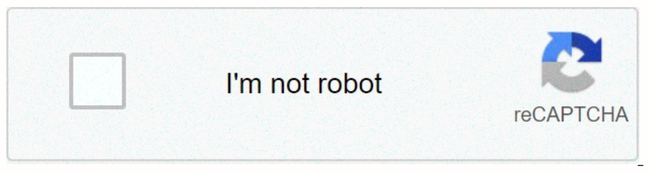

Continue

Angular form validation stackblitz

Example built with angular 8.0.0 Other available versions: Angle reactive formats: Angle 10, 9, 7, 6 Angle Pattern-driven formats: Angle 10, 9, 8, 7, 6 React Hook Form, Formik 2, Formik 1 Vue 3: VeeValidate Vue 2: Vuelidate, VeeValidate ASP.NET Core: Blazor WebAssembly This is a quick example of how to configure validation format validation in Angle 8 using reactive formats. The example is a simple registration form with several standard fields for the title, first name, last name, email, password, confirmation password, and a Ts & Cs acceptance check box. There is also a custom validator called MustMatch, which is used to validate that the password and password confirmation fields match. I have configured the form for validation during submission, and not once each field has changed, it is implemented with a property submitted to the application element that is set to true when the form is first submitted, and reset to false if you click the cancel button. Styling the example is all done with Bootstrap 4.3 CSS. Here it is in action: (See StackBlitz in Reactive App Component validation formats The application component defines the form fields and validators for our registration form by using an Angle FormBuilder to create an instance of a FormGroup stored in the registerForm property. RegisterForm then connects to the form in the following application template by using the [formGroup] directive. I also added a getter f as convenience property to make it easier to access form checks from the template. For example, you can access the confirmPassword field in the template by using f.confirmPassword instead of registerForm.controls.confirmPassword. import { Component, OnInit } from '@angular/core'; import { FormBuilder, FormGroup, Validators } from '@angular/forms'; enter a custom validator to validate this password and verify that the password fields match the {MustMatch } input from './_helpers/must-match.validator'. @Component({{ selector: 'app', templateUrl: 'app.component.html' }) the AppComponent export class implements OnInit { registerForm: FormGroup; submitted = false; constructor(private formBuilder: FormBuilder) { } ngOnInit() { this.registerForm = this.formBuilder.group({{ title: ['', Validators.required], name: ['', Validators.required], surname: ['', Validators.required], email: ['', [Validators.required, κωδικός πρόσβασης: ['', [Validators.required, Validators.minLength(6)]], confirmPassword: ['', Validators.required], acceptTerms: [false, Validators.requiredTrue] }, { validator: MustMatch('password', 'confirmPassword') }); } // getter ευκολίας για εύκολη πρόσβαση σε πεδία φόρμας get f() { return this.registerForm.controls; } onSubmit() { this.submit = true; // stop here if form is inc έγκυρο εάν (this.registerForm.invalid) { return; } // display form values values ειδοποίηση επιτυχίας ('ΕΠΙΤΥΧΙΑ!!!' + JSON.stringify (this.registerForm.value, null, 4)); } onReset() { this.subjed = false; this.registerForm.reset(); } Πρότυπο εφαρμογής επικύρωσης αντιδραστικών φορμών Το πρότυπο στοιχείου εφαρμογής περιέχει όλες τις σημειώσεις html για την εμφάνιση του παραδείγματος φόρμας δήλωσης στο πρόγραμμα περιήγησής σας. Το στοιχείο φόρμας χρησιμοποιεί την οδηγία [formGroup] για να συνδεθεί με την registerForm FormGroup στο παραπάνω στοιχείο εφαρμογής. Η φόρμα συνδέει το συμβάν υποβολής φόρμας στο πρόγραμμα χειρισμού onSubmit() στο στοιχείο εφαρμογής χρησιμοποιώντας τη σύνδεση γωνιακού συμβάντος (ngSubmit)=onSubmit(). Τα μηνύματα επικύρωσης εμφανίζονται μόνο αφού ο χρήστης προσπαθήσει να υποβάλει τη φόρμα για πρώτη φορά, αυτό ελέγχεται με την ιδιότητα που υποβάλλεται από το στοιχείο εφαρμογής. Το συμβάν κλικ του κουμπιού ακύρωσης είναι συνδεδεμένο με το πρόγραμμα χειρισμού onReset() στο στοιχείο εφαρμογής χρησιμοποιώντας τη σύνδεση γωνιακού συμβάντος (click)=onReset(). <!