    isTrusted: true
  ▶ path: (10) [form.ng-untouched.ng-pristine.ng-valid,
    returnValue: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

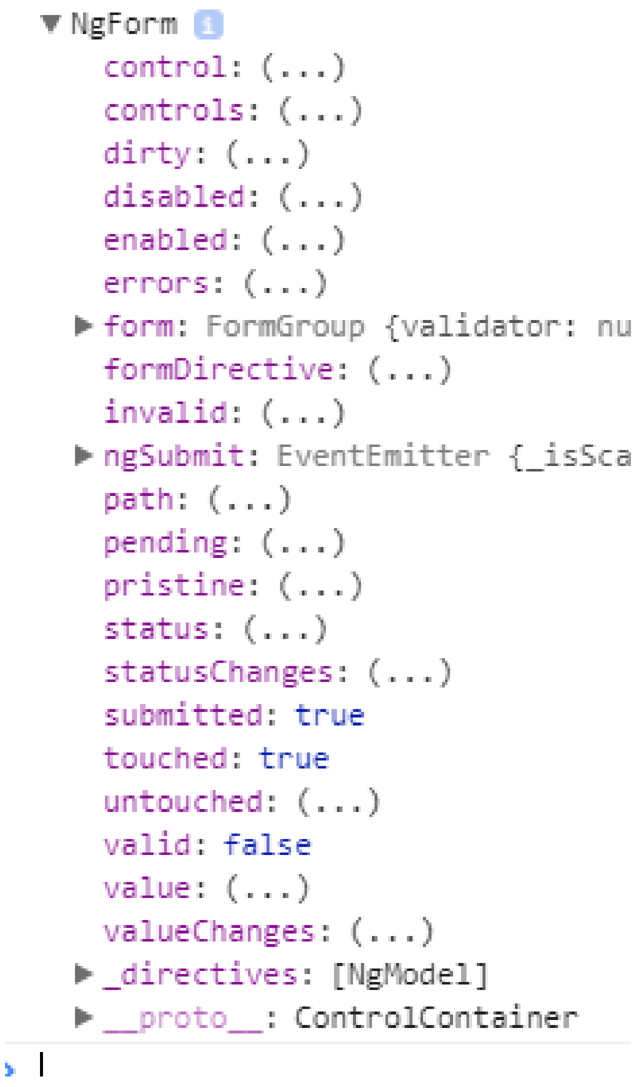
- Hat alle wichtigen Statusinfos
  - valid
  - touched
  - …
- Liefert Objekt als value



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

