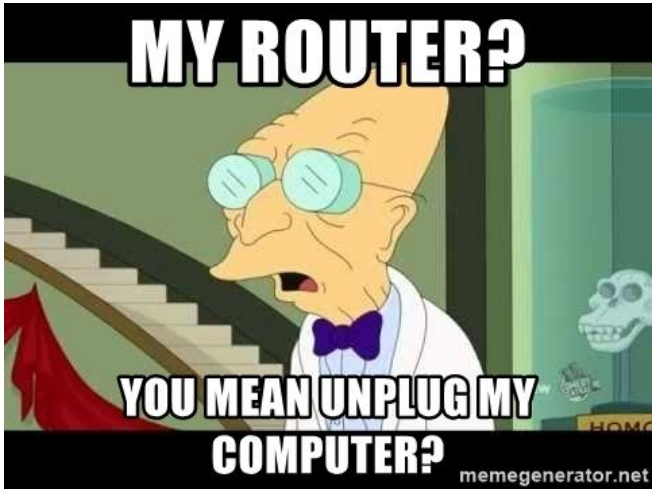
I THINK WE'RE GOING TO HAVE TO COMPLETELY REPLACE YOUR COMPUTER
IT ISN'T PUTTING OFF NEARLY ENOUGH HEAT FOR ME TO SLEEP ON IT

JUST PROGRAMMING THE NEW ROUTER


MY ROUTER?
YOU MEAN UNPLUG MY COMPUTER?
memegenerator.net

Vue Router

The official router for Vue.js.

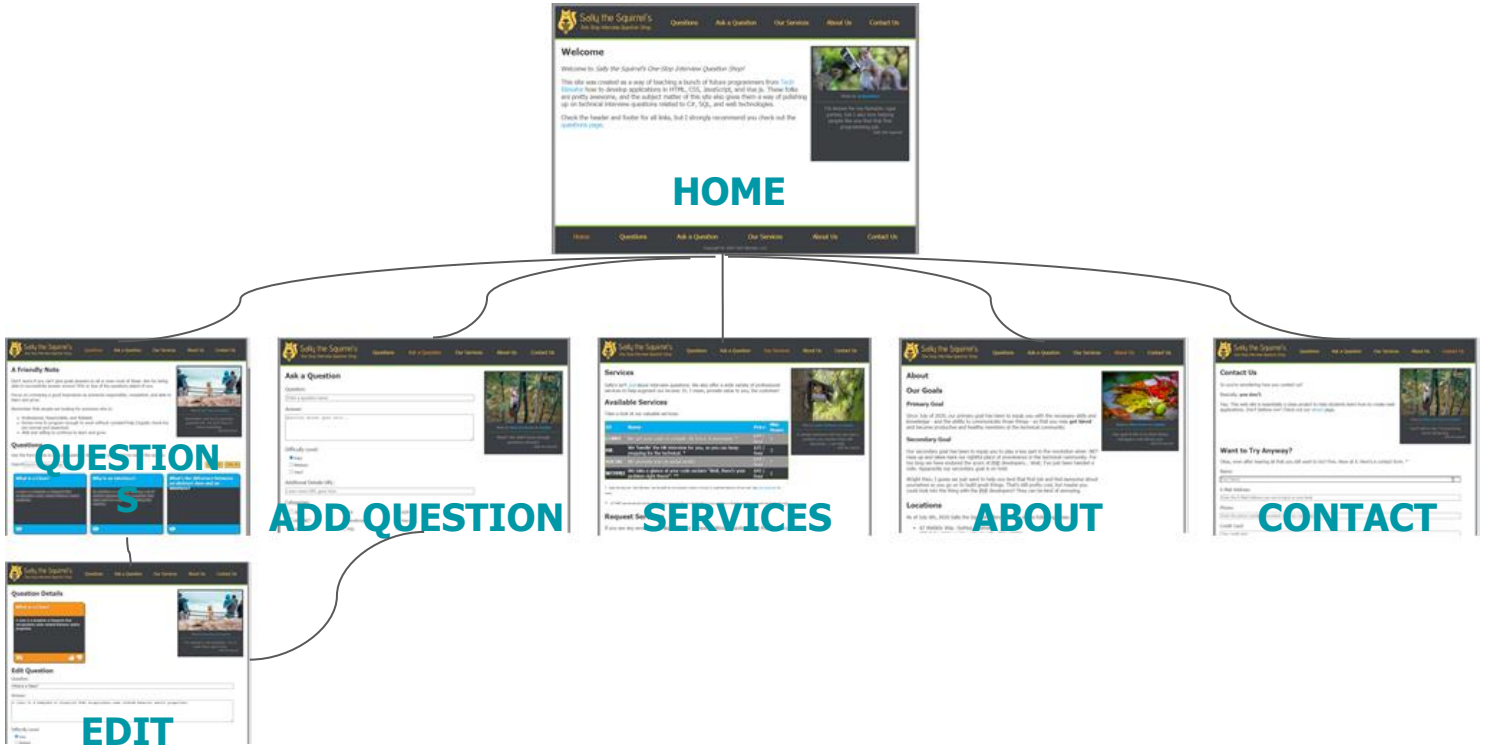
Module 3-14

VUE Routing

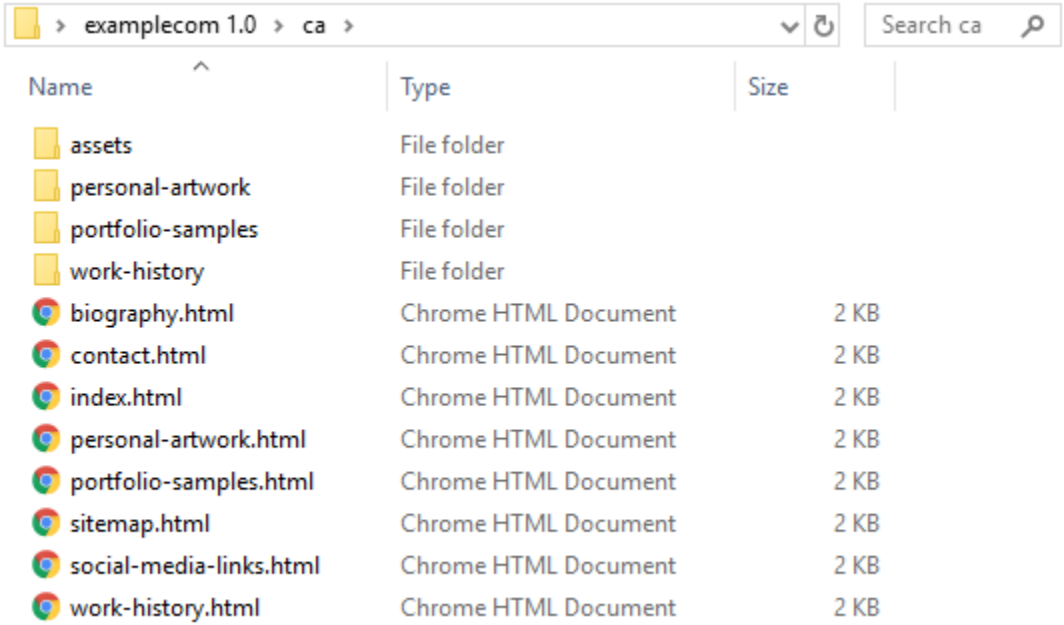
Objectives

- Why we need a “router” in a single page application (SPA)
- Define a new route
- Differences and uses of views and components directories
- Utilize router-link component
- Utilize router-view component
- Define dynamic routes
- this.$route object to access data from router
- this.$router object for programmatic navigation
- Lifecycle events

Web Navigation

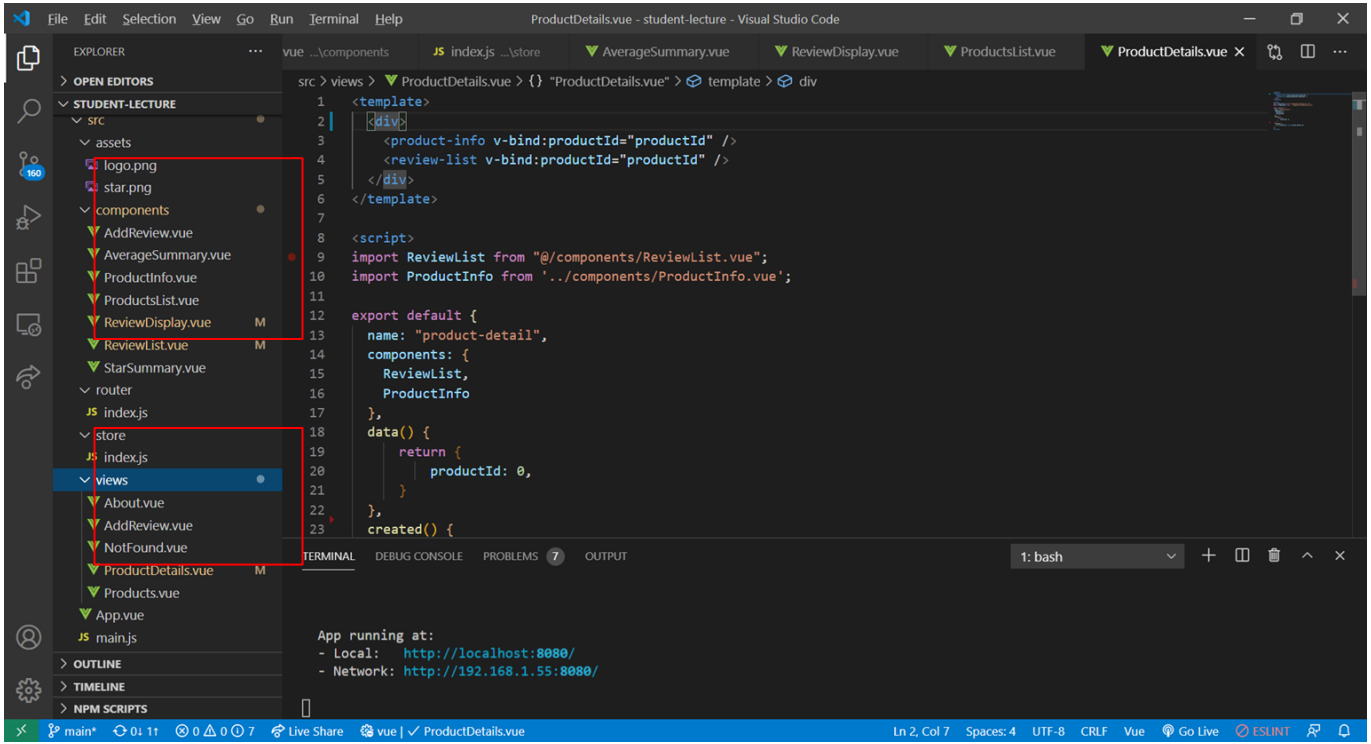


Web Navigation



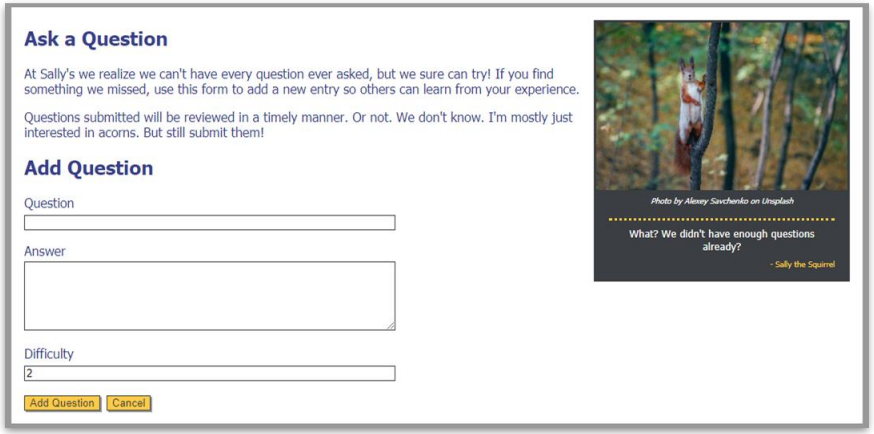
Vue.js Navigation

Vue.js Navigation

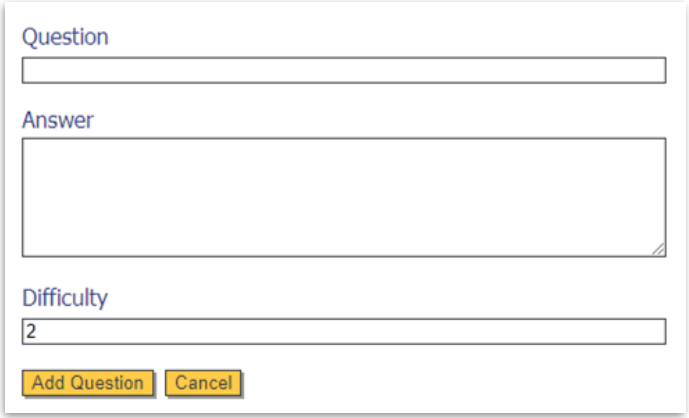


View vs. Components

VIEWS

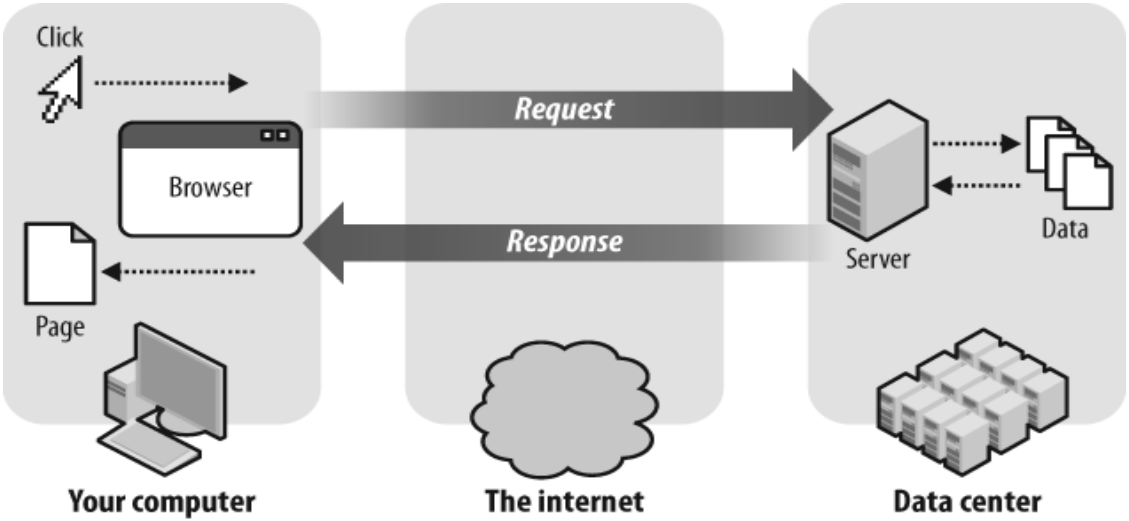


COMPONENTS



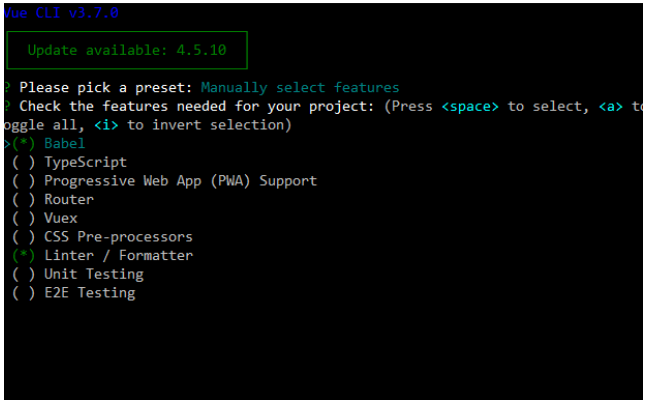
What is Routing?

- Routing allows users to be redirected to a certain component via a URL.
- Remember MVC Spring RequestMappings? Similar idea.

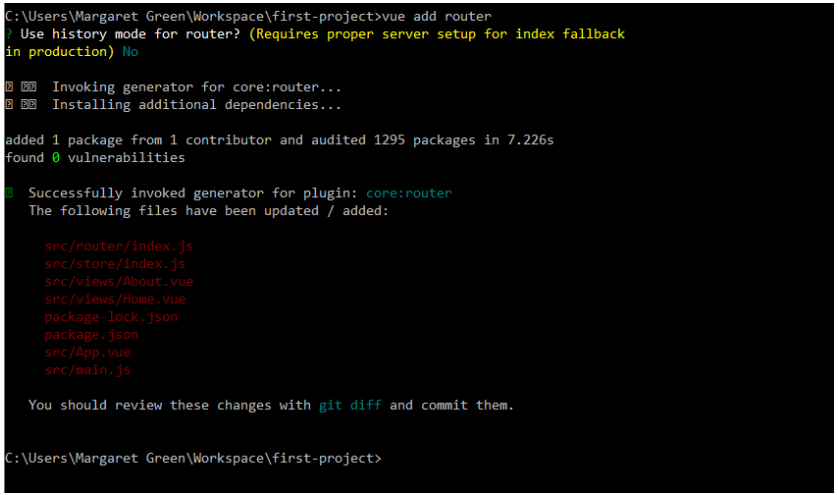


Installing Vuex Router

- During project creation, add Router.



- Add router to already created project



The src/router/index.js file overview

To define a route, an index.js file is needed.

