Vue 3 (Summary of considerations for the migration)

Release note: Some parts of the ecosystem are not ready yet. E.g. Vuetify.

💡 There is a migration guide in Vue 3 documentation.

Changes overview

- Vue instances are created differently

```
const app = Vue.createApp(config).mount(css selector e.g. ‘#app’)
```

- Data **must** be a method which return an object. (Methods, computed, and watch remain the unaltered)

```
data: function () { return{name: 'Manuel'})
```

- Components, directives and 3rd party modules are registered on “app” instead of the global object.

```
app.component('my-component', MyComponent)
```

- Transitions are v-enter is now v-enter-from

- Vue router is now created with createRouter() and mode history is set up differently.

```
//Importing
import { createRouter, createWebHistory } from 'vue-router';

const router = createRouter({
  history:
  //Same configuration object than Vue 2
});

//The app instance created with Vue.createApp()
app.use(router);
```

- Router link component no longer needs exact prop

- Router view component now wraps transition component

```
//Vue 2
<transition name="fakename" mode="fakemode">
  <router-view></router-view>
</transition>

//Vue 3
<router-view v-slot="slotProps">
  <transition name="fakename" mode="fakemode">
    <component :is="slotProps.Components"></ccomponent>
  </transition>
</router-view>
```

- Vuex is now created with with createStore()

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

//We use the same store config object than in Vue 2
const store = createStore(sameStoreObject);

//The app instance created with Vue.createApp()
app.use(store);
```

New features overview

- Teleport component: It allows us to render a component in a different place in the DOM. It helps to write semantically more correct HTML, which is beneficial for SEO.

```
//You need to use the css selector of the element in which the component must be rendered.

<teleport to="body">
  <dialog></dialog>
</teleport>

//In this case, it will be rendered right inside of the body element

<body>
  <dialog></dialog>
  <div id="app">
    //the rest of the app
  </div>
</body>
```

- Fragments

We can have more than one child in the template of a Vue component.

```
<template>
  <dialog></dialog>
  <li>
    <h3>Mock list item title</h3>
  </li>
</template>
```

- Composition API replaces Options API. This feature is 100% optional.

- Better Typescript support. For more info, check Vue 3 documentation.

Composition API

So far, we used the Options API for building Vue components. It is perfectly fine to use it, but it present some disadvantages in comparison with Composition API:

- Code that belongs together logically is split up across multiple options. (Data, methods, computed)
- Re-using logic across components can be tricky or cumbersome.

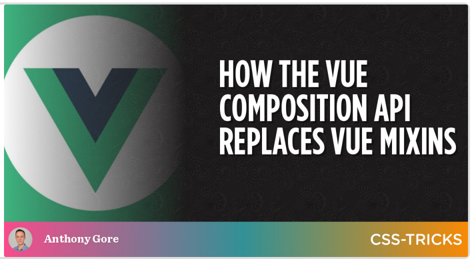
You may think issues could be resolve using mixins, but using them can result tricky as well.

For more info about these differences:

How the Vue Composition API Replaces Vue Mixins

Looking to share code between your Vue components? If you're familiar with Vue 2, you've probably used a mixin for this purpose. But the new Composition API, which is available now

https://css-tricks.com/how-the-vue-composition-api-replaces-vue-mixins/



You can find more info about the composition API here:

https://v3.vuejs.org/guide/composition-api-introduction.html