Pinia

https://pinia-ua.netlify.app/introduction.html

https://storage.googleapis.com/vue-mastery.appspot.com/flamelink/media/Pinia-Cheat-Sheet.pdf?GoogleAccessId=firebase-adminsdk-jyioc%40vue-mastery.iam.gserviceaccount.com&Expires=16725225600&Signature=kyMynndU2WMV1ie1HcAdGZkajToy0%2FuvB6xeObn6lTYLy2IRJWZxSqnlbB6P41f83y1lKczNxc5%2FXlk%2B%2B0BZ4zKpPSZAJVveEHzpROesCJ%2BJn%2FDw3tabttXgwwFVzWJC5yAmstJJE%2FNNULCJEmi1%2BOzPW7eyrok0Cg4HIqMhJoGVNp2DiUPZnqoYK71ugiNzMXos5bTk4%2FOwipsxuJeSVOeaD5N2CfDLQAOGNM0dVKCxQseiNUaYaKNe0bbAva9zmlbTlxVYNZlGPw3R0i0gs5WBAQiFHn0OlcQxcFP3F%2BKksqg3%2BHmlORkGDjnbiEdjX9pnZNh%2BqaVgg2p3ICXvQg%3D%3D

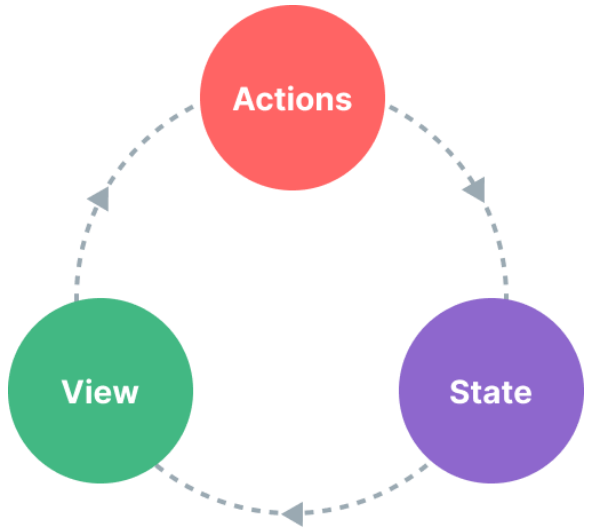
Що таке сховище?

Сховище (наприклад,Pinia) це сутність, яка містить стан і бізнес-логіку, яка не прив'язана до дерева компонентів.

Особливості використання сховища

1)Сховище має містити дані, до яких можна отримати доступ у вашому застосунку. Це включає дані, які використовуються в багатьох місцях, наприклад, інформація про користувача, яка відображається на навігаційній панелі, а також дані, які потрібно зберегти на сторінках, наприклад, дуже складна багатоступенева форма.

2)З іншого боку, вам слід уникати включення до сховища локальних даних, які замість цього можуть бути розміщені в компоненті, наприклад, видимість локального елемента сторінки.



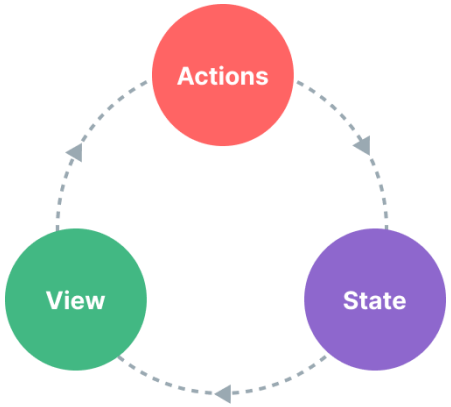
https://pinia-ua.netlify.app/getting-started.html

Переваги від використання Pinia:

- Підтримка Devtools
- Хронологія для відстеження дій, мутацій
- Сховища з'являються в компонентах, де вони використовуються
- Подорожі в часі та легше налагодження
- Гаряча заміна модулів
- Змінюйте свої сховища, не перезавантажуючи сторінку
- Зберігайте будь-який існуючий стан під час розробки
- Плагіни: розширюйте особливості Pinia за допомогою плагінів
- Коректна підтримка TypeScript або автозаповнення для користувачів JS
- Підтримка рендерингу на стороні сервера

Відмінності від Vuex ