Build an Online Shop with Vue: Lesson 13 of 33

Nested & Dynamic Routes

NEXT: TWO WAY BINDING (REACTIVITY)

Extreme cases demand that you nest related routes under a particular feature. Let me give you an example. Assuming an `admin` feature needs to have `create`, `edit`, and `list` pages -- this might be what we would do for the routes:

```js
export default new Router({
  routes: [
    {
      path: '/',
      name: 'Home',
      component: Home,
    },
    {
      path: '/admin',
      name: 'Admin',
      component: Admin,
    },
    {
      path: '/admin/create',
      name: 'CreateAdmin',
      component: CreateAdmin,
    },
    {
      path: '/admin/edit',
      name: 'EditAdmin',
      component: EditAdmin,
    },
    {
      path: '/Cart',
      name: 'Cart',
      component: Cart,
    }
  ]
});
```

This will work fine, but it won't scale well and feels rigid. We can't just start moving things around in-app whose routes are structured like the one we have above.

Nested Routes allow you to make `/admin` a parent route which can have numerous children like `/create`, `/edit`, and `/`. The endpoint will still resolve to `/admin/create`, `/admin/edit`, and `/admin`.

Table of Contents

To create nested routes, we just need to add a `children` property to the route's configuration. This property takes an array which can contain the nested routes:

```js
// src/routes/index.js
import Home from '@/pages/Home';
import Cart from '@/pages/Cart';

// Admin Components
import Index from '@/pages/admin/Index'
import New from '@/pages/admin/New'
import Products from '@/pages/admin/Products'
import Edit from '@/pages/admin/Edit'

export default new Router({
  routes: [
    {
      path: '/',
      name: 'Home',
      component: Home
    },
    {
      path: '/cart',
      name: 'Cart',
      component: Cart
    },
    {
      path: '/admin',
      name: 'Admin',

      // Parent routes still has a component
      component: Index,

      // Child routes
      children: [
        {
          path: 'new',
          name: 'New',
          component: New
        },
        {
          path: '',
          name: 'Products',
          component: Products
        },
        {
          path: 'edit/:id',
          name: 'Edit',
          component: Edit
        }
      ]
    }
  ]
});
```

The routes are set, but the components are not yet created. This part of the series is not concerned with how the components work. Therefore we can fill it up with simple templates as we have been doing:

