Lab: Multiple Page Layouts

With Vue JS

**Prerequisite:**

1. Install the latest version of vue/cli from an elevated command prompt (or your favorite command line tool).

npm install -g @vue/cli@latest

2. Create a new Vue app:

a. In command prompt, navigate to the folder where you want to create the project. If you run into issues or have questions, see Vue CLI docs.

b. To create a project with vue, run: vue create [your-project-name]

a. vue create multiple-layouts

b. Select **Manually Select Features** when prompted (use up / down arrow to move and space bar to select / deselect).

c. Select the following:

1. Babel

2. Router

3. Linter

d. When prompted for linting rule set, select Standard.

c. When Vue finishes creating the project it will tell you how to start it:

cd multiple-layouts

npm run serve

d. When serve finishes running, open browser to <http://localhost:8080/>

If you want your app to automatically open the browser for you:

* In package.json, add these to the scripts section:

"dev": "npm run lint && npm run dev-build && npm run serve",

"dev-build": "vue-cli-service build --mode development",

* In the serve script, add:

--open --modern --port 8080

Now you can run: npm run dev and it should automatically open the browser for you.

**Use cases for different page layouts**

* Migrating the UX to something dramatically different. Might need different style sheets, page layouts, etc.
* Login page vs authenticated page
* Marketing email campaign links to a landing page.
  + (In marketing terms, this is a page that is created specifically to gather user information like a form to sign up for newsletters or enter address to get a free sample by mail, etc. Generally, landing pages do not have any navigation. They tend to be simple forms and have a button. Then it might redirect to a page within your site when the user is done.
* Different layouts for different areas of your site.

**Instructions**

1. Open the new project in VS Code
2. Open App.vue and router.js from the root src folder
   1. Notice that App.vue has the navigation at the top.
   2. Notice the <router-view /> tag. This is what enables Vue's Router to route to pages from this component (which is the base component). See [docs](https://router.vuejs.org/api/#router-view).
3. In the src folder, create a new folder called **layouts**.
4. Create a new file called **default.vue**.
5. Add a template tag in default.vue.
6. Copy the contents of App.vue's div with the id "app" to the default.vue. (Leave div id="app" in App.vue).
7. Copy the styles from App.vue to default.vue. (**TIP**: in VS Code, use **SHIFT + ALT + F** to format the file.)
8. The contents of your App.vue should look like:

<template>

  <div id="app">

    <router-view/>

  </div>

</template>

1. The contents of /layouts/default.vue should look like the following:

<template>

  <div id="nav">

    <router-link to="/">Home</router-link> |

    <router-link to="/about">About</router-link>

  </div>

</template>

<style>

#app {

  font-family: 'Avenir', Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

}

#nav {

  padding: 30px;

}

#nav a {

  font-weight: bold;

  color: #2c3e50;

}

#nav a.router-link-exact-active {

  color: #42b983;

}

</style>

1. In order for Vue to render something in the default layout when routing to a component using that layout, we need to add the <router-view /> tag. Add the <router-view /> tag after the closing div of div id="nav".
   1. Notice, if you get red squiggly lines under the router-view, it's because we now have two outer elements and Vue only allows one outer element. (This will be changing in Vue 3).
   2. Let's add a <div> wrapper around all the elements.
2. In **router.js**, let's update our existing routes to use the new default layout.
   1. import DefaultLayout from '@/layouts/default'
   2. Wrap the existing routes inside an array called "children".
   3. Add curly braces around the children array you just created.
   4. Add a path of '/'
   5. Add component: DefaultLayout
   6. In the Home route, change the path to be an empty string. This will tell Vue Router that this child should be the default route when the parent path is hit.
3. Now, let's run the app to see if works.
   1. npm run dev (or npm run serve if you didn't create the dev script)
4. Now we'll add a second layout. In your layouts folder, create a new file called "second.vue".
   1. Let's set up the second page layout to use named router views, like our ERP apps use. Named router views are a great way to include multiple sections on your page, like a navigation header and separate content area. Vue Router will cache the navigation and have only one instance of the navigation instead of copying from page to page.
      1. Create a template tag.
      2. Inside the template tag, add a div with the id="app".
      3. Inside the div, create two named router-views. One with the name "header" and the other with "content".
      4. Second.vue should look like:

<template>

  <div id="app">

    <router-view name="header" />

    <router-view name="content" />

  </div>

</template>

1. In the **components** folder, create a new **header.vue** file. Copy the contents below into the file:

<template>

  <nav>

    <ul>

      <li>

        <router-link to="/">Home</router-link>

      </li>

      <li>

        <router-link to="/about">About</router-link>

      </li>

      <li>

        <router-link to="/second-layout">Shout Out</router-link>

      </li>

    </ul>

  </nav>

</template>

<script>

export default {

  name: 'Header'

}

</script>

<style scoped>

ul, li {

  list-style: none;

}

</style>

1. In the components folder, create a new file called **shout-out.vue**.
   1. Add a template tag.
   2. Add a div with the words Shout Out.
   3. Add a class to the div called "pink-background".
   4. Add a style section and add the new class pink-background with background-color: pink.
   5. Add a script section and add the component name "ShoutOut".
   6. Contents of shout-out.vue should look like:

<template>

  <div class="pink-background">

    Shout Out

  </div>

</template>

<script>

export default {

  name: 'ShoutOut'

}

</script>

<style scoped>

.pink-background {

  background-color: pink;

}

</style>

1. Now, let's update the router with our new page layout and shout-out route which will use the header component.
   1. Add import statements for SecondLayout, Header, and ShoutOut

import SecondLayout from '@/layouts/second'

import Header from '@/components/header'

import ShoutOut from '@/components/shout-out'

1. Add new parent page layout default to the router with a path of '/shout-out'.
2. Add the component SecondLayout to the route.
3. Add children array.
4. Add the route for ShoutOut. The path should be an empty string since this will be the default page for the parent path.
   1. path: ''
   2. name: 'shoutout'
   3. To use the multiple named components, instead of using a component node, we have to use the word components. Why? Dunno, but that's how Vue Router works.
      1. components: { }
      2. header: Header
      3. content: ShoutOut
5. The new rout should look like this:

{

      path: '/shout-out',

      component: SecondLayout,

      children: [

        {

          path: '',

          name: 'shoutout',

          components: {

            header: Header,

            content: ShoutOut

          }

        }

      ]

    }

1. To make it easier for us to test everything, let's add our shout-out route to the default layout nav.

<router-link to="/shout-out">Shout Out</router-link>

1. Now, let's run our app and test our new page layout.
2. If we navigate to About, we should see the same layout (navigation) as the Home page. If we navigate to Shout Out we should see our navigation styled differently and the text "Shout Out" should have a pink background.
3. Let's double check everything using Vue Dev tools.
4. Open Chrome dev tools and click on the Vue Tab. On the ShoutOut page, expand the App node.
   1. We should see a node called "Second". That's our second page layout. Expand that.
   2. We should see two nodes, one called Header and one called ShoutOut.
      1. If we click on the Header node, and we inspect the data $route, then click on ShoutOut, the routes should look identical. This is because we are on the ShoutOut page, but using two named components from the Second page layout.
5. Let's navigate back to the Home page and look at Vue Dev Tools.