Vue Workshop (2/2)

Workshop goal

Workshop goal

- Part 1
  - Migrate from Vue 2 to Vue 3
  - Learn about the Vue Migration build
  - Upgrade the Vue Router
  - Create reusable components
  - Learn about the composition API
  - Create composition functions
- Part 2
  - Use components Slots
  - Manage state using VueX
  - Explore the Vue Router
  - Use Vuetify components
  - Type check the code using TypeScript
  - Server side rendering of Vue applications using Nuxt

- Maurice de Beijer
- The Problem Solver
- Microsoft MVP
- Freelance lead/developer/instructor
- Twitter: @mauricedb
- Web: http://www.TheProblemSolver.nl
- E-mail: maurice.de.beijer@gmail.com

Type it out by hand?

"Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now!"

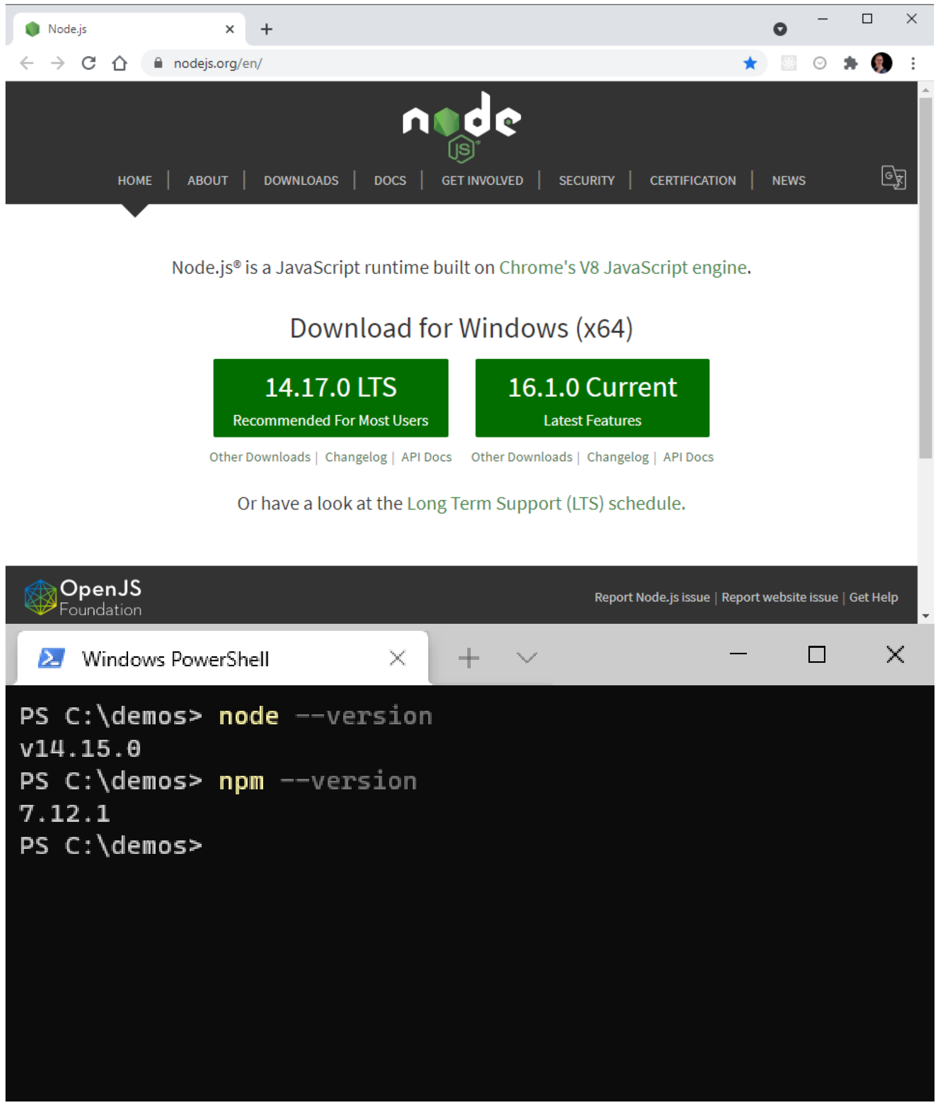
Prerequisites

Install Node, NPM & Vue CLI

Install the GitHub repository

Install Node.js & NPM

- Minimal:
  - Node version 14
  - NPM version 7


Node.js
nodejs.org/en/
HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for Windows (x64)

14.17.0 LTS
Recommended For Most Users

16.1.0 Current
Latest Features

Other Downloads | Changelog | API Docs

Or have a look at the Long Term Support (LTS) schedule.

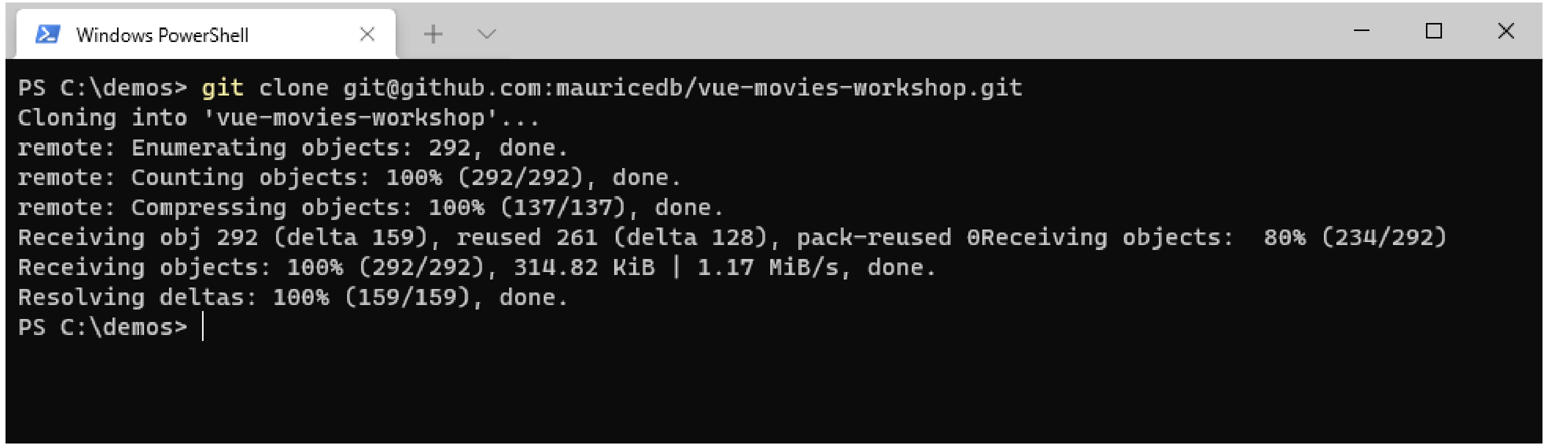
OpenJS Foundation
Report Node.js issue | Report website issue | Get Help
Windows PowerShell

```
PS C:\demos> node --version
v14.15.0
PS C:\demos> npm --version
7.12.1
PS C:\demos>
```

Vue CLI

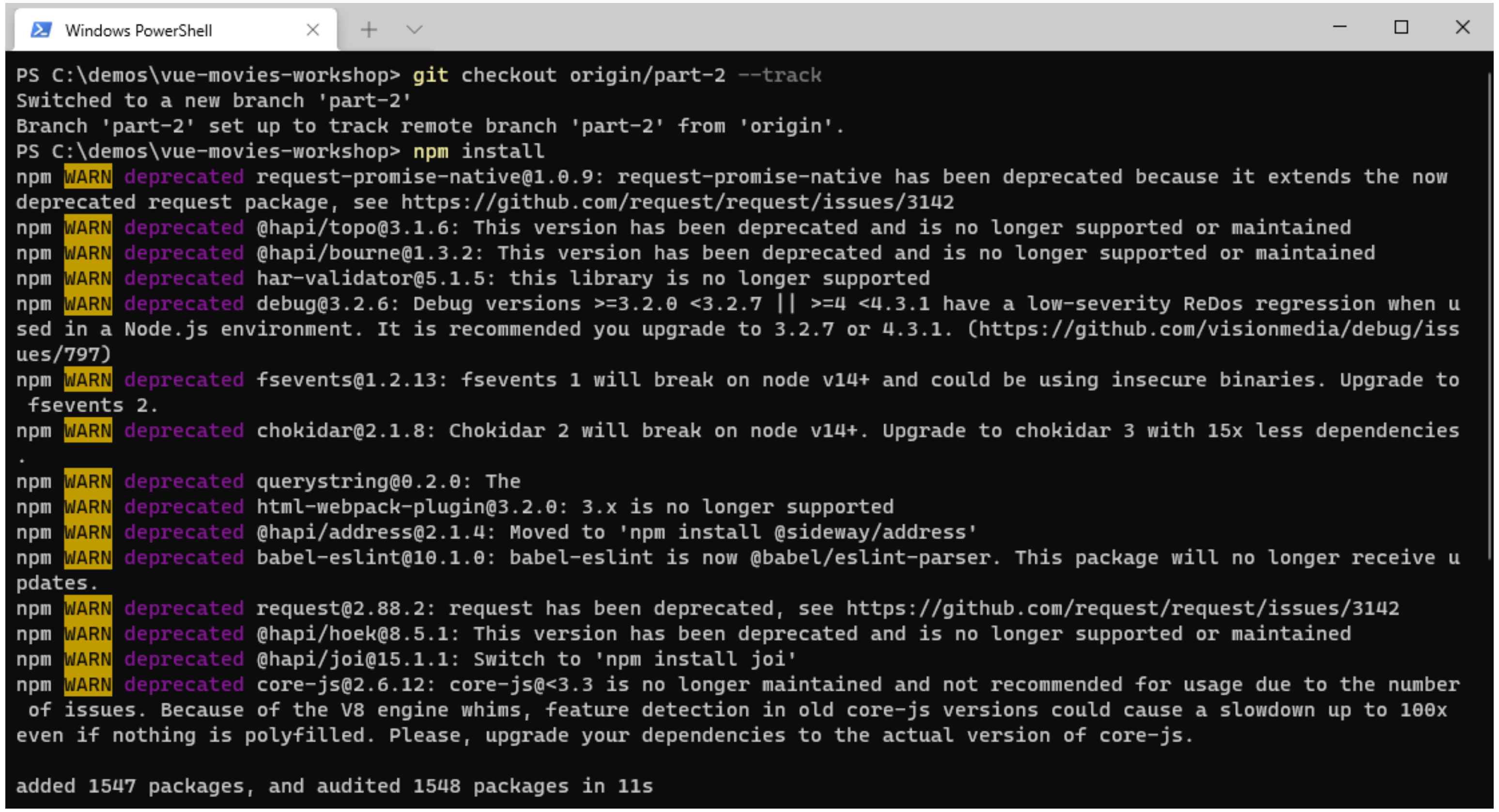
- Install the **Vue CLI** using NPM
  - `npm install -g @vue/cli`

Clone the GitHub Repository



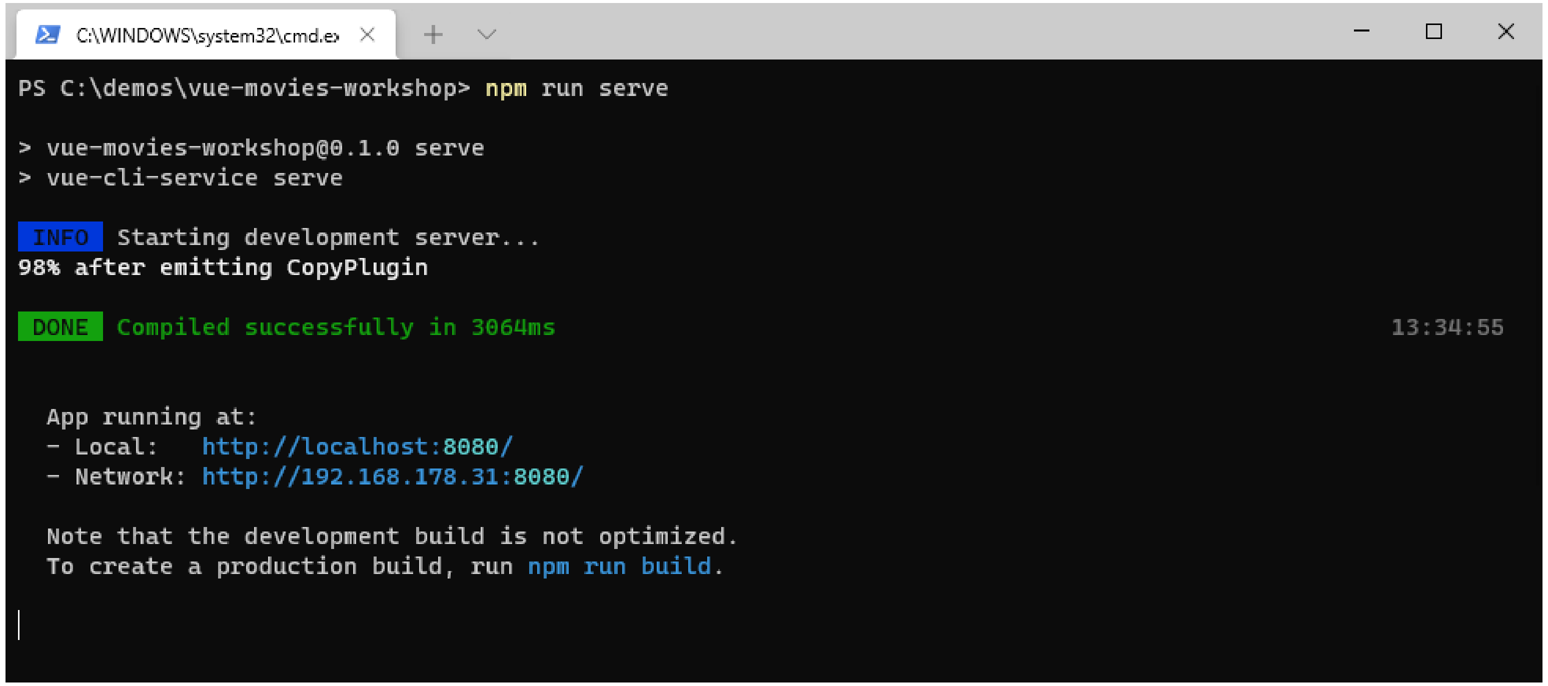
```
PS C:\demos> git clone git@github.com:mauricedb/vue-movies-workshop.git
Cloning into 'vue-movies-workshop'...
remote: Enumerating objects: 292, done.
remote: Counting objects: 100% (292/292), done.
remote: Compressing objects: 100% (137/137), done.
Receiving obj 292 (delta 159), reused 261 (delta 128), pack-reused 0Receiving objects:  80% (234/292)
Receiving objects: 100% (292/292), 314.82 KiB | 1.17 MiB/s, done.
Resolving deltas: 100% (159/159), done.
PS C:\demos>
```

Checkout branch & install NPM packages



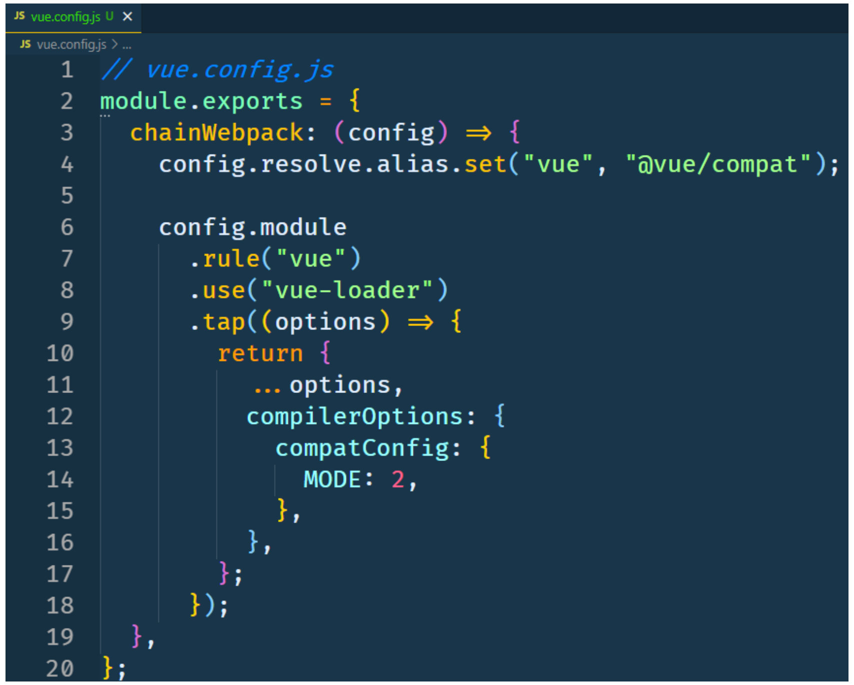
```
PS C:\demos\vue-movies-workshop> git checkout origin/part-2 --track
Switched to a new branch 'part-2'
Branch 'part-2' set up to track remote branch 'part-2' from 'origin'.
PS C:\demos\vue-movies-workshop> npm install
npm WARN deprecated request-promise-native@1.0.9: request-promise-native has been deprecated because it extends the now
deprecated request package, see https://github.com/request/request/issues/3142
npm WARN deprecated @hapi/topo@3.1.6: This version has been deprecated and is no longer supported or maintained
npm WARN deprecated @hapi/bourne@1.3.2: This version has been deprecated and is no longer supported or maintained
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated debug@3.2.6: Debug versions >=3.2.0 <3.2.7 || >=4 <4.3.1 have a low-severity ReDos regression when u
sed in a Node.js environment. It is recommended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/debug/iss
ues/797)
npm WARN deprecated fsevents@1.2.13: fsevents 1 will break on node v14+ and could be using insecure binaries. Upgrade to
 fsevents 2.
npm WARN deprecated chokidar@2.1.8: Chokidar 2 will break on node v14+. Upgrade to chokidar 3 with 15x less dependencies
.
npm WARN deprecated querystring@0.2.0: The
npm WARN deprecated html-webpack-plugin@3.2.0: 3.x is no longer supported
npm WARN deprecated @hapi/address@2.1.4: Moved to 'npm install @sideway/address'
npm WARN deprecated babel-eslint@10.1.0: babel-eslint is now @babel/eslint-parser. This package will no longer receive u
pdates.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated @hapi/hoek@8.5.1: This version has been deprecated and is no longer supported or maintained
npm WARN deprecated @hapi/joi@15.1.1: Switch to 'npm install joi'
npm WARN deprecated core-js@2.6.12: core-js@<3.3 is no longer maintained and not recommended for usage due to the number
 of issues. Because of the V8 engine whims, feature detection in old core-js versions could cause a slowdown up to 100x
even if nothing is polyfilled. Please, upgrade your dependencies to the actual version of core-js.

added 1547 packages, and audited 1548 packages in 11s
```

Start the application



```
PS C:\demos\vue-movies-workshop> npm run serve

> vue-movies-workshop@0.1.0 serve
> vue-cli-service serve

 INFO  Starting development server...
98% after emitting CopyPlugin

 DONE  Compiled successfully in 3064ms                                                    13:34:55


  App running at:
  - Local:   http://localhost:8080/
  - Network: http://192.168.178.31:8080/

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

Following Along


vue.config.js U ×
vue.config.js > ...

