WebApp
Conf

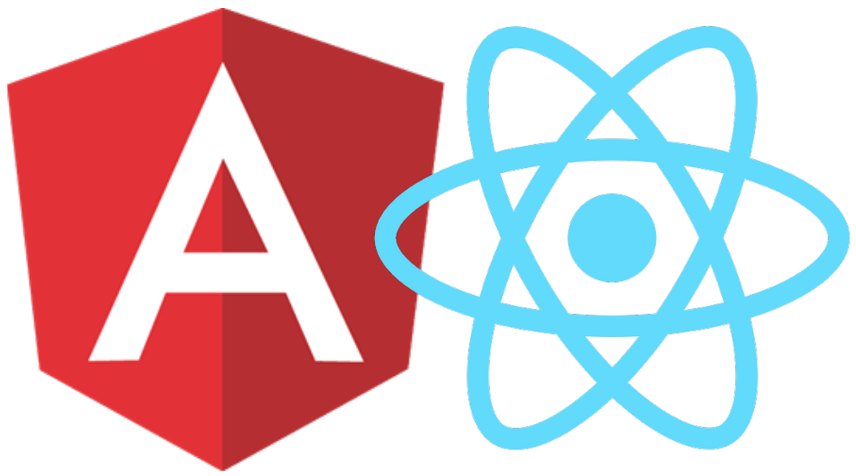
REACTIVE FORMS
for examples

Google Developers Experts

FABIO BIONDI .io

fabiobiondi.io

TRAINING AND MENTORING



FACEBOOK GROUPS

ANGULAR
developer italiani

JAVASCRIPT
developer italiani

REACT
developer italiani

OPPORTUNITÀ per DEVELOPER ITALIANI

(jobs / collaborations / training and more)

TOPICS

- Template-Driven Forms quick intro
- Forms & RxJS
- Reactive Forms Introduction
- Dynamic Input Validators
- Nested Form Groups
- Group Custom Validators
- Form Array
- Split Forms in components
- Dynamic Forms from JSON

TEMPLATE-DRIVEN-FORMS

INPUT VALIDATORS

```
@Component({
  selector: 'my-form',
  template: `
    <input
      placeholder="your email address"
      [ngModel]
      name="email"
      #emailInput="ngModel"
      required
      pattern="\\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,6}\\b"

      [style.background]="emailInput.valid ? 'green' : 'red'"
    >
     <button [disabled]="emailInput.invalid">Send</button>
  `
})
export class MyFormComponent {}
```

CUSTOM VALIDATORS

ITX37ABC

```
@Component({
  selector: 'my-form',
  template: `
    <div [ngClass]="{ 'error': ibanInput.errors?.invalidIBAN}">
      <input
        placeholder="your IBAN"
        [ngModel]
        name="iban"
        #ibanInput="ngModel"
        required
        IBANValidator
      >
      <button [disabled]="ibanInput.invalid">Send</button>
    </div>
  `
})
export class MyFormComponent {}
```

```
// i.e. IT28W8000000292100645211123
const regEx = /IT\d{2}[ ][a-zA-Z]\d{3}[ ]\d{4}[ ]\d{4}[ ]\d{4}[ ]\d{4}[ ]\d{3}|IT\d{2}[a-zA-Z]\d{22}/;

@Directive({
  selector: '[IBANValidator]',
  providers: [
    { provide: NG_VALIDATORS, useExisting: forwardRef(() => IBANValidator), multi: true }
  ]
})
export class IBANValidator implements Validator {
  validate(c: AbstractControl): { [key: string]: any } {
    let v = c.value;
    if (v && !v.match(regEx)) {
      return {'invalidIBAN': true};
    }
    return null;
  }
}
```