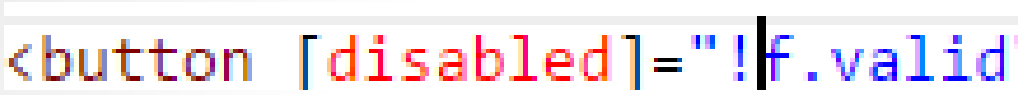


```
▼ {firstName: "dsasad", lastName: "sdsad"}
    firstName: "dsasad"
    lastName: "sdsad"
  ▶ __proto__: Object
```

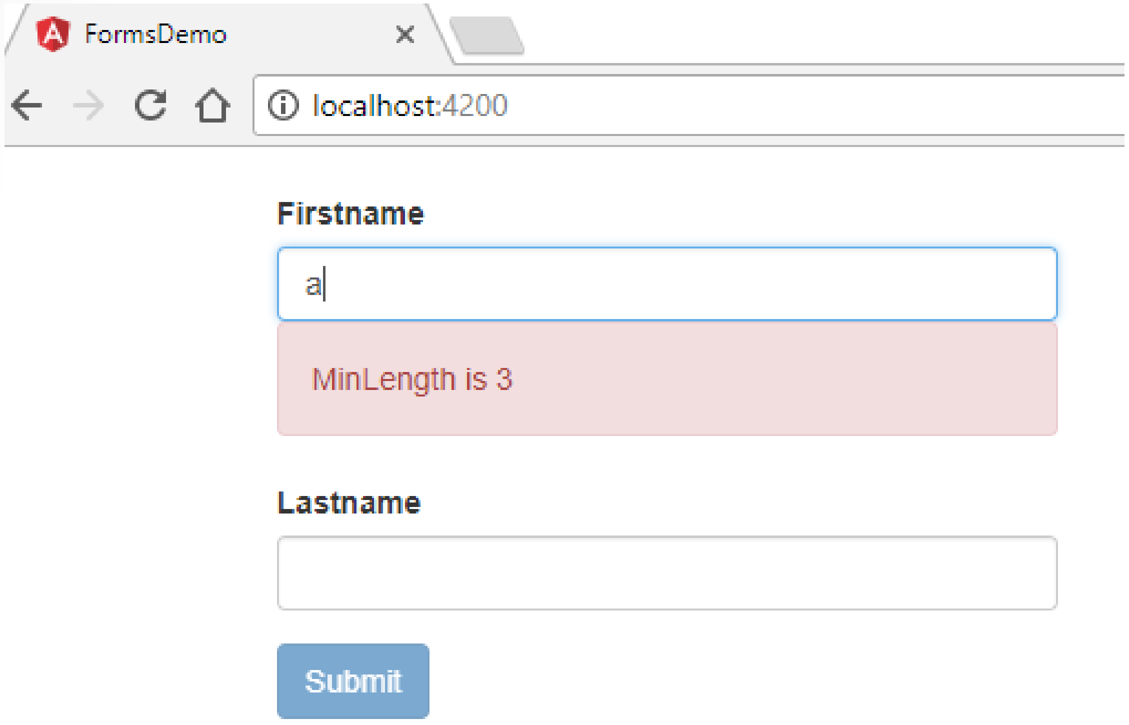


```
▼ NgForm
    control: (...)
    controls: (...)
    dirty: (...)
    disabled: (...)
