Vue Workshop (1/2)

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!"

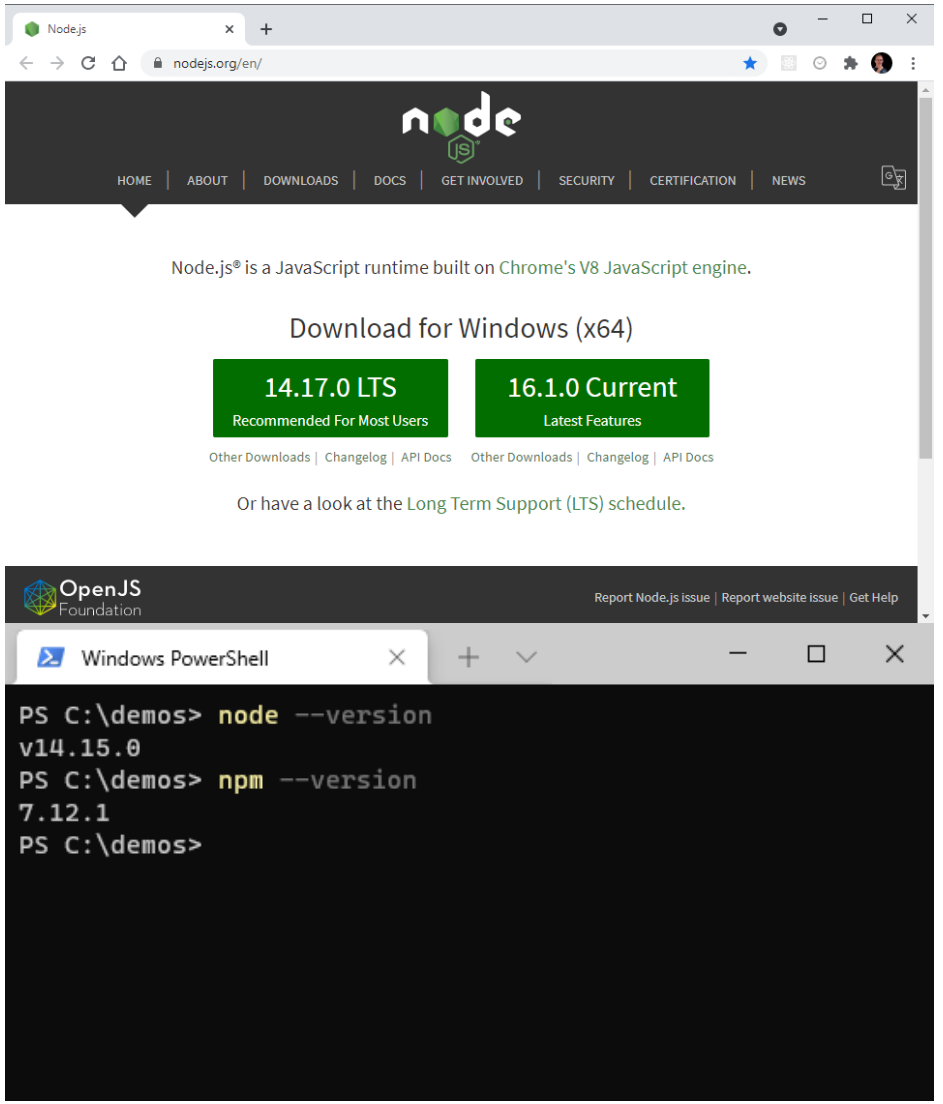
Prerequisites

Install Node & NPM

