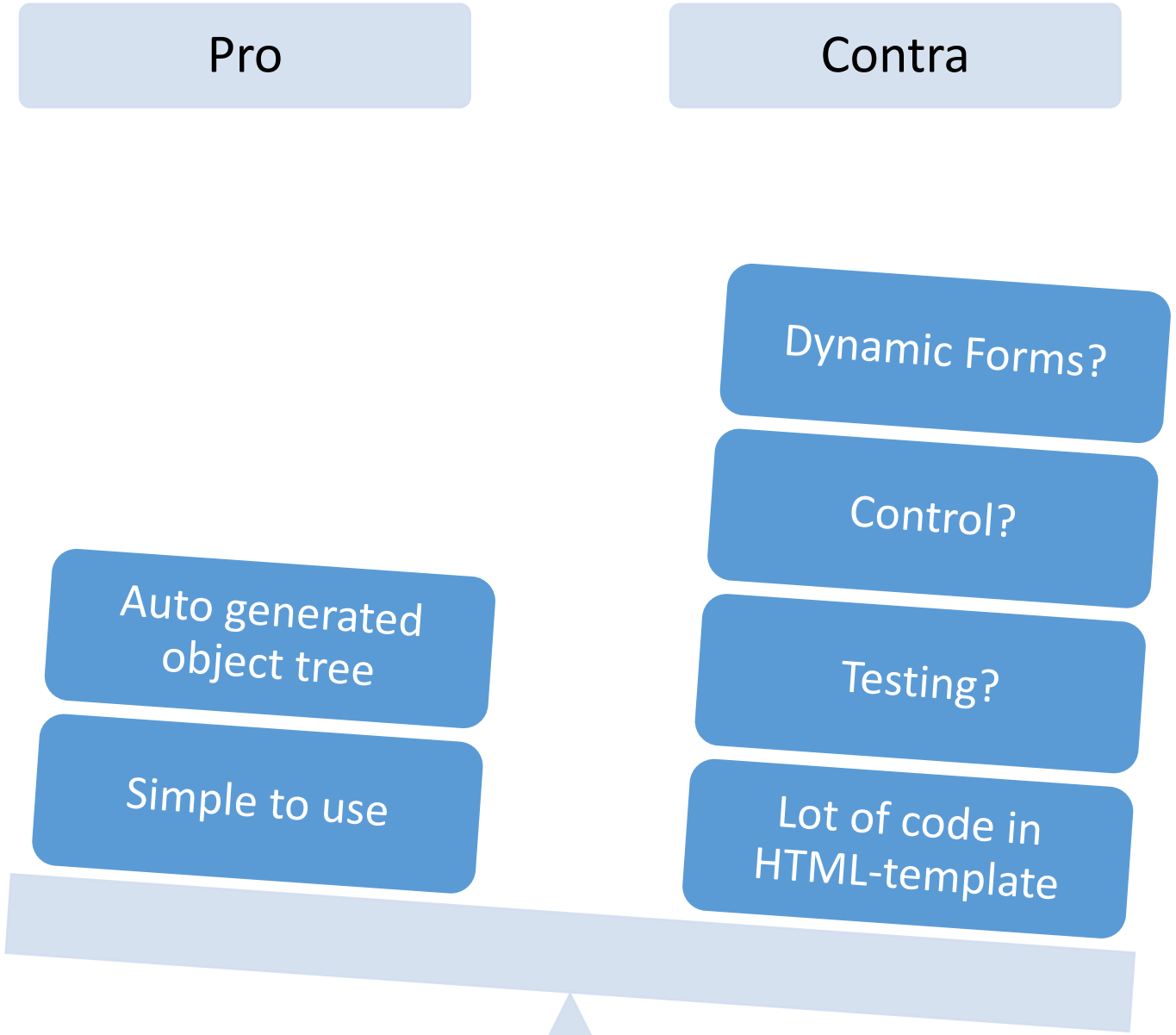
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Reactive Forms and Validation

Alex Thalhammer


Pro
Contra
Dynamic Forms?
Control?
Testing?
Lot of code in HTML-template
Auto generated object tree
Simple to use

Reactive Forms

ReactiveFormsModule

```
@NgModule({
   imports: [
      ReactiveFormsModule,
      CommonModule,
      SharedModule,
      [...]
   ],
   [...]
})
export class FlightBookingModule { }
```

Reactive Forms

```
export class FlightSearchComponent {

    searchForm: FormGroup;

    […]
}
```

Reactive Forms

```
export class FlightSearchComponent {

    searchForm: FormGroup;

    constructor(…) {
        const fromControl = new FormControl('Graz');
        const toControl = new FormControl('Hamburg');
        this.searchForm = new FormGroup({ from: fromControl, to: toControl});

        […]

    }
}
```

Reactive Forms

```
export class FlightSearchComponent {

    searchForm: FormGroup;

    constructor(…) {
        const fromControl = new UntypedFormControl('Graz');
        const toControl = new UntypedFormControl('Hamburg');
        this.searchForm = new FormGroup({ from: fromControl, to: toControl});

        […]

    }
}
```

Reactive Forms

```
export class FlightSearchComponent {

    searchForm: FormGroup;

    constructor(…) {
        const fromControl = new FormControl('Graz');
        const toControl = new FormControl('Hamburg');
        this.searchForm = new FormGroup({ from: fromControl, to: toControl});

        fromControl.validator = Validators.required;
        […]
    }
}
```