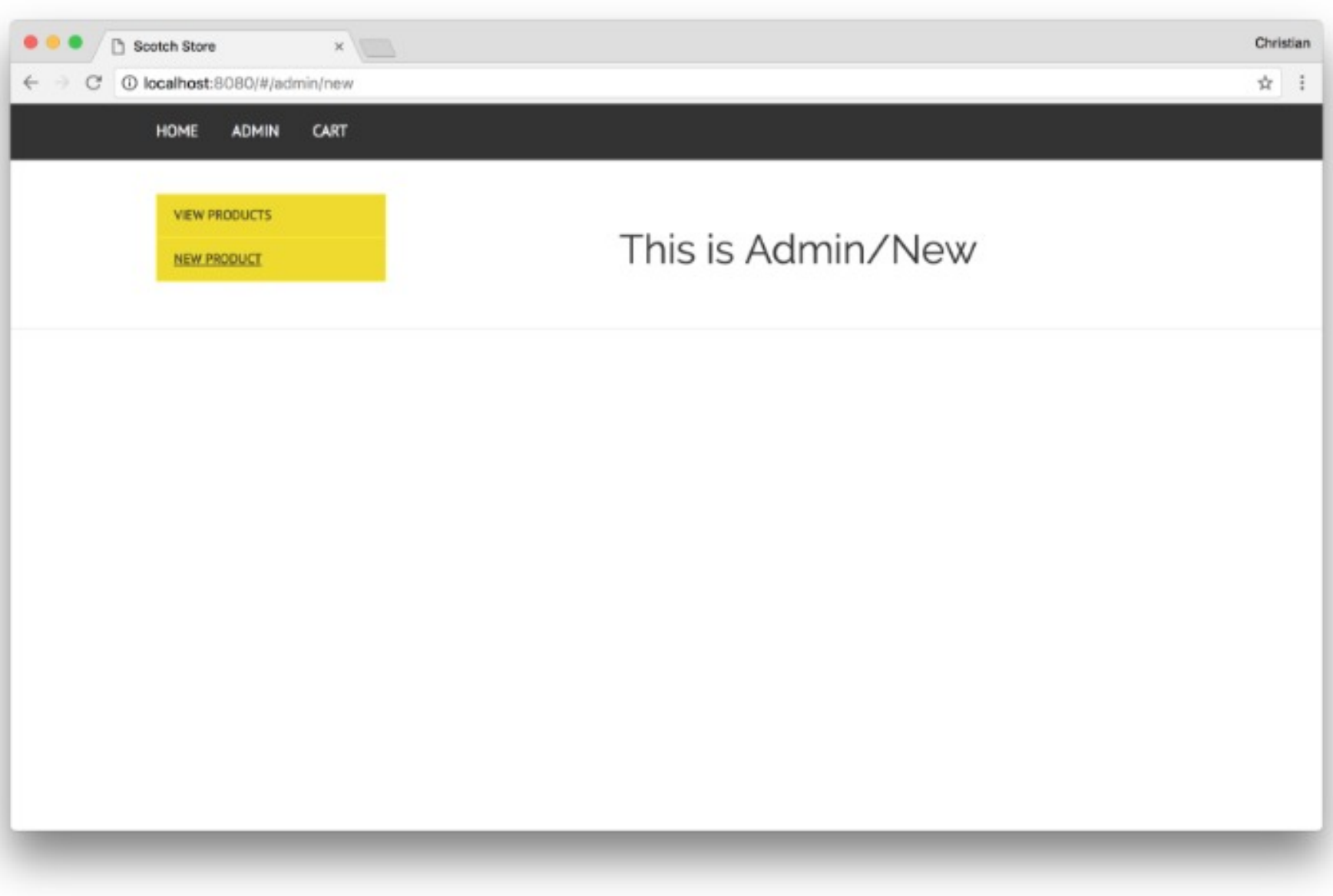
```html
<!-- ./src/pages/admin/Index.vue -->
<template>
  <div>
    <div class="admin-new">
      <div class="container">
        <div class="col-lg-3 col-md-3 col-sm-12 col-xs-12">
          <ul class="admin-menu">
            <!-- Links are to sibling routes -->
            <li><router-link to="/admin">View Products</router-link></li>
            <li><router-link to="/admin/new">New Product</router-link></li>
          </ul>
        </div>
        <!-- Outlet for children routes -->
        <router-view></router-view>
      </div>
    </div>
  </div>
</template>
```

The `Index.vue` above serves as the Admin parent route's entry point. Therefore, it's template is where the outlet for the children routes are specified.

Notice that we are only creating the template here. This is totally valid and will work since we don't need any data for these templates.