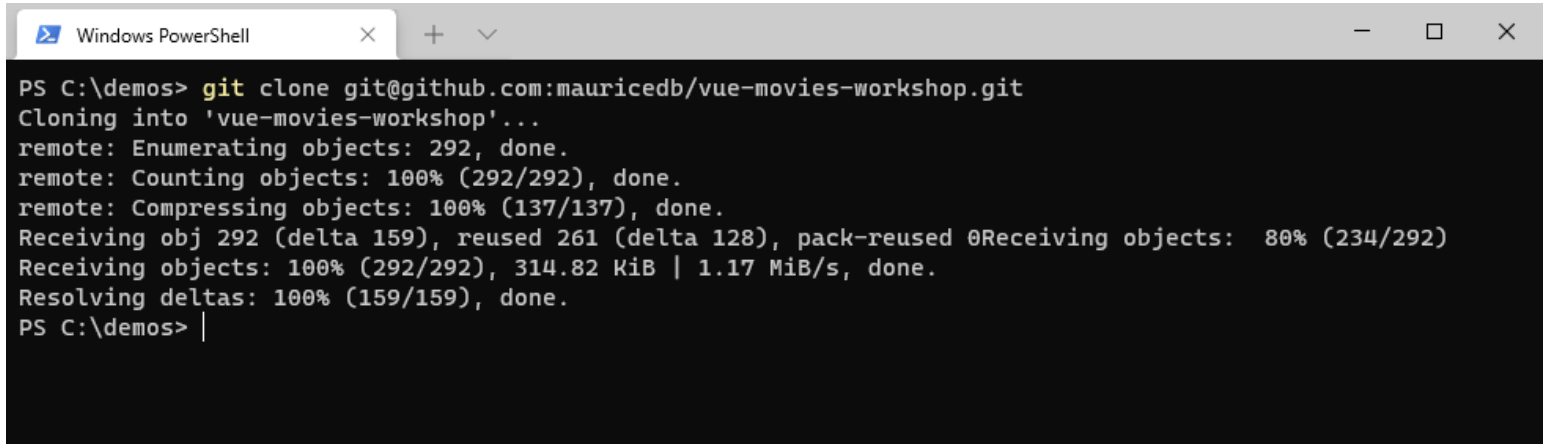
Install the GitHub repository

Install Node.js & NPM

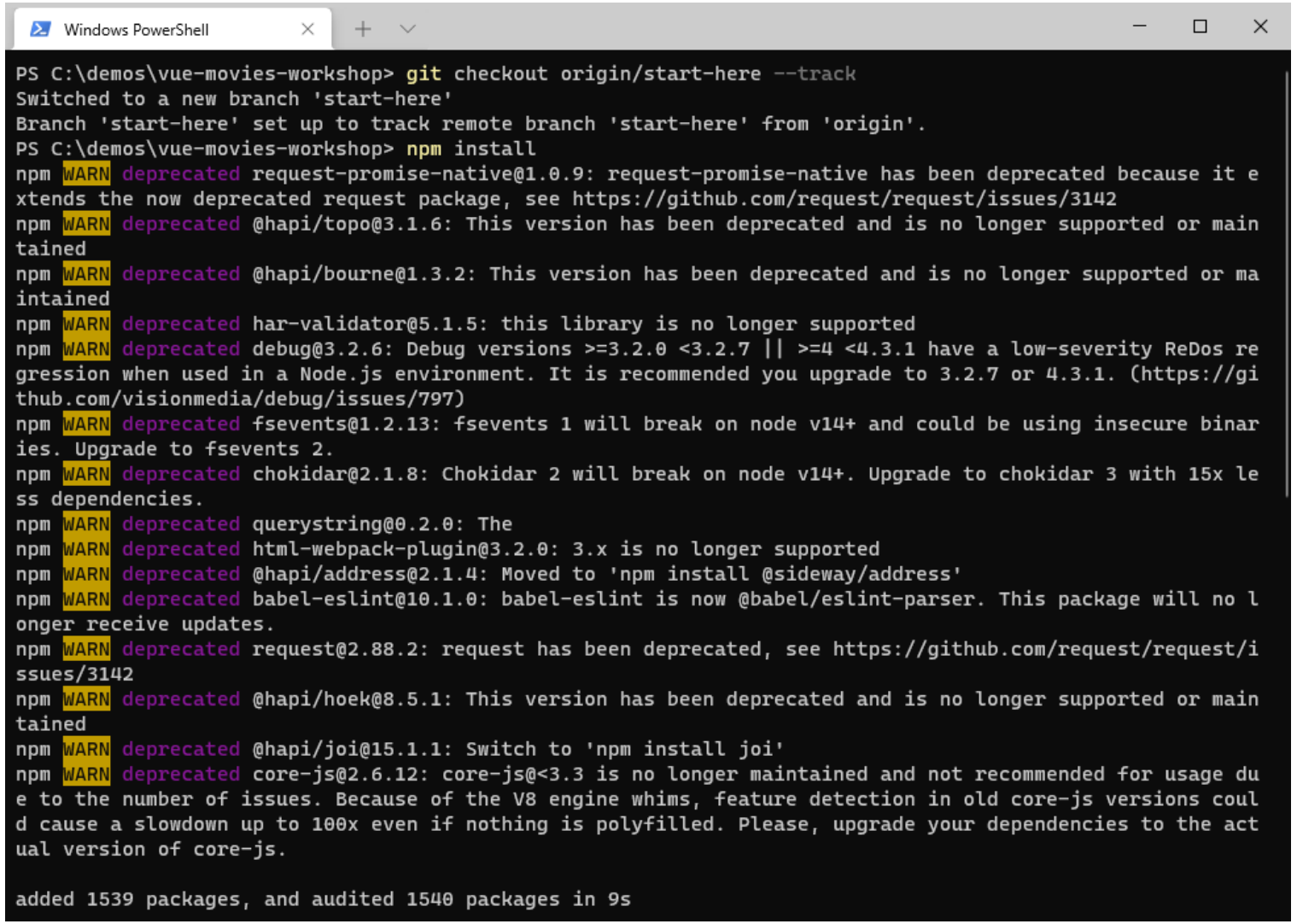
- Minimal:
  - Node version 14
  - NPM version 7



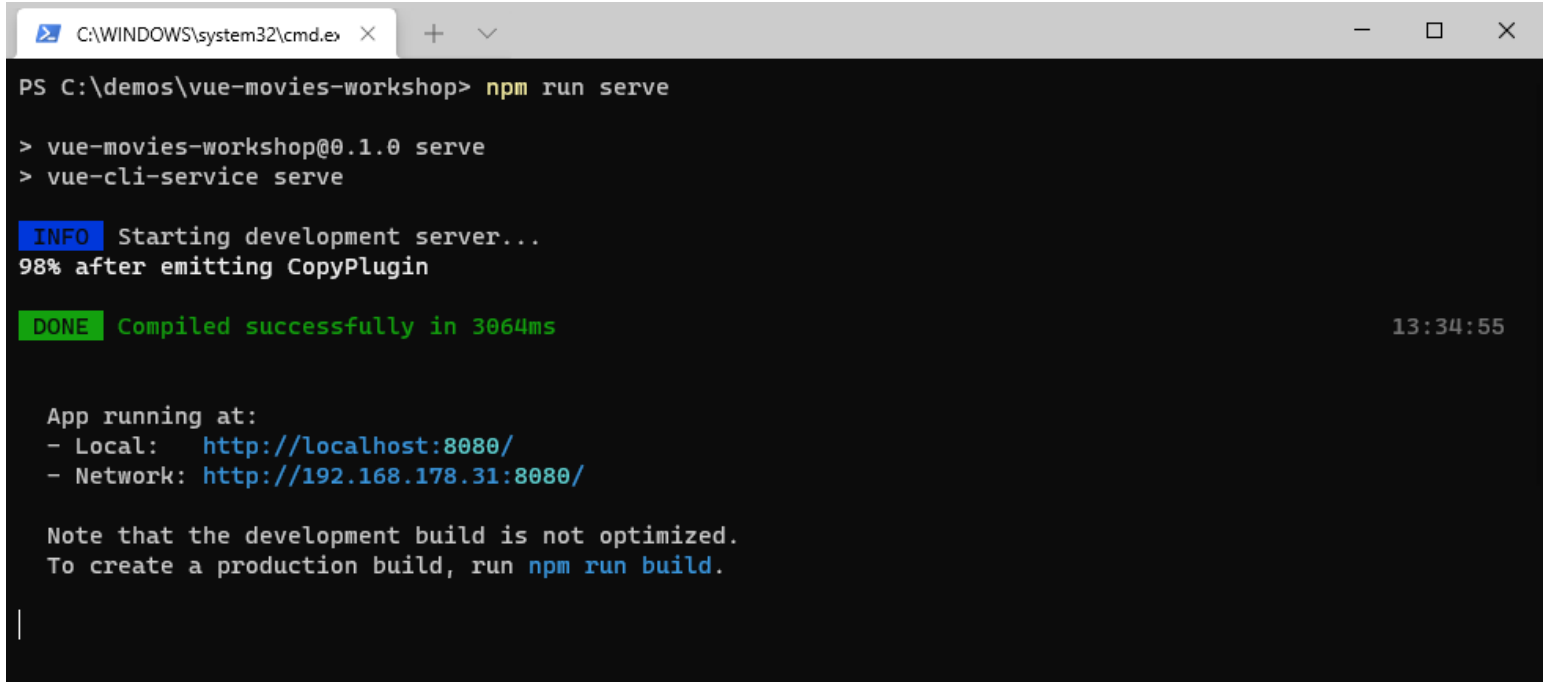
Clone the GitHub Repository



Checkout branch & install NPM packages



Start the application



Following Along


```vue.config.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-1.pdf

From Vue 2 to Vue 3

From Vue 2 to Vue 3

- Vue 3 is **not 100% backward compatible** with Vue2
  - Make sure to check the 3rd party Vue libraries for compatibility
- Mostly **adding new features**
  - Composition API
  - Teleport
  - Multiple root nodes in a template
- But **changing or dropping some features**
  - Application initialization
  - Reactivity
  - Slot naming
  - Filters
- Vue 3 dropped support for Internet Explorer 11

Vue Migration Build

- A **special version** of Vue 3
  - With support for most Vue 2 features

Workflow

- First upgrade **vue-cli**
  - `vue upgrade`
- **Uninstall Vue 2**
  - `npm uninstall vue vue-template-compiler @vue/test-utils`
- **Install Vue 3**
  - `npm install vue@3`
  - `npm install -D @vue/compiler-sfc @vue/test-utils@next`
- Install the **Vue migration build**
  - `npm install @vue/compat`
- Note: There is a **Vue CLI command** that automates this
  - `vue add vue-next`
  - Only works on simple applications

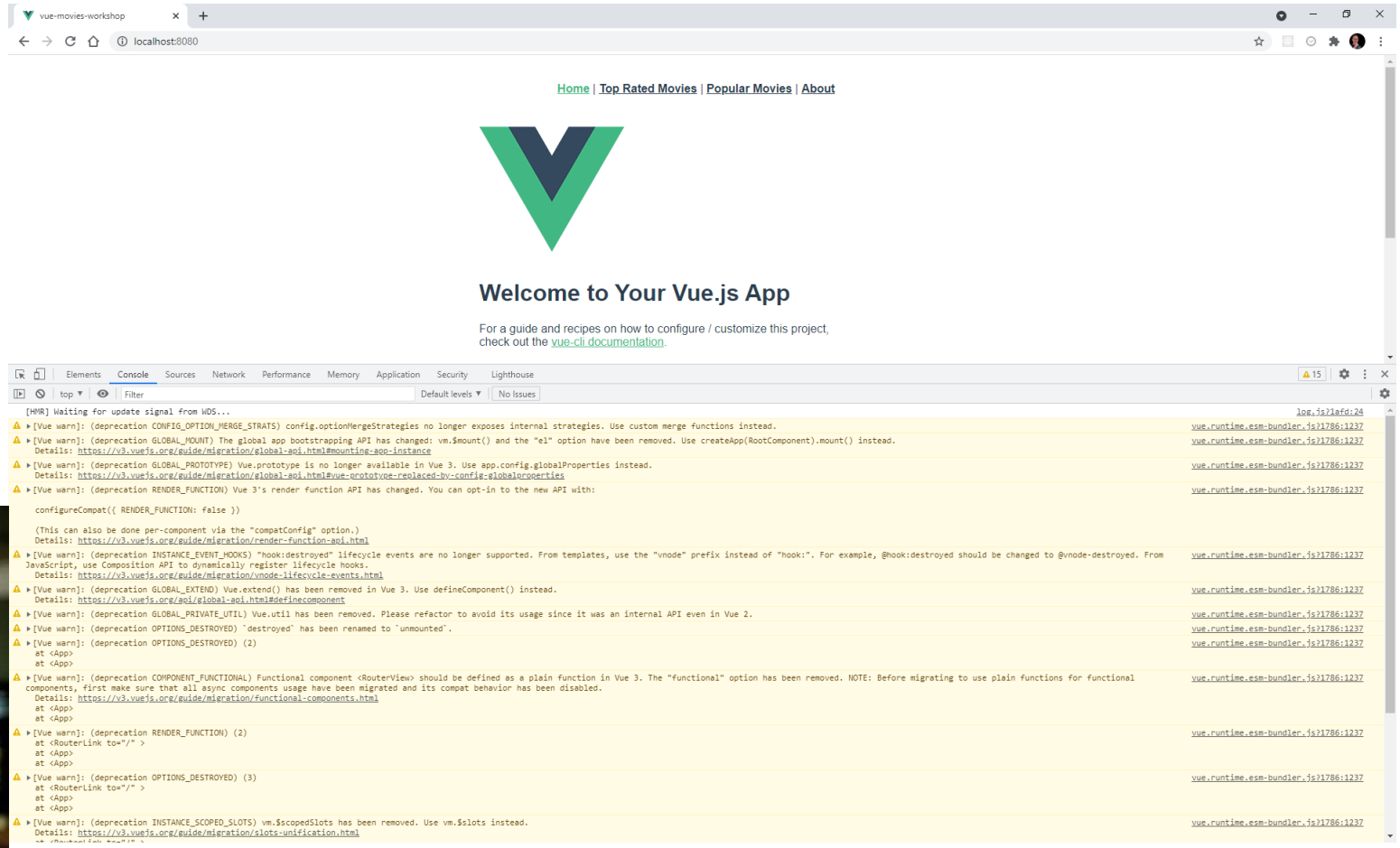
Add/Update vue.config.js

JS vue.config.js U ×

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 };
```

