Talk:

Hello my name is Nicu, I’m a frontend developer at yonder and in the last 1.5 year I worked with angular on building applications. Now the thing which I would like to talk about today is forms in particular @angular/forms or the angular opinion on how we should implement forms in angular. There are two modules FormsModule. Where you will write something like this.

<form #formRef="ngForm">

...

<input

type="text"

placeholder="Your full name"

required

name="name"

[(ngModel)]="user.name">

<button type="submit" [disabled]=" formRef.invalid">Sign up</button>

...

</form>

...

user: User = {

name: 'Todd Motto',

account: {

email: '',

confirm: ''

}

};

...

And of course ReactiveFormsModule. Where your code will look more like this

<form novalidate [formGroup]="myGroup">

Name: <input type="text" formControlName="name">

Location: <input type="text" formControlName="location">

</form>

ngOnInit() {

this.myGroup = new FormGroup({

name: new FormControl('Todd Motto'),

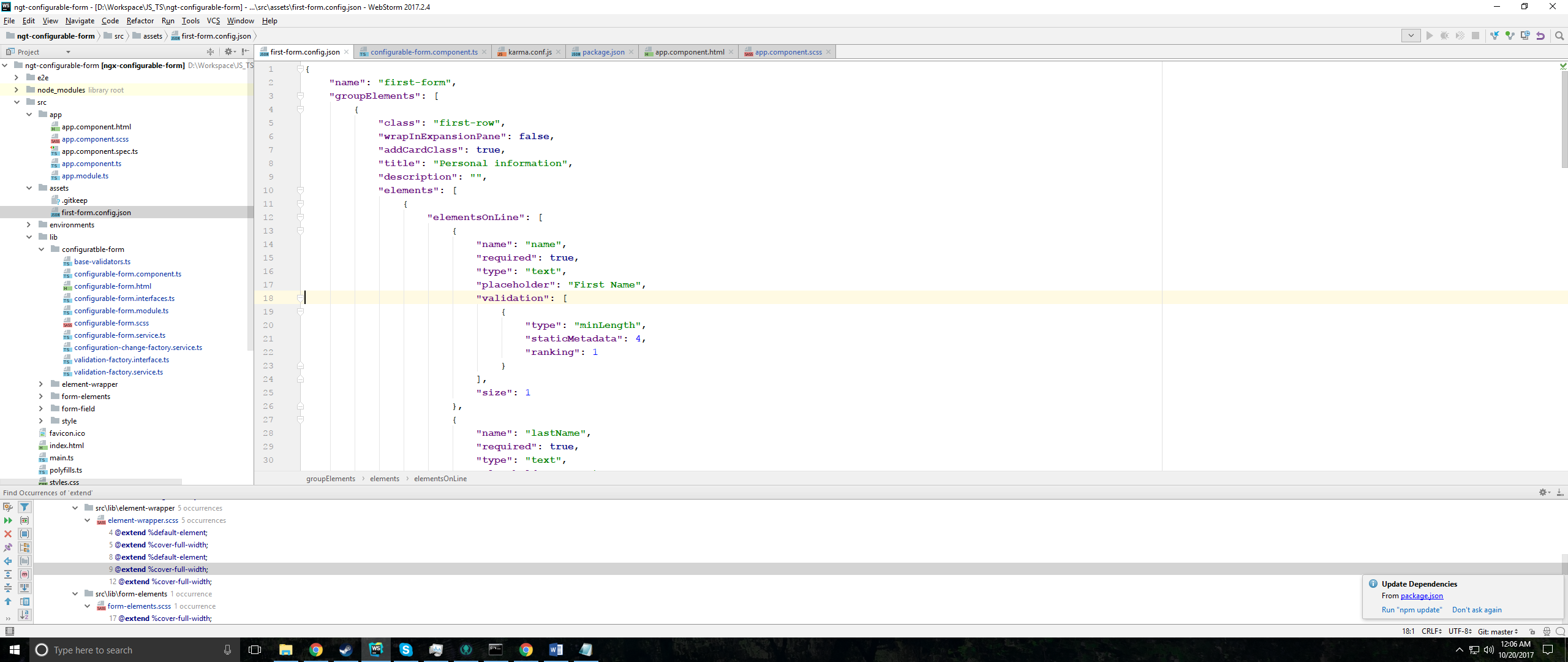
location: new FormControl('England, UK')

});

}

For a big application we build a lot of forms and we handle a lot of use cases each use case particular in it’s own way so usually we abstract those in a lot of components. And after one year we rich a large application which has inside 300+ components and let’s say that about 70+ of them are particular form implementations for each case in our app. 70+ components means a lot of specific bugs and also a lot of js code shipped to the browse. So how we can optimize this ?

We could build a configurable form which handles almost all the cases and reduce our components to less and less. For today I will present you my implementation for this. My proposal looks something like this.



At the first sight looks messier but I built a component which reads this configuration and renders. Also I started a linking system when you can define rules between fields. So this is stored in a json file and I use for this example a http request to fatch it, and my code looks something like this.

<ngt-configurable-form [formConfig]="config"

[fieldsDataProviders]="dataProviders">

</ngt-configurable-form>

So as examples for you I have implemented this using material design components. As the base for now I have the form configuration which can handle basic field linking, basic validations and data change events.

As the redux part what I noticed in the last time is that [(value)] and cycles in components like

set value(myVal: any) {

this.onMyValChanges.emit(myVal);

}

(onMyValChanges)=”emitActionValChanges($event)”

Are not really friendly with redux pattern in angular because will tringger at least two change detection cycles

set value(myVal: any) {

this.onMyValChanges.emit(new MyValue(myVal));

}

(onMyValChanges)=”emitActionValChanges($event)”

Will get you a nice infinite cycle. So what I wold like to do is create a basic integration with ngrx to prove the this component is fully compatible with redux pattern.