

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](https://twitter.com/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!"

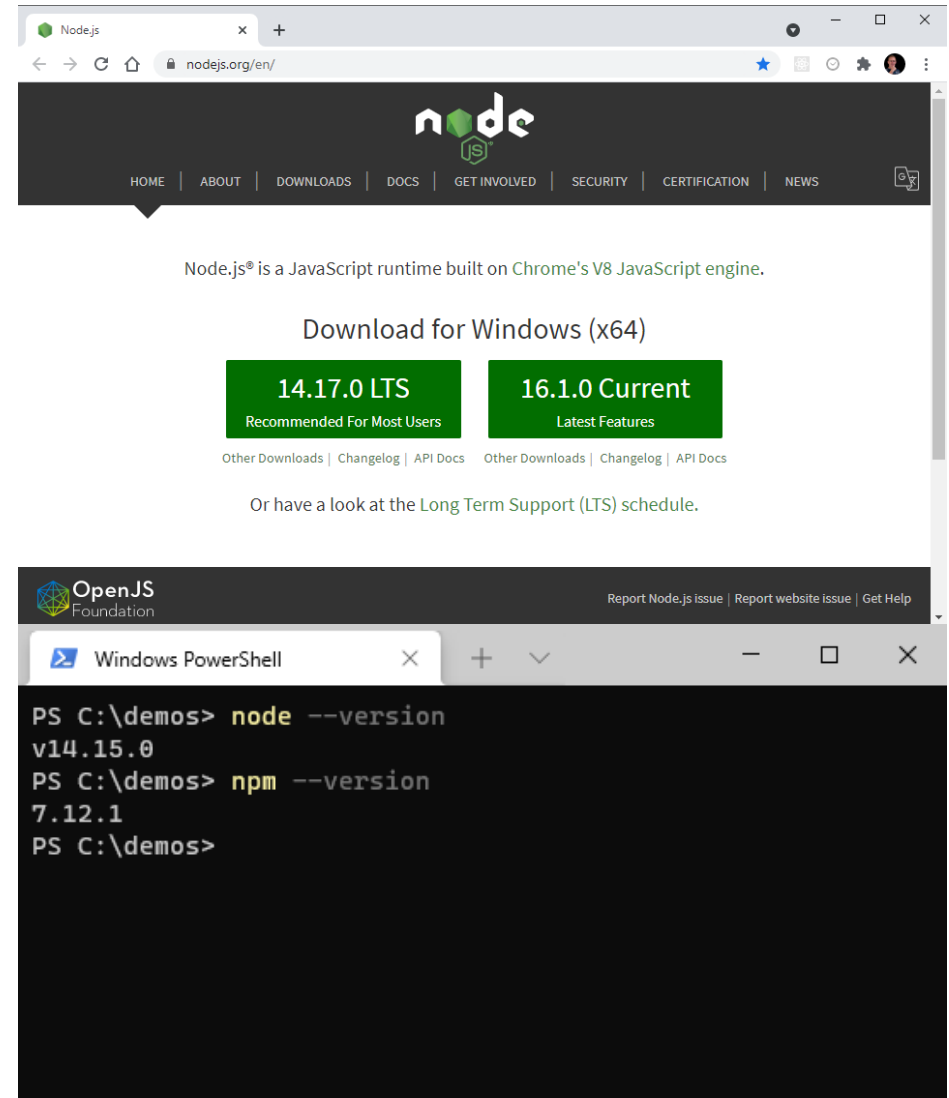
Prerequisites

Install Node, NPM & Vue CLI

Install the GitHub repository

Install Node.js & NPM

- Minimal:
 - Node version 14
 - NPM version 7



Vue CLI

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

Clone the GitHub Repository

```
Windows PowerShell
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> |
```