- There is some boiler plate code beyond the scope of this class, you can carry those over for now, they are highlighted in blue.
- Our focus will be on the sections in red, which we will define.

```
import Vue from 'vue'
import Router from 'vue-router'
import Home from './views/Home.vue'
import About from './views/About.vue'
import NotFound from './views/NotFound.vue'

Vue.use(Router)
const routes = [
  {
    path: '/',          // Required
    name: 'home',       // Recommended, but not required
    component: Home   // Required
  },
  {
    path: /about',
    name: 'about',
    component: About
  },
  {
    path: '*',
    name: 'notFound',
    component: NotFound
  }
]
const router = new VueRouter({
  routes
})
export default router
```

The index.js file in router folder: importing views

We need to first define the components a user can potentially be routed to, this can be achieved through imports:

```
import Home from '../views/Home.vue'
import About from '../views/About.vue'
import Books from '@/views/Books.vue'
```

Note that in this example the components are in a folder called views, this should make no difference, Home, About, and Books are VUE components.

The router/index.js file: importing components.

Next, we define the routes, these physically map the view components we imported.

```
import Home from './views/Home.vue'
import About from './views/About.vue'
import Books from '@/views/Books.vue'
```

```
routes: [
  {
    path: '/',
    name: 'home',
    component: Home
  },
  {
    path: '/about',
    name: 'about',
    component: About
  },
  {
    path: '/books',
    name: 'books',
    component: Books
  }
]
```

Each route is a JSON object, comprised of three key value pairs:

- **path**: what the user types into the URL
- **name**: This is how we will refer to the route within App.vue
- **component**: The component that was imported in, that the user will be redirected to

Adding router links to App.vue

We are almost done! The last step is want to define the routes within App.vue.

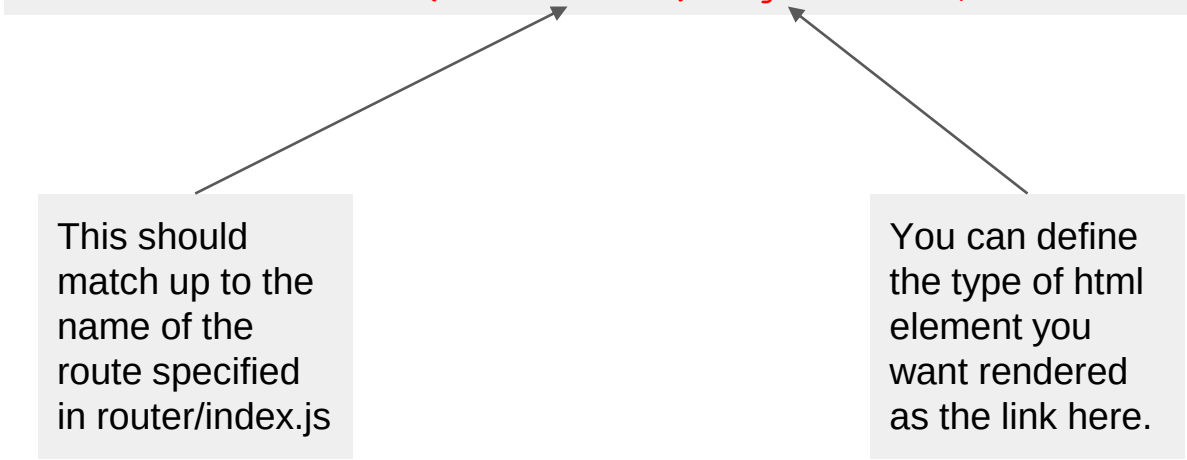
```
<template>
  <div id="app">
    <header>
      <ul class="nav">
        <router-link v-bind:to="{ name: 'home' }" tag="li">Home</router-link>
        <router-link to="/about" tag="li">About the Author</router-link>
        <router-link :to="{ name: 'books' }" tag="li">Related Books</router-link>
      </ul>
    </header>
    <router-view class="content"/>
  </div>
</template>
```

First we will talk about the red code – router-link, and the we will discuss later the blue code.