FORM VALIDATION

Mario Rossi

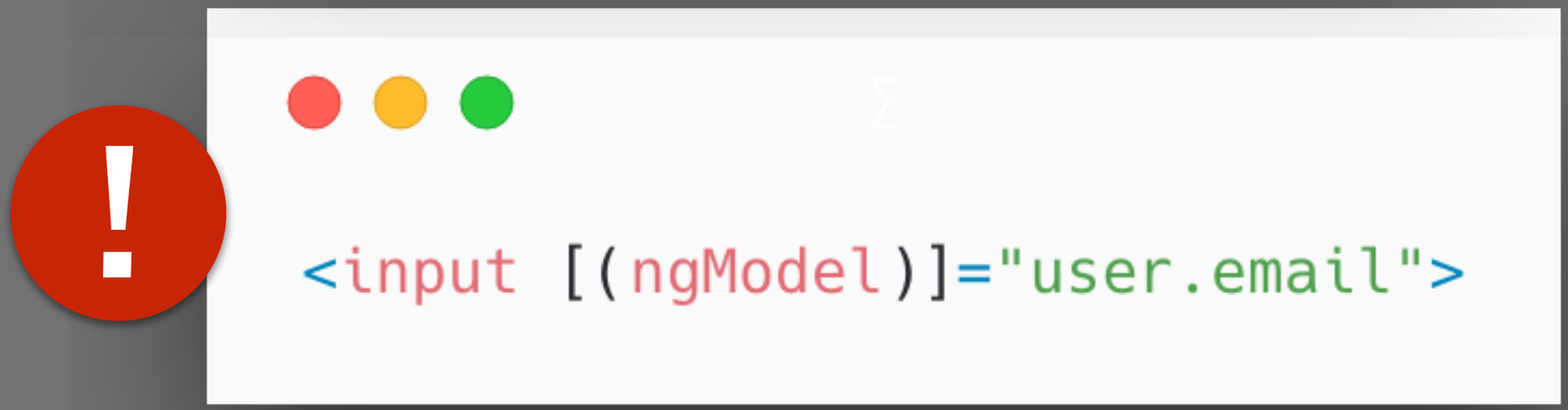
****

ADD

```
<form #f="ngForm" (ngSubmit)="add(f.value)">
  <img class="spinner" *ngIf="usernameRef.pending">
  <input type="text"
    name="username"
    [ngModel]
    #usernameRef="ngModel"
    required
    UsernameAsyncValidator
    [ngClass]="{
      'required': usernameRef.errors?.required,
      'exists': usernameRef.errors?.alreadyExists
    }"
  >
  <input
    type="password"
    name="password"
    [ngModel]
    #passwordRef="ngModel"
    required
    minlength="6"
  >
  <button [disabled]="f.invalid || f.pending">Add User</button>
</form>
```

1-WAY or 2-WAYS BINDING


<input [ngModel]="user.email">


<input [(ngModel)]="user.email">

REACTIVE FORMS

FORMS &  RXJS

Mario Biondi

```
@Component({
  selector: 'my-form',
  template: `
    <input type="text" #usernameRef />
  `
})
export class MyFormComponent implements OnInit {
  @ViewChild('usernameRef') username: ElementRef;

  ngOnInit(): void {
    fromEvent(this.username.nativeElement, 'keyup')
      .pipe(
        map(e => e.target.value),
        filter((text: string) => text.length > 3),
        debounceTime(1000),
        distinctUntilChanged()
      )
      .subscribe((text: string) => {
        console.log ('submit: ', text);
      });
  }
}
```

MY FIRST REACTIVE FORM

New York

```
@Component({
  selector: 'my-form',
  template: `
    <input type="text" [formControl]="term">
    <meteo-search [text]="search"></meteo-search>
`
})
export class MyFormComponent {
  search = 'Rome';
  term: FormControl;

  constructor() {
    this.term = new FormControl(this.search);
    this.term.valueChanges
      .pipe(
        debounceTime(400),
        distinctUntilChanged()
      )
      .subscribe(term => this.search = term );
  }
}
```