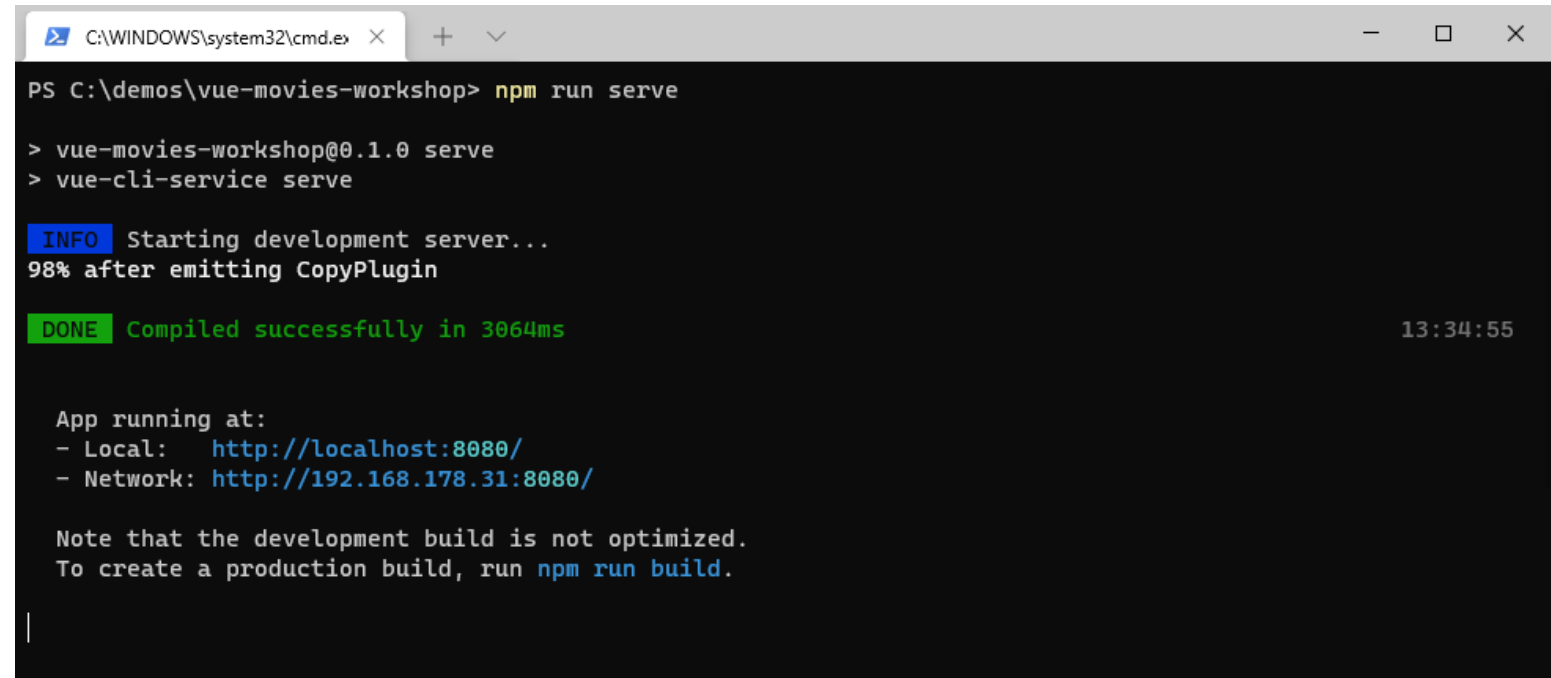


Checkout branch & install NPM packages

```
Windows PowerShell
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



```
C:\WINDOWS\system32\cmd.exe X + v
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



```
JS vue.config.js U x
JS vue.config.js > ...
1 // vue.config.js
2 module.exports = {
3   chainWebpack: (config) => {
4     config.resolve.alias.set("vue", "@vue/compat");
5
6     config.module
7       .rule("vue")
8       .use("vue-loader")
9       .tap((options) => {
10         return {
11           ... options,
12           compilerOptions: {
13             compatConfig: {
14               MODE: 2,
15             },
16           },
17         };
18       });
19   },
20 };
```

- 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 x
src > components > LabeledInput.vue > {} "LabeledInput.vue" > style

You, 19 hours ago | 1 author (You)
1 <template>
2   <div>
3     <label class="input-label" :for="$attrs.id">
4       {{ label }}
5     <slot />
6   </label>
7   <input
8     type="text"
9     class="form-control"
10    :value="modelValue"
11    @input="$emit('update:modelValue', $event.target.value)"
12    v-bind="$attrs"
13  />
14   <slot name="below" />
15 </div>
16 </template>
```

MovieDetails.vue



```
MovieDetails.vue x
src > views > MovieDetails.vue > {} "MovieDetails.vue" > template > div > form > fieldset.fieldset
You, 19 hours ago | 1 author (You)
1 <template>
2   <div>
3     <div v-if="error">{{ error }}</div>
4     <div v-else-if="loading">Loading ... </div>
5     <form v-else @submit.prevent="submitForm" @reset="resetForm" novalidate>
6       <fieldset class="fieldset" :disabled="saving">
7         <LabeledInput v-model="movie.title">
8           <b>Title</b>
9           <template v-slot:below v-if="!movie.title">
10            <div class="warning">The movie title is required</div>
11          </template>
12        </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



```
JS index.js A x
src > store > JS index.js > ...
You, seconds ago | 1 author (You)
1 import { createStore } from "vuex";
2
3 export default createStore({
4   state: {},
5   mutations: {},
6   actions: {},
7   modules: {},
8 });

JS main.js M x
src > JS main.js
You, seconds ago | 1 author (You)
1 import { createApp } from "vue";
2 import App from "./App.vue";
3 import router from "./router";
4 import store from "./store";
5
6 createApp(App).use(store).use(router).mount("#app");
```



Vuex Mutations

Synchronous updates

Vuex 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 x
src > store > JS index.js > ...
You, seconds ago | 1 author (You)
1 import { createStore } from "vuex";
2
3 const store = createStore({
4   state: {
5     now: new Date(),
6   },
7   mutations: {
8     tick(state) {
9       state.now = new Date();
10    },
11  },
12  actions: {},
13  modules: {},
14 });
15
16 setInterval(() => store.commit("tick"), 1000);
17
18 export default store;
```


Clock.vue



```
▼ Clock.vue M x
src > components > ▼ Clock.vue > {} "Clock.vue"

You, seconds ago | 1 author (You)
1 <template>
2   <span>{{ time }}</span>
3 </template>
4
5 <script>
6   import { computed } from "vue";
7   import { useStore } from "vuex";
8
9   export default {
10    setup() {
11      const store = useStore();
12      const time = computed(() => store.state.now.toLocaleTimeString());
13
14      return {
15        time,
16      };
17    },
18  };
19 </script>
```



Vuex Actions

Asynchronous updates

Vuex 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)

```
JS index.js M X
src > store > JS index.js > ...
20   actions: {
21     async fetchData(ctx, payload) {
22       if (!ctx.state[payload.key]) {
23         ctx.commit("dataLoading", payload);
24
25         try {
26           const rsp = await fetch(payload.url);
27
28           if (rsp.ok) {
29             ctx.commit("dataLoaded", {
30               data: await rsp.json(),
31               key: payload.key,
32             });
33           }
34         } catch (error) {
35           console.log(error?.message ?? "Failed to load data");
36         }
37       }
38     },
39   },
40   modules: {},
41 });
```

Store (2/2)

```
JS index.js M x
src > store > JS index.js > ...
You, seconds ago | 1 author (You)
1 import { createStore } from "vuex";
2
3 const store = createStore({
4   state: {
5     now: new Date(),
6   },
7   mutations: {
8     tick(state) {
9       state.now = new Date();
10    },
11    dataLoading(state, payload) {
12      state[payload.key] = null;
13      state[`${payload.key}Loading`] = true;
14    },
15    dataLoaded(state, payload) {
16      state[payload.key] = payload.data;
17      state[`${payload.key}Loading`] = false;
18    },
19  },
20 });
```

MovieList.vue



```
▼ MovieList.vue M x
src > views > ▼ MovieList.vue > {} "MovieList.vue"
20 import { computed, watchEffect } from "vue";
21 import { useStore } from "vuex";
22 import MovieCard from "../components/MovieCard.vue";
23 import { toRefs } from "@vue/reactivity";
24
25 export default {
26 >   components: { ...
28   },
29 >   props: { ...
42   },
43   setup(props) {
44     const store = useStore();
45     const url = toRefs(props).moviesUrl;
46     const key = toRefs(props).type;
47     watchEffect(() => {
48       store.dispatch("fetchData", { url: url.value, key: key.value });
49     });
50
51     const movies = computed(() => store.state[key.value]);
52     const error = computed(() => store.state[`${key.value}Error`]);
53     const loading = computed(() => store.state[`${key.value}Loading`]);
54
55     return {
56       error,
57       loading,
58       movies,
59     };
60   },
61 };

```



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
- **Bundle splitting** per route
- Use **Vue Router version 4** with Vue 3
 - Vue Router 3 is for Vue 2 😞

createRouter()

```
JS index.js x
src > router > JS index.js > ...
6  const routes = [
7    { ...
11  },
12  {
13    path: "/top-rated-movies",
14    name: "TopRatedMovies",
15    component: MovieList,
16    props: {
17      moviesUrl: `${process.env.VUE_APP_API_ORIGIN}/top-rated-movies`,
18      title: "Top Rated Movies",
19      type: "top-rated",
20    },
21  },
22  { ...
31  },
32  {
33    path: "/movie-details/:type/:id",
34    name: "MovieDetails",
35    component: MovieDetails,
36    props: true,
37  },
38  { ...
46  },
47  ];
48
49  const router = createRouter({
50    history: createWebHistory(process.env.BASE_URL),
51    routes,
52  });
```

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 `` 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 x
src > components > MovieCard.vue > {} "MovieCard.vue" > script > default
You, 2 days ago | 1 author (You)
1 <template>
2   <div>
3     <router-link :to="{ name: 'MovieDetails', params: { id: movie.id, type } }">
4       {{ movie.title }}
5       <span :title="movie.vote_average"> ({{ voteAsStars }}) </span>
6     </router-link>
7   </div>
8 </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

```
JS index.js M X
src > router > JS index.js > ...
6  const routes = [
7  > { ...
11  },
12 > { ...
21  },
22 > { ...
31  },
32  {
33    path: "/movie-details/:type/:id",
34    name: "MovieDetails",
35    component: MovieDetails,
36    props: true,
37    beforeEnter(to, from) {
38      const movieId = +to.params.id;
39      // Only navigate to movies with an even ID
40      return movieId % 2 === 0;
41    },
42  },
43 > { ...
51  },
52 ];
```

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

```
JS vuetify.js x
src > plugins > JS vuetify.js > ...
You, 3 hours ago | 1 author (You)
1 import "@mdi/font/css/materialdesignicons.css";
2 import "vuetify/lib/styles/main.sass";
3 import { createVuetify } from "vuetify";
4 import * as components from "vuetify/lib/components";
5 import * as directives from "vuetify/lib/directives";
6
7 export default createVuetify({
8   components,
9   directives,
10 });
```

main.js



```
JS main.js  X
src > JS main.js > ...
You, 3 hours ago | 1 author (You)
1  import { createApp } from "vue";
2  import vuetify from "./plugins/vuetify";
3  import App from "./App.vue";
4  import router from "./router";
5  import store from "./store";
6
7  const app = createApp(App);
8  app.use(router);
9  app.use(store);
10 app.use(vuetify);
11
12 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

```
MovieCard.vue M x
src > components > MovieCard.vue > {} "MovieCard.vue"
1 <template>
2   <v-card elevation="3" class="mx-auto my-3" max-width="500">
3     <router-link :to="{ name: 'MovieDetails', params: { id: movie.id, type } }">
4       <v-img :src="imageUrl" height="281" width="500" />
5       <v-card-title>
6         {{ movie.title }}
7         <span :title="movie.vote_average"> ({{ voteAsStars }}) </span>
8       </v-card-title>
9     </router-link>
10  </v-card>
11 </template>
12
13 <script>
14   export default {
15     props: {
16       movie: Object,
17       type: String,
18     },
19     computed: {
20       voteAsStars() {
21         return "".padEnd(this.movie.vote_average / 2, "★").padEnd(5, "☆");
22       },
23       imageUrl() {
24         return "https://image.tmdb.org/t/p/w500" + this.movie.backdrop_path;
25       },
26     },
27   };
28 </script>
```


MovieList.vue



```
MovieList.vue M x
src > views > MovieList.vue > {} "MovieList.vue"
You, seconds ago | 1 author (You)
1 <template>
2   <div>
3     <h2 class="text-center">{{ title }}</h2>
4     <div>
5       <div v-if="error">{{ error }}</div>
6       <div v-else-if="loading">Loading ... </div>
7       <v-row v-else>
8         <MovieCard
9           v-for="movie in movies"
10          :key="movie.id"
11          :movie="movie"
12          :type="type"
13        />
14      </v-row>
15    </div>
16  </div>
17 </template>
```



App Bar

App Bar

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

App.vue



```
App.vue M x
src > App.vue > {} "App.vue"
1 <template>
2   <v-app>
3     <v-app-bar>
4       <v-app-bar-title>Movies</v-app-bar-title>
5       <router-link to="/">
6         <v-btn text>Home</v-btn>
7       </router-link>
8       <router-link to="/top-rated-movies">
9         <v-btn text>Top Rated Movies</v-btn>
10      </router-link>
11      <router-link to="/popular-movies">
12        <v-btn text>Popular Movies</v-btn>
13      </router-link>
14      <router-link to="/about">
15        <v-btn text>About</v-btn>
16      </router-link>
17      <v-spacer></v-spacer>
18      <v-btn text>
19        <Clock />
20      </v-btn>
21    </v-app-bar>
22    <nav id="nav"></nav>
23
24    <v-main id="main">
25      <v-container>
26        <router-view />
27      </v-container>
28    </v-main>
29  </v-app>
30 </template>
```



Movie Rating

Movie Rating

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

MovieCard.vue



```
MovieCard.vue M x
src > components > MovieCard.vue > {} "MovieCard.vue"
1 <template>
2   <v-card elevation="3" class="mx-auto my-3" max-width="500">
3     <router-link :to="{ name: 'MovieDetails', params: { id: movie.id, type } }">
4       <v-img :src="imageUrl" height="281" width="500" />
5       <v-card-title>
6         {{ movie.title }}
7         <v-rating
8           :model-value="movie.vote_average / 2"
9           :title="movie.vote_average"
10            readonly
11            size="x-small"
12          />
13       </v-card-title>
14     </router-link>
15   </v-card>
16 </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`

vue add typescript

```
Windows PowerShell
PS C:\demos\vue-movies-workshop> vue add typescript

Installing @vue/cli-plugin-typescript...

added 59 packages in 6s

2 packages are looking for funding
  run `npm fund` for details
✓ Successfully installed plugin: @vue/cli-plugin-typescript

? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JSX)? Yes
? Convert all .js files to .ts? Yes
? Allow .js files to be compiled? No
? Skip type checking of all declaration files (recommended for apps)? Yes

🔧 Invoking generator for @vue/cli-plugin-typescript...
📦 Installing additional dependencies...

added 38 packages in 8s

2 packages are looking for funding
  run `npm fund` for details
- Running completion hooks...error: Duplicate key 'extends' (no-dupe-keys) at .eslintrc.js:38:3:
  36 |   ],
  37 |
> 38 |   extends: [
     |         ^
  39 |     "plugin:vue/essential",
  40 |     "eslint:recommended",
  41 |     "@vue/prettier",

1 error found.
PS C:\demos\vue-movies-workshop> |
```

store/index.ts

```
TS index.ts M X
src > store > TS index.ts > ...

You, seconds ago | 1 author (You)
1 import { createStore } from "vuex";
2
You, seconds ago | 1 author (You)
3 interface Store {
4   now: Date;
5   [key: string]: any;
6 }
7
8 const store = createStore<Store>({
9   state: {
10     now: new Date(),
11     dataLoading: null,
12   },
13   mutations: {
14     tick(state) {
15       state.now = new Date();
16     },
17     dataLoading(state, payload) {
18       state[payload.key] = null;
19       state[`${payload.key}Loading`] = true;
20     },

```

router/index.ts

```
TS index.ts M x
src > router > TS index.ts > routes
1 import {
2   createRouter,
3   createWebHistory,
4   RouteLocationNormalized,
5 } from "vue-router";
6 import Home from "../views/Home.vue";
7 import MovieList from "../views/MovieList.vue";
8 import MovieDetails from "../views/MovieDetails.vue";
9
10 const routes = [
11 >   { ...
15   },
16 >   { ...
25   },
26 >   { ...
35   },
36   {
37     path: "/movie-details/:type/:id",
38     name: "MovieDetails",
39     component: MovieDetails,
40     props: true,
41     beforeEnter(to: RouteLocationNormalized, from: RouteLocationNormalized) {
42       const movieId = +to.params.id;
43       // Only navigate to movies with an even ID
44       // return movieId % 2 === 0;
45     },
46   },

```

useFetchData.ts



```
TS useFetchData.ts M X
src > composables > TS useFetchData.ts > ...
1 import { ComputedRef, ref, watchEffect } from "vue";
2
3 export default function useFetchData(url: ComputedRef<string>) {
4   const error = ref<string | null>(null);
5   const loading = ref(true);
6   const data = ref(null);
7
8   async function fetchData() {
9     try {
10      const rsp = await fetch(url.value);
11
12      if (rsp.ok) {
13        data.value = await rsp.json();
14      } else {
15        error.value = rsp.statusText ?? "Failed to load data";
16      }
17    } catch (error) {
18      error.value = error?.message ?? "Failed to load data";
19    } finally {
20      loading.value = false;
21    }
22  }
}
```



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



