



Vue, Part 1

Recitation 6 2024

Plan



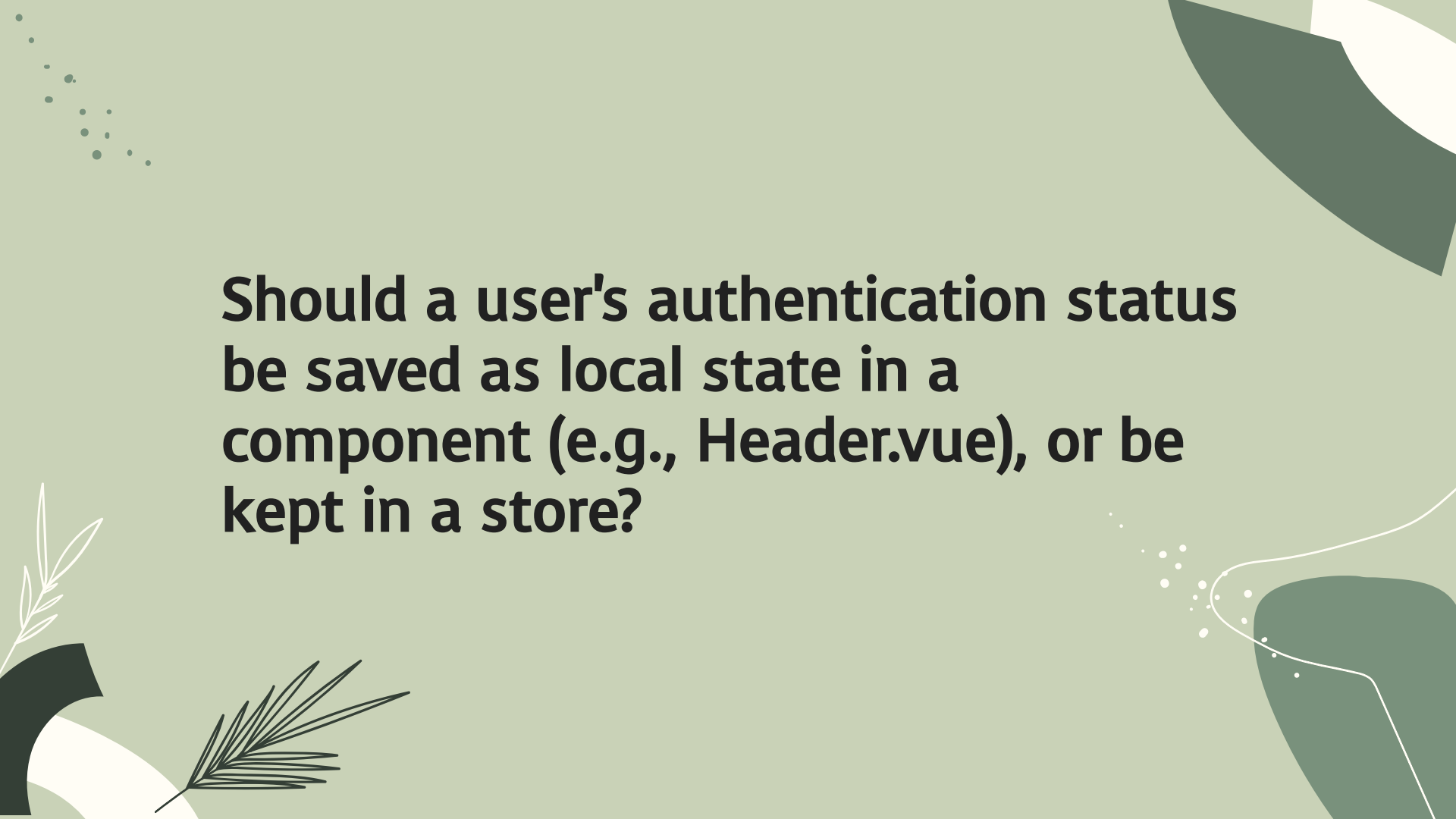
01 Front End Starter Code Walkthrough

02 Discussion / Resources


03 Questions / Wrap-up

Prep Review

The background is a solid dark green color. It features several decorative elements: a large black abstract shape in the top-left corner, a smaller black shape in the bottom-right corner, and a thin white circle on the left side. There are clusters of small white dots in the top-left, bottom-left, and top-right areas. A white line drawing of a leafy branch is positioned at the top center, and another similar branch is on the right side.



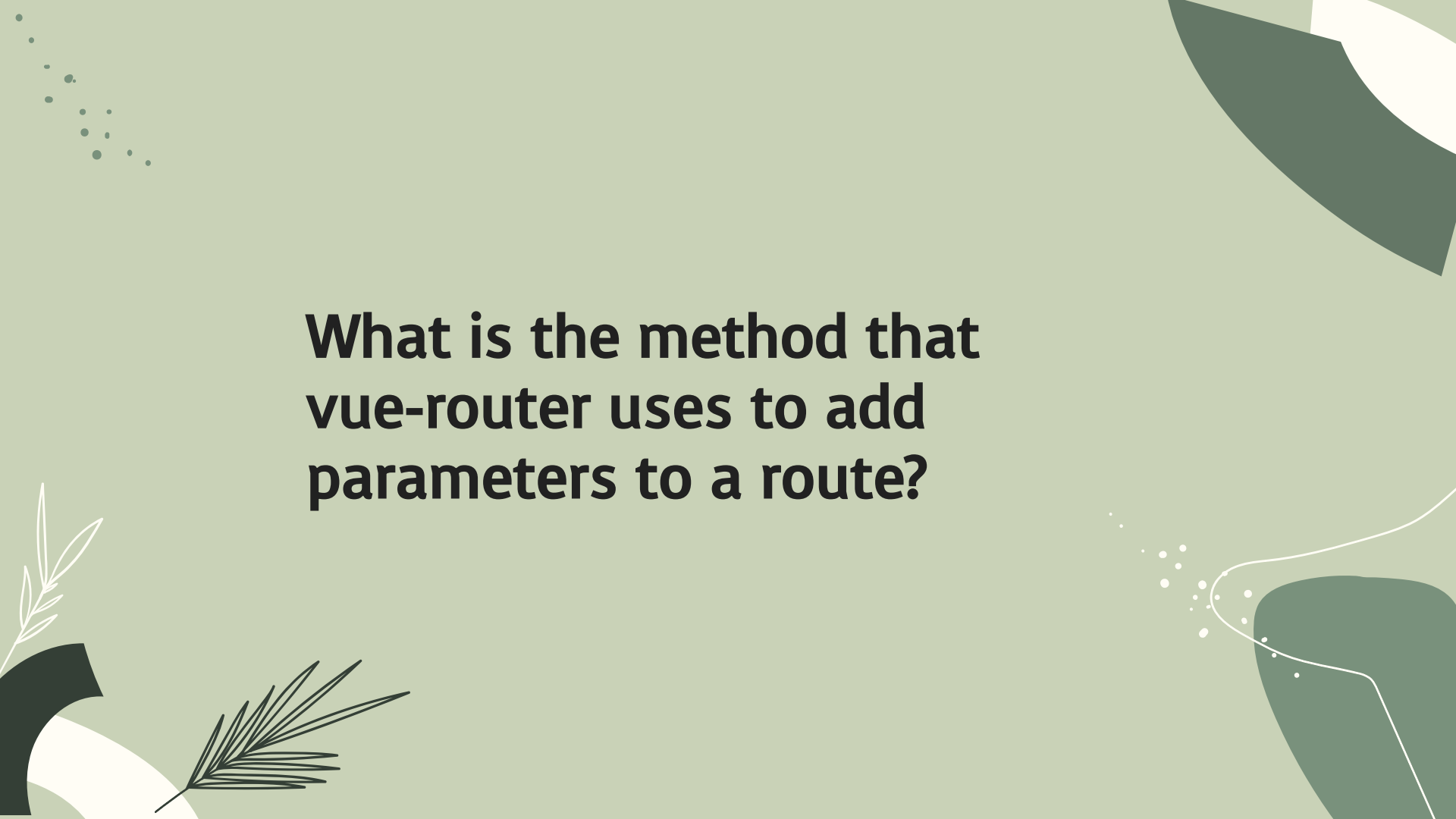
Should a user's authentication status be saved as local state in a component (e.g., Header.vue), or be kept in a store?



Are you allowed to use a third-party styling packages (e.g., Bootstrap, Material Design, etc.) for your assignments/projects in 6.1040?

The background is a light sage green color. It features several decorative elements: a cluster of small white dots in the top-left corner; a dark green leaf-like shape in the top-right corner; a white leaf-like shape in the bottom-left corner; a dark green leaf-like shape in the bottom-right corner; and a thin white line with a trail of small white dots curving across the bottom-right area.

**Are dynamic routing methods
supported by vue-router?**



**What is the method that
vue-router uses to add
parameters to a route?**



01 Front End
Starter Code!


Create Your Repo:

<https://github.com/61040-fa24/frontend-starter>




Getting Started

1. Run `npm install`
2. Copy over your `.env` file from your backend code into the root folder
3. Run `npm run dev:server` to start the server. Run `npm run dev:client` to start the client



After recitation: Read the rest of the README for instructions on transferring over your backend code and deployment!



Repo Contents



Overview – Non-client folder

```
frontend-starter/  
├── api/★  
├── client/  
│   ├── assets/  
│   │   └── images/  
│   ├── components/  
│   ├── router/  
│   ├── stores/  
│   ├── utils/  
│   ├── views/  
│   └── App.vue  
├── public/★  
├── server/★  
└── index.html/
```

api – connection with server-side routes (what you wrote last week)

public – top-level static assets

- Change your site favicon here!

server – your backend code!

index.html – app-level headers

Overview – client folder

```
frontend-starter/  
├── api/  
├── client/★  
│   ├── assets/  
│   │   └── images/  
│   ├── components/  
│   ├── router/  
│   ├── stores/  
│   ├── utils/  
│   ├── views/  
│   └── App.vue  
├── public/  
├── server/  
└── index.html/
```

client – all relevant frontend code

- **assets** – assets that are only compiled if necessary
 - store your images in **images/**
- **components** – your app's components
- **router** – page-level routing
- **stores** – keep track of app state (i.e. current user)
- **utils** – general utility functions
- **views** – your app's pages
- **App.vue** – app definition



client/assets

- **images** – where you should store all your relevant images
- Contains global css files (including css variables)

```
# main.css ×
client > assets > # main.css > ...
Grace Huang, 1 hour ago | 1 author (Grace Huang)
1 /* Add some global styles here! */
2
3 :root {
4   --red:   ■ rgb(208, 67, 12);
5   --base-bg: ■ #f1f3f5;
6 }
7 Grace Huang, 1 hour ago • Initial commit
8 button.btn-small {
9   font-size: 80%;
10 }
11
12 button.button-error {
13   color: ■ white;
```

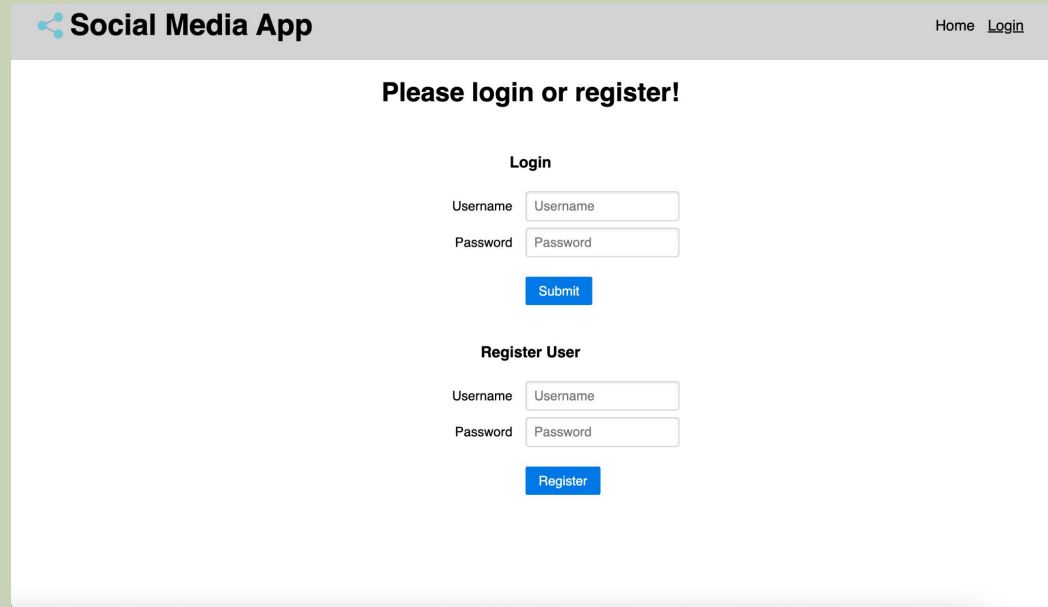


client/components

- All the components for your app
 - Each component will generally correspond to a certain section of your app
 - i.e. a login form, a list of posts
 - Not necessarily just overall pages
 - Structuring is up to your discretion, but keeping things modular will make your life a lot easier!
- 
- 

client/views

- Pages for your app, where each **view** corresponds to 1 page layout
- Example: user login page
- Use params to create pages that may show different content but are not necessarily completely different pages



The screenshot shows a web page for a "Social Media App". At the top left is the app's logo and name, and at the top right are links for "Home" and "Login". The main heading is "Please login or register!". Below this, there are two sections: "Login" and "Register User". Each section contains a "Username" input field, a "Password" input field, and a corresponding action button ("Submit" for login, "Register" for registration).

Social Media App [Home](#) [Login](#)

Please login or register!

Login

Username

Password

Register User

Username

Password

client/router



- Client-side routing to move between your different pages
- We use vue-router - documentation here: <https://router.vuejs.org/guide/>
- Supports dynamic routing!
 - Functionality to view other user profiles based on user id

```
TS index.ts 9+ x
client > router > TS index.ts > [e] router > routes
9
10 const router = createRouter({
11   history: createWebHistory(),
12   routes: [
13     {
14       path: "/",
15       name: "Home",
16       component: HomeView,
17     },
18     {
19       path: "/setting",
20       name: "Settings",
21       component: SettingView,
22       meta: { requiresAuth: true },
23     },
24     {
25       path: "/login",
26       name: "Login",
27       component: LoginView,
28       meta: { requiresAuth: false },
29       beforeEnter: (to, from) => {
30         const { isLoggedIn } = storeToRefs(useUserStore());
31         if (isLoggedIn.value) {
32           return { name: "Settings" };
33         }
34       },

```





client/stores

- Variables/functions for keeping track of the overall **state** of your app for the user
 - This should only store information that should persist across refreshes on a user's device on the client side (i.e. who is logged in rn)
 - **Not a substitute for mongodb!!**
 - What parts do we put in store vs components?
- 
- 



client/utls



- **Utility methods** that you might find useful across different components/concepts
 - *formatDate* helps format a date prettily on the frontend
 - Feel free to make your own functions! If you find yourself writing the same processing code for many different areas of the application, it'll make your life easier if you extract them into a utls file (6.102)
- 
- 

Component Deep Dive





client > components > Post

- An example of a set of components that helps a user create/edit/view posts - `CreatePostForm.vue`
 - Separation of the script from the template and style
- 
- 

Styling

Generally **scoped** within the file it's in for ease of reference

- Applies only to the **specific component**

```
∨ <style scoped>
∨ form {
  background-color: var(--base-bg);
  border-radius: 1em;
  display: flex;
  flex-direction: column;
  gap: 0.5em;
  padding: 1em;
}
```

Backend Communication

- Wrapper around fetch using `fetchy` (in `utils/fetchy.ts`)
- Each route structured as `api/___`
- `emit` communicates events back to their parent components
- Let's see how this works in `PostListComponent.vue`!

```
8   const createPost = async (content: string) => {
9     try {
10      await fetchy("api/posts", "POST", {
11        body: { content },
12      });
13    } catch (_) {
14      return;
15    }
16    emit("refreshPosts");
17    emptyForm();
18  };
```


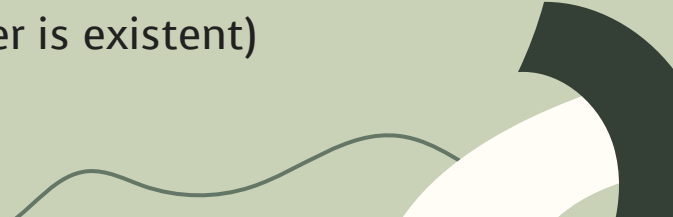


**02 Discussion /
Resources**



Client-side Validation



Best practices:

- Frontend shouldn't be more restrictive than the backend – if you allow certain responses in the backend, you should be able to handle that in the frontend as well
 - Validating things that can be **easily** checked on the frontend side can make you make less irrelevant requests
 - i.e. checking input format
 - But you shouldn't do things that would require a database call anyways (i.e. checking whether a queried user is existent)
- 
- 



General Coding Resources





- Vue 3 Tutorial:
<https://www.youtube.com/playlist?list=PL4cUxeGkcC9hYYGbV60Vq3IXYNfDk8At1>
 - Vue Style Guide: <https://v2.vuejs.org/v2/style-guide/?redirect=true>
 - Vue guide: <https://vuejs.org/guide/introduction.html>
 - Vue Cheatsheet: <https://devhints.io/vue>
 - Our router: <https://router.vuejs.org/guide/>
- 
- 



Styling/Formatting Resources



- Icon library: <https://fontawesome.com/>
 - MDN web docs:
<https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>
 - Pure.css: <https://purecss.io/>
 - What we currently use in the front end, but feel free to ignore it and just use plain CSS
- 
- 



03 Questions /
Wrap-up



<https://forms.gle/AXVqqkmaqSpppZ4E7>