Start the app
npm run serve



Update mount code

main.js

main.js M

src > 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
5  createApp({ ...App, router }).mount("#app");
```

Vue Router 3 to 4

Vue Router 4

- Update **Vue router** to version 4
  - `npm uninstall vue-router`
  - `npm install vue-router@4`

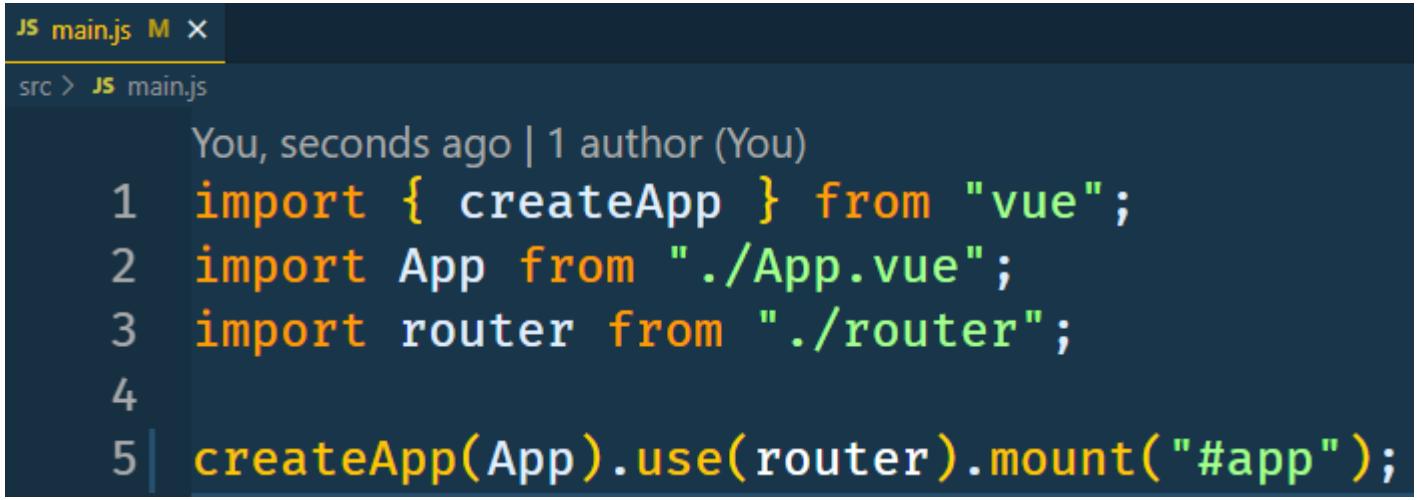
router/index.js

index.js M

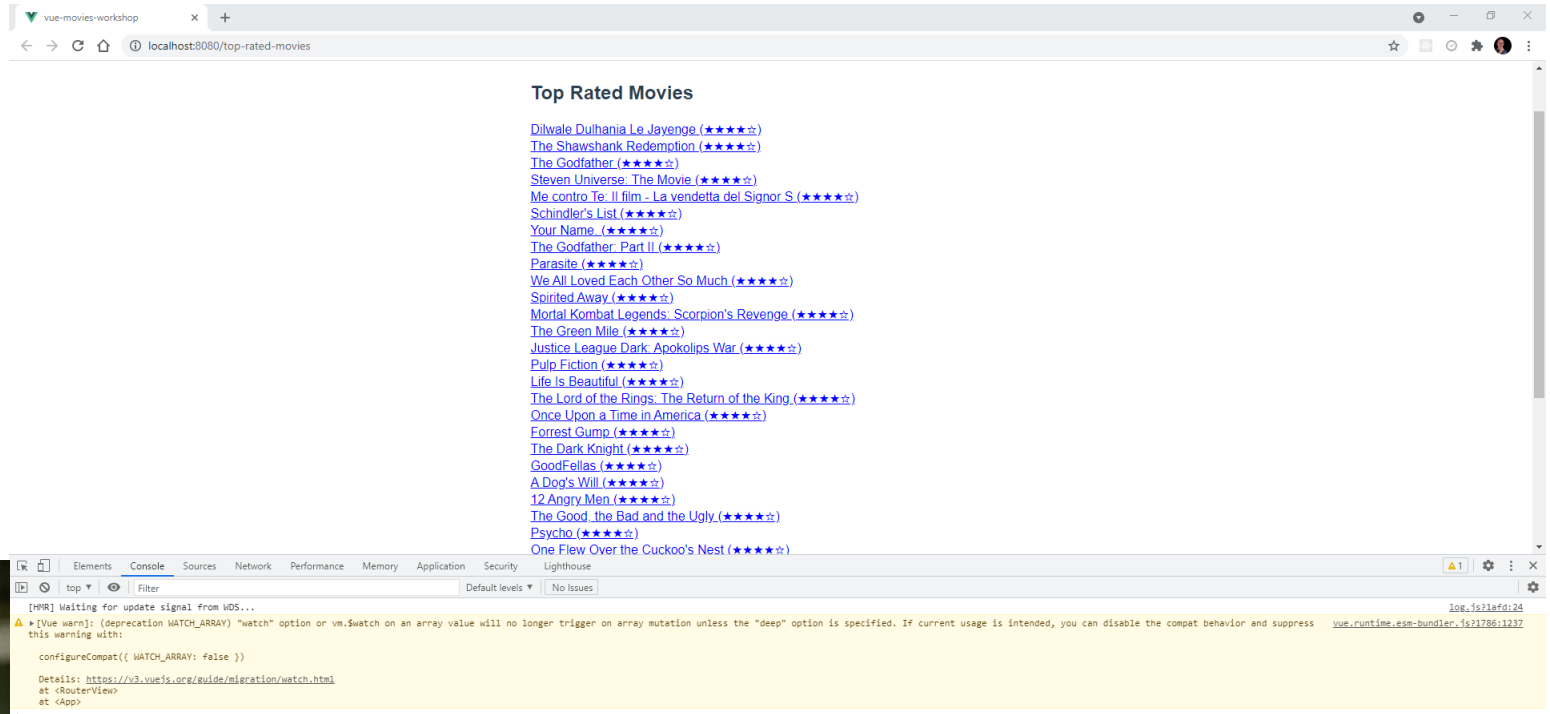
src > router > index.js > ...