```
▼ Clock.vue M x
src > components > ▼ Clock.vue > {} "Clock.vue"
1 <template>
2   <span>{{ time }}</span>
3 </template>
4
5 <script lang="ts">
6   import { computed } from "vue";
7   import { useStore } from "vuex";
8   import { Store } from "../store";
9
10  export default {
11    setup() {
12      const store = useStore<Store>();
13      const time = computed(() => store.state.now.toLocaleTimeString());
14
15      return {
16        time,
17      };
18    },
19  };
20 </script>
```

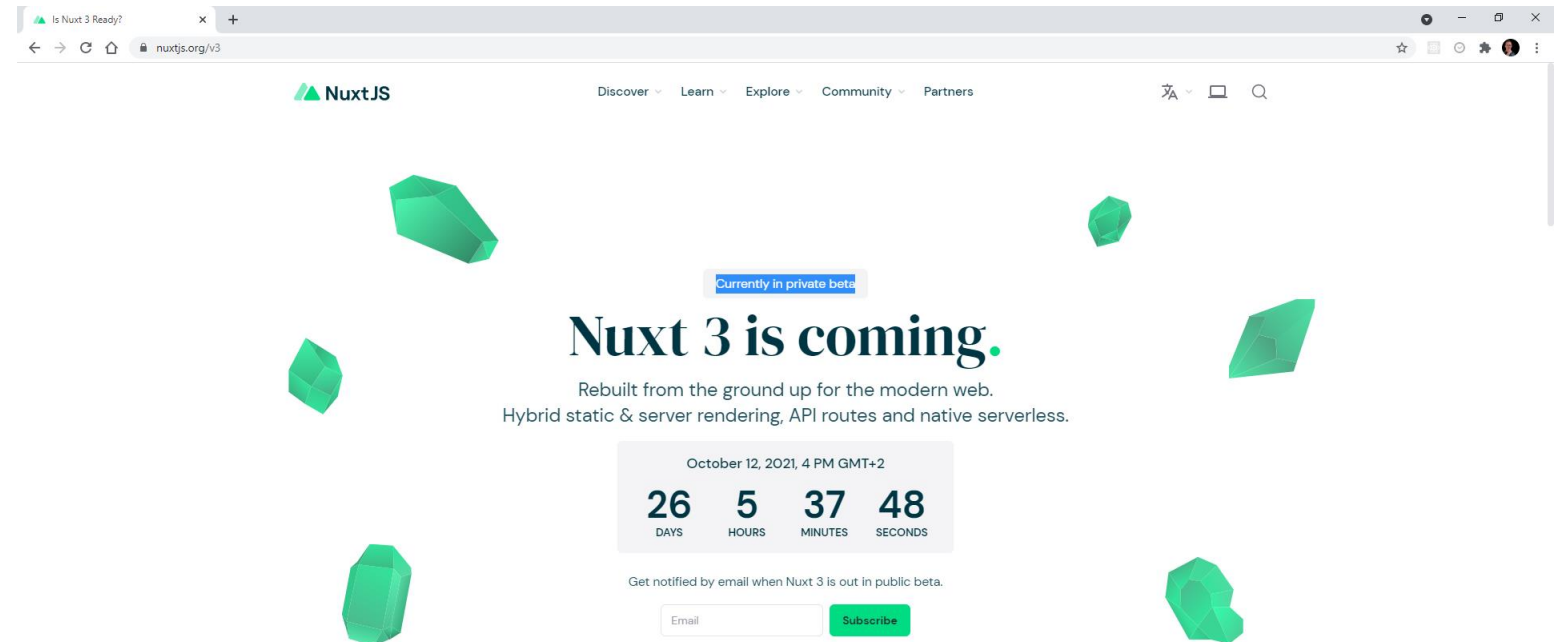



Getting started with Nuxt.js

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



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 x
pages > popular-movies.vue > {} "popular-movies.vue"
You, a day ago | 1 author (You)
1 <template>
2   <div>
3     Top Rated Movies
4     <MovieList :movies-url="moviesUrl" title="Popular Movies" type="popular" />
5   </div>
6 </template>
7
8 <script>
9   import MovieList from '../components/MovieList.vue'
10  export default {
11    components: { MovieList },
12    computed: {
13      moviesUrl() {
14        return `${process.env.apiOrigin}/popular-movies`
15      },
16    },
17  }
18 </script>
```

MovieList.vue

```
MovieList.vue x
components > MovieList.vue > {} "MovieList.vue"
You, a day ago | 1 author (You)
1 <template>
2   <div>
3     <h2>{{ title }}</h2>
4     <div>
5       <div v-if="error">{{ error }}</div>
6       <div v-else-if="loading">Loading ... </div>
7       <v-row v-else>
8         <MovieCard
9           v-for="movie in movies"
10          :key="movie.id"
11          :movie="movie"
12          :type="type"
13        />
14      </v-row>
15    </div>
16  </div>
17 </template>
18
19 <script>
20 import MovieCard from './MovieCard.vue'
21
22 export default {
23   components: {
24     MovieCard,
25   },
```

MovieList.vue

```
▼ MovieList.vue X
components > ▼ MovieList.vue > {} "MovieList.vue"
26 props: {
27   moviesUrl: {
28     type: String,
29     required: true,
30   },
31   title: {
32     type: String,
33     required: true,
34   },
35   type: {
36     type: String,
37     required: true,
38   },
39 },
40 data() {
41   return {
42     error: null,
43     loading: true,
44     movies: [],
45   }
46 },
47 watch: {
48   moviesUrl() {
49     this.fetchMovies()
50   },
51 },
52 mounted() {
53   this.fetchMovies()
54 },
```

MovieList.vue



```
MovieList.vue X
components > ▼ MovieList.vue > {} "MovieList.vue"
55   methods: {
56     async fetchMovies() {
57       try {
58         const rsp = await fetch(this.moviesUrl)
59
60         if (rsp.ok) {
61           this.movies = await rsp.json()
62         } else {
63           this.error = rsp.statusText ?? 'Failed to load data'
64         }
65       } catch (error) {
66         this.error = error?.message ?? 'Failed to load data'
67       } finally {
68         this.loading = false
69       }
70     },
71   },
72 }
73 </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