FORM BUILDER

username

password

confirm pass

```
@Component({
  selector: 'lost-pass',
  template: `
    <div>{{email.errors | json}}</div>
    <form [formGroup]="myForm" (ngSubmit)="submit()" >
      <input formControlName="email">
      <button [disabled]="myForm.invalid">SUBMIT</button>
    </form>`
})
export class LostPassComponent implements OnInit{
  myForm: FormGroup;
  email: AbstractControl;

  constructor( private fb: FormBuilder) {
    this.myForm = fb.group({
      'email':  ['', Validators.compose([
        Validators.required, Validators.minLength(4), myCustomValidator
      ])],
    });
    this.myForm.valueChanges.subscribe(data => console.log(data));
  }

  submit() { console.log (this.myForm.value); }
}
```

CUSTOM VALIDATORS

f@b1o_b1Ond1

```
constructor(private fb: FormBuilder) {
  this.form = fb.group({
    'company': [
      null,
      Validators.compose([Validators.required, companyValidator])
    ],
  });
}
```

```
export function companyValidator(c: FormControl): { [s: string]: boolean } {
  const regEx = /^[ANY_REGEX]$/;
  if (c.value && !c.value.match(regEx)) {
    return {wrongName: true};
  }
}
```

ENABLE / DISABLE VALIDATORS

Mario Biondi

```
constructor(private fb: FormBuilder) {
  this.form = fb.group({
    'company': [null],
  });
  this.form.get('company').disable();
}

enableField() {
  this.form.get('company').enable();
}
```

UPDATE VALIDATORS

☒ is a company?

your company name

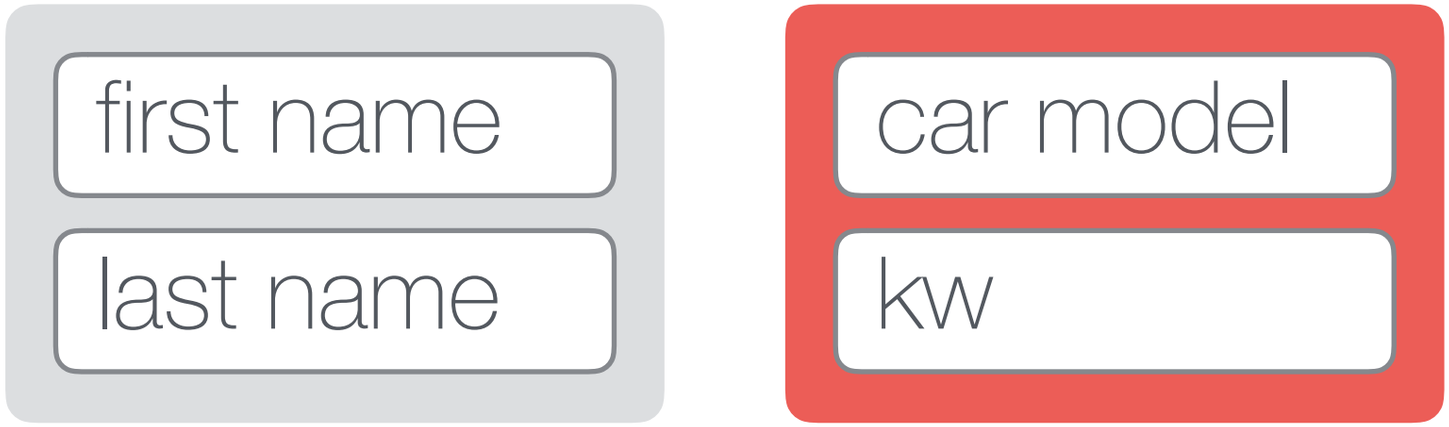
Your name

```
initForm() {
  // ...
  this.inputName = this.form.get('name');
  this.form.get('isCompanyCheckbox')
    .valueChanges
    .subscribe(value => {
      if (value) {
        this.setCompanyValidator();
      } else {
        this.setUserValidator()
      }
      this.inputName.updateValueAndValidity();
    });
}

setCompanyValidator() {
  this.inputName.setValidators(
    Validators.compose([Validators.required, companyValidator])
  );
}

setUserValidator() {
  this.inputName.setValidators([userValidator]);
}
```

