How to rerun validation on submit in React Final Form

Asked 5 years, 5 months ago Modified 4 years, 11 months ago

Viewed 22k times

5

Once a form's fields have been validated, submitting doesn't trigger a rerun of the validation. Is there a way I can trigger a rerun of the validation when the form is submitted?

I have a form field whose value can become invalid if it's not submitted within a particular timeframe. It's not async; I'm just trying to cover a scenario in which the user doesn't click submit for a while, and when they eventually do, the value would have become invalid. Final form remembers the result of the validation that happens immediately after the value is changed, which means that the unchanged value remains valid regardless of how much time passes between the validation and the submission. This is the behavior I want to hook into and change; the intervening time matters in my use case. I have tried using the `beforeSubmit` listener from the `final-form-submit-listener` package but it only gives access to the `FormApi` object. I tried using the `pauseValidation` and `resumeValidation` functions from `FormApi` but they couldn't achieve what I want, or maybe I'm not using them correctly. I have a feeling it's painfully obvious how to do this, but I can't figure it out. 😩

I created this Sandbox to demonstrate what I mean.

Thanks!

UPDATE: Some additional information:

- This is for a time picker. If you're picking times for today, you may pick a time that is 15 minutes from now. It's valid now because it's currently in the future. If you don't touch the form for the next 20 minutes then click submit, the submission should be prevented because your selected time is now 5 minutes in the past.
- I have considered just adding the validation directly in the submit handler. Two answers here do this. However, it is not ideal for me because Final Form doesn't receive the errors and pass them to the `meta` object for the form fields. My codebase is complex and relies heavily upon the `meta` object to display error messages. Trying to replicate that functionality in the submit handler may work but it's hacky and goes against the convention used throughout the codebase.

javascript **reactjs** **react-final-form** **final-form**

edited Jul 11, 2019 at 12:35

asked Jul 11, 2019 at 11:19

Uche Ozoemena

926 ● 2 ● 11 ● 33

If it's validated once why revalidate? – its4zahoor Jul 11, 2019 at 11:23

@its4zahoor As I said above, the intervening time matters in my use case. The value can become invalid after a while, so I want to make sure that it gets checked just before it's submitted. – Uche Ozoemena Jul 11, 2019 at 11:25

Okay, let me see sandbox – its4zahoor Jul 11, 2019 at 11:29

5 Answers

Sorted by: Highest score (default)

21

Library author here. I'm always fascinated by new ways people can invalidate my assumptions. I mean that in a sincerely positive way, as it results in learning.

🏁 **Final Form makes the assumption that your validation functions are "pure" or "idempotent"**, i.e. will always return the same result when given the same values. This is why it doesn't run the synchronous validation again (just to double check) before allowing the submission: because it's already stored the results of the last time it ran it. By using an outside timer, you've invalidated that assumption.

> If you have a better/simpler/"more official" solution, I'd still love to see it!

No need for mutators or decorators for this problem.

The more official way to do this would be to run the check (you could even reuse `this.validate`) in `onSubmit`. The only tricky part is that the error will come back as `meta.submitError`, so you need to check for *both* when displaying your error. Like so:

Edit in CodeSandbox

Share Improve this answer

Follow

answered Jul 12, 2019 at 11:59

Erik R.

7,272 ● 1 ● 31 ● 39

Fascinating! I never even thought to look into `submitError`. I'll implement it in my real code first just to be sure it works as I expect it to. Thanks a lot!
– Uche Ozoemena Jul 12, 2019 at 12:13

1 This results in redundant code per component. A `validateOnSubmit` property would be convenient.
– Steven Vachon Oct 20, 2019 at 18:40

1 Some user experience issue with the proposed solution: This won't show up the inline field error message if you just hit on submit and other fields already contained errors since the code on the onSubmit won't be hit – Braulio Jan 9, 2020 at 15:04

4

So I have found a way to do this! 🎉 I use a mutator and use it's `changeValue` function to 'change' the value of the relevant field (I supply the same value). This in turn

notifies all relevant parties of the change to the form's state, and a validation is triggered. The key is to call the mutator inside the submit handler, which therefore ensures that the validation is performed when the form is submitted. Have a look at this new Sandbox.

The relevant bits are as follows:

```
// this is a stateful component
...
...
  mutateValue([name], state, { changeValue }) {
    // change the value to the same value, thus
    // triggering a revalidation of the same value
    changeValue(state, name, value => value);
  }

  handleSubmit(values) {
    alert("submitted");
  }

  render() {
    return (
  ...
  ...
        <Form
          onSubmit={this.handleSubmit}
          mutators={{ mutateValue: this.mutateValue }}
          render={({
            handleSubmit,
            form: {
              mutators: { mutateValue }
            }
          }) => {
            const mutateBeforeSubmit = values => {
              // supply the name of the relevant form
              mutateValue("revalidate");
              // submit handler gets called if revalid
              handleSubmit(values);
            };
            return (
```

```
            <form onSubmit={mutateBeforeSubmit}>
            ...
            ...
            </form>
          );
        }}
      />
      ...
      ...
```

And because it's triggering the same validation mechanism, `meta` gets used accordingly!

If you have a better/simpler/"more official" solution, I'd still love to see it!

Share Improve this answer Follow

edited Jul 12, 2019 at 8:21

answered Jul 12, 2019 at 8:10

Uche Ozoemena

926 ● 2 ● 11 ● 33

2

You're already putting a function inside `onSubmit`, why not just add the functionality you want to it? `event.preventDefault()` and then work with your validate function, it' s a part of the component and accessible to you.

```
handleOnSubmit(e){
  let value = document.querySelector("input").value
  if (!!this.validate(value)){
    e.preventDefault();
    alert("Prevented submit event")
```

```
  } else{
    alert("Form submitted")
  }
}
```

now just use this function in the form `onSubmit` prop(I put in bot since i wasn't sure about the component structure):

```
<Form onSubmit={this.handleOnSubmit}>...</Form>
<form onSubmit={this.handleOnSubmit}>
```

And **remove** the submitListener decorator from the Form component:

```
decortaor={submitListener}
```

Now it will check the validation before submitting and prevent it if not validated.

Share Improve this answer Follow

edited Jul 11, 2019 at 12:03

answered Jul 11, 2019 at 11:29

aviya.developer

3,573 ● 3 ● 20 ● 46

Oh sorry just noticed i'm passing the same form button submit input into validate so let me fix that by finding the required element – aviya.developer Jul 11, 2019 at 11:31

also i don't see any `handleSubmit` declaration in your file – its4zahoor Jul 11, 2019 at 11:32

1 Yeah s/he is just using an arrow function in the onSubmit prop, i assumed based on the code already shown that it's a given for them to put it. But i'll add to the answer anyways. – aviya.developer Jul 11, 2019 at 11:34

Yes @aviya.developer there's an actual handler there in the real code, I only used a simple arrow function for the sandbox. Let me try this out and see. – Uche Ozoemena Jul 11, 2019 at 11:39

1 @its4zahoor it's based on both. It's a time picker, and if you're picking times for today, you may pick a time that is 15 minutes from now. It's valid now because it's currently in the future. If you don't touch the form for the next 20 minutes then click submit, the submission should be prevented because your selected time is now 5 minutes in the past. – Uche Ozoemena Jul 11, 2019 at 11:54

1

Got another use case where manual triggering validation is needed: In this case we got an student form, in case we got a new student, a field called *temporary password* must be fullfilled, if not that field is not mandatory, to keep it simple we update the validation schema on the fly (we had a similar case with a validation that needs to be included whenever some fetch operation was completed).

We applied @uche-ozoemena and it worked fine, but @erik-r is this solution considered a hack or is something that we can use as an approved workaround in scenarios where we need to manually trigger validations?

We got something like (using fonk-final-form):

```
React.useEffect(() => {
  if (isUserCreation) {
    const newDataFormvalidationSchema = {
      ...dataFormvalidationSchema,
      field: {
        ...dataFormvalidationSchema.field,
        temporaryInitialPassword: [
          ...dataFormvalidationSchema.field.temporar
          Validators.required,
        ],
      },
    };

    dataFormValidation.updateValidationSchema(newDat
  }
}, [isUserCreation]);

return (
  <Form
    initialValues={initialData}
    mutators={{ mutateValue: mutateValue }}
    onSubmit={values => {
      save(values);
    }}
    validate={dataFormValidation.validateForm}
```

Share Improve this answer

Follow

answered Jan 9, 2020 at 13:41

Braulio

1,728 ● 14 ● 24

0

Since you want to enforce re-validation OR **stop submission** of form based on the `interval`, why not use `disabled` on submit button?

```
// interval less than 900 = 15 minutes
<button type="submit" disabled={this.state.interval>=9
```

```
    Submit
</button>
```

I have also read docs of `react-final-form` but I thinks it's more easy unless you have a specific case to address using `meta`.

Share Improve this answer Follow

edited Jul 12, 2019 at 8:06

answered Jul 11, 2019 at 12:05

its4zahoor

1,791 ● 1 ● 16 ● 23

1 Updated :-). I thought on changes it creates fork autmatically :/ – its4zahoor Jul 11, 2019 at 12:10

1 `meta` ? can you elaborate meta a little? are you talking about `meta tags` – its4zahoor Jul 11, 2019 at 12:16

1 Oh no not meta tags lol. It's an object supplied to each `Field` component - see the docs here github.com/final-form/react-final-form#fieldrenderprops. I created an issue for this on GitHub yesterday. I'll wait until the author of the project responds. Thanks a lot for the effort though! I'll upvote your answer. :-) – Uche Ozoemena Jul 11, 2019 at 12:23

1 Nice! Yeah it's great. I just posted the solution I came up with. Have a look. :-) It plays nicely with `meta` so I'm happy. :-D – Uche Ozoemena Jul 12, 2019 at 8:11

1 Just saw your update to the answer. Yeah that's another nice idea actually, but it doesn't offer a way to leave an error message using `meta`. That `meta` part is key, and it only happens when you hook into Final Form's validation

mechanism. My solution does this nicely. :-)
– Uche Ozoemena Jul 12, 2019 at 8:18