```
MovieList.vue x
components > ▼ MovieList.vue > {} "MovieList.vue"
19 <script>
20 import MovieCard from './MovieCard.vue'
21
22 export default {
23 > components: { ...
25 },
26 > props: { ...
39 },
40 data() {
41   return {
42     movies: [],
43   }
44 },
45 async fetch() {
46   const rsp = await fetch(this.moviesUrl)
47
48   if (rsp.ok) {
49     this.movies = await rsp.json()
50   } else {
51     throw new Error(rsp.statusText ?? 'Failed to load data')
52   }
53 },
54 }
55 </script>
```

MovieList.vue



```
MovieList.vue x
components > MovieList.vue > {} "MovieList.vue"
You, a day ago | 1 author (You)
1 <template>
2   <div>
3     <h2>{{ title }}</h2>
4     <div>
5       <div v-if="$fetchState.error">{{ $fetchState.error.message }}</div>
6       <div v-else-if="$fetchState.pending">Loading ... </div>
7       <v-row v-else>
8         <MovieCard
9           v-for="movie in movies"
10          :key="movie.id"
11          :movie="movie"
12          :type="type"
13        />
14      </v-row>
15    </div>
16  </div>
17 </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

```
top-rated-movies.vue U x
pages > ▼ top-rated-movies.vue > {} "top-rated-movies.vue"
1  <template>
2    <div>
3      Top Rated Movies
4      <MovieList
5        :movies-url="moviesUrl"
6        title="Top Rated Movies"
7        type="top-rated"
8      />
9    </div>
10 </template>
11
12 <script>
13 import MovieList from '../components/MovieList.vue'
14 export default {
15   components: { MovieList },
16   computed: {
17     moviesUrl() {
18       return `${process.env.apiOrigin}/top-rated-movies`
19     },
20   },
21 }
22 </script>
```

default.vue



```
default.vue M X
layouts > default.vue > {} "default.vue"
66 export default {
67   data() {
68     return {
69       clipped: false,
70       drawer: false,
71       fixed: false,
72       items: [
73 >     { ...
77       },
78       {
79         title: 'Top Rated Movies',
80         to: '/top-rated-movies',
81       },
82       {
83         title: 'Popular Movies',
84         to: '/popular-movies',
85       },
```



The MovieCard

The MovieCard

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

MovieCard.vue

```
MovieCard.vue 2, M x
components > MovieCard.vue > {} "MovieCard.vue"
1 <template>
2   <v-card elevation="3" class="mx-auto my-3" max-width="500">
3     <nuxt-link
4       :to="{ name: 'movie-details-type-id', params: { id: movie.id, type } }"
5     >
6       <v-img :src="imageUrl" height="281" width="500" />
7       {{ movie.title }}
8       <v-rating
9         :value="movie.vote_average / 2"
10        :title="movie.vote_average"
11        readonly
12        size="x-small"
13      />
14    </nuxt-link>
15  </v-card>
16 </template>
```