NESTED FORM GROUPS



```
@Component({
  selector: 'insurance',
  template: `
    <form [formGroup]="form" (submit)="send()">
      <input type="text" formControlName="username">
      <div formGroupName="carInfo"
           [ngClass]="{'invalid': form.get('carInfo').invalid}>
          <input type="text" formControlName="model">
          <input type="number" formControlName="kw">
      </div>
      <button [disabled]="form.invalid">SEND</button>
    </form>
  `,
})
export class RegistrationComponent {
  form: FormGroup;

  constructor(private fb: FormBuilder) {
    this.form = fb.group({
      'username': ['', Validators.required]),
      'carInfo': fb.group(
        {
          'model': ['', Validators.required],
          'kw': ['', Validators.required]
        }
      )
    });
  }
  send() { console.log(this.form.value) }
}
```

GROUP CUSTOM VALIDATORS

first name

last name

****

******

passwords don't match

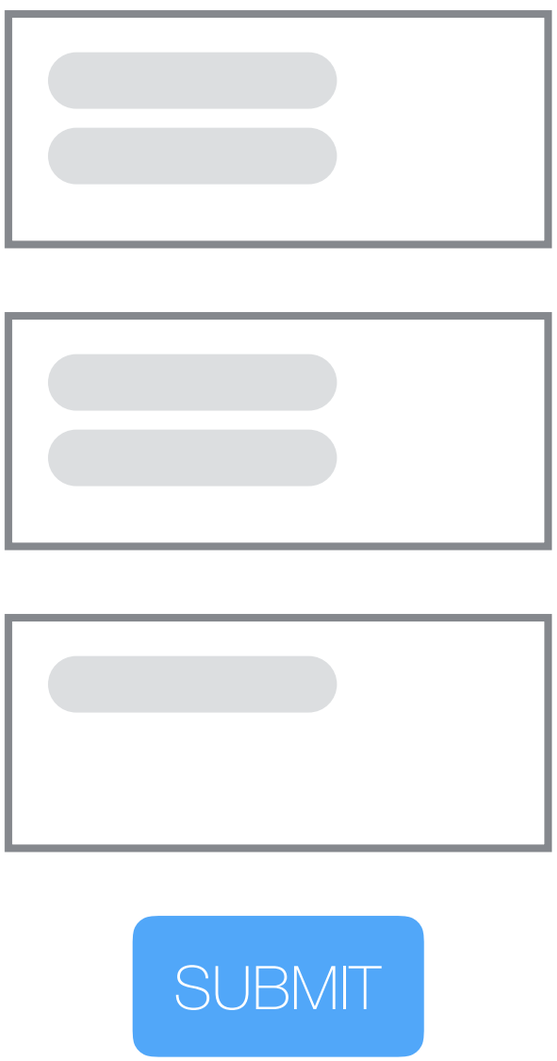
```
<form [formGroup]="signupForm" (submit)="register()">
  <input type="text" formControlName="username">
  <div formGroupName="passwords"
    [ngClass]="{'invalid': signupForm.get('passwords').errors?.notTheSame}">
      <input type="text" formControlName="password1" placeholder="password">
      <input type="text" formControlName="password2" placeholder="confirm password">
  </div>
  ...
</form>
```

```
loginForm: FormGroup;

initForm(private fb: FormBuilder) {
  this.loginForm = fb.group({
    'username': ['', Validators.compose([
          Validators.required, Validators.minLength(4), usernameValidator
      ])
    ],
    'passwords': fb.group({
        'password1': ['', Validators.compose([Validators.required, Validators.minLength(3)])],
        'password2': ['', Validators.compose([Validators.required, Validators.minLength(3)])]
      },
      { validator: passwordMatch('password1', 'password2') }
    )
  });
}
```

```
export function passwordMatch(field1, field2) {
  return (group) => {
    if (group.get(field1).value !== group.get(field2).value) {
      group.get(field2).setErrors({notTheSame: true});
    } else {
      group.get(field2).setErrors(null);
    }
  };
}
```