-- main app container --><div class=card m-3><h5 class=card-header>Γωνιακή επικύρωση 8 αντιδραστικής μορφής</h5><div class=card-body><form [formgroup]=registerForm (ngsubmit)=onSubmit()><div class=form-row><div class=form-group col><label>Τίτλος</label><select formcontrolname=title class=form-control [ngclass]={ 'is-invalid': submitted && f.title.errors }><option value=><option value=Mr>Κύριε<option value=Mrs>Κα<option value=Miss>Χάσετε<option value=Ms>Κα</select><div *ngif=submitted && f.title.errors class=invalid-feedback><div *ngif=f.title.errors.required>Απαιτείται τίτλος</div></div></div><div class=form-group col-5><label>Όνομα</label><input type=text formcontrolname=firstName class=form-control [ngclass]={ 'is-invalid' : submitted && f.firstName.errors }><div *ngif=submitted && f.firstName.errors class=invalid-feedback><div *ngif=f.firstName.errors.required>Απαιτείται όνομα</div></div></div><div class=form-group col-5><label>Επώνυμο</label><input type=text formcontrolname=lastName class=form-control [ngclass]={ 'is-invalid': submitted && f.lastName.errors }><div *ngif=submitted && f.lastName.errors class=invalid-feedback><div *ngif=f.lastName.errors.required>Απαιτείται επώνυμο</div></div></div></div><div class=form-group><label>Ηλεκτρονικό ταχυδρομείο</label><input type=text formcontrolname=email class=form-control [ngclass]={ 'is-invalid': submitted && f.email.errors }><div *ngif=submitted && f.email.errors class=invalid-feedback><div *ngif=f.email.errors.required>Απαιτείται ηλεκτρονικό ταχυδρομείο</div><div *ngif=f.email.errors.email>Το ηλεκτρονικό ταχυδρομείο πρέπει να είναι έγκυρη διεύθυνση ταχυδρομείου</div></div></div><div class=form-row><div class=form-group col><label>Κωδικός πρόσβασης</label><input type=password formcontrolname=password class=form-control [ngclass]={ 'is-invalid': submitted && f.password.errors }><div *ngif=submitted && f.password.errors class=invalid-feedback><div *ngif=f.password.errors.required>Απαιτείται κωδικός πρόσβασης</div><div ></div ></div></div></div></form></div></div> ></div></div></div></form></div></div> πρέπει να έχει τουλάχιστον 6 χαρακτήρες<div class=form-group col><label>Επιβεβαίωση κωδικού πρόσβασης</label><input type=password formcontrolname=confirmPassword class=form-control [ngclass]={ 'is-invalid': submitted && f.confirmPassword.errors }><div *ngif=submitted && f.confirmPassword.errors class=invalid-feedback><div *ngif=f.confirmPassword.errors.required>Επιβεβαίωση ότι απαιτείται κωδικός πρόσβασης</div></div><div *ngif=f.confirmPassword.errors.mustMatch>Οι κωδικοί πρόσβασης πρέπει να ταιριάζουν</div></div></div></div><div class=form-group form-check><input type=checkbox formcontrolname=acceptTerms id=acceptTerms class=form-check-input [ngclass]={ 'is-invalid': submitted && f.acceptTerms.errors }><label for=acceptTerms class=form-check-label>Αποδοχή Όρων και Προϋποθέσεων</label><div *ngif=submitted && f.acceptTerms.errors class=invalid-feedback>Απαιτείται αποδοχή ts & cs</div></div><div class=text-center><button class=btn btn-primary mr-1>Ακύρωση καταχώρησης</button> <button class=btn btn-secondary type= reset (click)=onReset()></button></div>Reactive Custom Must Match Validator Formats The custom MustMatch validator is used in this example to validate that both password fields - password and Password confirmation - match. However, it can be used to validate that any pair of fields matches (e.g. e-mail and e-mail confirmation fields). It works slightly differently from a standard custom validator because I'm setting the error in the second field instead of returning to be set to the form group. I did it this way because I think it makes the template a little cleaner and more intuitive, the mustMatch validation error appears under the confirmPassword field so I think it makes sense that the error is attached to the confirmPassword check form. import { FormGroup } from '@angular/forms'; custom validation to check that two fields match the MustMatch(controlName: string, matchingControlName: string) { return (formGroup: FormGroup) => { control = formGroup.controls[controlName]; const matchingControl = formGroup.controls[matchingControlName]; if (matchingControl.errors & !matchingControl.errors.mustMatch) { // return if another validator has already found an error on the matchingControl return? } // set error on matchingControl, if (control.value !== matchingControl.value) { matchingControl.setErrors({ mustMatch: true }); Reactive Forms Validation App Module There is not much going on in the app section other than standard stuff, the main thing to remember about using reactive formats in Angle is to import the ReactiveFormsModule from @angular/formats and include in the import order of @NgModule decorator. import { NgModule } from '@angular/core'; import { BrowserModule } from the '@angular/platform browser'. import { ReactiveFormsModule } from '@angular/forms'. import { AppComponent } from @NgModule({ imports: [BrowserModule, ReactiveFormsModule], statements: [AppComponent], startup: [AppComponent] }) Export category AppModule { } Subscribe to my YouTube channel or follow me on Twitter or GitHub to be notified when I post new content. Content.