    enabled: (...)
    errors: (...)
  ▶ form: FormGroup {validator: nu
    formDirective: (...)
    invalid: (...)
  ▶ ngSubmit: EventEmitter {_isSca
    path: (...)
    pending: (...)
    pristine: (...)
    status: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

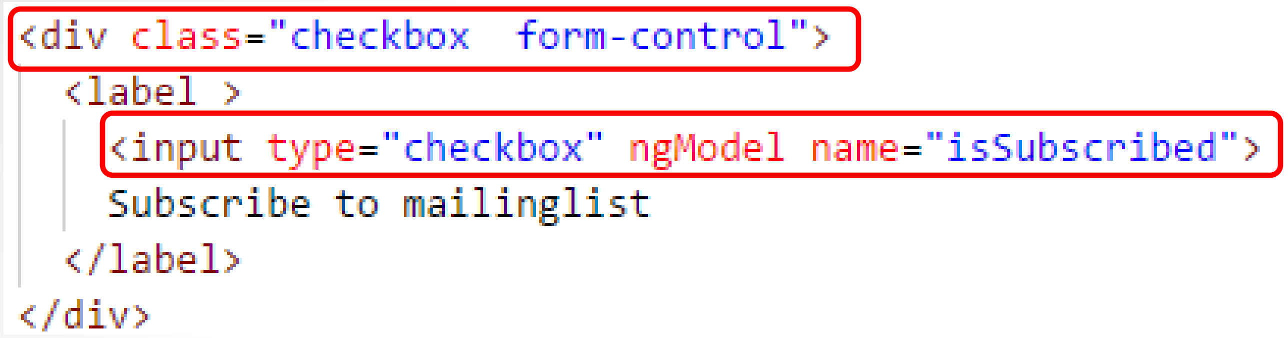
