Displaying Actual Errors

The first step to getting error messages to show up when validation fails

WE'LL COVER THE FOLLOWING

- Changelist
- Changelist

We now have a basic form, some JavaScript to execute validation on each input at the correct time, a way to distinguish which field is which, and a set of validation rules to apply to each one. Let's put the pieces together now!

Output

JavaScript

HTML

CSS (SCSS)

Name

First Last

Choose your username

Create a password

Confirm your password

Birthday

Month

Day Year

Mobile phone

Your current email address

Changelist

- Create a unique error class
  - This lets us do things only with expected errors that arise from validation by checking `instanceof`.
- Introduce a `try/catch` for validation.
  - The `catch` clause is where the error view is rendered.
  - We could have used enums and passed back something like `{status: 'error', message: 'blah'}` for errors and `{status: 'success'}` to signal passing validation, but if the set of return possibilities are many different ways of specifying a type of error and one way of specifying success, we should make use of error handling which covers this exact case. We want to avoid defining a new ad-hoc object, its schema (what fields it contains) is just another thing to keep track of.
- Each input maps to the validation rule through a mapping.
  - Cleaner than a switch statement.
- `validate` functions that match the previous lesson, except they throw an error when validation fails instead of returning `false`.

Hopefully, this all makes sense. I skipped `password` and `confirmPassword` for now because that uses some special handling. Try playing around with this form and check for yourself.

When you implement a new rule, how do you check that it works? Testing it manually doesn't give us confidence, so let's adjust our test and see if it passes.

Output

JavaScript

HTML

Test Results

validateName(asdf) passes
validateName(Alfred) passes
validateName(ALFRED) passes

validateName(@) fails with message: Please enter a valid name
validateName() fails with message: Please enter a valid name
validateName(blah$) fails with message: Please enter a valid name
validateName(123) fails with message: Please enter a valid name

validateEmail(asdf@asdf.com) passes
validateEmail(what@what.au) passes
validateEmail(a@a.c) passes

validateEmail(@asdf.com) fails with message: Please enter a valid email
validateEmail(what@what) fails with message: Please enter a valid email
validateEmail() fails with message: Please enter a valid email
validateEmail(..) fails with message: Please enter a valid email

The source code JavaScript is in a hidden file, but note that this test runner HTML would import `js` files from the source before the test so that the test can reference functions in the source.

Changelist

- Instead of passing in a function that returns a boolean, properly pass in the `validateX` functions.
- Instead of calling the functions in the runner and interpreting them as boolean results, we use the `try` branch for success and `catch` for failure.
- Messages are wrapped in a span that colors the text red if the test failed.
- A test failure means the passing of validation does not match what actually happened. This occurs when the `try` block runs completely, but failure was expected, or the `catch` block runs, but success was expected.