Getting Started

1 Introduction

2 Tools

3 Our Task

4 Hello World

Vue Basics

5 Vue CLI

7 Template Syntax

9 Styles & Classes

10 Single Page Apps

11 Creating Routes

15 Form Validation

API Backend

17 Prelude

18 Setup

Vuex

20 Enabling CORS

21 Schemas & Models

22 Routes and Controllers

23 Testing with POSTman

24 The Current Problem

25 State Management

Using Store In Components

30 LAB: Product List

31 LAB: Product Details

32 LAB: Admin Features

HTML

HTML

29 State and Actions on Components Front

33 LAB: Spinner, Cart & Strore Subscrip

26 Getters

27 Mutations

28 Actions

12 Route Outlets & Links

13 Nested & Dynamic Routes

14 Two Way Binding (Reactivity) Free

16 Handling Form Submissions Free

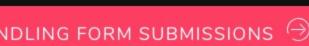
19 Provisioning a MongoDB Database F

Routing

Forms

6 Components and the Vue Instance Fr

8 Conditional & List Rendering Free



Validating forms is not a simple task on any platform. It's even more difficult when building websites because user experience is demanding these days, and we need to quench the thirst. People expect real-time immediate-updating super-informative forms.

If you have been coding for 5+ years, you might have seen the era when validation was done only on the server. Your form has to send a post request, the server validates the information, and if there is an error, it throws your payload back at you with the errors. This is poor because the user has to wait for the server trip only to be notified that information provided is invalid.

JavaScript validation on the client was around in that era, but it wasn't a norm until user experience started becoming more and more demanding. Unfortunately, it's not such an easy task to validate a form with JavaScript. For this reason, a lot of libraries and framework dependent libraries try to make this simple. In fact, a lot of them do a great job at that. Vue is not left out, and vee-validate is one of its validation libraries.

Let's install vee-validate in our Vue app:

```
BASH
npm install --save vee-validate
```

Then import and configure in the entry file:

```
JS
// ./src/main.js
import VeeValidate from 'vee-validate'
Vue.use(VeeValidate)
```

validate our form field:

You can go ahead to use the vee-validate APIs and directives. For example, let's

```
HTML
<input type="text" placeholder="Name" v-model="model.name" v-validate="'required'" name="name" />
```

Now you can check if there are errors via the error property:

```
JS
export default {
 methods: {
   saveProduct() {
     // prints all the errors
     console.log(this.errors);
```

You can use the power of class or style binding to style invalid inputs differently:

```
HTML
<input type="text" placeholder="Name" v-model="model.name" v-validate="'required'" name="name" :cl</pre>
If errors.name exists, an error class will be added to the input. This is how this class
is styled:
                                                                                               CSS
.form-control.error {
 border-color: #FF3333;
 box-shadow: inset 0 1px 1px rgba(0,0,0,.075), 0 0 8px rgba(255, 71, 71, 0.6);
```

Red signs might not be enough. We need to let the user know what they did wrong. This can be achieved with an error message:

Adding a red border and inset red box shadow to the element.

```
You can update the rest of the elements to validate them:
```

Name is required

<template>

<form @submit.prevent="saveProduct">

```
<div class="col-lg-5 col-md-5 col-sm-12 col-xs-12">
      <div class="form-group">
        <label>Name</label>
        <input type="text" placeholder="Name" v-model="model.name" v-validate="'required'" name="n</pre>
        <span class="small text-danger" v-show="errors.has('name')">Name is required</span>
      </div>
      <div class="form-group">
        <label>Price</label>
        <input type="number" class="form-control" placeholder="Price" v-model="model.price" v-vali</pre>
        <span class="small text-danger" v-show="errors.has('price')">Price is required</span>
      </div>
      <div class="form-group">
        <label>Manufacturer</label>
        <select type="text" class="form-control" v-model="model.manufacturer" v-validate="'require</pre>
          <template v-for="manufacturer in manufacturers">
            <option :value="manufacturer._id" :selected="manufacturer._id == (model.manufacturer &</pre>
          </template>
        </select>
        <span class="small text-danger" v-show="errors.has('manufacturer')">Manufacturer is requir
      </div>
    </div>
    <div class="col-lg-4 col-md-4 col-sm-12 col-xs-12">
      <div class="form-group">
        <label>Image</label>
        <input type="text" lass="form-control" placeholder="Image" v-model="model.image" v-validat</pre>
        <span class="small text-danger" v-show="errors.has('image')">Image is required and must be
      </div>
      <div class="form-group">
        <label>Description</label>
        <textarea type="number" class="form-control" placeholder="Description" rows="5" v-model="m
        <span class="small text-danger" v-show="errors.has('description')">Description is required
      </div>
      <div class="form-group new-button">
        <button class="button">
          <i class="fa fa-pencil"></i></i>
          <span v-if="isEditing">Update Product</span>
          <span v-else>Add Product</span>
        </button>
      </div>
    </div>
  </form>
</template>
Errors will be shown when we have invalid fields as shown in the image below:
    Scotch Store
                                                                              ☆ :
```

```
HOME ADMIN CART
                               Name
  NEW PRODUCT
                               Price
                                                                                  Description
                                Price
                                                                                   Description
                               Manufacturer
                                                                                                          ADD PRODUCT
```

NOTE Client side validation does not guarantee that what is

sent to the server is valid. A user can send values to the

server via other clients that you don't have control of like

Postman. Be sure to validate on the server too. too.