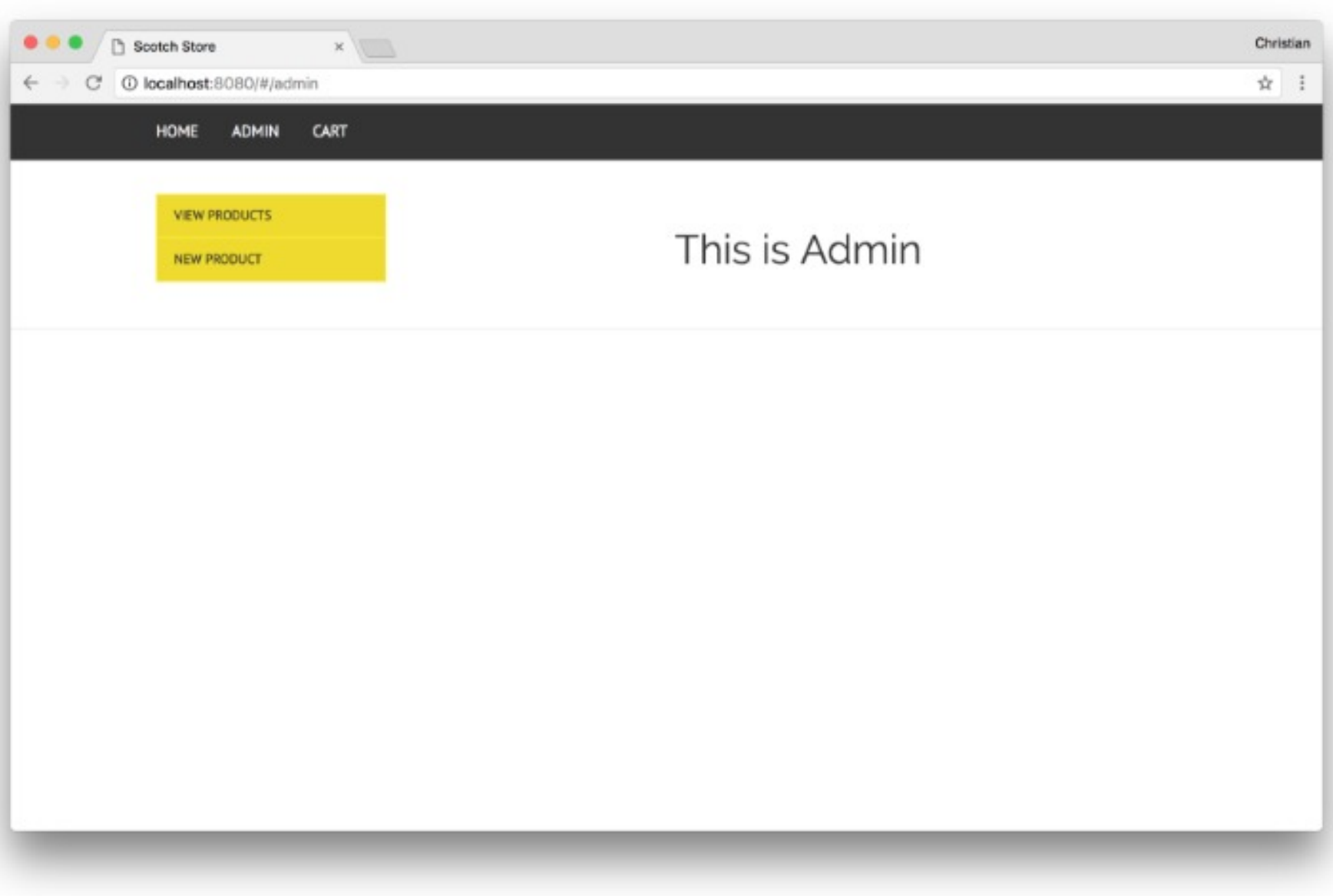
The rest of the components are Admin's children components:

```html
<!-- ./src/pages/admin/New.vue -->
<template>
  <div>
    <div class="title">
      <h1>This is Admin/New</h1>
    </div>
  </div>
</template>
```