Submit-Button disablen



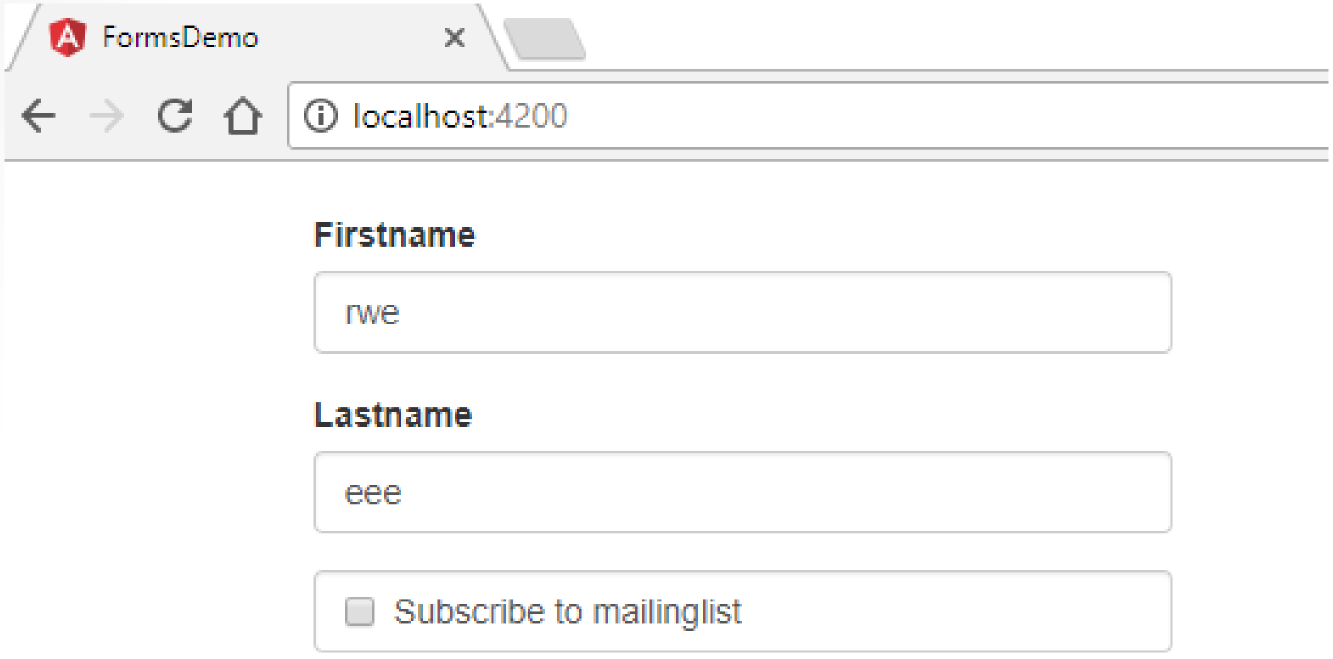
```
<button [disabled]="!f.valid
```



Beispiel CheckBox



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



Firstname

rwe

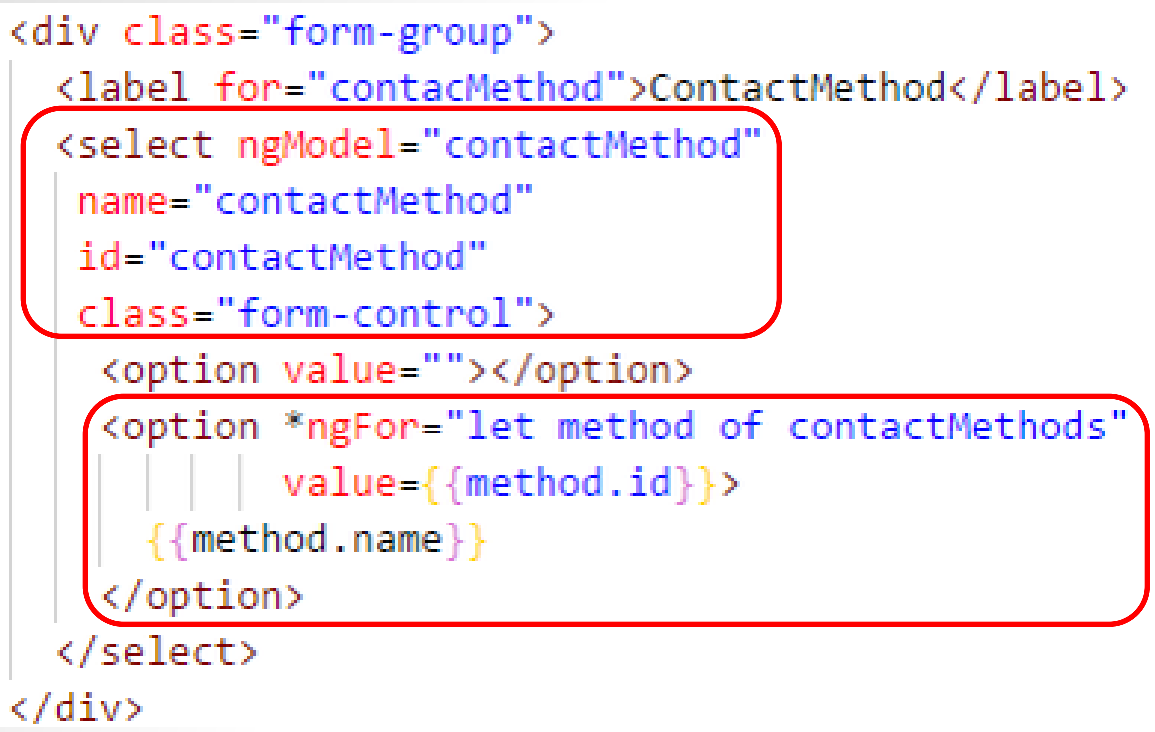
Lastname

eee

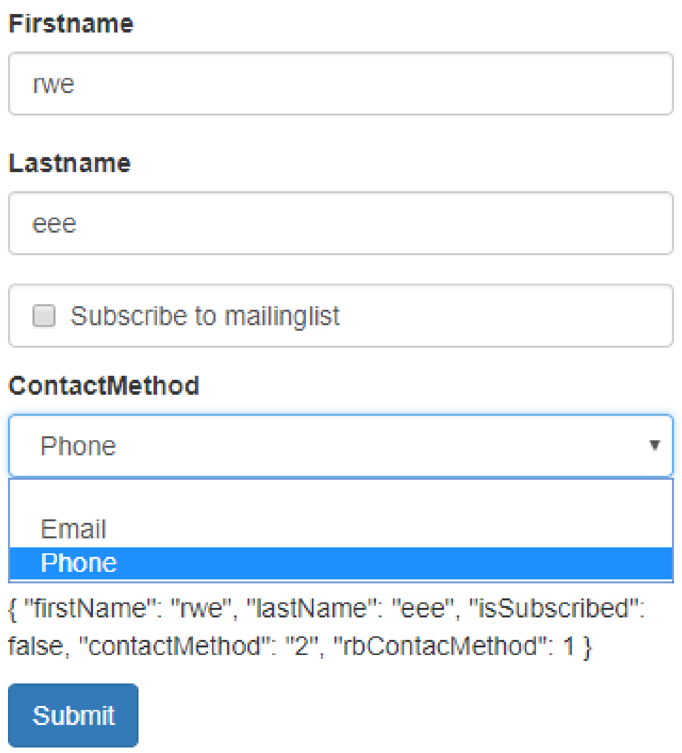