SPLIT FORMS IN COMPONENTS



```html
<div>
  <form [formGroup]="form" (submit)="add(form.value)">
    <input type="text" formControlName="name">
    <insurance-step-1 [group]="form"></insurance-step-1>
    <insurance-step-2 [group]="form"></insurance-step-2>
    <input type="submit" [disabled]="form.invalid">
  </form>
</div>
```

```ts
form: FormGroup;

initForm(private fb: FormBuilder) {
  this.form = fb.group({
    'name': [null, [Validators.required, Validators.minLength(3)]],
    'step1': fb.group({
      'brand': [null, Validators.required],
      'model': [null, Validators.required],
    }),
    'step2': fb.group({
      'kw': [null, [Validators.required, Validators.min(0), Validators.max(700)]],
      'year': [2018, [Validators.required, Validators.min(1970), Validators.max(2018)]],
    }),
  });
}
```

```
@Component({
  selector: 'insurance-step-1',
  template: `
    <div [formGroup]="group.get('step1')">
        <i class="fa fa-check" *ngIf="group.get('step1').valid"></i>
        <div [ngClass]="{
          'has-error': group.get('step1').controls['brand'].invalid
        }">
          <input type="text" formControlName="brand">
        </div>

        <div [ngClass]="{
          'has-error': group.get('step1').controls['model'].invalid
        }">
          <input type="text" formControlName="model">
        </div>
    </div>
  `
})
export class InsuranceStep1Component {
  @Input('group') group;
}
```

FORM ARRAY

Nutella | 4.99 ⊖

Biscotti | 2.50 ⊖

item name | price ⊕

```
orderForm: FormGroup;
items: FormArray;

initForm(private fb: FormBuilder) {
  this.orderForm = this.fb.group({
    customer: '',
    items: this.fb.array([ this.createItem() ]) as AbstractControl
  });
  this.items = this.orderForm.get('items') as FormArray;
}

submit() { console.log (this.orderForm.value); }
createItem(): FormGroup { return this.fb.group({ name: '', price: '' }) }
addItem() { this.items.push(this.createItem()); }
removeItem(item) {
  const index = this.items.controls.indexOf(item);
  this.items.removeAt(index);
}
```

```
<form [formGroup]="orderForm">
  <input formControlName="customer" required>
  <div formArrayName="items" *ngFor="let item of items.controls; let i = index">
    <div [formGroupName]="i" [ngClass]="{'ok': items.controls[i].valid}">
      <input formControlName="name" required>
      <input formControlName="price" required>
    </div>
  </div>
  ...
</form>
```

DYNAMIC FORMS
from JSON
(simplest way)

```
config = [
  {
    type: 'text',
    label: 'Full name',
    key: 'name',
    placeholder: 'Enter your name',
    required: true
  },
  {
    type: 'list',
    label: 'Car Brand',
    key: 'brands',
    options: [ {value: 'BMW', key: 0 },  {value: 'Fiat', key: 1 }],
    placeholder: 'Select an option',
    required: true
  },
  {
    label: 'Submit',
    key: 'submit',
    type: 'button',
  },
];
```

```
<form (ngSubmit)="onSubmit()" [formGroup]="form">
  <div *ngFor="let control of config">
    <div [ngSwitch]="control.type">
      <input
        *ngSwitchCase="'text'"
        [formControlName]="control.key"
        [type]="control.type"
        [required]="control.required"
      >

      <select
        *ngSwitchCase="'list'"
        [formControlName]="control.key"
      >
        <option *ngFor="let opt of control.options" [value]="opt.key">
          {{opt.value}}
        </option>
      </select>
    </div>
  </div>
</form>
```