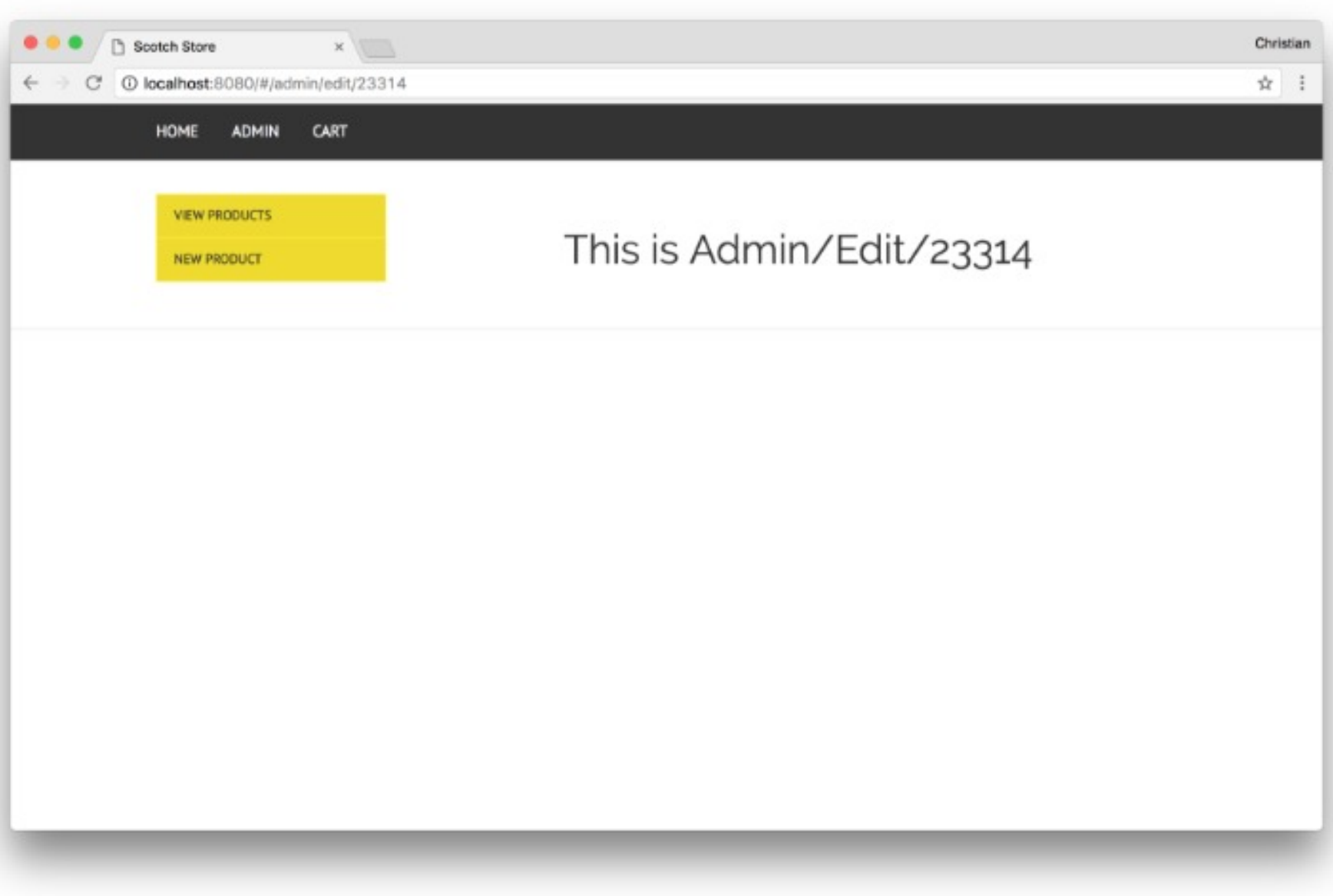


[ADMIN NEW IMAGE HERE]

```html
<!-- ./src/pages/admin/Products.vue -->
<template>
  <div>
    <div class="title">
      <h1>This is Admin</h1>
    </div>
  </div>
</template>
```



```html
<!-- ./src/pages/admin/Edit.vue -->
<template>
  <div>
    <div class="title">
      <h1>This is Admin/Edit/{{$route.params.id}}</h1>
    </div>
  </div>
</template>
```



#Dynamic Routes

Previously, we saw one of the Admin's child component that looked like this:

```js
{
  path: 'edit/:id',
  name: 'Edit',
  component: Edit
}
```

The route's path has a dynamic portion, `:id`. This part of the path is used to manipulate the state of the route's component using whatever value is passed via `:id`, thus making the route *dynamic*.

In the component, you can have access to the value passed in via the `params` object on `$route`. Example:

```js
export default {
  created() {
    console.log(this.$route.params.id) // prints value of :id
  }
}
```

We will see more practical use-cases while building the functional parts of our Admin and Cart pages. We will also have a Details page that shows detailed information about a selected product. It will also require a dynamic route.