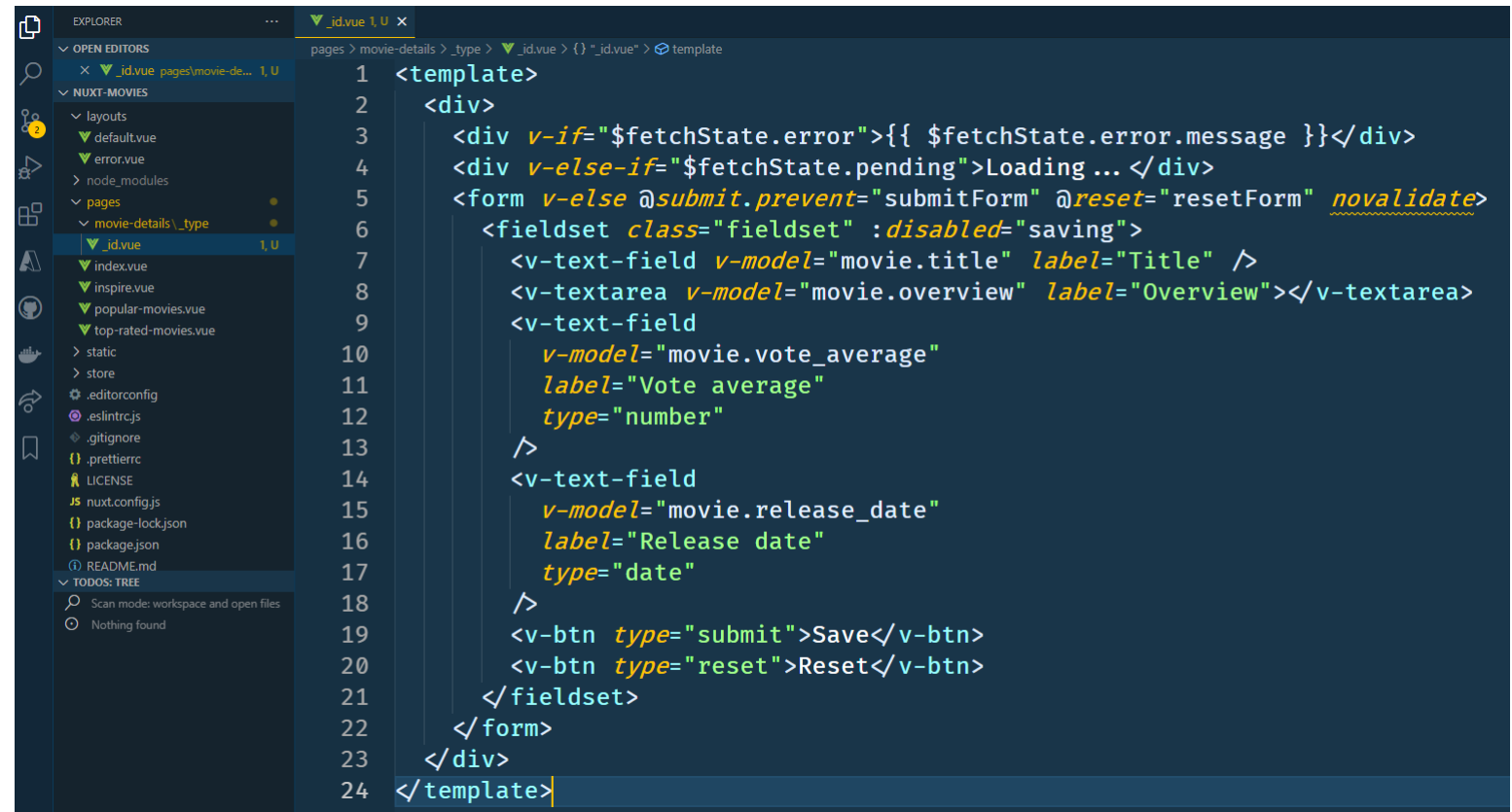


Movie Details

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



The screenshot shows a code editor with a dark theme. On the left, the Explorer sidebar displays a file tree for a project named 'NUXT-MOVIES'. The 'pages' directory is expanded, showing 'movie-details/_type' selected. The main editor area shows the template of the file 'pages/movie-details/_type/_id.vue'. The code is as follows:

```
1 <template>
2   <div>
3     <div v-if="$fetchState.error">{{ $fetchState.error.message }}</div>
4     <div v-else-if="$fetchState.pending">Loading ... </div>
5     <form v-else @submit.prevent="submitForm" @reset="resetForm" novalidate>
6       <fieldset class="fieldset" :disabled="saving">
7         <v-text-field v-model="movie.title" label="Title" />
8         <v-textarea v-model="movie.overview" label="Overview"></v-textarea>
9         <v-text-field
10           v-model="movie.vote_average"
11           label="Vote average"
12           type="number"
13         />
14         <v-text-field
15           v-model="movie.release_date"
16           label="Release date"
17           type="date"
18         />
19         <v-btn type="submit">Save</v-btn>
20         <v-btn type="reset">Reset</v-btn>
21       </fieldset>
22     </form>
23   </div>
24 </template>
```

pages/movie-details/
_type/_id.vue

```
pages > movie-details > _type > _id.vue > {} "_id.vue"
26 <script>
27 export default {
28   data() {
29     return {
30       movie: null,
31       saving: false,
32     }
33   },
34   async fetch() {
35     const rsp = await fetch(this.movieUrl)
36
37     if (rsp.ok) {
38       this.movie = await rsp.json()
39     } else {
40       throw new Error(rsp.statusText ?? 'Failed to load data')
41     }
42   },
43   computed: {
44     movieUrl() {
45       return `${process.env.apiUrl}/${this.uriTypeFragment}/${this.$nuxt.$route.params.id}`
46     },
47     uriTypeFragment() {
48       return this.$nuxt.$route.params.type === 'popular'
49         ? 'popular-movies'
50         : 'top-rated-movies'
51     },
52   },

```

pages/movie-details/
_type/_id.vue



```
pages > movie-details > _type > ▼ _id.vue > {} "_id.vue"
53  methods: {
54    async saveMovie() {
55      try {
56        this.saving = true
57        const rsp = await fetch(this.movieUrl, {
58          method: 'put',
59          headers: {
60            'Content-Type': 'application/json',
61          },
62          body: JSON.stringify(this.movie),
63        })
64
65        if (rsp.ok) {
66          this.movie = await rsp.json()
67        } else {
68          this.error = rsp.statusText ?? 'Failed to save data'
69        }
70      } catch (error) {
71        this.error = error?.message ?? 'Failed to save data'
72      } finally {
73        this.saving = false
74      }
75    },
76    async submitForm() {
77      await this.saveMovie()
78      if (!this.error) {
79        this.$router.push(`/${this.uriTypeFragment}`)
80      }
81    },

```

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