```
1  import { createRouter, createWebHistory } from "vue-router";
2  import Home from "../views/Home.vue";
3  import MovieList from "../views/MovieList.vue";
4  import MovieDetails from "../views/MovieDetails.vue";
5
6  const routes = [ ···
47 ];
48
49 const router = createRouter({
50   history: createWebHistory(process.env.BASE_URL),
51   routes,
52 });
53
54 export default router;
```

main.js


main.js M
src > 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
5 createApp(App).use(router).mount("#app");

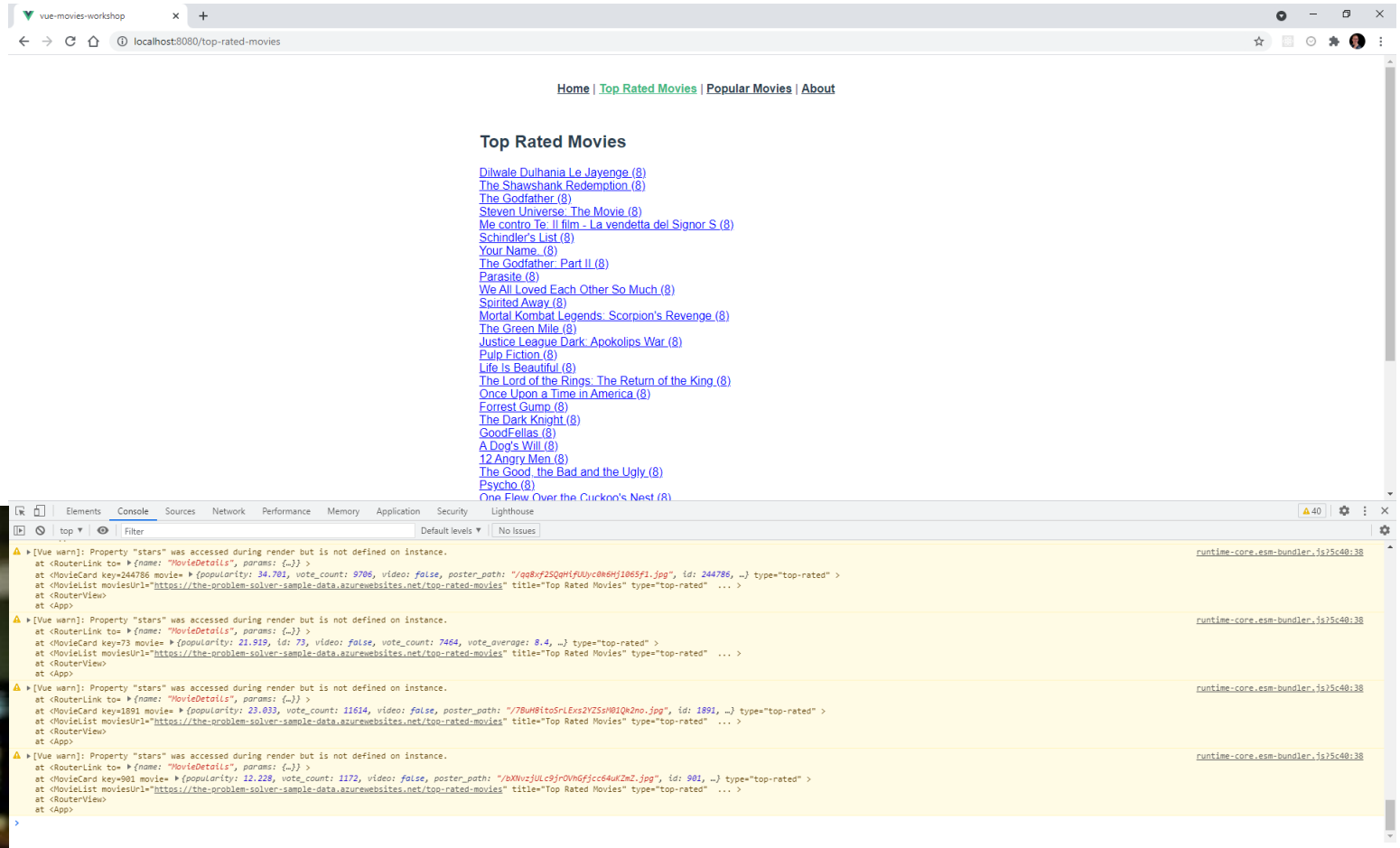
Start the app
npm run serve



Remove the migration build

Workflow

- **Remove the migration build**
  - `npm uninstall @vue/compat`
  - Remove/Update vue.config.js
- Note: The rating stars filter is now broken


vue-movies-workshop
localhost:8080/top-rated-movies
Home | Top Rated Movies | Popular Movies | About
Top Rated Movies
Dilwale Dulhania Le Jayenge (8)
The Shawshank Redemption (8)
The Godfather (8)
Steven Universe: The Movie (8)
Me contro Te: Il film - La vendetta del Signor S (8)
Schindler's List (8)
Your Name. (8)
The Godfather: Part II (8)
Parasite (8)
We All Loved Each Other So Much (8)
Spirited Away (8)
Mortal Kombat Legends: Scorpion's Revenge (8)
The Green Mile (8)
Justice League Dark: Apokolips War (8)
Pulp Fiction (8)
Life Is Beautiful (8)
The Lord of the Rings: The Return of the King (8)
Once Upon a Time in America (8)
Forrest Gump (8)
The Dark Knight (8)
GoodFellas (8)
A Dog's Will (8)
12 Angry Men (8)
The Good, the Bad and the Ugly (8)
Psycho (8)
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
[Vue warn]: Property "stars" was accessed during render but is not defined on instance.
runtime-core.esm-bundler.js?5c40:38


MAKE IT SO
memegenerator.net

Fix the rating stars

Removed APIs

Removed APIs

- `keyCode` support as `v-on` modifiers
- $on, $off and $once instance methods
- Filters
- Inline templates attributes
- `$children` instance property
- `propsData` option
- `$destroy` instance method. Users should no longer manually manage the lifecycle of individual Vue components.
- Global functions `set` and `delete`, and the instance methods `$set` and `$delete`. They are no longer required with proxy-based change detection.

Filters

- **Filters have been removed** from Vue 3
  - Replace with computed property
  - Or with a function call

MovieCard.vue

MovieCard.vue M

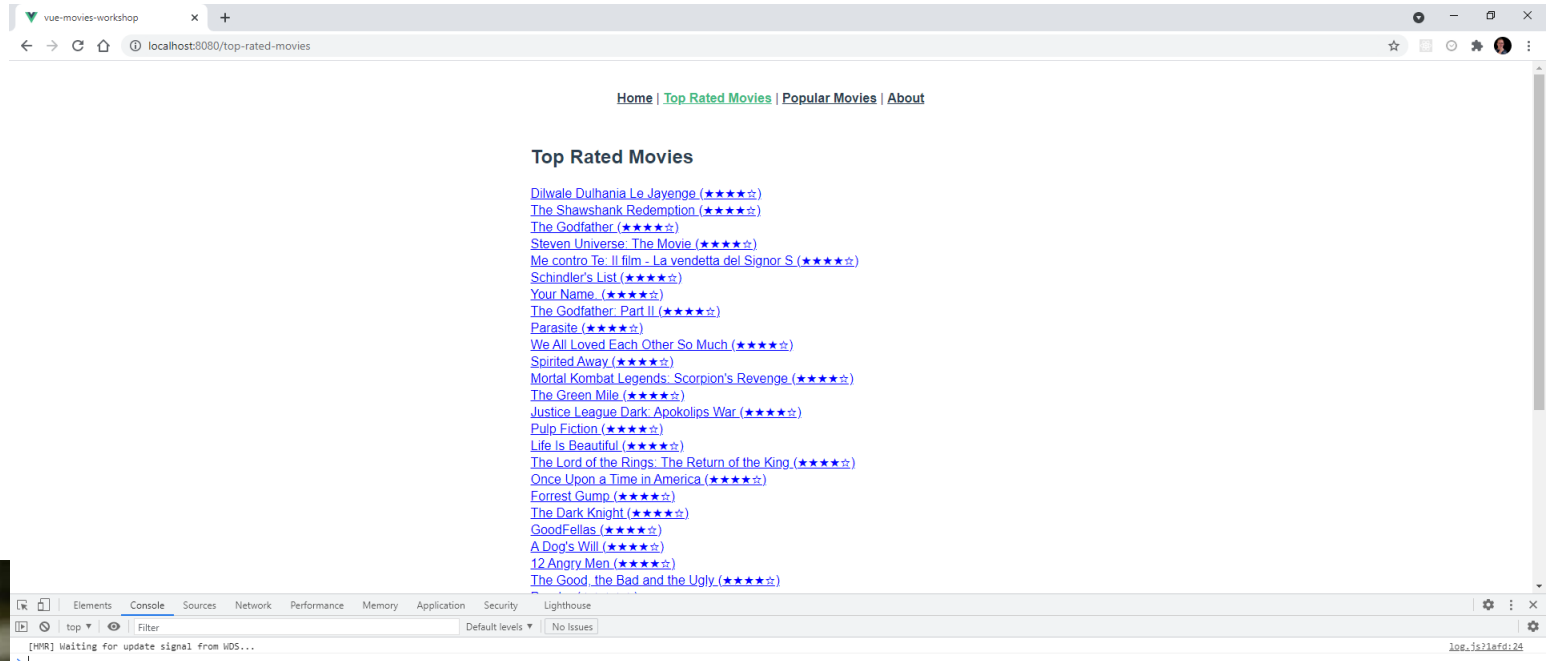
src > views > MovieCard.vue > { } "MovieCard.vue"

You, 2 minutes ago | 1 author (You)