DYNAMIC FORMS

(create components @ runtime)

```
initForm() {
  this.form = this.createGroup();
}

createGroup() {
  const group = this.fb.group({});
  this.config.forEach(control => group.addControl(
    control.name, this.createControl(control)
  ));
  return group;
}

createControl(config: FieldConfig) {
  const { value = '' } = config;
  return this.fb.control(value);
}
```

```ts
import {Type} from '@angular/core';
import {Field} from '../models/field.interface';
import {FormInputComponent} from './form-input.component';
import {FormButtonComponent} from './form-button.component';
import {FormSelectComponent} from './select.component';

export const components: {[type: string]: Type<Field>} = {
  text: FormInputComponent,
  list: FormSelectComponent
  submit: FormButtonComponent
};
```

```
<form
  [formGroup]="form"
  (submit)="handleSubmit($event)">
  <ng-container
    *ngFor="let field of config;"
    dynamicField
    [fieldConfig]="field"
    [group]="form">
  </ng-container>
</form>
```

```
@Directive({
  selector: '[dynamicField]'
})
export class DynamicFieldDirective implements OnInit {
  @Input() group: FormGroup;
  @Input() fieldConfig: any;
  component: ComponentRef<Field>;

  constructor(
    private resolver: ComponentFactoryResolver,
    private container: ViewContainerRef
  ) {}

  ngOnInit() {
    const cFactory =
      this.resolver.resolveComponentFactory<Field>(
        components[this.fieldConfig.type]
      );
    this.component = this.container.createComponent(cFactory);
    this.component.instance.config = this.fieldConfig;
    this.component.instance.group = this.group;
  }
}
```

```
@Component({
  selector: 'form-input',
  template: `
    <div [formGroup]="group">
      <label>{{ config.label }}</label>
      <input
        [type]="config.type"
        [required]="config.required"
        [attr.placeholder]="config.placeholder"
        [formControlName]="config.name" />
    </div>
  `,
})
export class FormInputComponent implements Field {
  group: FormGroup;
  config: any;
}
```

FACEBOOK GROUPS

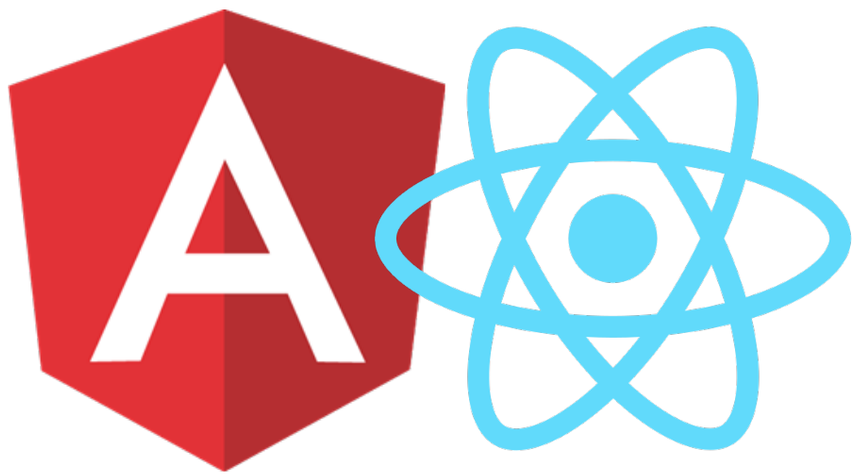
ANGULAR
developer italiani

JAVASCRIPT
developer italiani

REACT
developer italiani

OPPORTUNITÀ per DEVELOPER ITALIANI

(jobs / collaborations / training and more)



fabiobiondi.io