What is a FormGroup?

A **FormGroup** aggregates the values of each child FormControl into one object, with each control name as the key. It calculates its status by reducing the status values of its children. For example, if one of the controls in a group is invalid, the entire group becomes invalid.

What is FormGroup and FormControl in angular why is it used for?

It is **used** to create **angular** reactive form. ... **FormControl**: It is a class that is **used**to get and set values and validation of a **form control** such as <input> and <select> tag. **FormGroup**: It has the role to track value and validity state of group of **FormControl**

What is the difference between template driven and reactive forms?

Summary. Angular provides with us ways to build **forms**: **Template Driven and Reactive**. ... The **Reactive** approach removes validation logic from the **template**, keeping the **templates** clean **of** validation logic. But also allows **for** a whole**different** way **of** building

[FormGroup](https://angular.io/docs/ts/latest/api/forms/index/FormGroup-class.html):

A FormGroup aggregates the values of each child FormControl into one object, with each control name as the key.

const form = new FormGroup({

first: new FormControl('Nancy', Validators.minLength(2)),

last: new FormControl('Drew'),

});

[FormArray](https://angular.io/docs/ts/latest/api/forms/index/FormArray-class.html):

A FormArray aggregates the values of each child FormControl into an array.

const arr = new FormArray([

new FormControl('Nancy', Validators.minLength(2)),

new FormControl('Drew'),

]);

<section>

<p>Any special requests?</p>

<ul formArrayName="specialRequests">

<li \*ngFor="let item of orderForm.controls.specialRequests.controls; let i = index">

<input type="text" formControlName="{{i}}">

<button type="button" title="Remove Request" (click)="onRemoveSpecialRequest(i)">Remove</button>

</li>

</ul>

<button type="button" (click)="onAddSpecialRequest()">

Add a Request

</button>

</section>

constructor () {

this.orderForm = new FormGroup({

firstName: new FormControl('Nancy', Validators.minLength(2)),

lastName: new FormControl('Drew'),

specialRequests: new FormArray([

new FormControl(null)

])

});

}

onSubmitForm () {

console.log(this.orderForm.value);

}

onAddSpecialRequest () {

this.orderForm.controls

.specialRequests.push(new FormControl(null));

}

onRemoveSpecialRequest (index) {

this.orderForm.controls['specialRequests'].removeAt(index);

}

FormArray is a variant of FormGroup. The key difference is that its data gets serialized as an array (as opposed to being serialized as an object in case of FormGroup). This might be especially useful when you don’t know how many controls will be present within the group, like dynamic forms.

In order to create a reactive forms, a parent FormGroup is must. This FormGroup can further contain formControls, child formGroups or formArray

FormArray can further contain array of formControls or a formGroup itself.

this.tripForm = this.fb.group({

name: [name, Validators.required],

cities: new FormArray(

[0] ---> new FormGroup({

name: new FormControl('', Validators.required),

places: new FormArray(

[0]--> new FormGroup({

name: new FormControl('', Validators.required),

}),

[1]--> new FormGroup({

name: new FormControl('', Validators.required),

})

)

}),

[1] ---> new FormGroup({

name: new FormControl('', Validators.required),

places: new FormArray(

[0]--> new FormGroup({

name: new FormControl('', Validators.required),

}),

[1]--> new FormGroup({

name: new FormControl('', Validators.required),

})

)

}))

})

let map : { [key: string]: boolean} = {};

map["foo"] = true;

map["bar"] = false;

map["foobar"] = "foo"; // Throws exception

map[1] = true; // Curiously doesn't throws exception

map.foo = true; // Throws exception