```js
// vue.config.js
module.exports = {
  chainWebpack: (config) => {
    config.resolve.alias.set("vue", "@vue/compat");

    config.module
      .rule("vue")
      .use("vue-loader")
      .tap((options) => {
        return {
          ...options,
          compilerOptions: {
            compatConfig: {
              MODE: 2,
            },
          },
        };
      });
  },
};
```

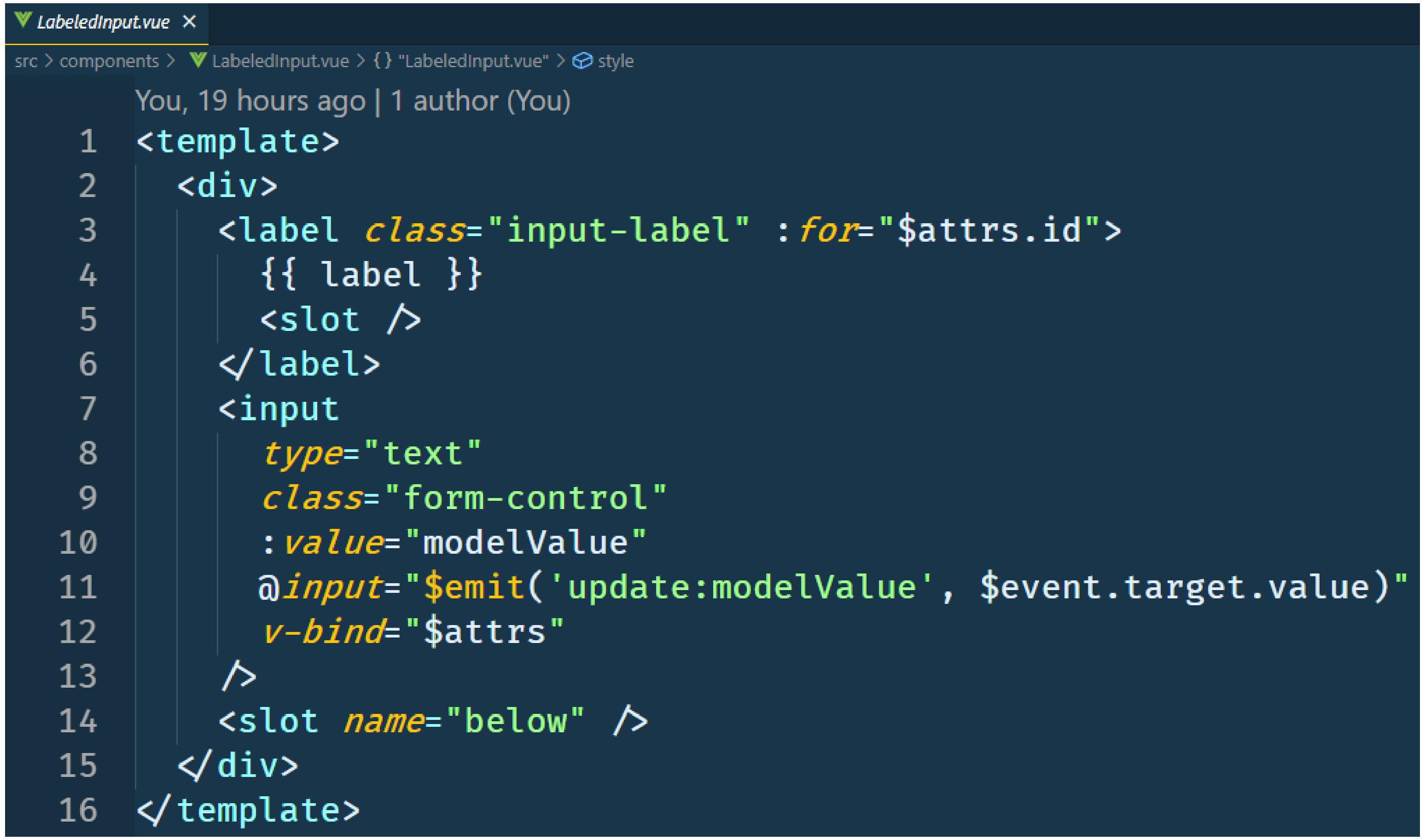
- Repository: https://github.com/mauricedb/vue-movies-workshop
- Slides: http://theproblemsolver.nl/vue-workshop-part-2.pdf

Slots

Slots

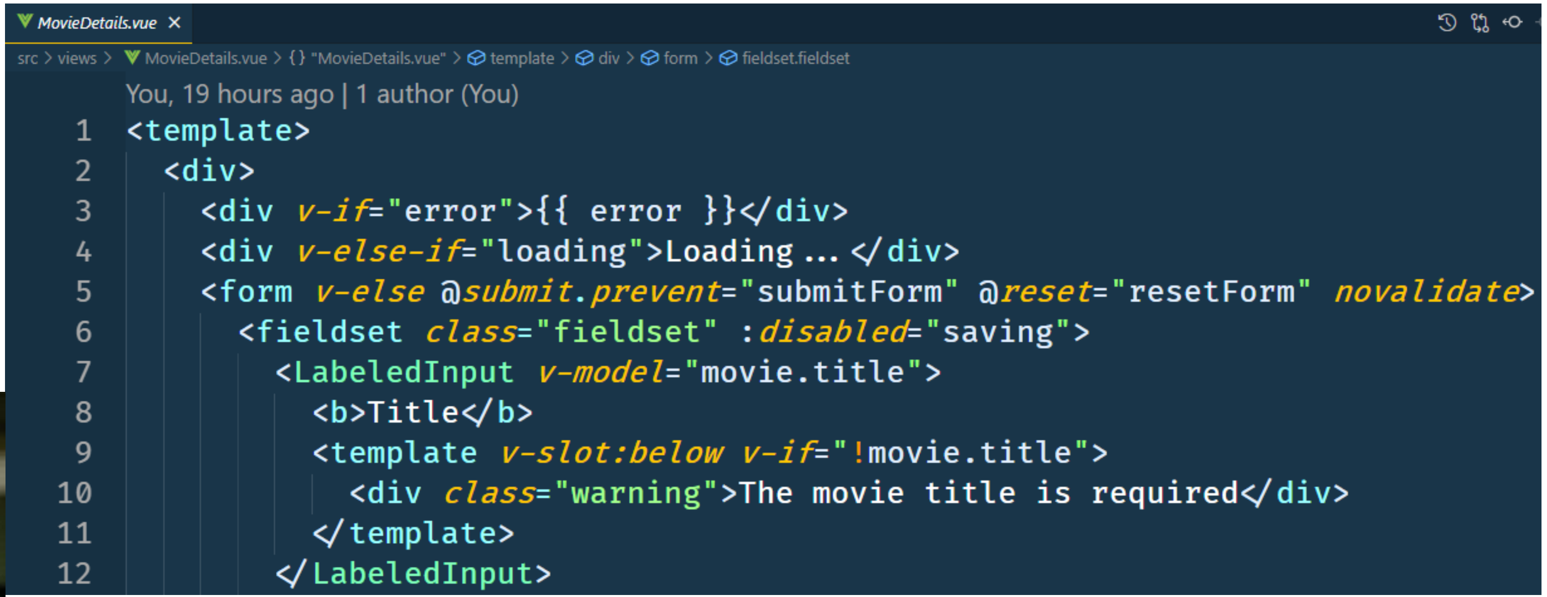
- Slots can be used to **render component content**
  - Content can be markup and/or other Vue components
- Multiple **named slots** can be used if required
  - An unnamed slot is the default slot
  - Use a `<template v-slot:[name]>` to target a named slot

LabeledInput.vue


LabeledInput.vue ×
src > components > LabeledInput.vue > { } "LabeledInput.vue" > style
You, 19 hours ago | 1 author (You)

```
<template>
  <div>
    <label class="input-label" :for="$attrs.id">
      {{ label }}
      <slot />
    </label>
    <input
      type="text"
      class="form-control"
      :value="modelValue"
      @input="$emit('update:modelValue', $event.target.value)"
      v-bind="$attrs"
    />
    <slot name="below" />
  </div>
</template>
```

MovieDetails.vue


MovieDetails.vue
src > views > MovieDetails.vue > {} "MovieDetails.vue" > template > div > form > fieldset.fieldset
You, 19 hours ago | 1 author (You)

```
<template>
  <div>
    <div v-if="error">{{ error }}</div>
    <div v-else-if="loading">Loading ... </div>
    <form v-else @submit.prevent="submitForm" @reset="resetForm" novalidate>
      <fieldset class="fieldset" :disabled="saving">
        <LabeledInput v-model="movie.title">
          <b>Title</b>
          <template v-slot:below v-if="!movie.title">
            <div class="warning">The movie title is required</div>
          </template>
        </LabeledInput>
```

VueX

State management

VueX

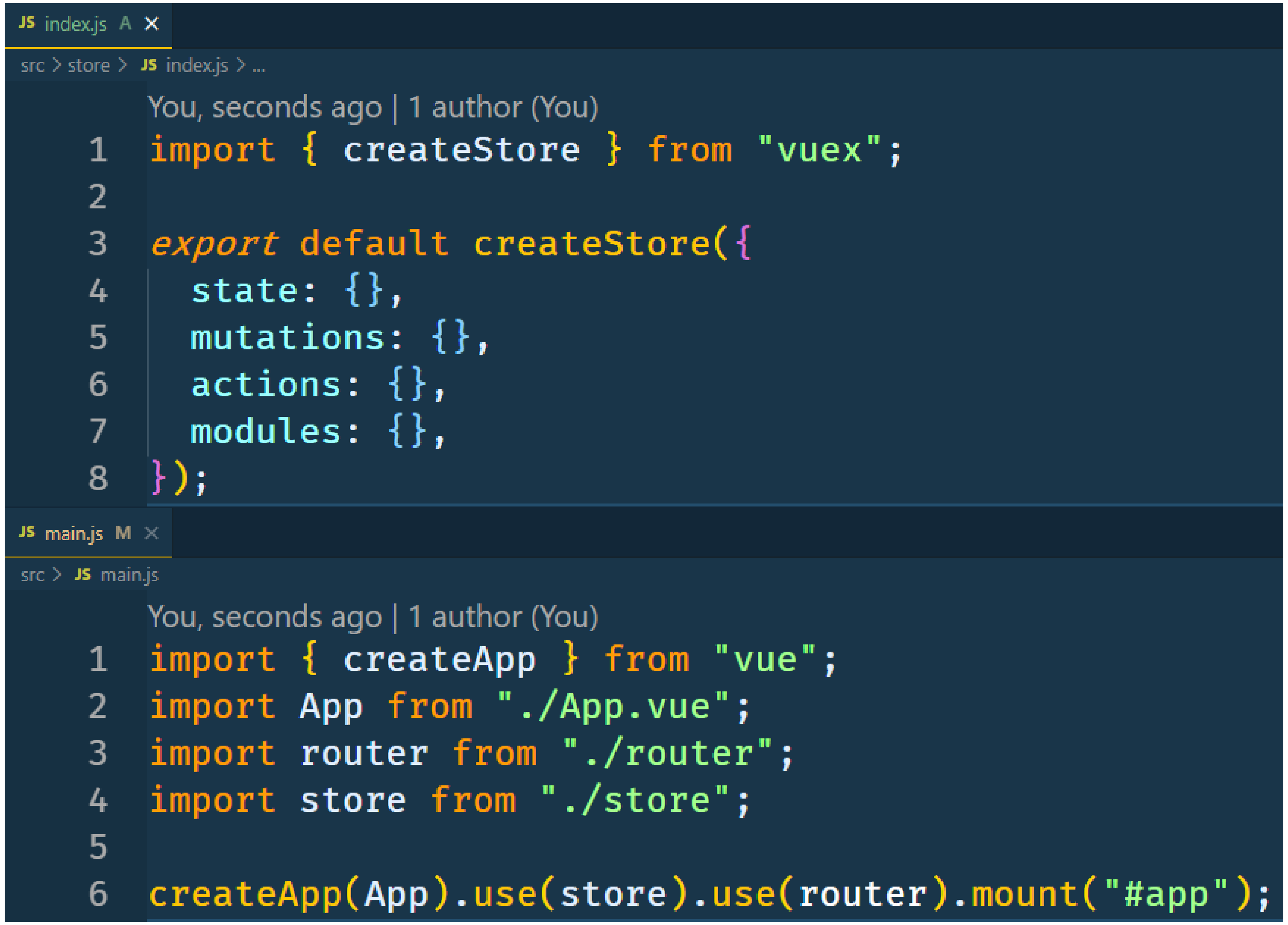
- **VueX** is a **state management** pattern + library
- Makes working with **shared state** more **predictable**
  - By externalizing state from components
- A store can split into **multiple store modules**
  - When the state becomes large
- Use **VueX version 4** with Vue 3
  - VueX 3 is for Vue 2 ☹

When to use?

- Using VueX makes sense in **larger applications**
  - With state that is used in multiple components
- **Small applications** don’t need VueX
  - Neither do components with local state

Installing VueX

- **VueX can be added** with: `vue add vuex`
  - Installs version 4.0.0-0
  - Use `npm install vuex@next` to update to the latest
- **Creates a starter VueX store**
  - And registers it with the Vue application


index.js A
src > store > index.js > ...
You, seconds ago | 1 author (You)

```javascript
import { createStore } from "vuex";

export default createStore({
  state: {},
  mutations: {},
  actions: {},
  modules: {},
});
```

main.js M
src > main.js
You, seconds ago | 1 author (You)

```javascript
import { createApp } from "vue";
import App from "./App.vue";
import router from "./router";
import store from "./store";

createApp(App).use(store).use(router).mount("#app");
```

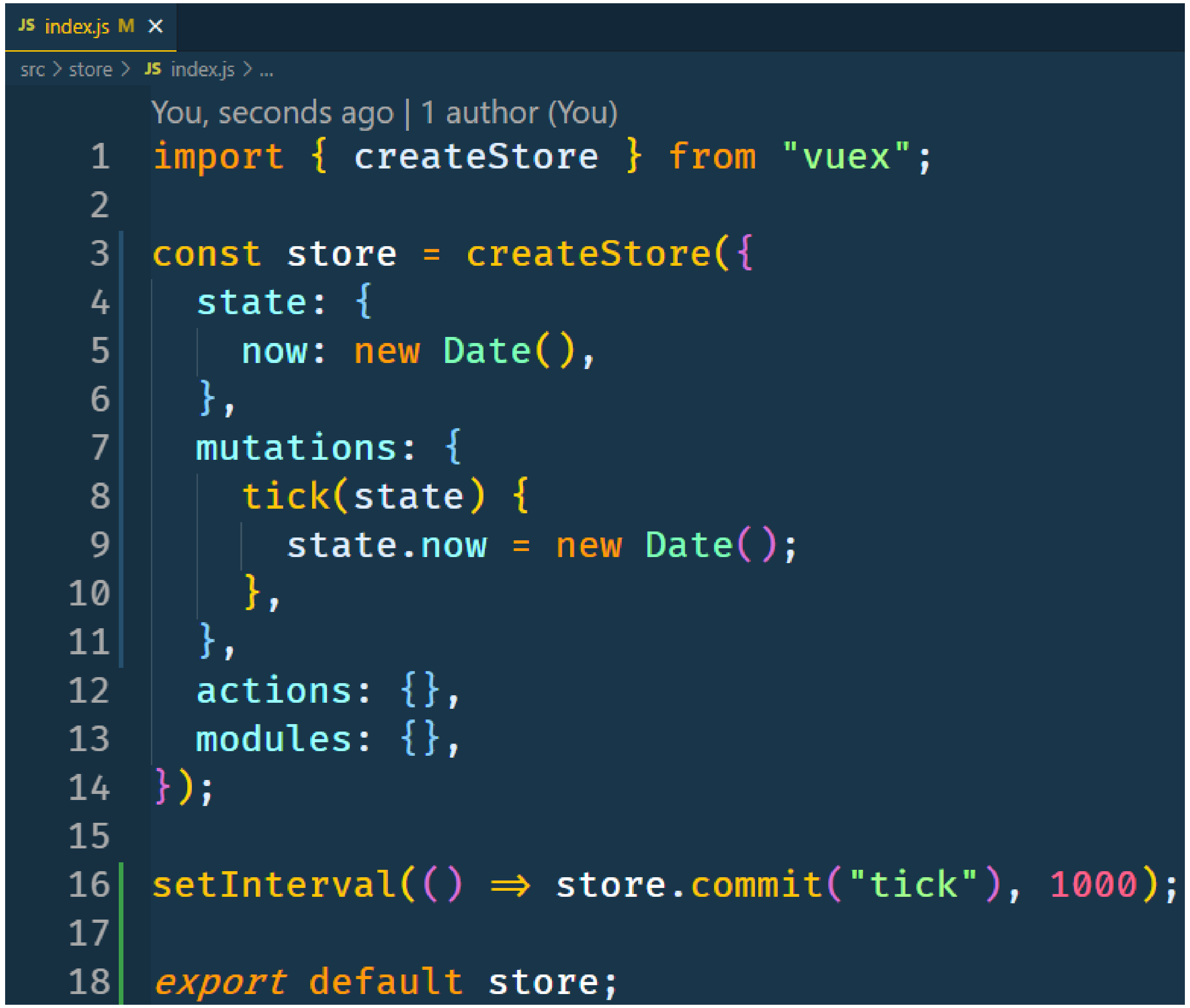
Vue Mutations

Synchronous updates

Mutations

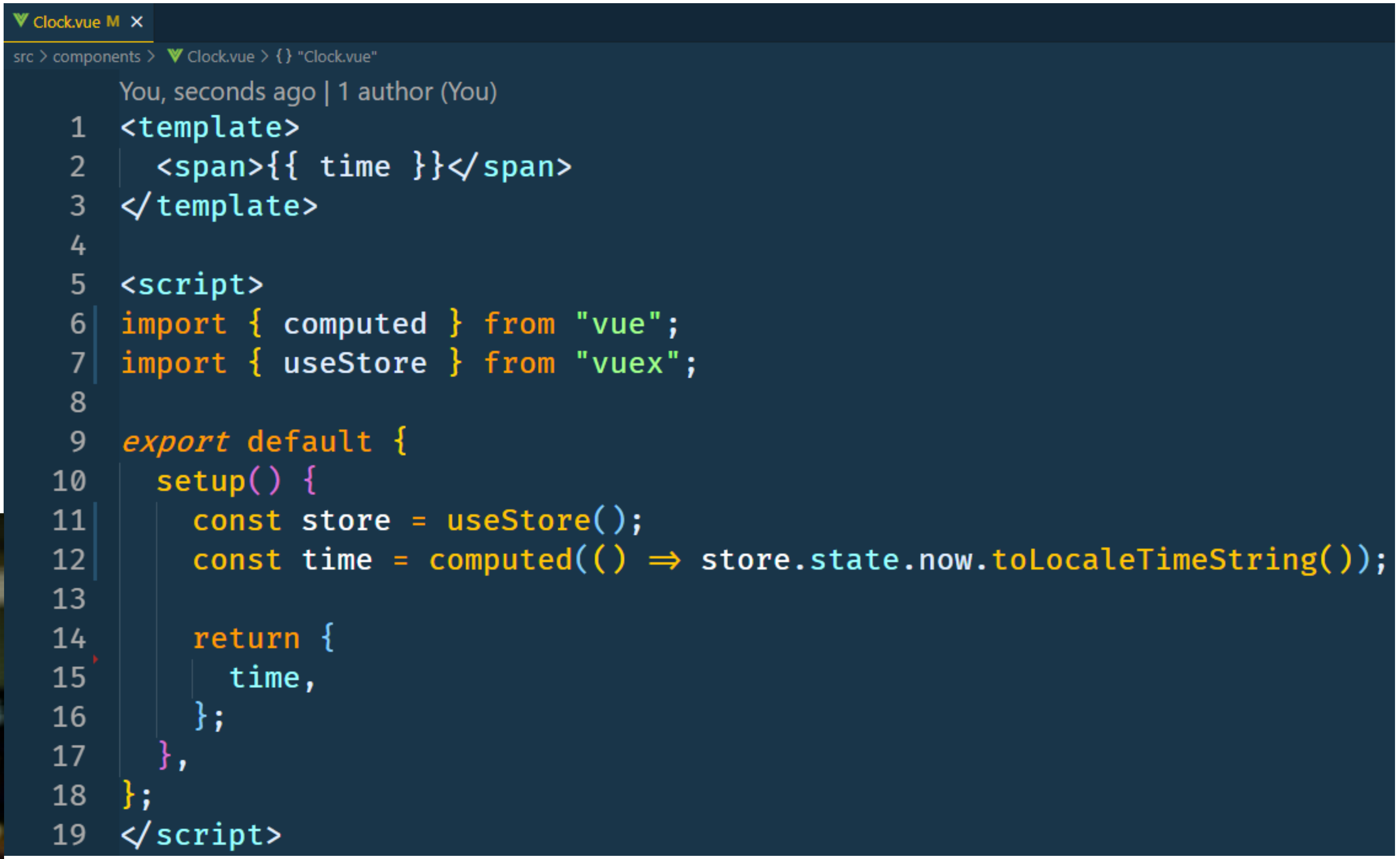
- **Mutations** are for **synchronous** store **updates**
  - Can be parameterized if needed
- Trigger the update when needed
  - Using `store.commit("[mutation name]", payload)`
- The **mutation function** receives
  - The current state as the first parameter
  - The optional payload as the second parameter

Store


JS index.js M ×
src > store > JS index.js > ...
You, seconds ago | 1 author (You)

```javascript
import { createStore } from "vuex";

const store = createStore({
  state: {
    now: new Date(),
  },
  mutations: {
    tick(state) {
      state.now = new Date();
    },
  },
  actions: {},
  modules: {},
});

setInterval(() ⇒ store.commit("tick"), 1000);

export default store;
```

Clock.vue


Clock.vue M ×
src > components > Clock.vue > { } "Clock.vue"
You, seconds ago | 1 author (You)