Reactive Forms

```
export class FlightSearchComponent {

    searchForm: FormGroup;

    constructor(…) {
        const fromControl = new FormControl('Graz');
        const toControl = new FormControl('Hamburg');
        this.searchForm = new FormGroup({ from: fromControl, to: toControl});

        fromControl.validator =
                Validators.compose([Validators.required, Validators.minLength(3)]);
    }
}
```

Reactive Forms

```
export class FlightSearchComponent {

    searchForm: FormGroup;

    constructor(…) {
        [...]

        fromControl.validator =
                Validators.compose([Validators.required, Validators.minLength(3)]);

        fromControl.asyncValidator = Validators.composeAsync([…]);
    }
}
```

FormBuilder

```
export class FlightSearchComponent {

    searchForm: FormGroup;

    constructor(private fb: FormBuilder, …) {
        this.searchForm = this.fb.group({
            from: ['Graz', Validators.required],
            to: ['Hamburg', Validators.required]
        });
        […]
    }

}
```

FormBuilder

```
export class FlightSearchComponent {

    searchForm: FormGroup;

    constructor(private fb: FormBuilder, …) {
        this.searchForm = this.fb.group({
            from: ['Graz', [Validators.required, Validators.minLength(3)]],
            to: ['Hamburg', Validators.required]
        });
        […]
    }
}
```

FormBuilder

```
export class FlightSearchComponent {

    searchForm: FormGroup;

    constructor(private fb: FormBuilder, …) {
        this.searchForm = this.fb.group({
            from: ['Graz', [Validators.required, Validators.minLength(3)], [ /* asyncValidator */ ] ],
            to: ['Hamburg', Validators.required]
        });
        […]
    }

}
```

FormBuilder

```
export class FlightSearchComponent {

    searchForm = this.fb.group({
        from: ['Graz', [Validators.required, Validators.minLength(3)]],
        to: ['Hamburg', Validators.required]
    });

    constructor(private fb: FormBuilder, …) {}
}
```

API

```
this.searchForm.valueChanges.subscribe(change => {
    console.debug('form value has changed', change);
});
```

```
this.searchForm.controls['from'].valueChanges.subscribe(change => {
    console.debug('from input has changed ', change);
});
```

```
let fromValue = this.searchForm.controls['from'].value; // directly via control
const toValue = this.searchForm.controls[to'].value;
```

```
const formValue = this.searchForm.value;
fromValue = formValue.from; // via form value object
```

Reactive Forms

```
<form [formGroup]="searchForm">

   <input formControlName="from" id="from" class="form-control" type="text">

   […]

</form>
```

Reactive Forms

```
<form [formGroup]="searchForm">

   <input formControlName="from" id="from" class="form-control" type="text">
   <div *ngIf="searchForm.controls['from'].invalid">…Error…</div>

   […]

</form>
```

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

LAB

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Validators for Reactive Forms

Reactive validators === functions

A simple validator

```
export function validate(c: AbstractControl): ValidationErrors | null  {
    if (c.value === 'Graz' || c.value === 'Hamburg') {
        return null;
    }

    return { city: true };
}
```

Apply validators

```
this.form = fb.group({
    from: [
        'Graz',
        [
            validate
        ],
        [
            /* asyncValidator */
        ]
    ],
    to: ['Hamburg', Validators.required]
});
```

Parameterizable validators

```
export function validateWithParams(validCities: string[]): ValidatorFn {

    […]

}
```

Parameterizable validators

```
export function validateWithParams(validCities: string[]): ValidatorFn {

    return (c: AbstractControl): ValidationErrors | null => {

        […]
    };
}
```

Parameterizable validators

```
export function validateWithParams(validCities: string[]): ValidatorFn {

    return (c: AbstractControl): ValidationErrors | null => {

        if (c.value && !validCities.includes(c.value)) {
            return {
                city: {
                    actualCity: c.value,
                    validCities: validCities.join(', ')
                }
            };
        }

        return null;
    };
}
```

Use validators

```
this.form = fb.group({
    from: [
        'Graz',
        [Validators.required, validateWithParams(['Graz', 'Hamburg'])],
    ],
    to: ['Hamburg', Validators.required]
});
```

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Asynchronous validators

```
export function cityValidatorAsync(flightService): AsyncValidatorFn {

    return (ctrl: AbstractControl): Observable<ValidationErrors | null> => {

        […]

        return observable;
    }
}
```

Use async validators

```
this.form = fb.group({
    from: [
        'Graz',
        [Validators.required, validateWithParams(['Graz', 'Hamburg'])],
        cityValidatorAsync(this.flightService)
    ],
    to: ['Hamburg', Validators.required]
});
```

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Multifield validators

```
export function validateMultiField([…]): ValidationFn {

    return (control: AbstractControl): ValidationErrors | null {

        const formGroup = control as FormGroup;

        […]
    }

};
```

Use multifield validators

```
this.form = fb.group({ … });

this.form.validator = validateRoundTrip;

this.form.validator = validators.compose([validateRoundTrip, …])
```

```
this.form = fb.group(
  { … },
  { validators: validateRoundTrip }
);
```

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

LAB

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT