

CIS 371-01 Web Application Programming

Pinia II

App State Management



GRAND VALLEY
STATE UNIVERSITY®

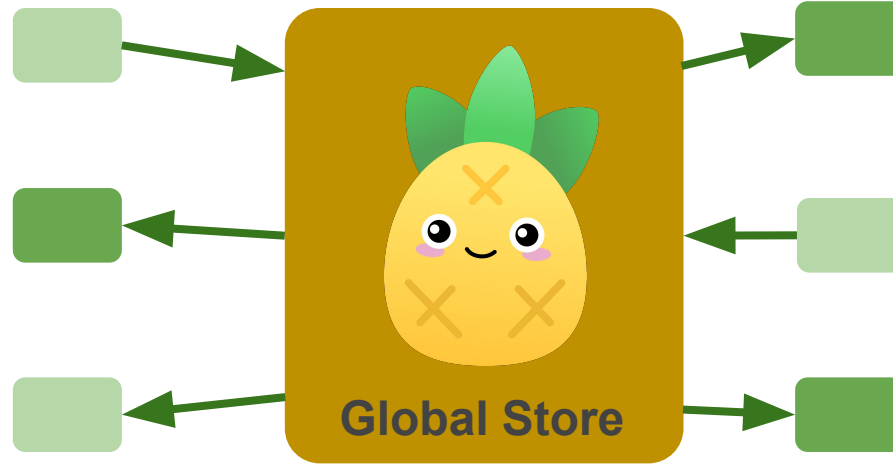
Lecturer: **Dr. Yong Zhuang**

Last time

What is state management?

What is Pinia?

- State,
- Actions,
- Getters



Using Stores in Other Stores

CareStore.ts

```
import { defineStore } from "pinia";
import { useItemStore } from "../ItemStore";

export const useCartStore = defineStore("CartStore", {
  state: () => {
    return {
      items: [],
    };
  },
  getters: {
    allProducts() {
      const itemStore = useItemStore();
      return itemStore.products;
    },
  },
  actions: {
    addItem(itemId, count) {
      // set the count for the proper item in the state above
    },
  },
});
```

ItemStore.ts

```
import { defineStore } from "pinia";
import products from "../data/products.json";
export const useItemStore = defineStore("ItemStore", {
  state: () => {
    return { products };
  },
});
```

Use actions from another store in an action

```
import { defineStore } from "pinia";
import { useItemStore } from "../ItemStore";

export const useCartStore = defineStore("CartStore", {
  state: () => {
    return {
      items: [],
    };
  },
  actions: {
    reloadProducts() {
      const itemStore = useItemStore();
      return itemStore.fill();
    },
  },
});
```

CartStore.ts

```
import { defineStore } from "pinia";
import products from "../data/products.json";

export const useItemStore = defineStore("ItemStore", {
  state: () => {
    return { products };
  },
  actions: {
    async fill() {
      //...
    },
  },
});
```

ItemStore.ts

Use actions from another store in an action

CareStore.ts

```
import { defineStore } from "pinia";
import { useItemStore } from "../ItemStore";

const itemStore = useItemStore();
export const useCartStore = defineStore("CartStore", {
  state: () => {
    return {
      items: [],
    };
  },
  actions: {
    reloadProducts() {
      return itemStore.fill();
    },
  },
});
```



```
✖ ▶ Uncaught Error: [🐞]: getActivePinia was called with no active Pinia. Did you forget to install pinia?
    const pinia = createPinia()
    app.use(pinia)
    This will fail in production.
    at useStore (pinia.esm-browser.js:1638:19)
```

Subscribe to Pinia Stores

Notify me when something changes in this store, so I can update the UI or perform other actions in response.

- Watch state for changes.
- Monitor actions for calls.
- Perform side effects.
- Measure how long your actions take to run.
- Trigger user notifications.
- Log errors to third-party services.

Subscribe to actions

```
unsubscribe = userStore.$onAction(  
  ({  
    name, // name of the action  
    store, // store instance, same as `someStore`  
    args, // array of parameters passed to the action  
    after, // hook after the action returns or resolves  
    onError, // hook if the action throws or rejects  
  }) => {  
    // Action-related logic code here...  
  }  
);
```

AnyVueComponent.vue

Use conditional to run on select actions

```
unsubscribe = userStore.$onAction(  
  ({  
    name, // name of the action  
    store, // store instance, same as `someStore`  
    args, // array of parameters passed to the action  
    after, // hook after the action returns or resolves  
    onError, // hook if the action throws or rejects  
  }) => {  
    if (name === "fill") {  
      // more code  
    }  
  }  
);
```

AnyVueComponent.vue

After and onError

AnyVueComponent.vue

UserStore.ts

```
import { defineStore } from "pinia";
import { RandomUser } from "../types/user";

export const useUserStore = defineStore("UserStore", {
  state: () => {
    return { users: {} as RandomUser };
  },

  getters: {
    count: (state) => state.users.results.length,
  },
  actions: {
    async fill() {
      const res = await fetch(
        "https://randomuser.me/api?results=5&nat=gb,fr&inc=name,email,picture"
      );
      this.users = await res.json();
      return this.count;
    },
  },
});
```

```
unsubscribe = userStore.$onAction(
  ({
    name, // name of the action
    store, // store instance, same as `someStore`
    args, // array of parameters passed to the action
    after, // hook after the action returns or resolves
    onError, // hook if the action throws or rejects
  }) => {
    const startTime = Date.now();
    // this will trigger before an action on `store` is executed
    console.log(`Start "${name}" with params [${args.join(", ")}].`);
    if (name === "fill") {
      // after() will trigger if the action succeeds and after it has fully run.
      after((result) => {
        console.log(
          `Finished "${name}" after ${
            Date.now() - startTime
          }ms.\nResult: ${result}.`
        );
      });
    }
    // onError() will trigger if the action throws or returns a promise that rejects
    onError((error) => {
      console.warn(
        `Failed "${name}" after ${
          Date.now() - startTime
        }ms.\nError: ${error}.`
      );
    });
  });
}
```

Demo

Subscribe to the state

AnyVueComponent.vue

```
onMounted(() => {  
  userStore.$subscribe((mutation, state) => {  
    // 'direct' | 'patch object' | 'patch function'  
    mutation.type;  
  
    // same as userStore.$id  
    mutation.storeId;  
  
    // only available with mutation.type === 'patch object'  
    mutation.payload; // patch object passed to $patch()  
  });  
});
```

Demo

The 'Patch Object' in Pinia

The `$patch()` method is a convenient way to apply multiple state changes at once. It simplifies state management by allowing batch updates.

- When to use: Handling complex state structures or when multiple state properties need to be updated simultaneously.
- Value Setting:
 - For each key in the patch object, the value is the new value that you want to set for that corresponding state property.
 - The `$patch()` method ensures that these updates are reactive and efficiently managed by Vue's reactivity system.

The 'Patch Object' in Pinia

```
import { defineStore } from "pinia";

export const useUserStore = defineStore("userStore", {
  state: () => ({
    name: "",
    age: 0,
    email: "",
  }),
  // actions, getters, etc.
});
```

UserStore.ts

[Demo](#)

[Online Doc](#)

```
<script setup lang="ts">
import { onMounted } from "vue";
import { useUserStore } from "../stores/UserStore";
const userStore = useUserStore();

onMounted(() => {
  userStore.$subscribe((mutation, state) => {
    if (mutation.type === "patch object") {
      console.log("Patch object:", mutation.payload);
    }
  });
  userStore.$patch({
    name: "John Doe",
    email: "john@example.com",
  });
});
</script>

<template>
  <ul>
    <li>{{ userStore.name }}</li>
    <li>{{ userStore.age }}</li>
    <li>{{ userStore.email }}</li>
  </ul>
</template>
```

AnyVueComponent.vue

Exercises

Complete the store-app implementation with the following functionalities:

- Implement a CartStore using Pinia to temporarily store the user's selected items for checkout.
- Utilize the \$patch method to add items into cart.
- Displays all items currently in the cart.

[store-app](#)

Persist data?

Refresh-proof your Pinia Stores