```
<template>
  <span>{{ time }}</span>
</template>

<script>
import { computed } from "vue";
import { useStore } from "vuex";

export default {
  setup() {
    const store = useStore();
    const time = computed(() => store.state.now.toLocaleTimeString());

    return {
      time,
    };
  },
};
</script>
```

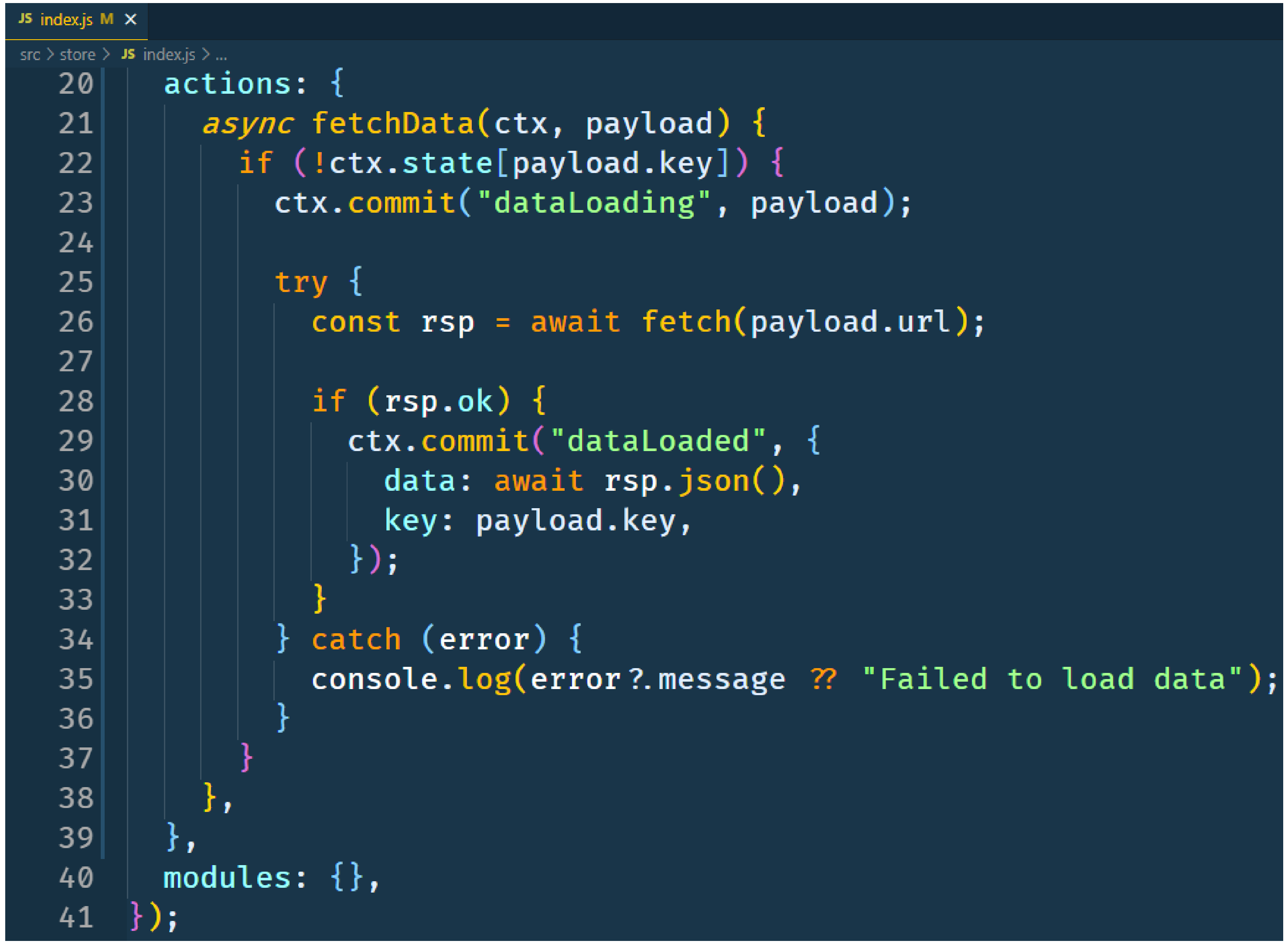
Vue Actions

Asynchronous updates

Actions

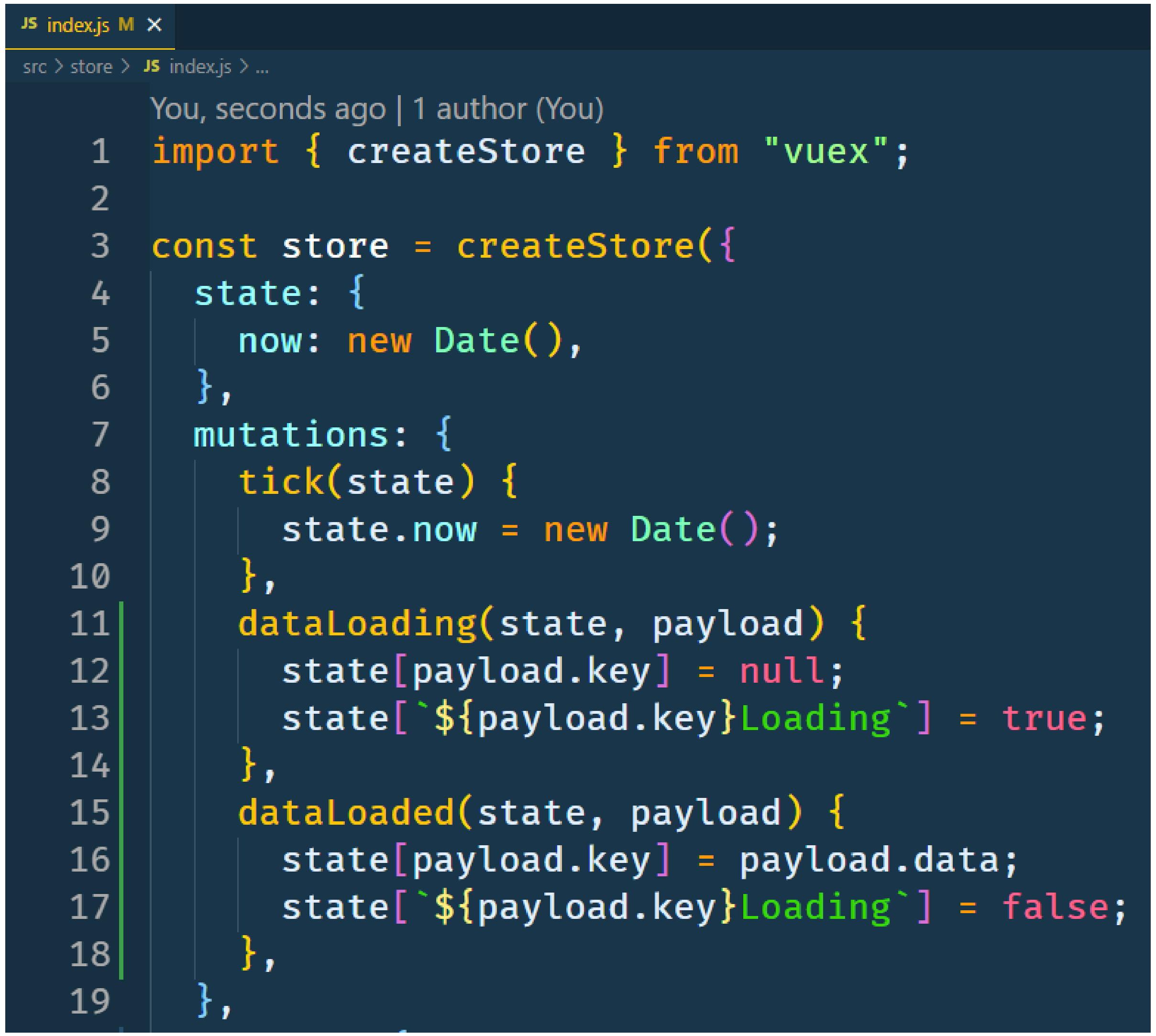
- Actions are for **asynchronous** store **updates**
  - Don’t mutate state directly but commit actions
- Receives a context object as parameter
  - state, dispatch(), commit() etc.

Store (1/2)



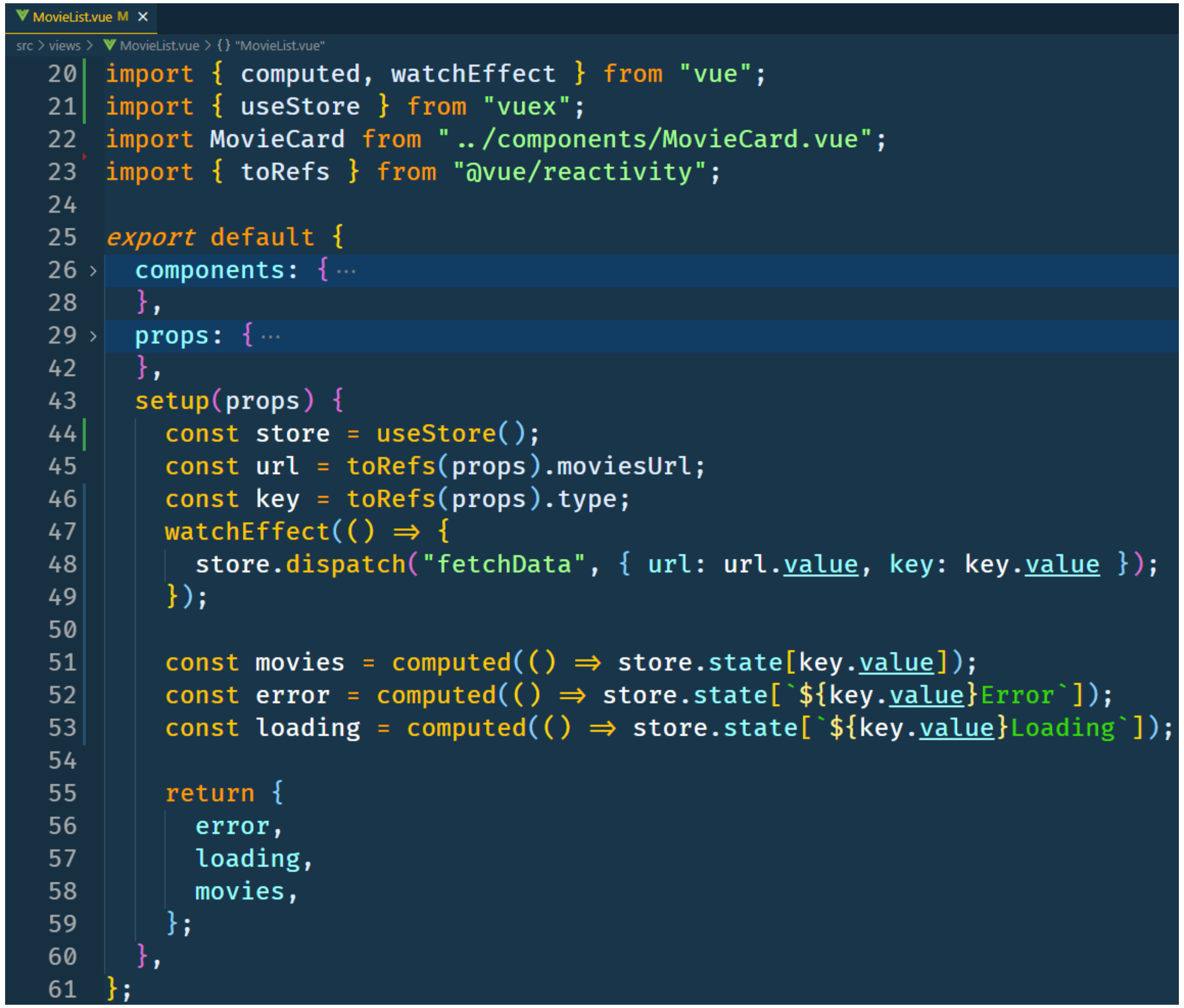
```
  actions: {
    async fetchData(ctx, payload) {
      if (!ctx.state[payload.key]) {
        ctx.commit("dataLoading", payload);

        try {
          const rsp = await fetch(payload.url);

          if (rsp.ok) {
            ctx.commit("dataLoaded", {
              data: await rsp.json(),
              key: payload.key,
            });
          }
        } catch (error) {
          console.log(error?.message ?? "Failed to load data");
        }
      }
    },
  },
  modules: {},
});
```

Store (2/2)


JS index.js M ×
src > store > JS index.js > ...
You, seconds ago | 1 author (You)

```javascript
import { createStore } from "vuex";

const store = createStore({
  state: {
    now: new Date(),
  },
  mutations: {
    tick(state) {
      state.now = new Date();
    },
    dataLoading(state, payload) {
      state[payload.key] = null;
      state[`${payload.key}Loading`] = true;
    },
    dataLoaded(state, payload) {
      state[payload.key] = payload.data;
      state[`${payload.key}Loading`] = false;
    },
  },
```

MovieList.vue


MovieList.vue M ×

src > views > MovieList.vue > {} "MovieList.vue"

```
import { computed, watchEffect } from "vue";
import { useStore } from "vuex";
import MovieCard from "../components/MovieCard.vue";
import { toRefs } from "@vue/reactivity";

export default {
  components: {…
  },
  props: {…
  },
  setup(props) {
    const store = useStore();
    const url = toRefs(props).moviesUrl;
    const key = toRefs(props).type;
    watchEffect(() => {
      store.dispatch("fetchData", { url: url.value, key: key.value });
    });

    const movies = computed(() => store.state[key.value]);
    const error = computed(() => store.state[`${key.value}Error`]);
    const loading = computed(() => store.state[`${key.value}Loading`]);

    return {
      error,
      loading,
      movies,
    };
  },
};
```

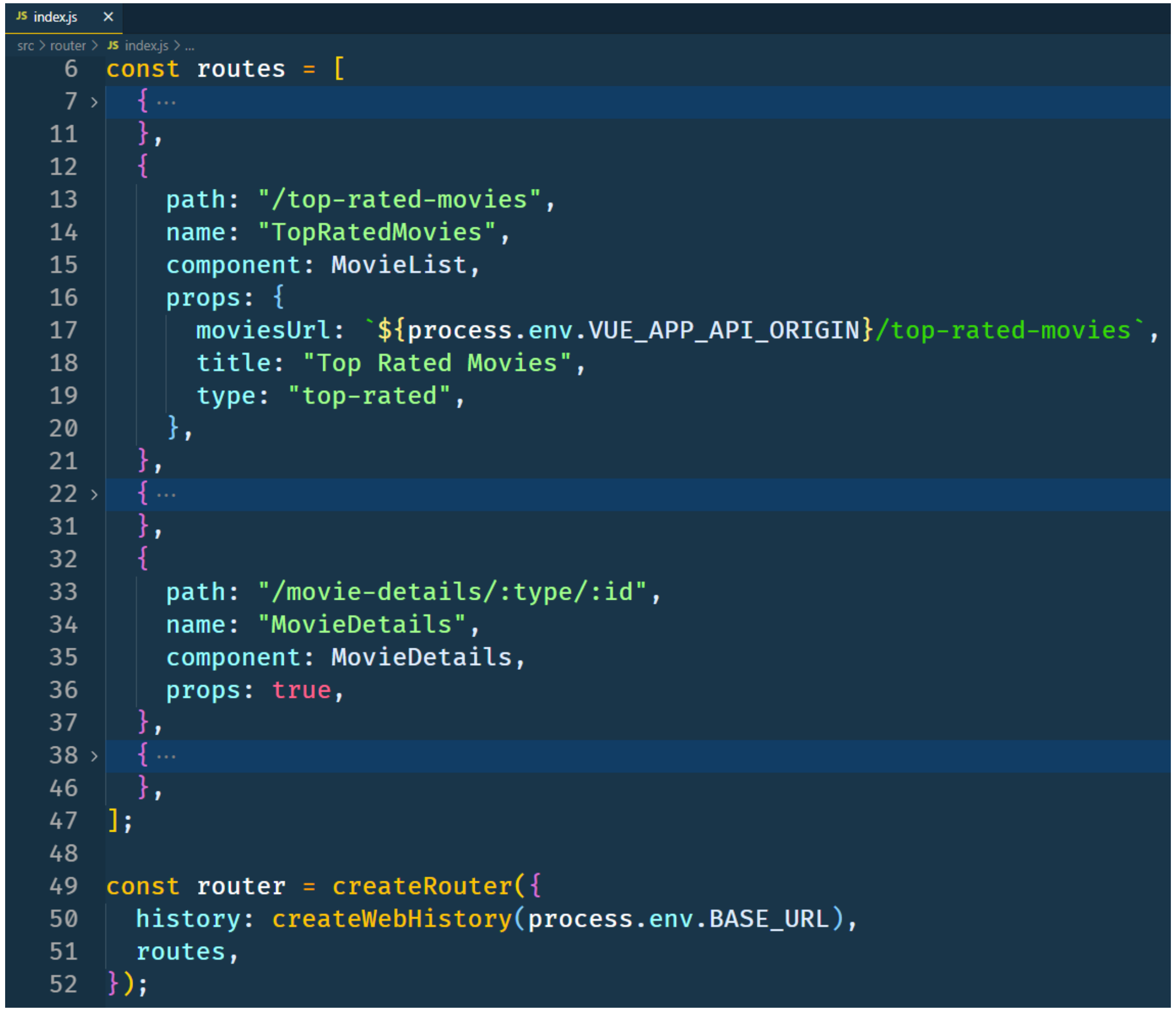
Vue Router

Single Page Routing & Navigation

Vue Router

- Map the **browser path to a component**
  - Use the `createRouter()` function to create the router
- The `<router-view />`
  - Determines where the component is rendered
- Use nested and/or named views in more complex layouts
  - Use the `<router-view>` `name` prop when using multiple views
- Guard against routes being activated
- Use **Vue Router version 4** with Vue 3
  - Vue Router 3 is for Vue 2 ☹

createRouter()



```
const routes = [
  {…
  },
  {
    path: "/top-rated-movies",
    name: "TopRatedMovies",
    component: MovieList,
    props: {
      moviesUrl: `${process.env.VUE_APP_API_ORIGIN}/top-rated-movies`,
      title: "Top Rated Movies",
      type: "top-rated",
    },
  },
  {…
  },
  {
    path: "/movie-details/:type/:id",
    name: "MovieDetails",
    component: MovieDetails,
    props: true,
  },
  {…
  },
];

const router = createRouter({
  history: createWebHistory(process.env.BASE_URL),
  routes,
});
```

Dynamic Routes

- Routes can be **dynamic** with one or more **parameters**
  - Like `{ path: "/movie-details/:type/:id“ }`

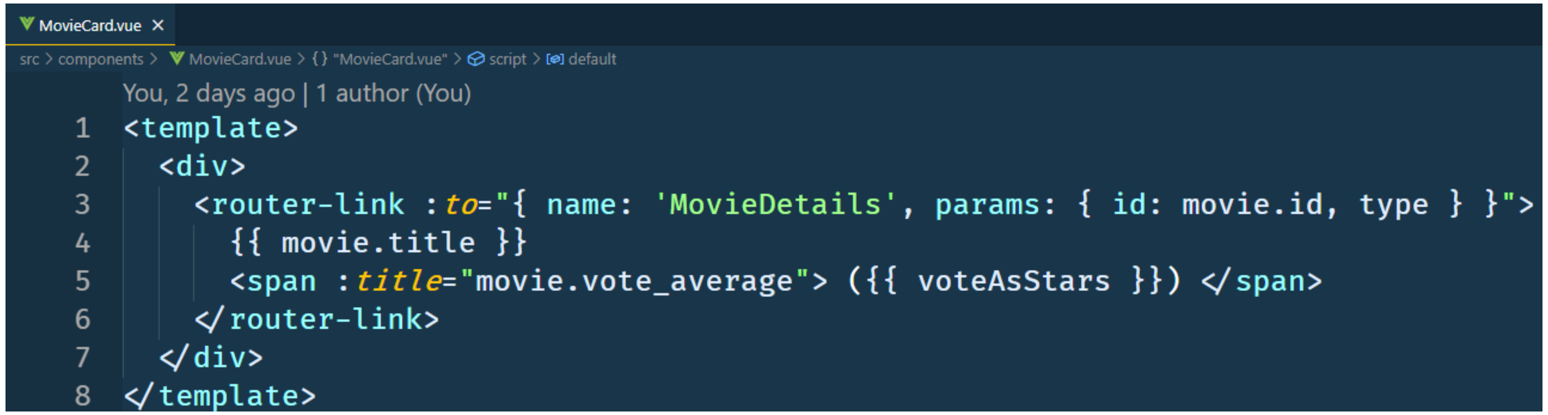
Navigating

- Use the `<router-link to="...">` in a **template**
  - Instead of an anchor tag
  - Still renders like an <a href="..."> in the browser
- **Programmatic navigation** with the Composition API
  - Use `const router = useRouter()` and `router.push("...")`
- Or `this.$router.push("...")` with the **Options API**

Named Routes

- **Named routes** are more **flexible**
  - Automatic encoding/decoding of params
  - No hardcoded URLs throughout the application

Named Routes/Links



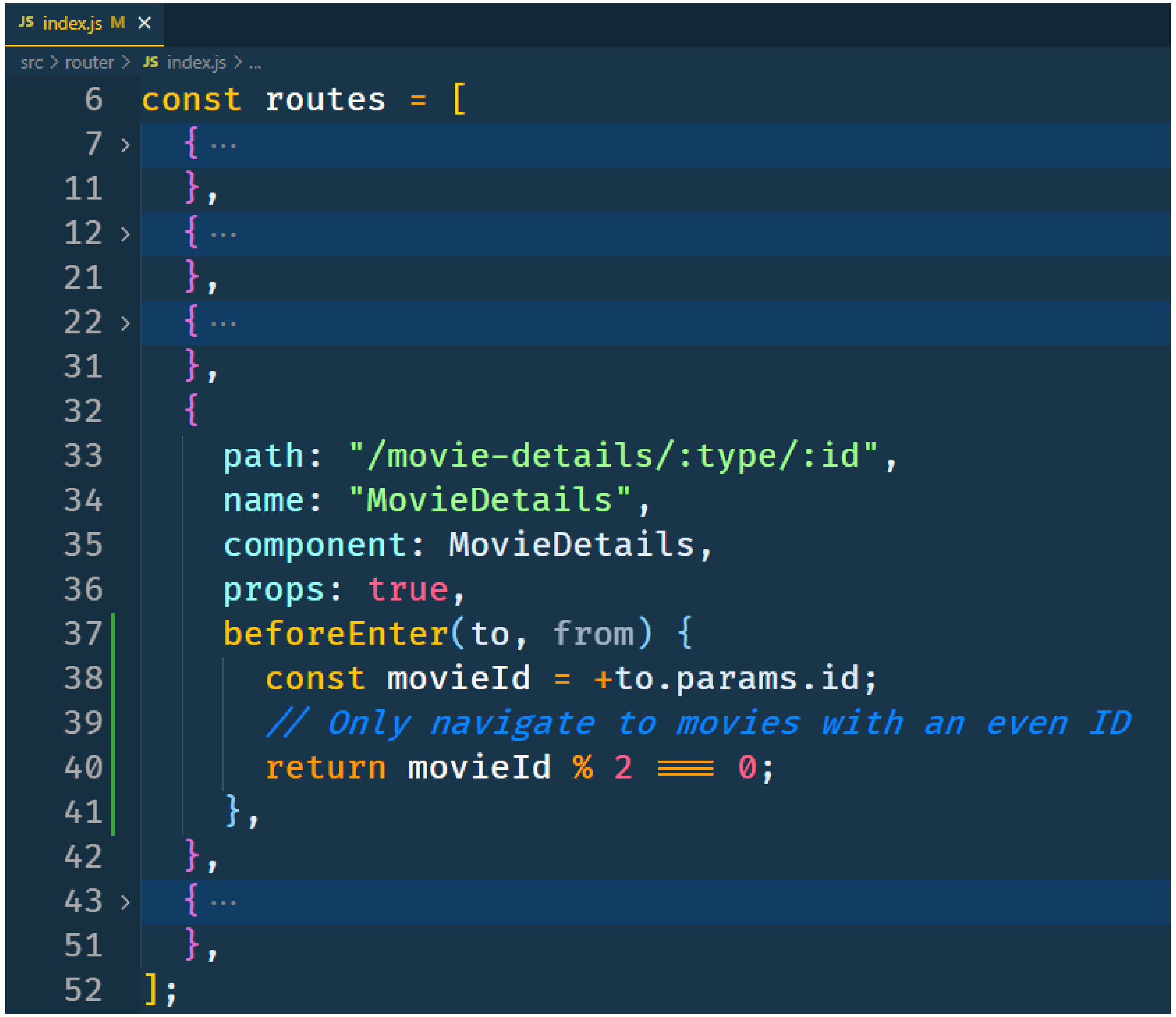
MovieCard.vue
src > components > MovieCard.vue > {} "MovieCard.vue" > script > default
You, 2 days ago | 1 author (You)

```
<template>
  <div>
    <router-link :to="{ name: 'MovieDetails', params: { id: movie.id, type } }">
      {{ movie.title }}
      <span :title="movie.vote_average"> ({{ voteAsStars }}) </span>
    </router-link>
  </div>
</template>
```

Navigation Guards

- Navigation **guards** can be **per route**
  - The `route.beforeEnter()`
- There is also a **global navigation guard**
  - The `router.beforeEach()`
- Each function receives the **`to`** and **`from`** location as parameter
- **Called when entering** a route from another route
  - Not when route params change
- Don't use the **`next`** parameter in version 4
  - It's deprecated and will be removed

Navigation Guard



```
const routes = [
  { ···
  },
  { ···
  },
  { ···
  },
  {
    path: "/movie-details/:type/:id",
    name: "MovieDetails",
    component: MovieDetails,
    props: true,
    beforeEnter(to, from) {
      const movieId = +to.params.id;
      // Only navigate to movies with an even ID
      return movieId % 2 === 0;
    },
  },
  { ···
  },
];
```

Async Navigation Guards

- Navigation guards **can be asynchronous**
  - Using async/await or returning a promise
- The **navigation state is pending** until the promise resolves

Vuetify

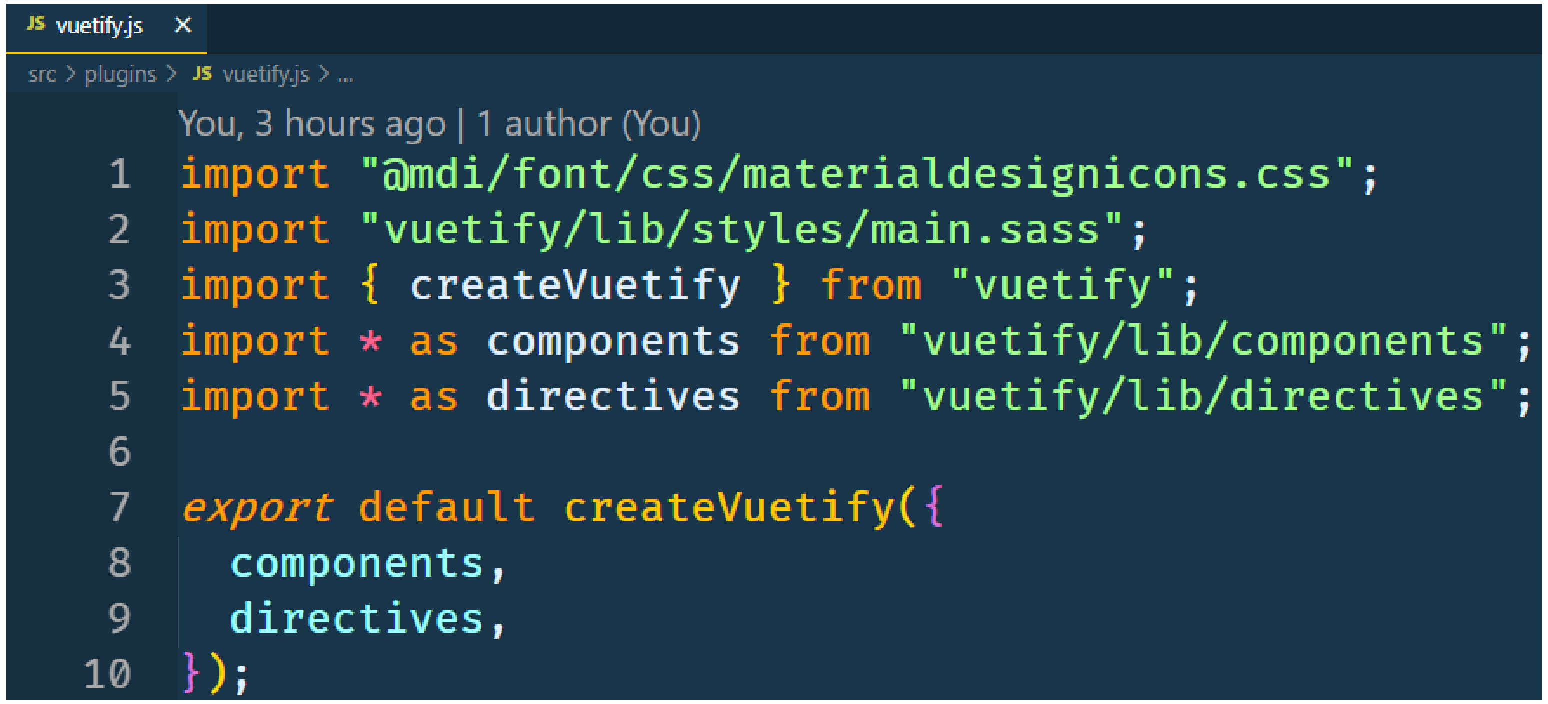
Compatibility

- **Vuetify for Vue 3** has not been released yet
    - Same is true for many other libraries
- See: Vue3 compatibility status of central Vue libraries

Adding Vuetify

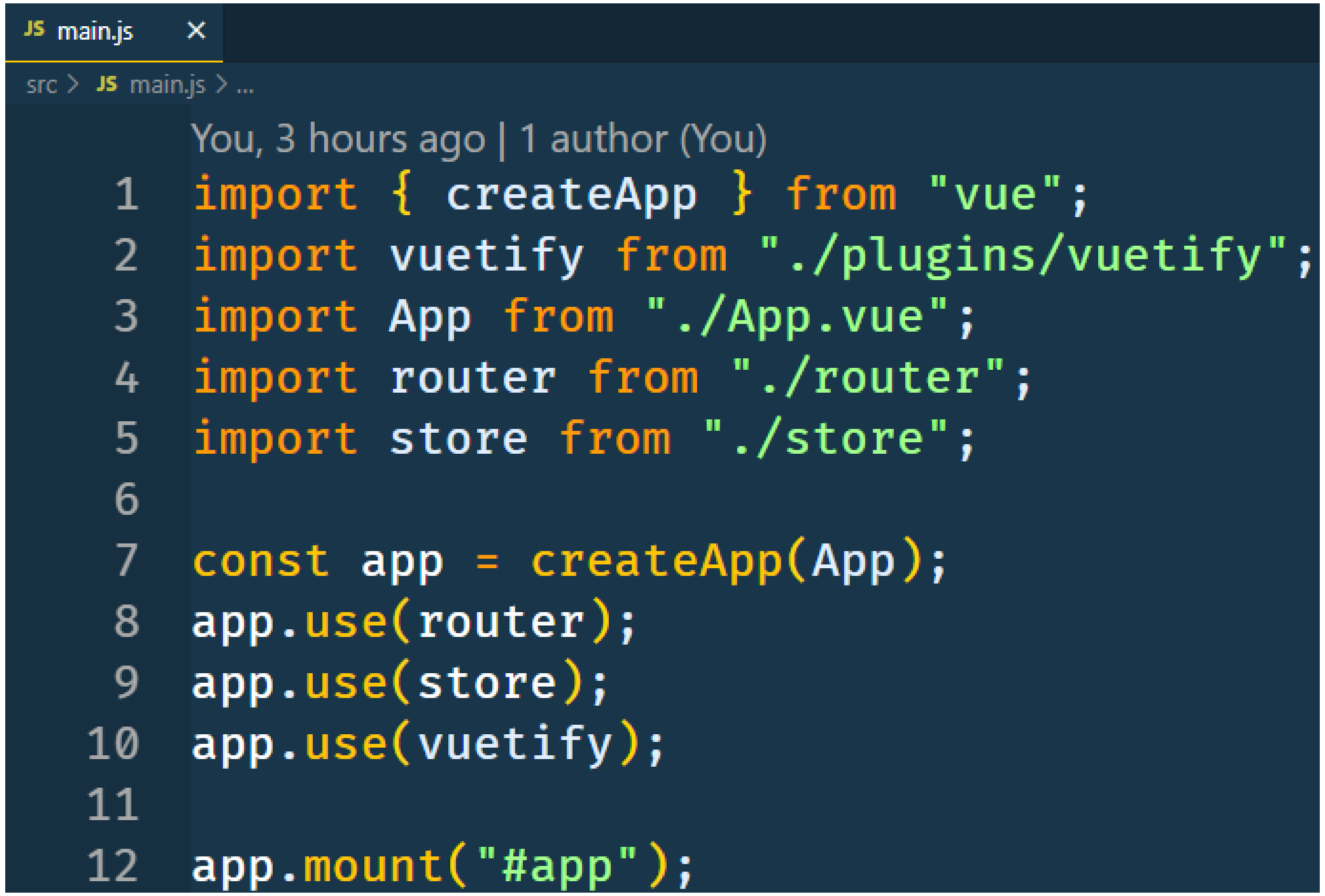
- Can be added using the **Vue CLI**
  - `vue add vuetify`
  - ~~`npm install vuetify@next`~~
  - `npm install vuetify@3.0.0-alpha.10`
- Adds and configures Vuetify
  - 👉Overwrites some files like App.vue

plugins/vuetify.js


vuetify.js
src > plugins > vuetify.js > ...
You, 3 hours ago | 1 author (You)

```javascript
import "@mdi/font/css/materialdesignicons.css";
import "vuetify/lib/styles/main.sass";
import { createVuetify } from "vuetify";
import * as components from "vuetify/lib/components";
import * as directives from "vuetify/lib/directives";

export default createVuetify({
  components,
  directives,
});
```

main.js



```
import { createApp } from "vue";
import vuetify from "./plugins/vuetify";
import App from "./App.vue";
import router from "./router";
import store from "./store";

const app = createApp(App);
app.use(router);
app.use(store);
app.use(vuetify);

app.mount("#app");
```

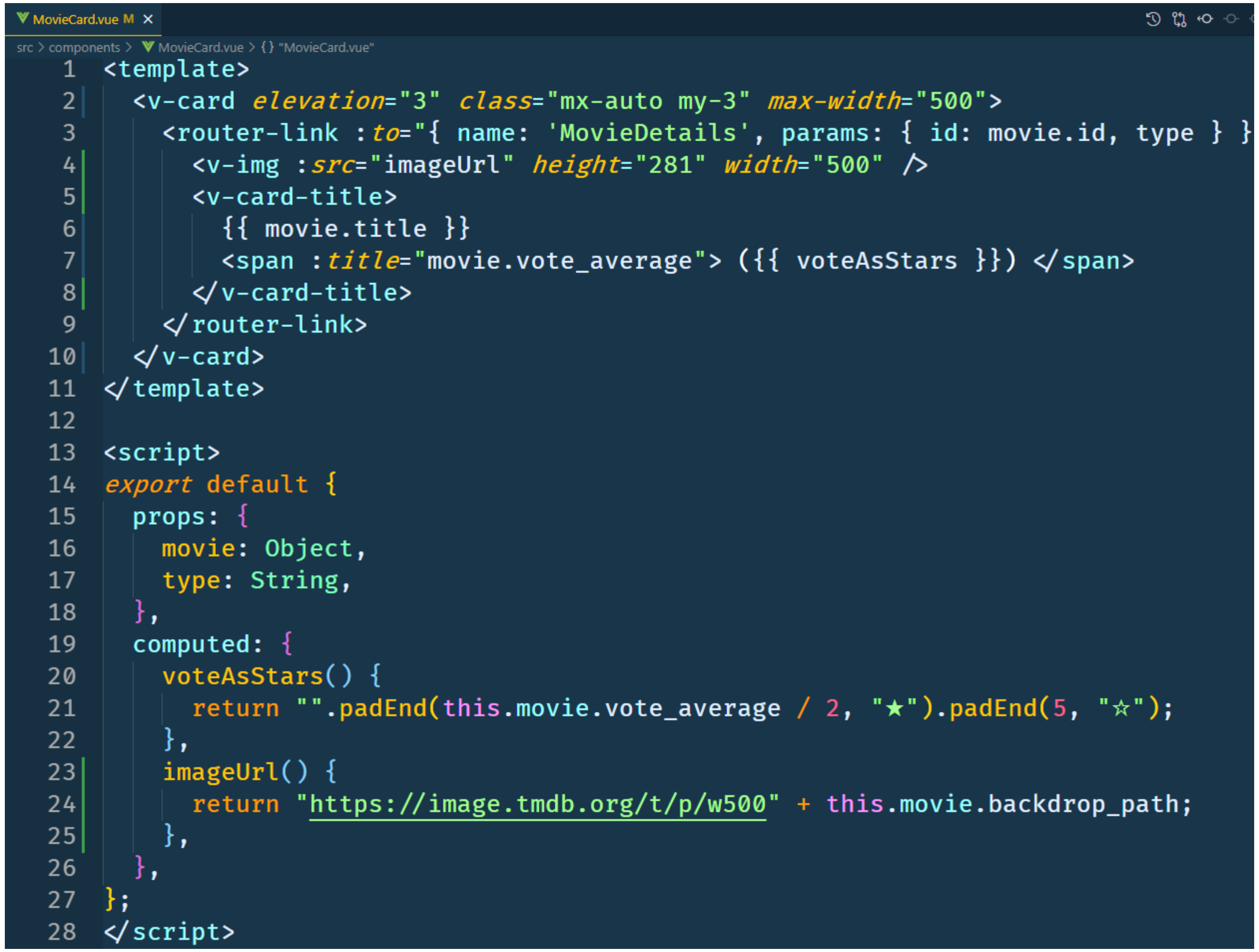
Movie Cards

Using Vuetify

Movie Cards

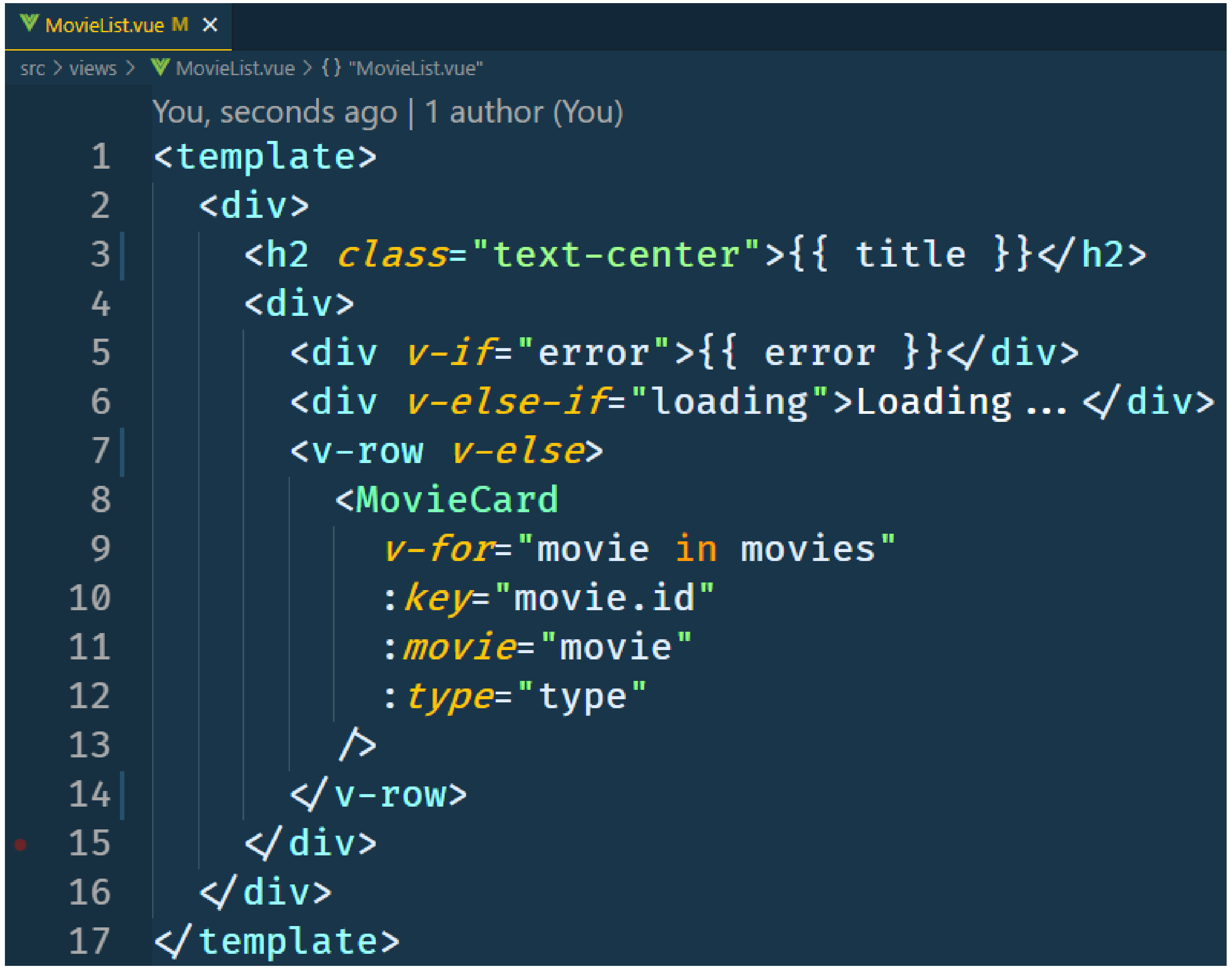
- Movie cards can be done with a `<v-card />` component
  - With a `<v-img />` to add images
- Place inside of a `<v-container />` `<v-row />` for a grid layout

MovieCard.vue



```
<template>
  <v-card elevation="3" class="mx-auto my-3" max-width="500">
    <router-link :to="{ name: 'MovieDetails', params: { id: movie.id, type } }
      <v-img :src="imageUrl" height="281" width="500" />
      <v-card-title>
        {{ movie.title }}
        <span :title="movie.vote_average"> ({{ voteAsStars }}) </span>
      </v-card-title>
    </router-link>
  </v-card>
</template>

<script>
export default {
  props: {
    movie: Object,
    type: String,
  },
  computed: {
    voteAsStars() {
      return "".padEnd(this.movie.vote_average / 2, "★").padEnd(5, "☆");
    },
    imageUrl() {
      return "https://image.tmdb.org/t/p/w500" + this.movie.backdrop_path;
    },
  },
};
</script>
```

MovieList.vue


MovieList.vue M
src > views > MovieList.vue > { } "MovieList.vue"
You, seconds ago | 1 author (You)

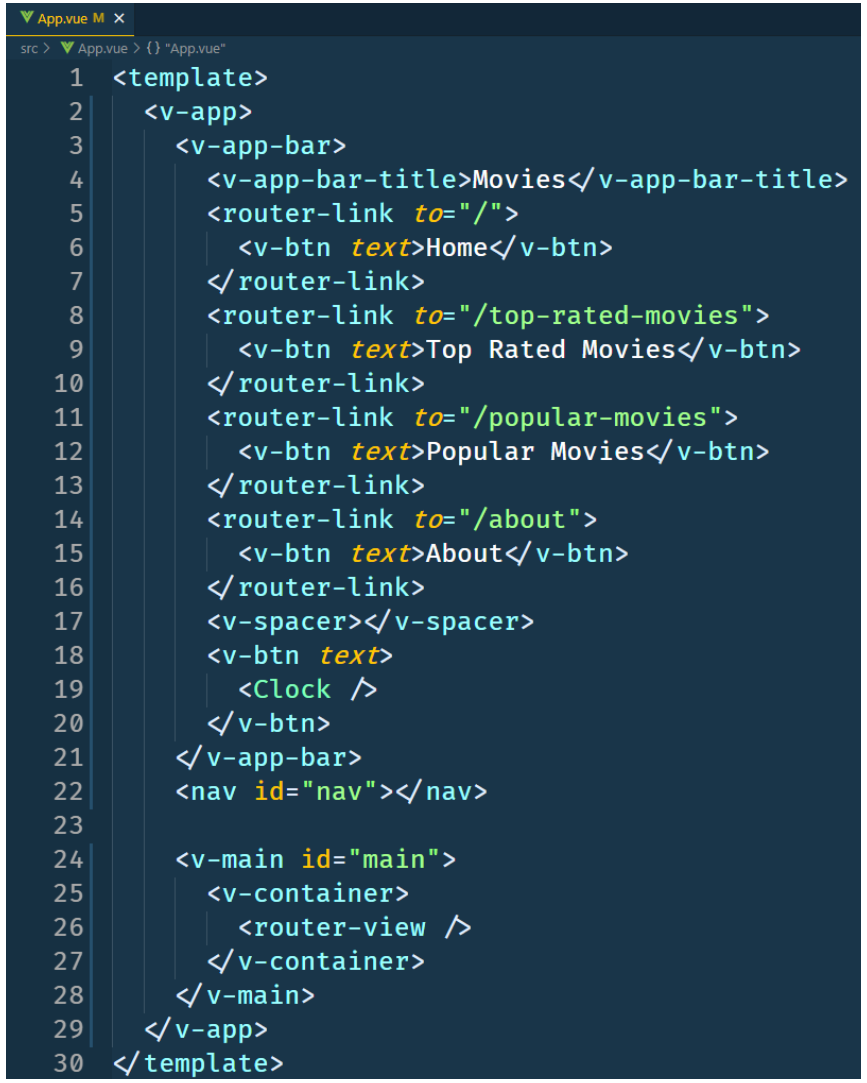
```
<template>
  <div>
    <h2 class="text-center">{{ title }}</h2>
    <div>
      <div v-if="error">{{ error }}</div>
      <div v-else-if="loading">Loading...</div>
      <v-row v-else>
        <MovieCard
          v-for="movie in movies"
          :key="movie.id"
          :movie="movie"
          :type="type"
        />
      </v-row>
    </div>
  </div>
</template>
```

App Bar

App Bar

- The <v-app-bar> can be used as an navigation bar

App.vue



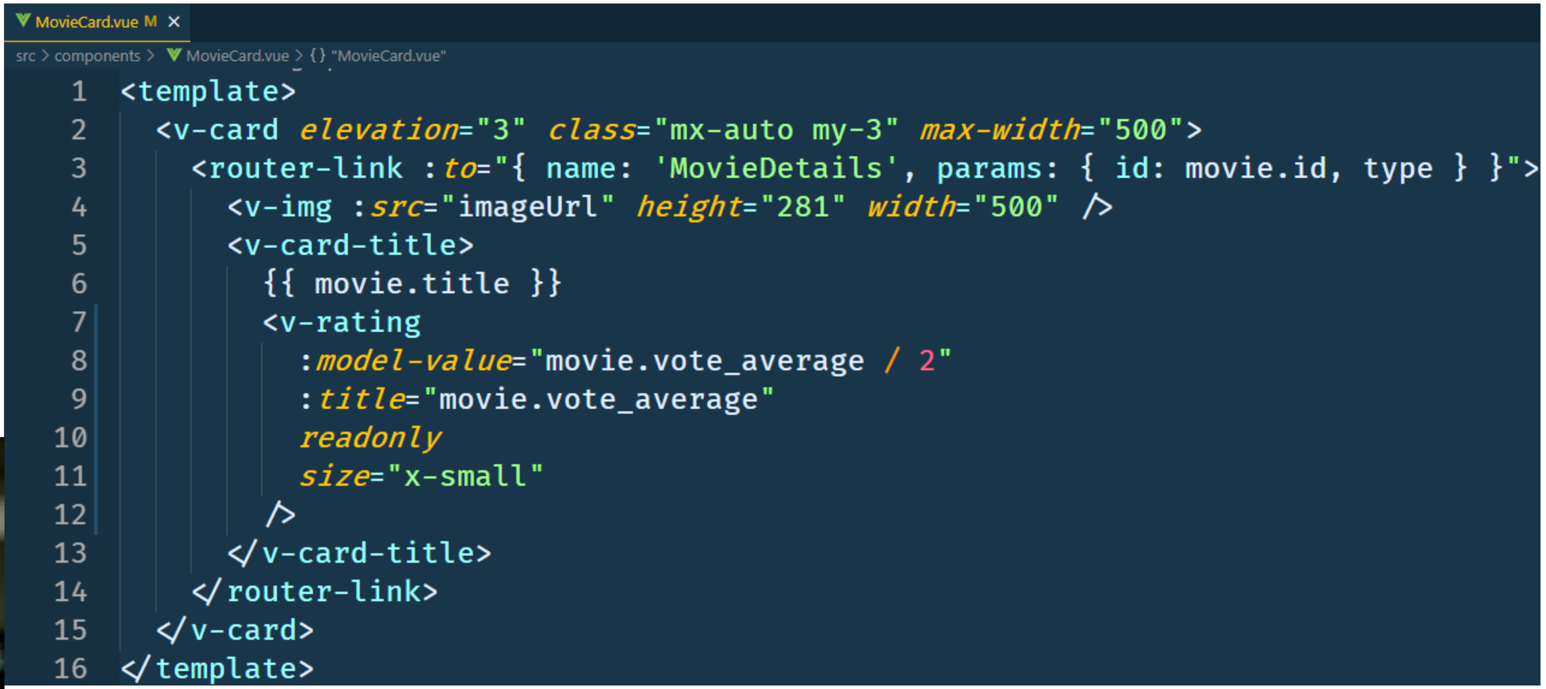
```
<template>
  <v-app>
    <v-app-bar>
      <v-app-bar-title>Movies</v-app-bar-title>
      <router-link to="/">
        <v-btn text>Home</v-btn>
      </router-link>
      <router-link to="/top-rated-movies">
        <v-btn text>Top Rated Movies</v-btn>
      </router-link>
      <router-link to="/popular-movies">
        <v-btn text>Popular Movies</v-btn>
      </router-link>
      <router-link to="/about">
        <v-btn text>About</v-btn>
      </router-link>
      <v-spacer></v-spacer>
      <v-btn text>
        <Clock />
      </v-btn>
    </v-app-bar>
    <nav id="nav"></nav>

    <v-main id="main">
      <v-container>
        <router-view />
      </v-container>
    </v-main>
  </v-app>
</template>
```

Movie Rating

Movie Rating

- Vuetify has a <v-rating /> component
    - Useful for the movie rating

MovieCard.vue



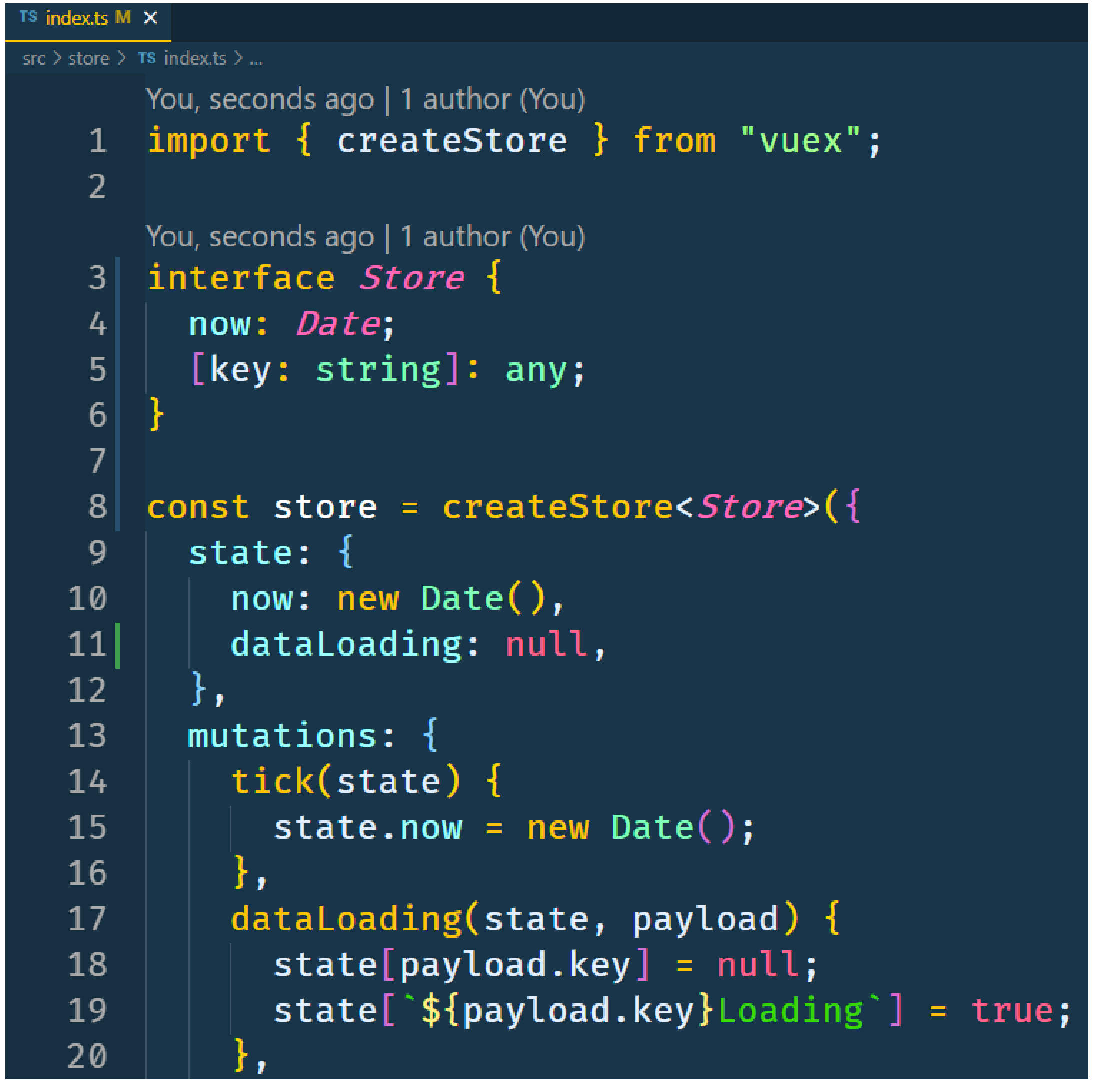
```
<template>
  <v-card elevation="3" class="mx-auto my-3" max-width="500">
    <router-link :to="{ name: 'MovieDetails', params: { id: movie.id, type } }">
      <v-img :src="imageUrl" height="281" width="500" />
      <v-card-title>
        {{ movie.title }}
        <v-rating
          :model-value="movie.vote_average / 2"
          :title="movie.vote_average"
          readonly
          size="x-small"
        />
      </v-card-title>
    </router-link>
  </v-card>
</template>
```

TypeScript

TypeScript

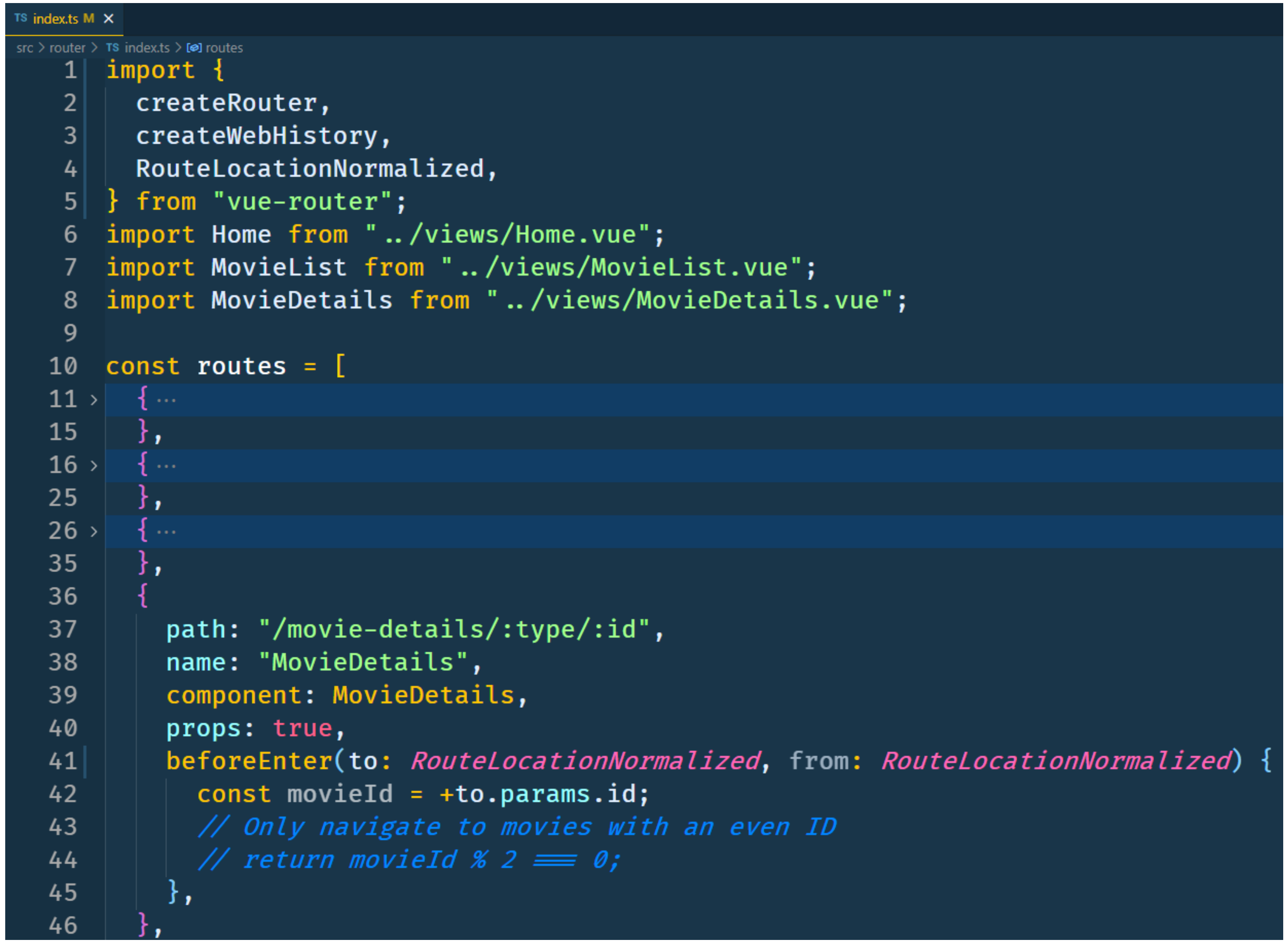
- Vue 3 is **written in TypeScript**
  - But you don’t have to
- TypeScript will help a lot when writing **reliable Vue applications**
  - Compile time validation of code
- **Adding TypeScript is easy** but not perfect using the Vue CLI
  - `vue add typescript`

store/index.ts


index.ts M
src > store > index.ts > ...
You, seconds ago | 1 author (You)

```
import { createStore } from "vuex";

interface Store {
  now: Date;
  [key: string]: any;
}

const store = createStore<Store>({
  state: {
    now: new Date(),
    dataLoading: null,
  },
  mutations: {
    tick(state) {
      state.now = new Date();
    },
    dataLoading(state, payload) {
      state[payload.key] = null;
      state[`${payload.key}Loading`] = true;
    },
```

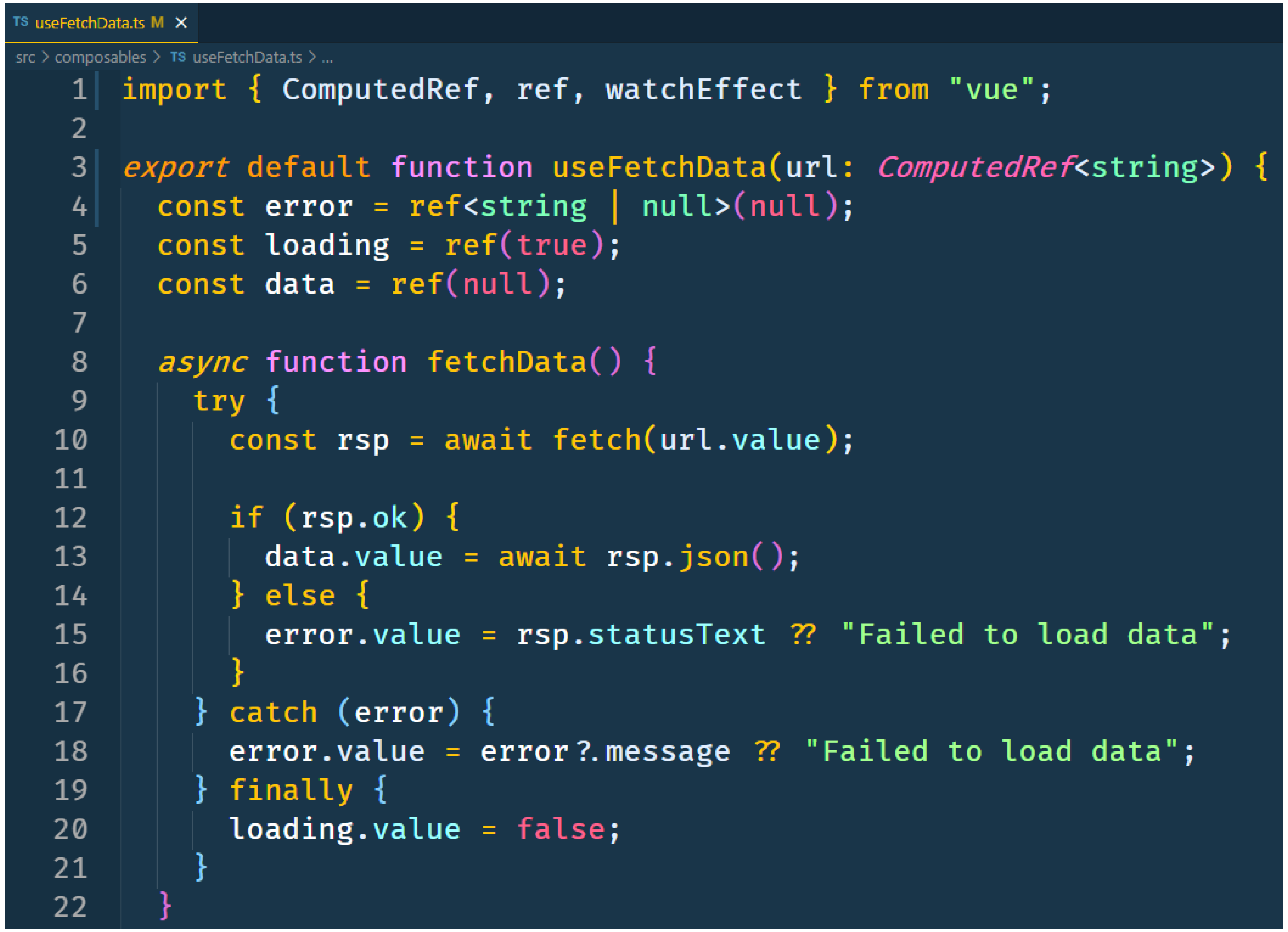
router/index.ts


index.ts M ×

src > router > index.ts > routes

```
import {
  createRouter,
  createWebHistory,
  RouteLocationNormalized,
} from "vue-router";
import Home from "../views/Home.vue";
import MovieList from "../views/MovieList.vue";
import MovieDetails from "../views/MovieDetails.vue";

const routes = [
  {…
  },
  {…
  },
  {…
  },
  {
    path: "/movie-details/:type/:id",
    name: "MovieDetails",
    component: MovieDetails,
    props: true,
    beforeEnter(to: RouteLocationNormalized, from: RouteLocationNormalized) {
      const movieId = +to.params.id;
      // Only navigate to movies with an even ID
      // return movieId % 2 === 0;
    },
  },
```

useFetchData.ts



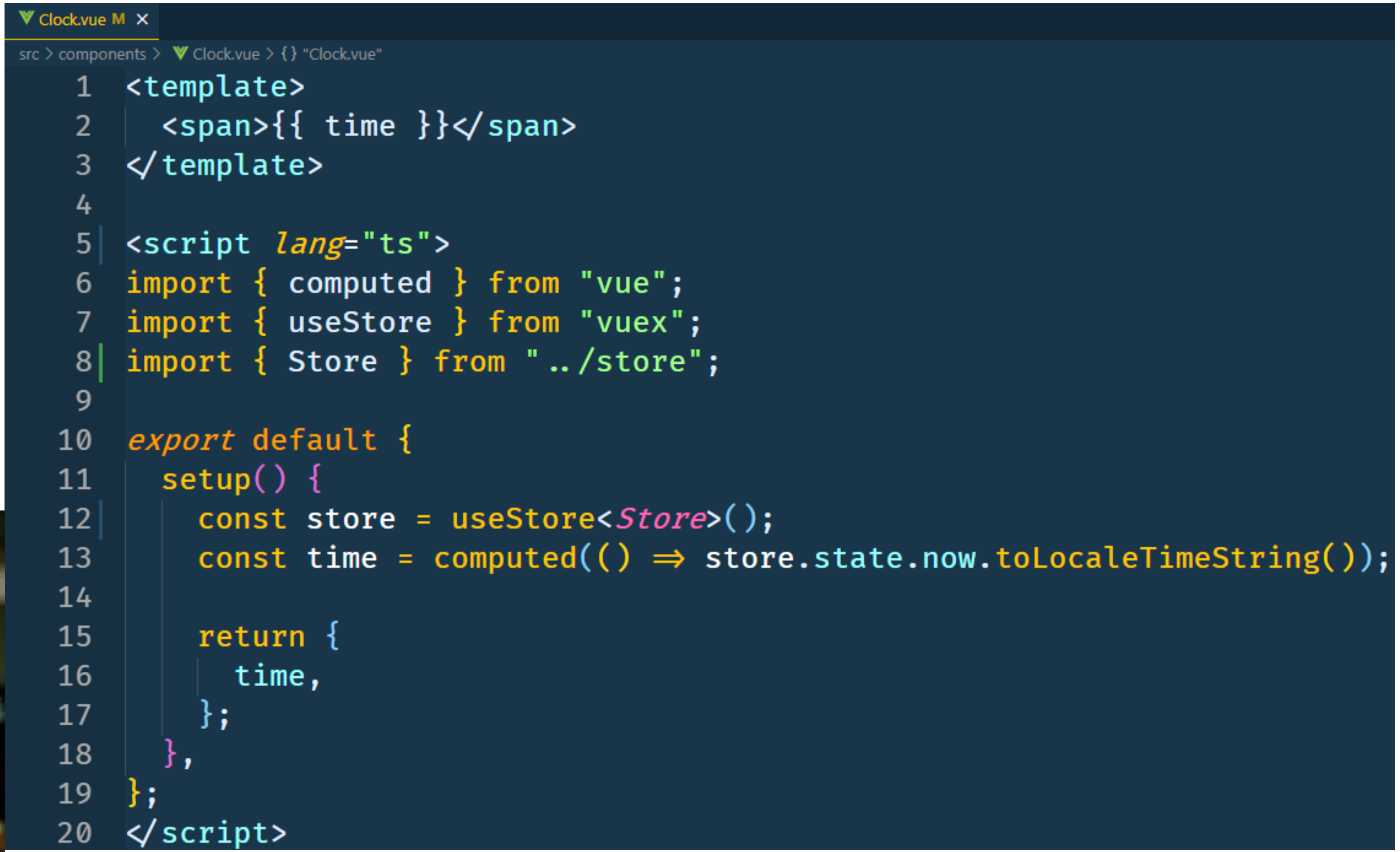
```ts
import { ComputedRef, ref, watchEffect } from "vue";

export default function useFetchData(url: ComputedRef<string>) {
  const error = ref<string | null>(null);
  const loading = ref(true);
  const data = ref(null);

  async function fetchData() {
    try {
      const rsp = await fetch(url.value);

      if (rsp.ok) {
        data.value = await rsp.json();
      } else {
        error.value = rsp.statusText ?? "Failed to load data";
      }
    } catch (error) {
      error.value = error?.message ?? "Failed to load data";
    } finally {
      loading.value = false;
    }
  }
```

Convert other components to TypeScript

Conversion

- Not all components are converted to TypeScript
  - For a number there is not much point
- Add `lang="ts"` to the `<script>` element

Clock.vue



```
<template>
  <span>{{ time }}</span>
</template>

<script lang="ts">
import { computed } from "vue";
import { useStore } from "vuex";
import { Store } from "../store";

export default {
  setup() {
    const store = useStore<Store>();
    const time = computed(() => store.state.now.toLocaleTimeString());

    return {
      time,
    };
  },
};
</script>
```

Getting started with Nuxt.js

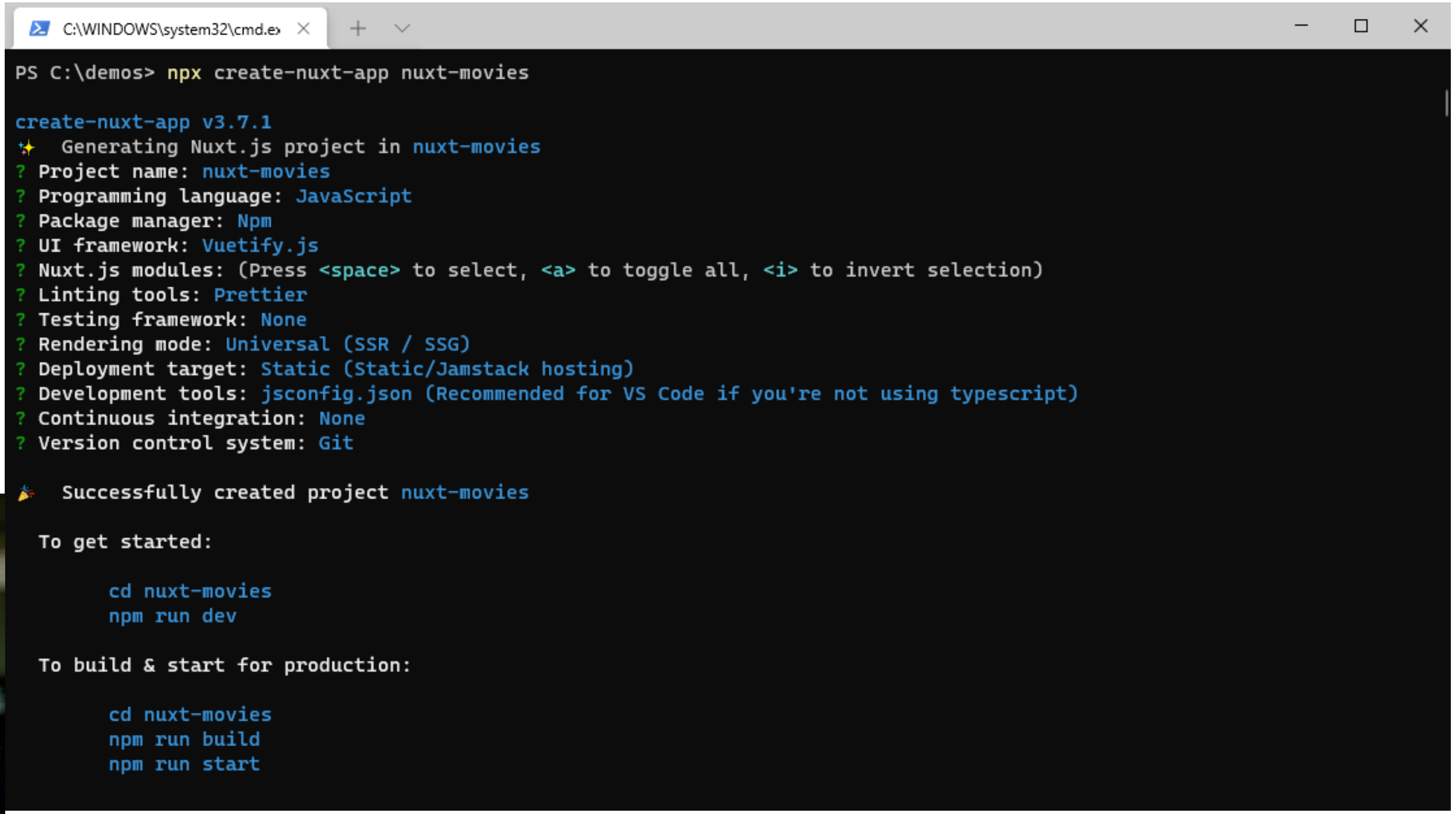
Nuxt.js

- Nuxt.js is a an **application framework** build on top of Vue
  - Inspired by the popular Next.js framework for react
- Supports **server side rendering** and much more
  - Great for performance and SEO
- Many **build in** and external **modules**
  - Makes integration much easier
- Create a new Nuxt.js application using the **Nuxt CLI**
  - `npx create-nuxt-app nuxt-movies`

Nuxt and Vue 3

- Nuxt.js **doesn’t support Vue 3** yet
  - Has optional dependencies on many external libraries like Vuetify
  - An alpha version is due any day

create-nuxt-app



```
PS C:\demos> npx create-nuxt-app nuxt-movies

create-nuxt-app v3.7.1
✨  Generating Nuxt.js project in nuxt-movies
? Project name: nuxt-movies
? Programming language: JavaScript
? Package manager: Npm
? UI framework: Vuetify.js
? Nuxt.js modules: (Press <space> to select, <a> to toggle all, <i> to invert selection)
? Linting tools: Prettier
? Testing framework: None
? Rendering mode: Universal (SSR / SSG)
? Deployment target: Static (Static/Jamstack hosting)
? Development tools: jsconfig.json (Recommended for VS Code if you're not using typescript)
? Continuous integration: None
? Version control system: Git

🎉  Successfully created project nuxt-movies

  To get started:

        cd nuxt-movies
        npm run dev

  To build & start for production:

        cd nuxt-movies
        npm run build
        npm run start
```

Nuxt Routing

Nuxt Routing

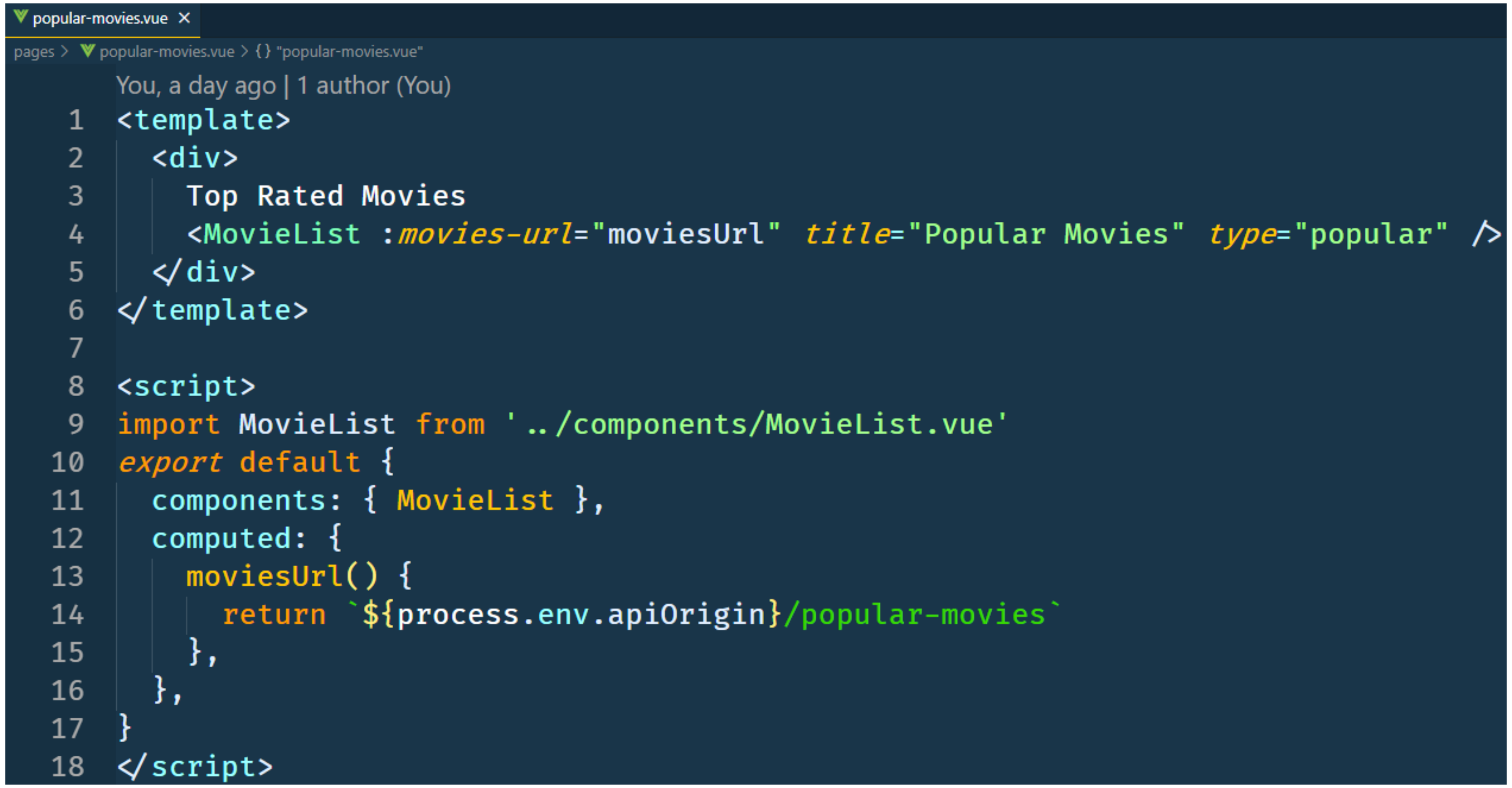
- Nuxt uses **file based routing**
  - Create Vue components in the pages folder
  - The router config will be generated from there
- Use the `<NuxtLink />` component for **anchor tags**
  - Extends the Vue Router `<RouterLink />`
  - Adds smart prefetching behavior by default

The popular movies

Popular movies

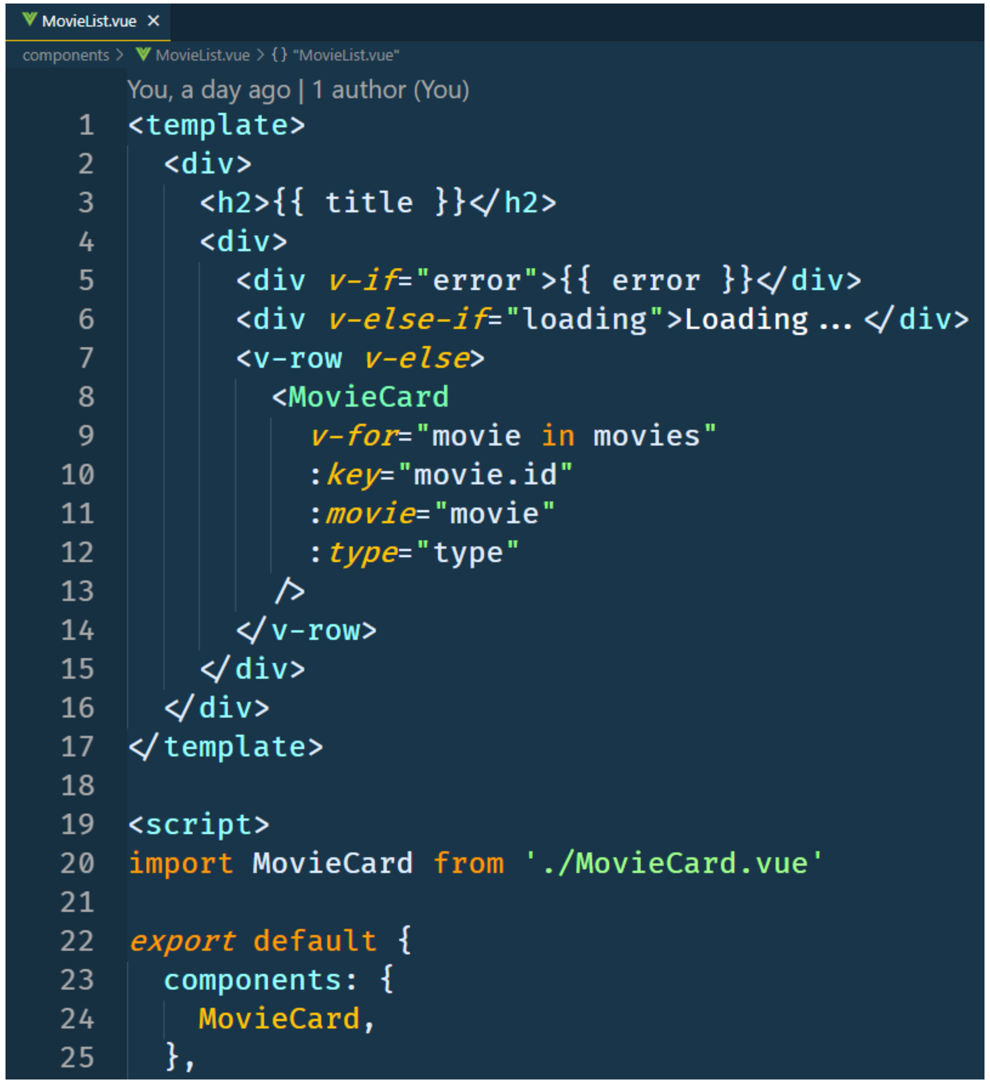
- Create a `popular-movies.vue` in the pages directory
  - And render the `<MovieList />` component
  - Note: Copied from the original Vue 2 workshop code

pages/popular-movies.vue


popular-movies.vue ×
pages > popular-movies.vue > { } "popular-movies.vue"
You, a day ago | 1 author (You)

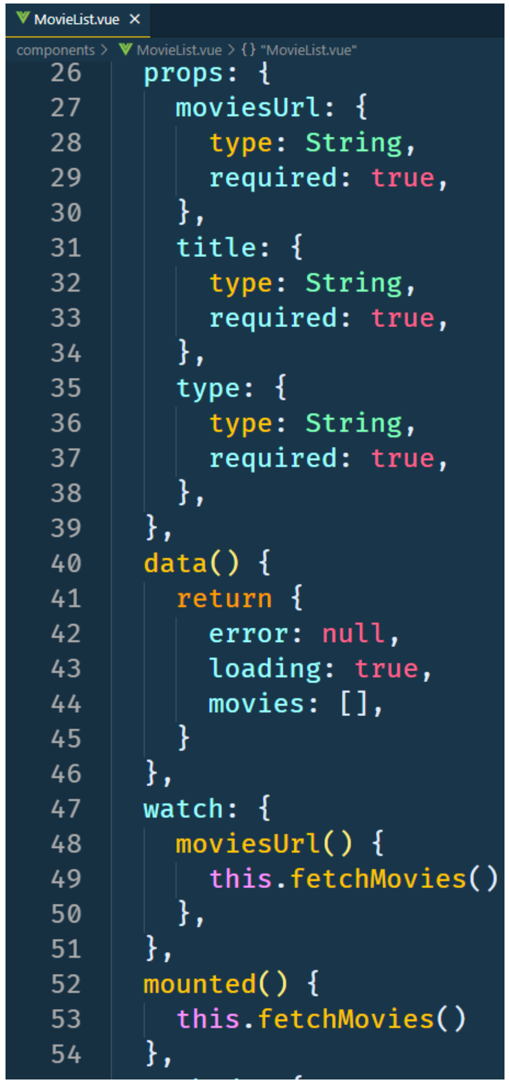
```
<template>
  <div>
    Top Rated Movies
    <MovieList :movies-url="moviesUrl"  title="Popular Movies"  type="popular"  />
  </div>
</template>

<script>
import MovieList from '../components/MovieList.vue'
export default {
  components: { MovieList },
  computed: {
    moviesUrl() {
      return  `${process.env.apiOrigin}/popular-movies`
    },
  },
}
</script>
```

MovieList.vue


MovieList.vue ×
components > MovieList.vue > {} "MovieList.vue"
You, a day ago | 1 author (You)

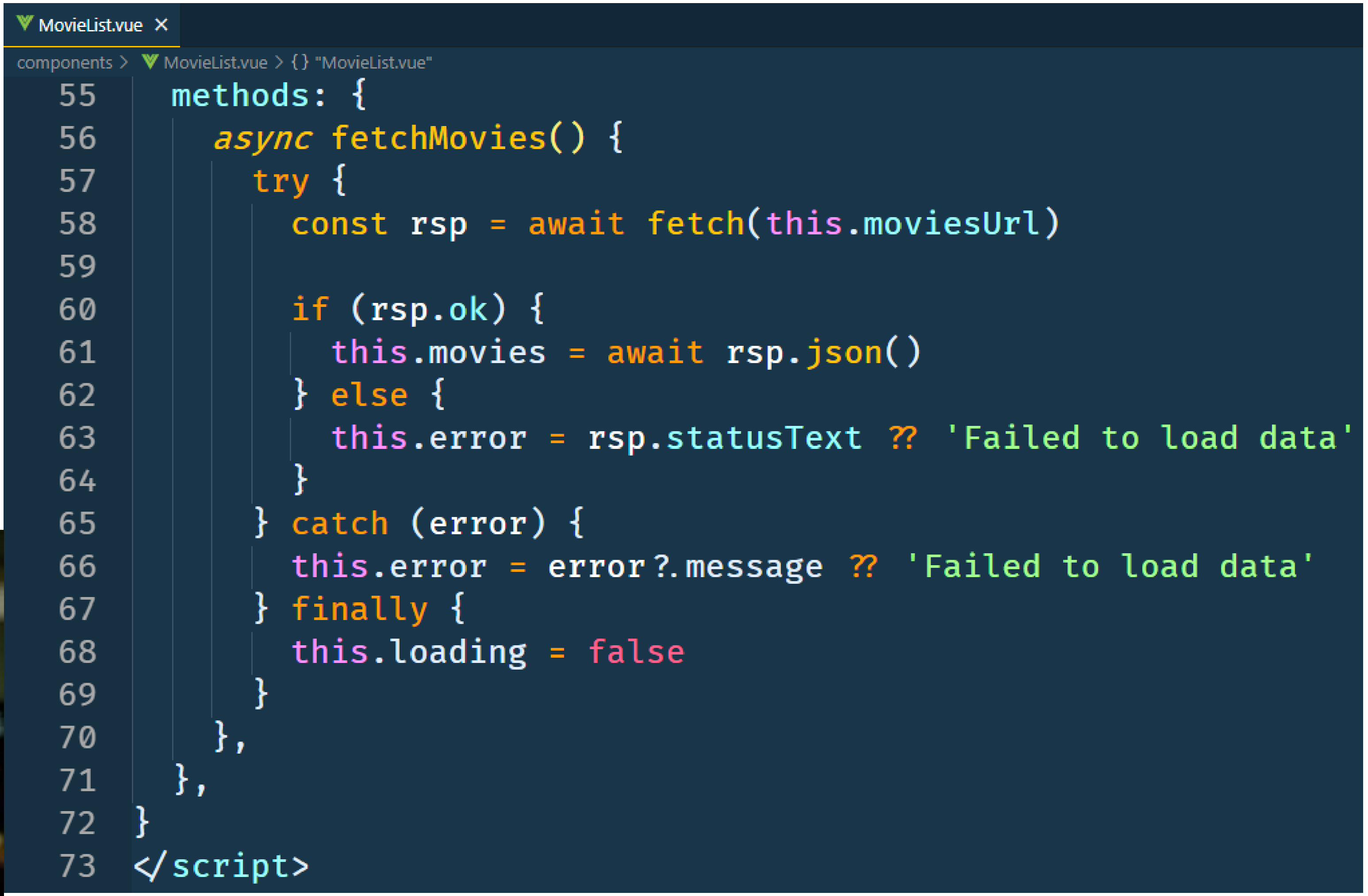
```
<template>
  <div>
    <h2>{{ title }}</h2>
    <div>
      <div v-if="error">{{ error }}</div>
      <div v-else-if="loading">Loading ... </div>
      <v-row v-else>
        <MovieCard
          v-for="movie in movies"
          :key="movie.id"
          :movie="movie"
          :type="type"
        />
      </v-row>
    </div>
  </div>
</template>

<script>
import MovieCard from './MovieCard.vue'

export default {
  components: {
    MovieCard,
  },
```

MovieList.vue



```
props: {
  moviesUrl: {
    type: String,
    required: true,
  },
  title: {
    type: String,
    required: true,
  },
  type: {
    type: String,
    required: true,
  },
},
data() {
  return {
    error: null,
    loading: true,
    movies: [],
  }
},
watch: {
  moviesUrl() {
    this.fetchMovies()
  },
},
mounted() {
  this.fetchMovies()
},
```

MovieList.vue



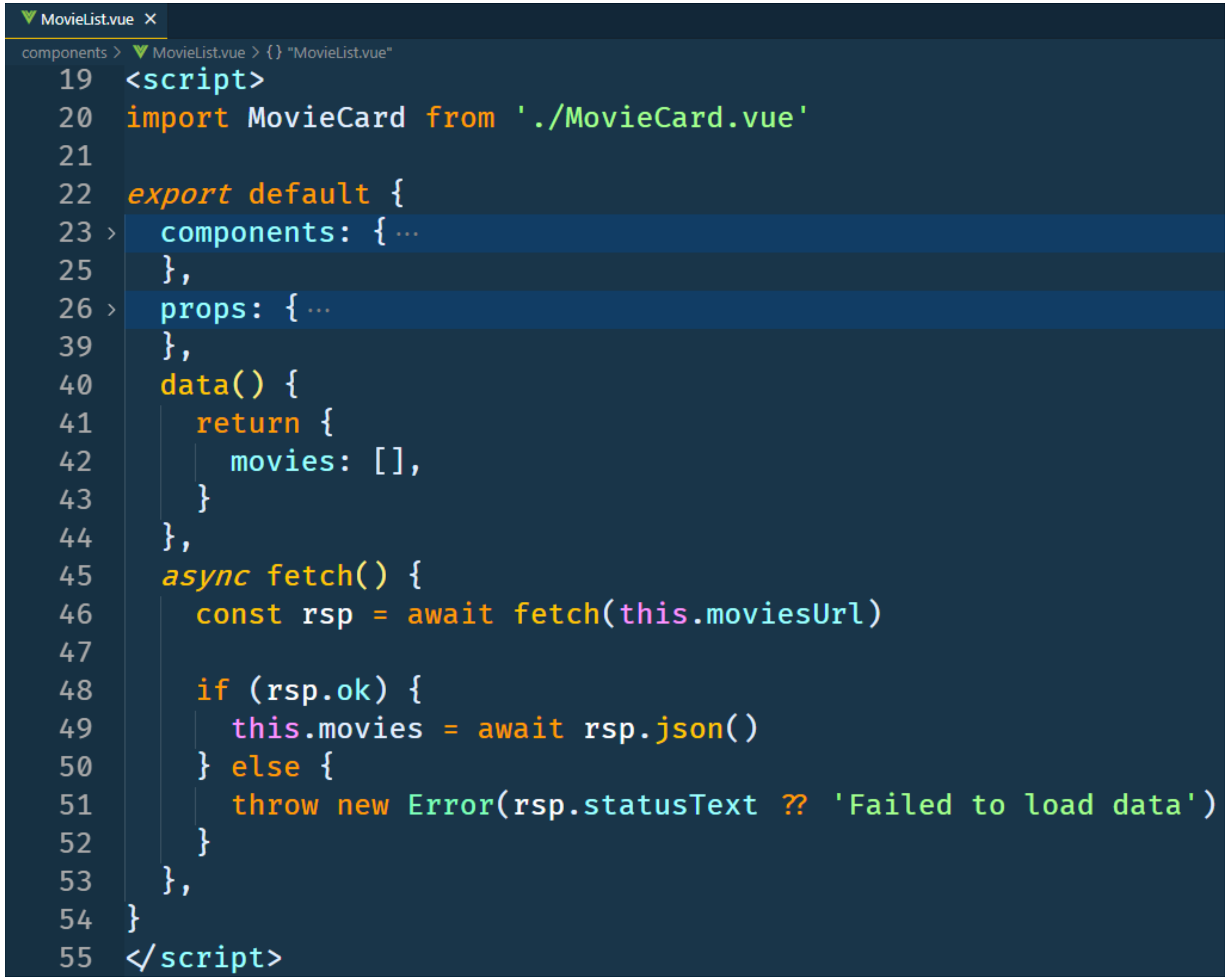
```
  methods: {
    async fetchMovies() {
      try {
        const rsp = await fetch(this.moviesUrl)

        if (rsp.ok) {
          this.movies = await rsp.json()
        } else {
          this.error = rsp.statusText ?? 'Failed to load data'
        }
      } catch (error) {
        this.error = error?.message ?? 'Failed to load data'
      } finally {
        this.loading = false
      }
    },
  },
}
</script>
```

The Nuxt fetch hook

Nuxt fetch hook

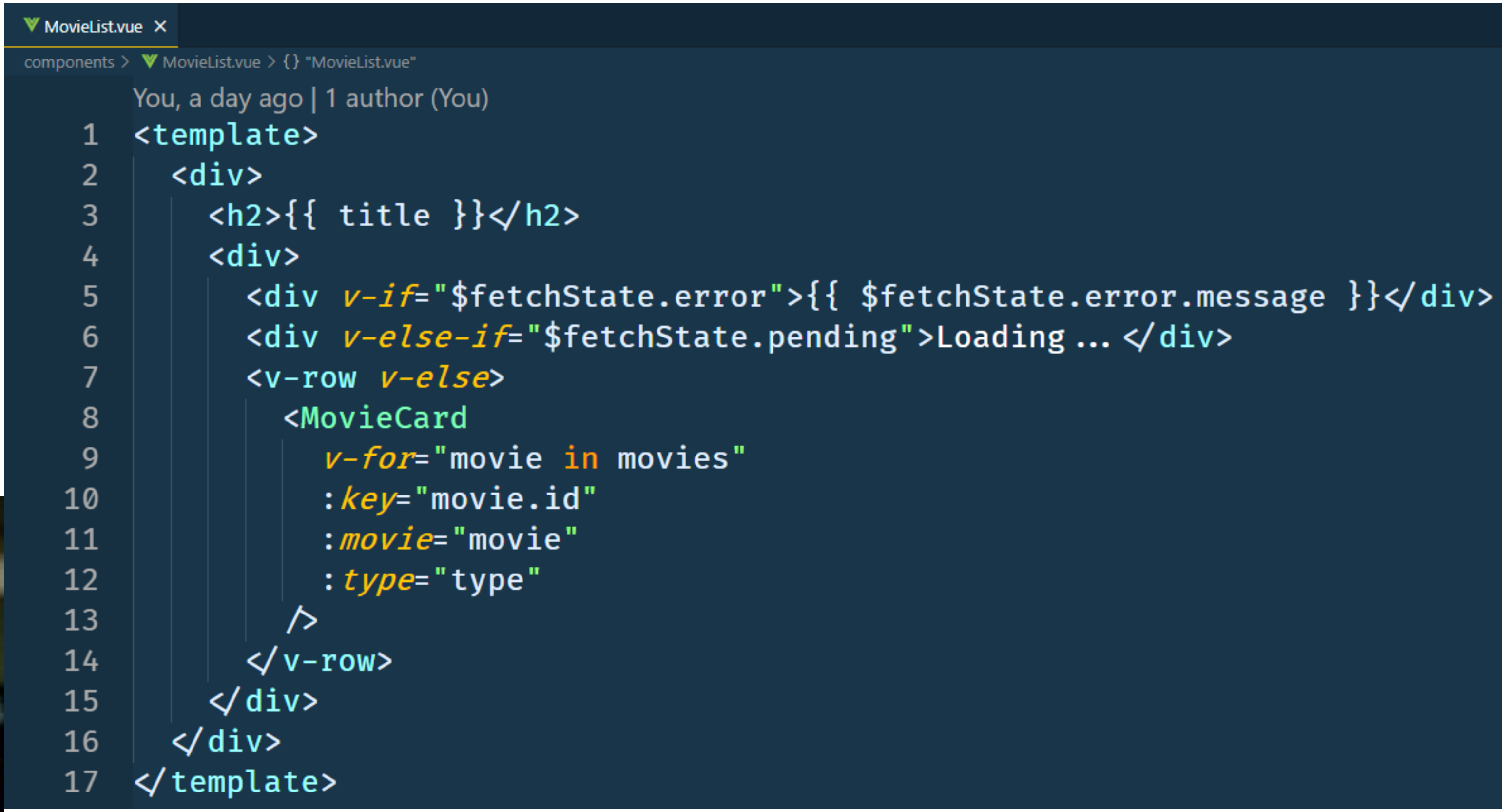
- Fetching data in the **mounted lifecycle hook is not optimal**
  - Always done on the client
- The Nuxt fetch hook will **server render where possible**
  - Or else on the clients browser
- Use the **`$fetchState`**
  - It has a `pending` and `error` property.

MovieList.vue



```
<script>
import MovieCard from './MovieCard.vue'

export default {
  components: {…
  },
  props: {…
  },
  data() {
    return {
      movies: [],
    }
  },
  async fetch() {
    const rsp = await fetch(this.moviesUrl)

    if (rsp.ok) {
      this.movies = await rsp.json()
    } else {
      throw new Error(rsp.statusText ?? 'Failed to load data')
    }
  },
}
</script>
```

MovieList.vue


MovieList.vue
components > MovieList.vue > {} "MovieList.vue"
You, a day ago | 1 author (You)

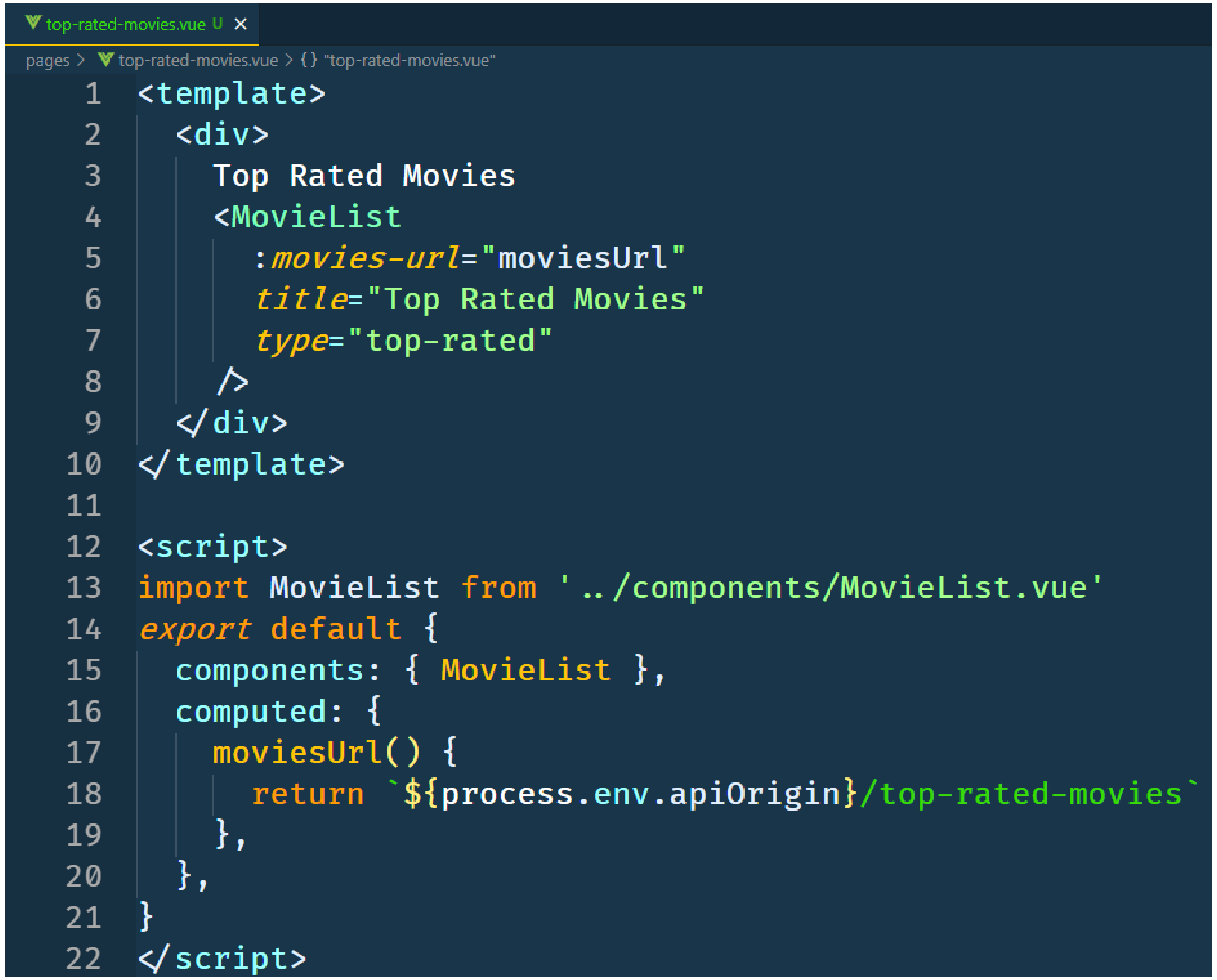
```
<template>
  <div>
    <h2>{{ title }}</h2>
    <div>
      <div v-if="$fetchState.error">{{ $fetchState.error.message }}</div>
      <div v-else-if="$fetchState.pending">Loading ... </div>
      <v-row v-else>
        <MovieCard
          v-for="movie in movies"
          :key="movie.id"
          :movie="movie"
          :type="type"
        />
      </v-row>
    </div>
  </div>
</template>
```

Top Rated Movies

Top Rated Movies

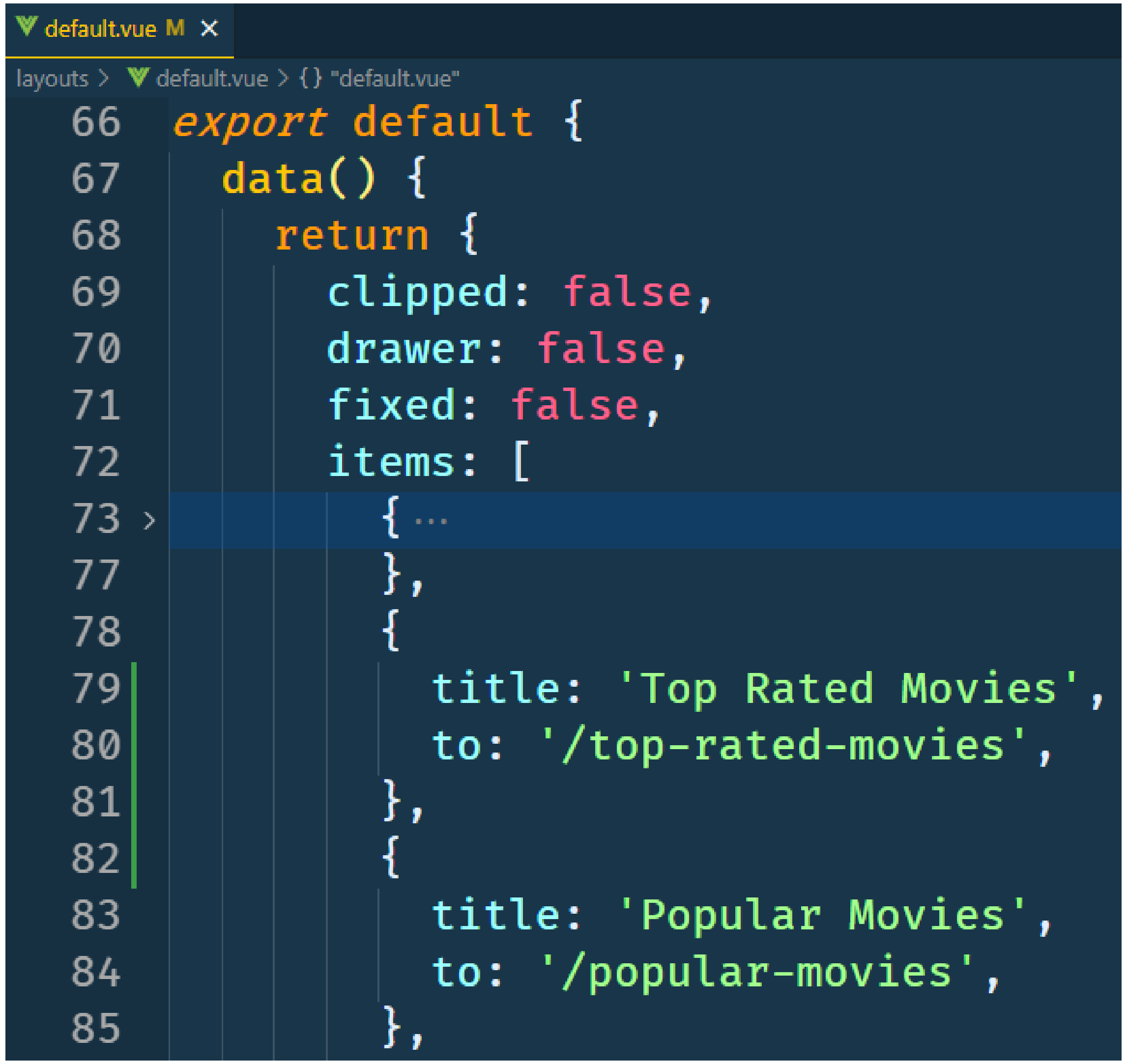
- Create a **new page** for the top rated movies
  - Render the `<MovieList>` with the proper URL
- Note: A copy of Popular Movies with a few changes

top-rated-movies.vue



```
<template>
  <div>
    Top Rated Movies
    <MovieList
      :movies-url="moviesUrl"
      title="Top Rated Movies"
      type="top-rated"
    />
  </div>
</template>

<script>
import MovieList from '../components/MovieList.vue'
export default {
  components: { MovieList },
  computed: {
    moviesUrl() {
      return `${process.env.apiOrigin}/top-rated-movies`
    },
  },
}
</script>
```

default.vue



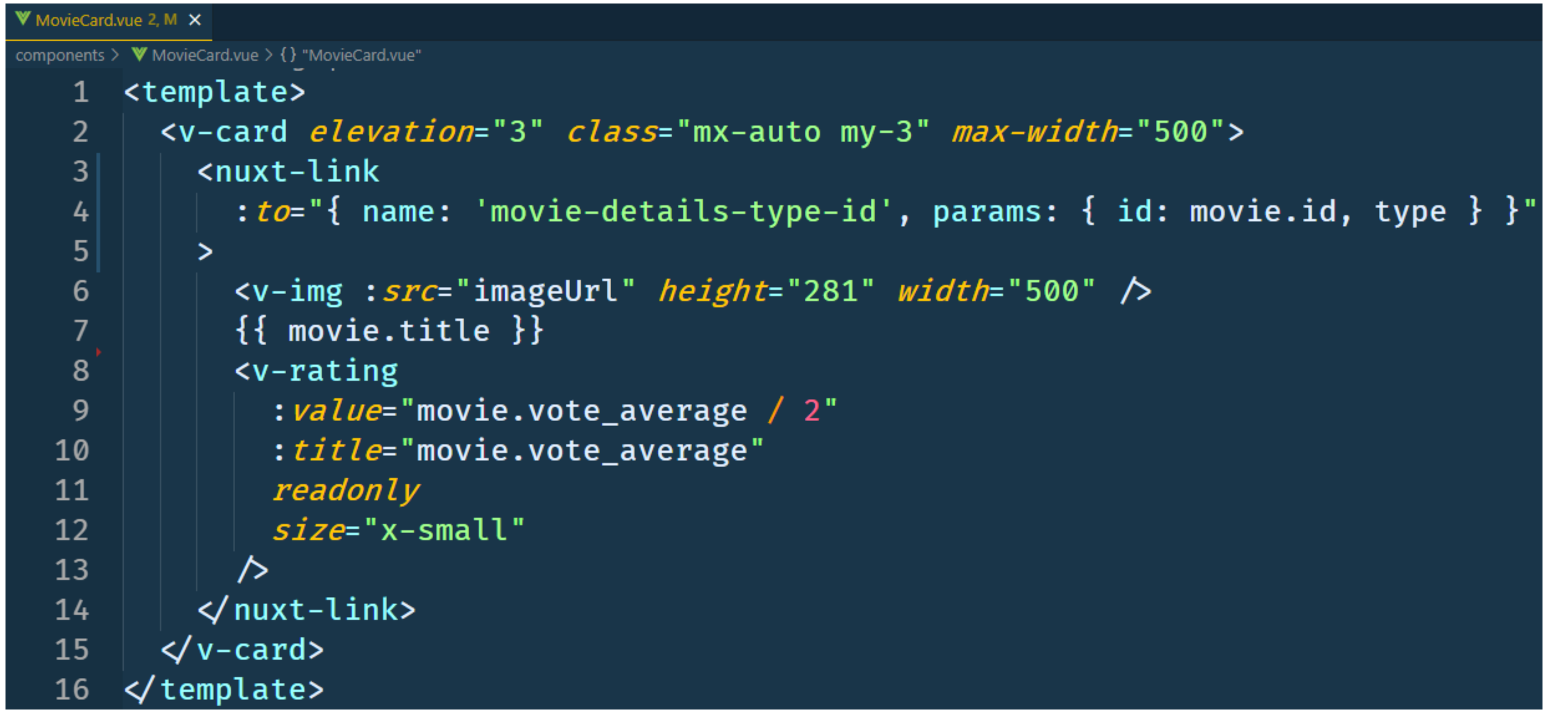
```
export default {
  data() {
    return {
      clipped: false,
      drawer: false,
      fixed: false,
      items: [
        { ...
        },
        {
          title: 'Top Rated Movies',
          to: '/top-rated-movies',
        },
        {
          title: 'Popular Movies',
          to: '/popular-movies',
        },
```

The MovieCard

The MovieCard

- The `<MovieCard />` is simple using Vuetify

MovieCard.vue



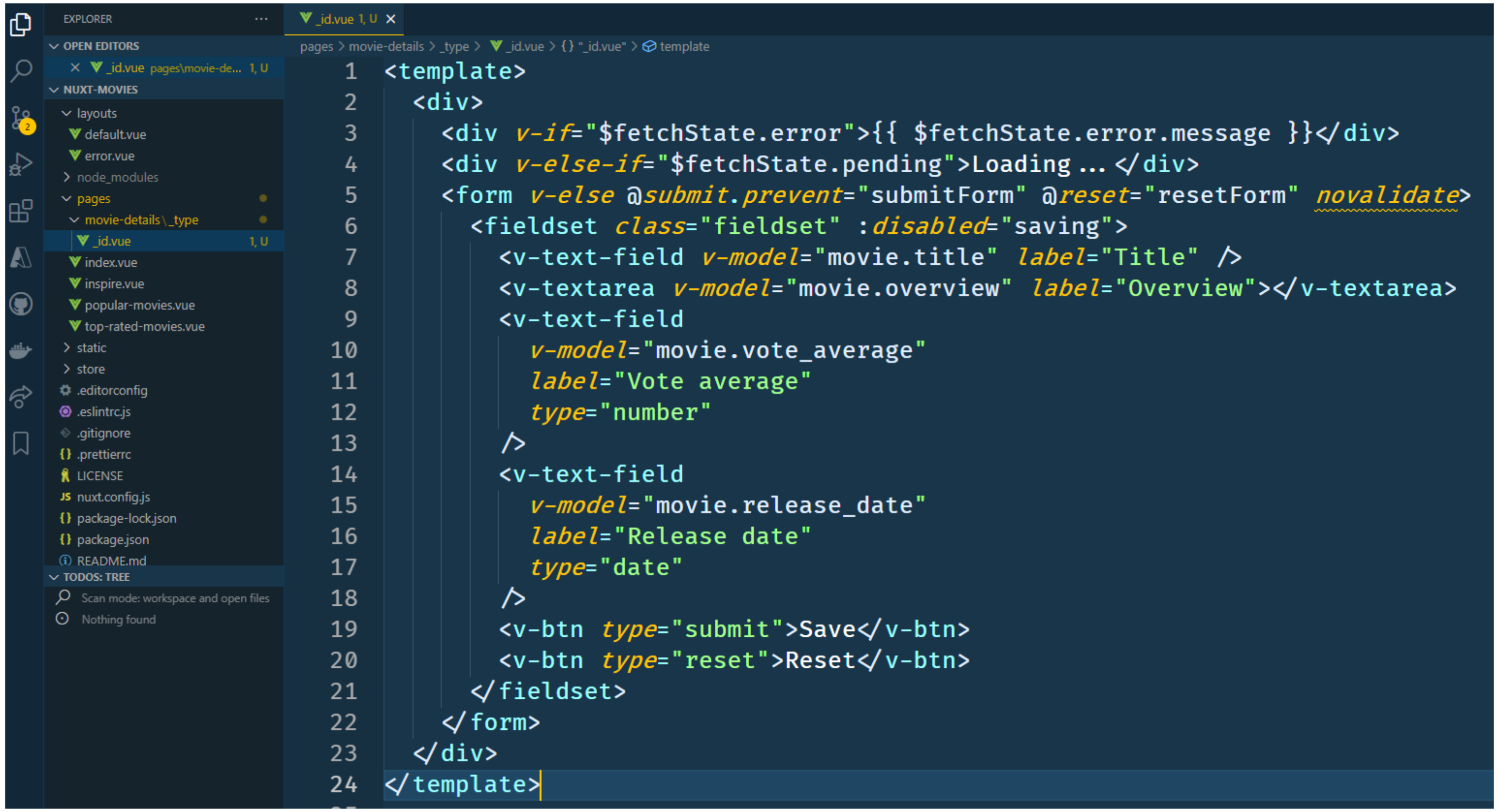
```
<template>
  <v-card elevation="3" class="mx-auto my-3" max-width="500">
    <nuxt-link
      :to="{ name: 'movie-details-type-id', params: { id: movie.id, type } }"
    >
      <v-img :src="imageUrl" height="281" width="500" />
      {{ movie.title }}
      <v-rating
        :value="movie.vote_average / 2"
        :title="movie.vote_average"
        readonly
        size="x-small"
      />
    </nuxt-link>
  </v-card>
</template>
```

Movie Details

Movie Details

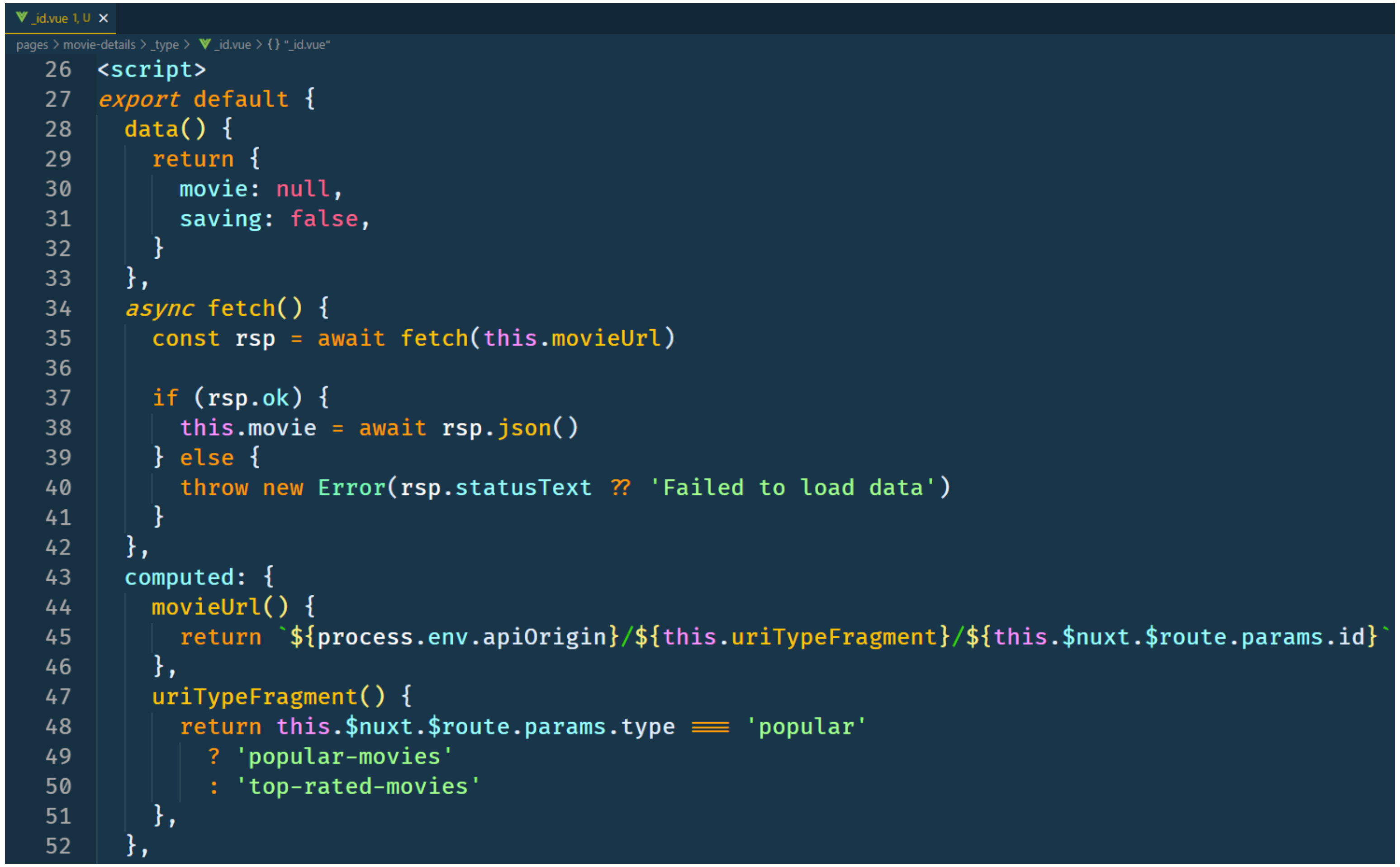
- Use **nested folders** to specify the route structure
  - Use an _ to indicate route params
- Use Vuetify to create a simple form

pages/movie-details/
_type/_id.vue



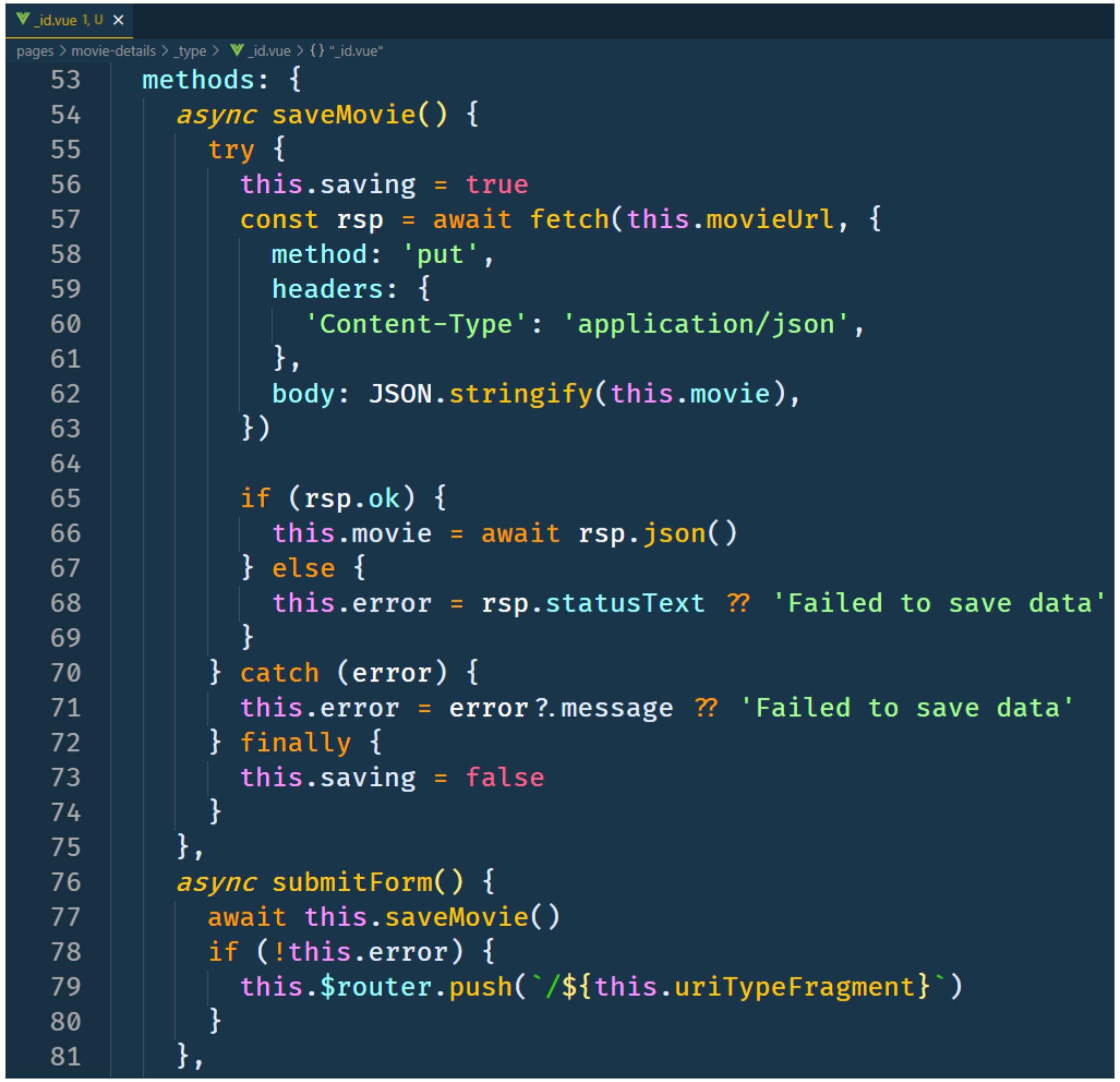
```
<template>
  <div>
    <div v-if="$fetchState.error">{{ $fetchState.error.message }}</div>
    <div v-else-if="$fetchState.pending">Loading ...</div>
    <form v-else @submit.prevent="submitForm" @reset="resetForm" novalidate>
      <fieldset class="fieldset" :disabled="saving">
        <v-text-field v-model="movie.title" label="Title" />
        <v-textarea v-model="movie.overview" label="Overview"></v-textarea>
        <v-text-field
          v-model="movie.vote_average"
          label="Vote average"
          type="number"
        />
        <v-text-field
          v-model="movie.release_date"
          label="Release date"
          type="date"
        />
        <v-btn type="submit">Save</v-btn>
        <v-btn type="reset">Reset</v-btn>
      </fieldset>
    </form>
  </div>
</template>
```

pages/movie-details/ _type/_id.vue



```
<script>
export default {
  data() {
    return {
      movie: null,
      saving: false,
    }
  },
  async fetch() {
    const rsp = await fetch(this.movieUrl)

    if (rsp.ok) {
      this.movie = await rsp.json()
    } else {
      throw new Error(rsp.statusText ?? 'Failed to load data')
    }
  },
  computed: {
    movieUrl() {
      return `${process.env.apiOrigin}/${this.uriTypeFragment}/${this.$nuxt.$route.params.id}`
    },
    uriTypeFragment() {
      return this.$nuxt.$route.params.type === 'popular'
        ? 'popular-movies'
        : 'top-rated-movies'
    },
  },
```

pages/movie-details/_type/_id.vue



```
  methods: {
    async saveMovie() {
      try {
        this.saving = true
        const rsp = await fetch(this.movieUrl, {
          method: 'put',
          headers: {
            'Content-Type': 'application/json',
          },
          body: JSON.stringify(this.movie),
        })

        if (rsp.ok) {
          this.movie = await rsp.json()
        } else {
          this.error = rsp.statusText ?? 'Failed to save data'
        }
      } catch (error) {
        this.error = error?.message ?? 'Failed to save data'
      } finally {
        this.saving = false
      }
    },
    async submitForm() {
      await this.saveMovie()
      if (!this.error) {
        this.$router.push(`/${this.uriTypeFragment}`)
      }
    },
```

Conclusion

- Slots can make reusable components more flexible
  - The API has been changed since Vue 2
- Extracting global state using VueX can be helpful
  - Don’t extract local component state to VueX
- The Vue Router is very flexible
  - Use so the end user can use deep linking
  - Use route guards to ensure authentication
- Libraries like Vuetify can speed UI development
  - But most are not released for Vue 3 yet
- Using TypeScript can make code more reliable
  - But templates are not validated
- Using Nuxt can have a lot of benefits
  - A framework build on top of a framework
  - More productive and consistent

Maurice de Beijer

@mauricedb

maurice.de.beijer
@gmail.com