☐ Subscribe to mailinglist

HTL LEONDING

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

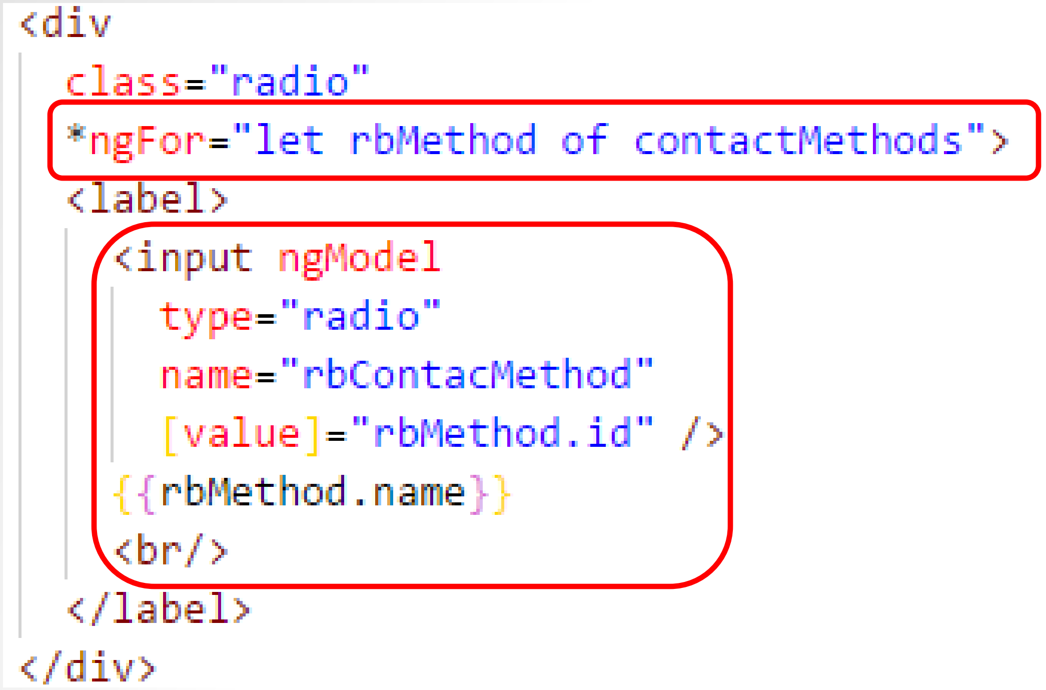
Phone

Email

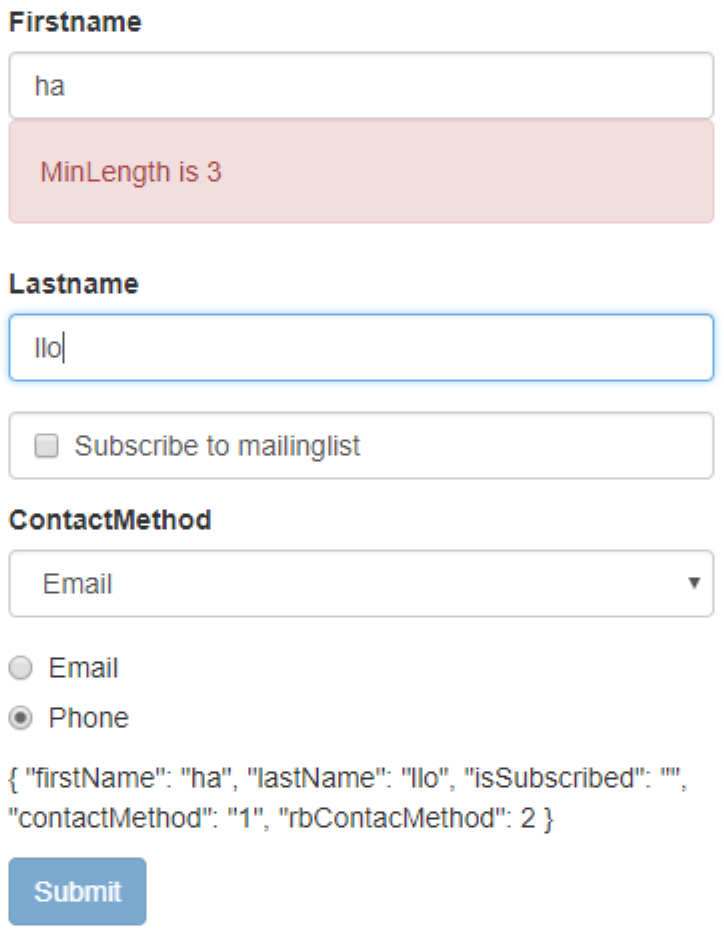
Phone

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

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 }