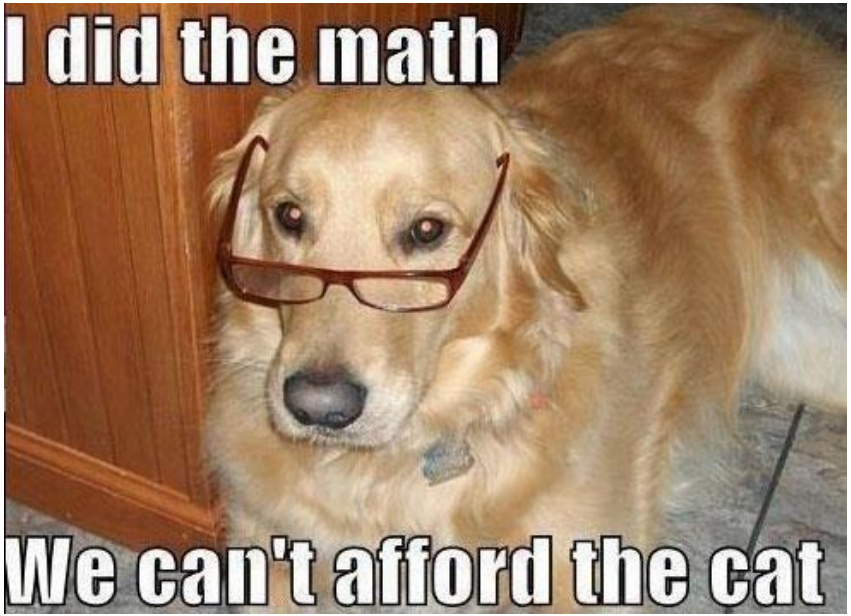
Adding the router links to App.vue

This is the basic structure of a router-link

```
<router-link v-bind:to="{ name: 'home' }" tag="li">Home</router-link>
```



Let’s Implement These Routes!



Dynamic Routing

- Sometimes data is encapsulated within a URL path:
    - Consider the following URL: (account/**135**)
    - The value 135 could very well be an ID that is associated with a “row” of data.

- The purpose of dynamic routing is to implement these URL parameters.

- Remember MVC Spring PathParams? Similar idea.

Dynamic Routing : router/index.js

We must first implement a route , where the path parameter contains a placeholder for the path variable.

```
{
  path: '/users/:id',
  name: 'user',
  component: User
}
```

Dynamic Routing : Defining the Router Links

We can now define router links

```
<tr v-for="user in users" :key="user.id">
      <td><router-link :to="{name: 'users', params: {id: user.id}}">{{user.id}}</router-
link></td>
      <td>{{user.name}}</td>
</tr>
```

Template section

```
data() {
  return {
    users: [
      {
        "id": 1,
        "name": "Leanne Graham"
      },
      {
        "id":2,
        "name": "Ervin Howell"
      }
    ]
  }
```

Script section

Here we have a v-for that will iterate through every object in the users array, each time it does so it generates a new router-link with its respective id value.

Two links are generated:

- /users/1
- /users/2

Router-View

```
1 <template>
2   <div id="app">
3     <AppHeader />
4     <router-view /> <!-- The currently active view will be presented here -->
5     <app-footer />
6   </div>
7 </template>
```

Let’s Implement Dynamic Routes!

Router-link with parameters

```
1 <router-link v-bind:to="{name:'product-details', params: {id: product.id}}">
2   View Product Details
3 </router-link>
```

This $route

```
created() {
    const id = this.$route.params.id; // Grabs the route parameter named id, if it was present
    this.question = this.$store.state.questions.find(q => q.id === id);

    if (!this.question) {
        this.$router.push({name: 'NotFound'});
    }
}
```

Routing

SomeComponent.vue

```vue
<router-link v-bind:to="{name:'product-details', params: {id: product.id}}">
  View Product Details
</router-link>
```

router/index.js

```js
const routes = [
  {
    path: '/',
    name: 'products',
    component: Products,
  },
  {
    path: '/products/:id',
    name: 'product-details',
    component: ProductDetails,
  },
  {
    path: '*',
    name: 'not-found',
    component: NotFound,
  }
]
```

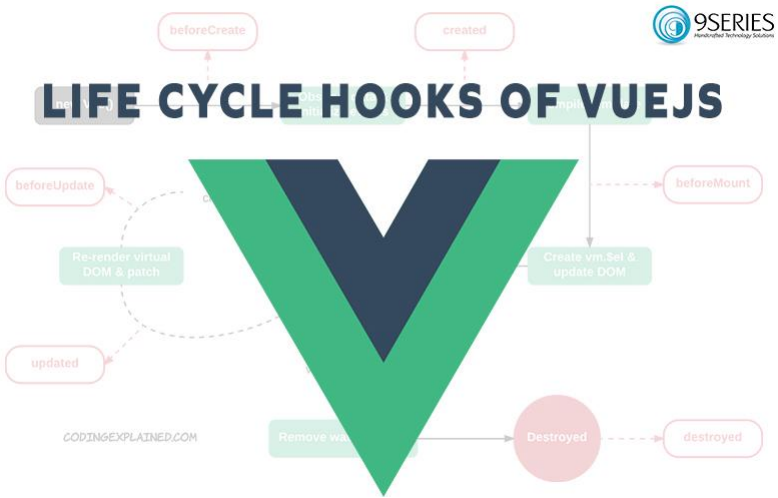
ProductDetails.vue

```vue
<script>
export default {
  name: "product-detail",
  data() {
      return {
          productId: 0,
      }
  },
  created() {
    this.productId = this.$route.params.id;
  }
};
</script>
```

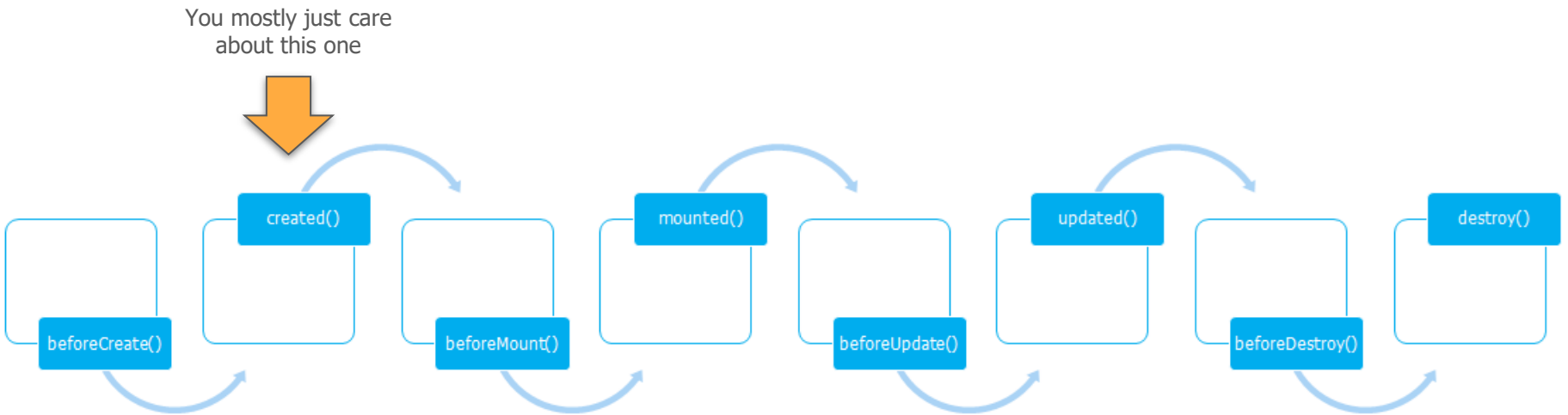
Adding reviews with this.$router

```
 6
 7 // named route
 8 router.push({ name: 'user', params: { userId: '123' } })
 9
10 // with query, resulting in /register?plan=private
11 router.push({ path: 'register', query: { plan: 'private' } })
```

Life cycle hooks



Life cycle events

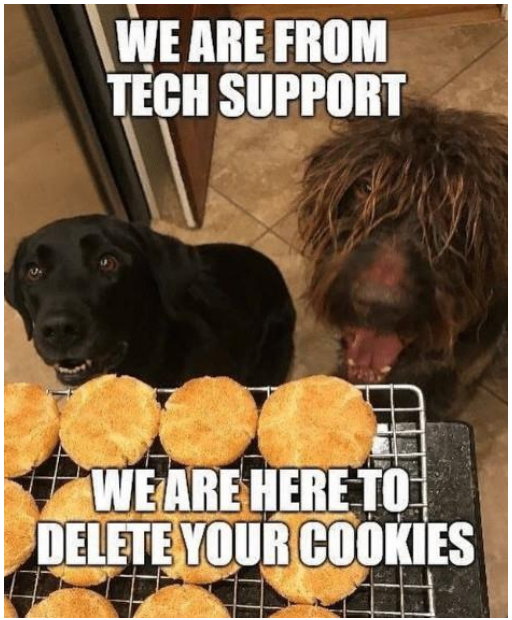


created()

```
// This fires after the component is created but before it renders.
// You can access any data, props, and $route info you need to here.
// This is also a good place to kick off requests for data your component will eventually need
created() {
    const id = this.$route.params.id; // Grabs the route parameter named id, if it was present
    this.question = this.$store.state.questions.find(q => q.id === id);

    if (!this.question) {
        this.$router.push({name: 'NotFound'});
    }
}
```

Let’s Code! But first – RECAP!