```vue
<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>

<script>
export default {
  props: {
    movie: Object,
    type: String,
  },
  computed: {
    voteAsStars() {
      return "".padEnd(this.movie.vote_average / 2, "★").padEnd(5, "☆");
    },
  },
};
</script>
```

Computed stars



Reusable components

Reusable components

- Create **reusable components** as building blocks
- Add **v-model support** where needed
  - Receive the modelValue prop
  - Submit the update:modelValue event

LabeledInput.vue

LabeledInput.vue M

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

You, seconds ago | 1 author (You)

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

<script>
export default {
  props: ["label", "modelValue"],
  emits: ["update:modelValue"],
};
</script>

<style scoped>
.input-label {
  display: inline-block;
  margin-top: 1em;
  margin-bottom: 0.5em;
}

.form-control {
  width: 100%;
```

MovieDetails.vue

MovieDetails.vue M

src > views > MovieDetails.vue > { } "MovieDetails.vue" > template > div > form > fieldset.fieldset

You, seconds ago | 1 author (You)

```vue
<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 label="Title" v-model="movie.title" />
        <div>
          <label>
            <span class="input-label">Overview</span>
            <textarea v-model="movie.overview" rows="5" class="form-control" />
          </label>
        </div>
        <LabeledInput
          label="Vote average"
          v-model.number="movie.vote_average"
          type="number"
        />
        <LabeledInput
          label="Release date"
          v-model="movie.release_date"
          type="date"
        />

        <div class="button-row">
          <button type="submit">Save</button>
          <button type="reset">Reset</button>
        </div>
```

Using inheritAttrs

- Use **inheritAttrs** to control where additional attributes are added
  - By default at the components root HTML tag
- Use v-bind="$attrs" to add them to any required HTML tag

LabeledInput.vue

LabeledInput.vue M

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

You, seconds ago | 1 author (You)

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

<script>
export default {
  inheritAttrs: false,
  props: ["label", "modelValue"],
  emits: ["update:modelValue"],
};
</script>

<style scoped>
.input-label {
  display: inline-block;
  margin-top: 1em;
  margin-bottom: 0.5em;
}
```

Composition API

Composition API

- The Composition API is an **alternative** way of creating components
  - Doesn't replace the Options API!
- Makes most sense in more **complex components**
  - But can be used everywhere
- Export a **setup() function**
  - Receives the props and context as parameters
  - Returns an object with items available in the template
- Note: There is a Vue 2 plugin for Composition API

Lifecycle Hooks

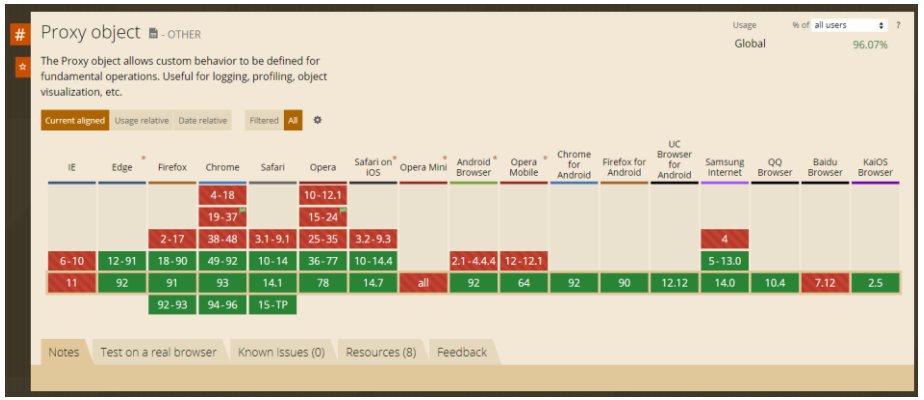
Lifecycle Hooks

- Import and call with a callback function **in the setup() function**
- All functions start with on…
  - onMounted()
  - onUnmounted()
  - Etc.
- The setup() function itself replaces **the `created()` hook**

Reactivity

Reactivity

- The Vue reactivity is **completely rewritten** in Vue 3
  - Now based on JavaScript proxy objects
- More flexible and fewer edge cases
  - But only works in modern browsers



Reactive Helper Functions

- ref()
- reactive()
- toRefs()
- computed()
- watch()
- watchEffect()

ref()

- Use a `ref()` to create a **reactive variable**
  - Use the `ref.value` in the code
  - Use the `ref` directly in the template
- Works both for **objects and primitive** values

reactive()

- Can be used to create a **reactive object**
  - No wrapping `.value` property
- Using a `ref()` calls `reactive()` internally for an object
  - Not often used as ref() achieves almost the same result

computed()

- Creates a **reactive value based on other reactive values**
  - Automatically recomputes the result when the dependencies update
- Computed values are **usually read only**
  - But they can be read/write when needed

watch()
watchEffect()

- **`watch()` a reactive value** and execute when updated
  - The source can be a function or a ref
  - Pass in an array to watch multiple sources
  - Similar as `this.$watch()` in the Options API
- There is a new and similar `watchEffect()` in Vue 3
  - Similar to `computed()`
  - Often the **better option** in Vue 3

toRefs()

- **Converts** a reactive object to an **object with reactive properties**
  - The `toRefs()` function turns a new object
- The **props** passed to the `setup()` function are not reactive
  - With `toRefs()` you can watch for prop updates

toRaw()

- Turn a reactive object back **into a normal object**
  - With the `toRaw()` function

MovieList using the Composition API

MovieList.vue

MovieList.vue M

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

```vue
<script>
import { ref, onMounted, watch } from "vue";
import MovieCard from "./MovieCard.vue";

export default {
  components: {
    MovieCard,
  },
  props: {…
  },
  setup(props) {
    const error = ref(null);
    const loading = ref(true);
    const movies = ref([]);

    async function fetchMovies() {
      try {
        const rsp = await fetch(props.moviesUrl);

        if (rsp.ok) {
          movies.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;
      }
```

MovieList.vue

MovieList.vue M

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

