Model-based Form Validation with Vue.js and Vuelidate

alligator.io/vuejs/model-form-validation-vuelidate

While the most beginner-friendly approach to form validation in Vue.js might be through template-based forms, a much more flexible way is to validate the model instead. Model-based validation tends to be easier to understand and change for larger apps, moving the visual clutter from the template to the model and reducing it significantly. The most well-known library to accomplish this in *Vue* is through Vuelidate.

Installation

As usual, *Vuelidate* can be installed from NPM or Yarn.

```
# Yarn
$ yarn add vuelidate

# NPM
$ npm install vuelidate --save
```

Then, in your app bootstrap, enable the *Vuelidate* plugin.

main.js

```
import Vue from 'vue';
import Vuelidate from 'vuelidate';
import App from 'App.vue';

Vue.use(Vuelidate);

new Vue({
  el: '#app',
  render: h => h(App)
});
```

Field Validation

To validate fields, add a definition object for properties in your *data* model to the *validations* property of your component. Then import validation functions from *vuelidate/lib/validations* (or custom ones you create). You will then be able to bind to the data property via *v-model* as usual. Validation information will be stored in *this.$v[propertyName]*.

$v's Schema:

```
$v[propertyName]: {
  $dirty: boolean,
  $invalid: boolean,
  $error: boolean,
  $pending: boolean,
  $each: object
  [YourValidationName]: boolean
  [Nested]: object
}
```

```
<template>
  <form>
    <input type="text" v-model="emailValue"/>
    <p v-if="!$v.emailValue.required">The email field is required!</p>
    <p v-if="!$v.emailValue.email">The input must be a proper email!</p>
  </form>
</template>

<script>
import { required, email } from 'vuelidate/lib/validations'

export default {
  data() {
    return {
      emailValue: ''
    }
  },

  validations: {
    emailValue: {
      required,
      email
    }
  }
}
</script>
```

Built-In Validators

Vuelidate provides these validators by default. They can be imported from *vuelidate/lib/validators*. If you're using Webpack 2 or Rollup with tree shaking, any validators not used in your project will not be bundled with

it.

- *required()* - This field cannot be empty.
- *minLength(length: number)* - Minimum length of the field. Works on strings and arrays.
- *maxLength(length: number)* - Maximum length of the field. Works on strings and arrays.
- *between(min, max)* - Requires a number to be between the minimum and maximum values.
- *alpha()* - Only allows alphabetic characters.
- *alphaNum()* - Only allows alphanumeric characters.
- *email()* - Allow only valid emails.
- *sameAs(fieldName: string | getFieldName: function -> string)* - Allows you to require the input to be the same as another field, specified by fieldName or a function.
- *or(validators...)* - Valid when at least one of the specified validators is valid.
- *and(validators...)* - Valid when all of the specified validators are valid.

Custom Validatiors

A custom validator in Vuelidate is simply a function that returns a boolean or a promise that resolves to a boolean.

If you want to ensure that fields contain only the word “tom”, you might write a validator like this.

tom-validator.js

```
export const TomValidator = (value, component) => {
  return value === 'tom';
}
```

MyComponent.vue

```
...
<script>
import { TomValidator } from './tom-validator';
export default {
  data() {
    return {
      inputField: ''
    }
  },

  validators: {
    inputField: {
      TomValidator
    }
  }
}
</script>
```

You might want to go a step further and allow a string to be specified that must be matched. In that case, just create a higher-order function that returns the validator function, like so.

match-validator.js

```
export const MatchValidator = (stringToMatch) => {
  return (value, component) => value === stringToMatch;
}
```

MyComponent.vue

```
...
<script>
import { MatchValidator } from './match-validator';
export default {
  data() {
    return {
      inputField: ''
    }
  },

  validators: {
    inputField: {
      MatchValidator: MatchValidator('rupert')

    }
  }
}
</script>
```