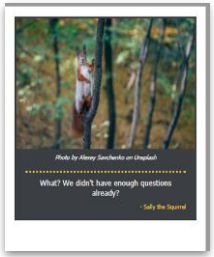


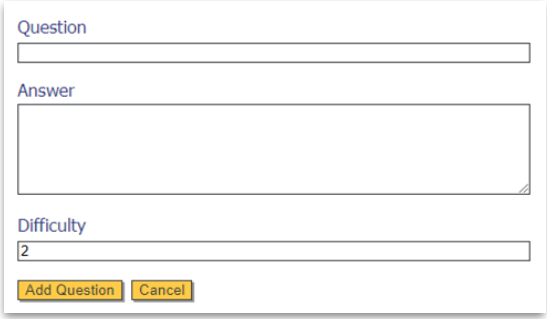
Objectives

- Differences and uses of views and components directories

View

Components





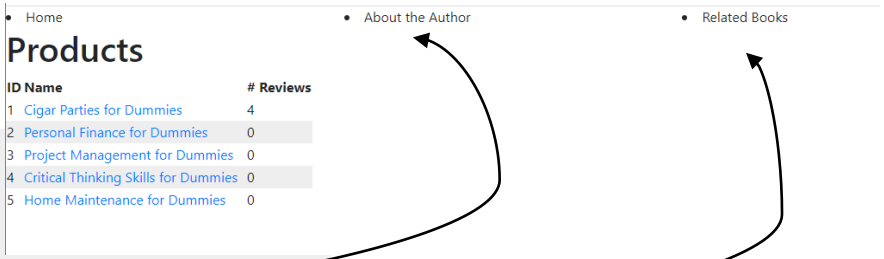
Objectives

- Differences and uses of views and components directories
- Utilize router-link component

```
<router-link v-bind:to="{ name: 'home' }" tag="li">Home</router-link>
```

Objectives

- Differences and uses of views and components directories
- Utilize router-link component
- Utilize router-view component




```
<template>
  <div id="app">
    <header>
      <ul class="nav">
        <router-link :to="{ name: 'home' }" tag="li">Home</router-link>
        <router-link :to="{ name: 'about' }" tag="li">About the Author</router-link>
        <router-link :to="{ name: 'books' }" tag="li">Related Books</router-link>
      </ul>
    </header>
    <router-view class="content"/>
  </div>
</template>
```

Where component will be displayed

Objectives

- Differences and uses of views and components directories
- Utilize router-link component
- Utilize router-view component
- Define dynamic routes

```
{
  path: '/users/:id',
  name: 'user',
  component: User
}
```

```
<tr v-for="user in users" :key="user.id">
        <td><router-link :to="{name: 'users', params: {id: user.id}}">{{user.id}}</router-
link></td>
        <td>{{user.name}}</td>
</tr>
```

Objectives

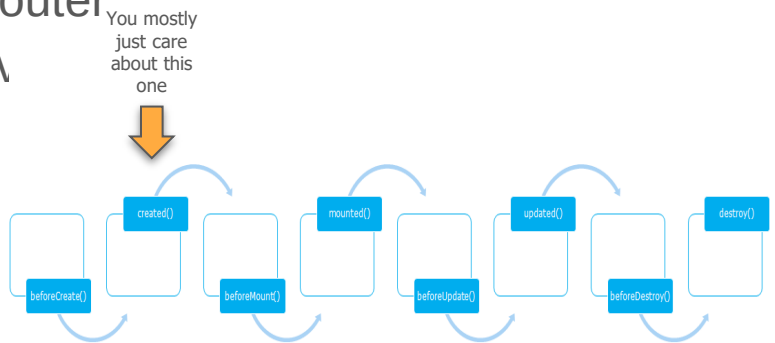
- Differences and uses of views and components directories
- Utilize router-link component
- Utilize router-view component
- Define dynamic routes
- this.$route object to access data from router

```
<script>
export default {
  name: "product-detail",
  data() {
      return {
          productId: 0,
      }
  },
  created() {
    this.productId = this.$route.params.id;
  }
};
</script>
```

Objectives

- Differences and uses of views and components directories
- Utilize router-link component
- Utilize router-view component
- Define dynamic routes
- this.$route object to access data from router
- this.$router object for programmatic navigation

```
 8     if (!this.question) {
 9         this.$router.push({name: 'NotFound'});
10     }
11 }
```

Objectives

- Differences and uses of views and components directories
- Utilize router-link component
- Utilize router-view component
- Define dynamic routes
- this.$route object to access data from router
- this.$router object for programmatic nav
- Lifecycle events