```
48            const rsp = await fetch(props.moviesUrl);
49
50            if (rsp.ok) {
51              movies.value = await rsp.json();
52            } else {
53              error.value = rsp.statusText ?? "Failed to load data";
54            }
55          } catch (error) {
56            error.value = error?.message ?? "Failed to load data";
57          } finally {
58            loading.value = false;
59          }
60        }
61
62        onMounted(() => fetchMovies());
63
64        watch(
65          () => props.moviesUrl,
66          () => fetchMovies()
67        );
68
69        return {
70          error,
71          loading,
72          movies,
73        };
74      },
75    };
76    </script>
```

MovieDetails using the Composition API

MovieDetails.vue

MovieDetails.vue M

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

```vue
<script>
import { ref, onMounted, watch, computed } from "vue";
import { useRouter } from "vue-router";
import LabeledInput from "./LabeledInput.vue";

export default {
  components: { LabeledInput },
  props: {
    id: { type: [Number, String], required: true },
    type: { type: String, required: true },
  },
  setup(props) {
    const router = useRouter();

    const error = ref(null);
    const loading = ref(true);
    const saving = ref(false);
    const movie = ref(null);

    const uriTypeFragment = computed(() =>
      props.type === "popular" ? "popular-movies" : "top-rated-movies"
    );

    const movieUrl = computed(
      () =>
        `${process.env.VUE_APP_API_ORIGIN}/${uriTypeFragment.value}/${props.id}`
    );

```

MovieDetails.vue

MovieDetails.vue M

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

```
63      async function fetchMovie() {
64        try {
65          loading.value = true;
66          const rsp = await fetch(movieUrl.value);
67
68          if (rsp.ok) {
69            movie.value = await rsp.json();
70          } else {
71            error.value = rsp.statusText ?? "Failed to load data";
72          }
73        } catch (error) {
74          error.value = error?.message ?? "Failed to load data";
75        } finally {
76          loading.value = false;
77        }
78      }
79
80      onMounted(() => fetchMovie());
81
82      watch(movieUrl, () => fetchMovie());
```

MovieDetails.vue

MovieDetails.vue M

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

```
84      async function saveMovie() {
85        try {
86          saving.value = true;
87          const rsp = await fetch(movieUrl.value, {
88            method: "put",
89            headers: {
90              "Content-Type": "application/json",
91            },
92            body: JSON.stringify(movie.value),
93          });
94
95          if (rsp.ok) {
96            movie.value = await rsp.json();
97          } else {
98            error.value = rsp.statusText ?? "Failed to save data";
99          }
100       } catch (error) {
101         error.value = error?.message ?? "Failed to save data";
102       } finally {
103         saving.value = false;
104       }
105     }
```

MovieDetails.vue

MovieDetails.vue M

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

```
107      async function submitForm() {
108        await saveMovie();
109        if (!error.value) {
110          router.push(`/${uriTypeFragment.value}`);
111        }
112      }
113      function resetForm() {
114        fetchMovie();
115      }
116
117      return {
118        error,
119        loading,
120        saving,
121        movie,
122        submitForm,
123        resetForm,
124      };
125    },
126  };
127  </script>
```

Composition Functions

Composition Functions

- **Extract** functionality into **reusable functions**
  - Use anything you would use in the `setup()`
- **More reliable** than using mixins
  - No accidental naming collisions
- Composition API **naming convention**
  - Prefix a function with “use”

useFetchData.js

useFetchData.js M

src > composables > useFetchData.js > ...

```
1  import { ref, onMounted, watch } from "vue";
2
3  export default function useFetchData(url) {
4    const error = ref(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   }
23
24   onMounted(() => fetchData());
25
26   watch(url, () => fetchData());
27
28   return { error, loading, data, fetchData };
29 }
```

MovieList.vue

MovieList.vue M

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

```vue
<script>
import MovieCard from "./MovieCard.vue";
import useFetchData from "../composables/useFetchData";
import { toRefs } from "@vue/reactivity";

export default {
  components: {
    MovieCard,
  },
  props: {…
  },
  setup(props) {
    const url = toRefs(props).moviesUrl;
    const { error, loading, data: movies } = useFetchData(url);

    return {
      error,
      loading,
      movies,
    };
  },
};
</script>
```

MovieDetails.vue

MovieDetails.vue M

src > views > MovieDetails.vue > {} "MovieDetails.vue" > script > default > setup > saveMovie

```
48   setup(props) {
49     const router = useRouter();
50
51     const saving = ref(false);
52
53     const uriTypeFragment = computed(() =>
54       props.type === "popular" ? "popular-movies" : "top-rated-movies"
55     );
56
57     const movieUrl = computed(
58       () =>
59         `${process.env.VUE_APP_API_ORIGIN}/${uriTypeFragment.value}/${props.id}`
60     );
61
62     const {
63       error,
64       loading,
65       data: movie,
66       fetchData: fetchMovie,
67     } = useFetchData(movieUrl);
68
69     async function saveMovie() {
```

watchEffect()

watchEffect()

- **Automatically watches** all dependencies
  - Even deep dependencies
  - Run the watch function immediately

useFetchData.js

useFetchData.js

```
You, 2 days ago | 1 author (You)
import { ref, watchEffect } from "vue";

export default function useFetchData(url) {
  const error = ref(null);
  const loading = ref(true);
  const data = ref(null);

  async function fetchData() {…
  }

  watchEffect(() => fetchData());

  return { error, loading, data, fetchData };
}
```

Conclusion

- Migrating from Vue 2 to 3 is usually not hard
  - The migration build helps a lot
- Most external libraries need to be updated
  - Not all are compatible with Vue 3 yet
- The Composition API is an improvement over the Options API
  - But the Options API still works
  - Reuse code with composition functions
- The Reactive API has been completely rewritten
  - More flexible and fewer edge cases

Maurice de Beijer

@mauricedb

maurice.de.beijer@gmail.com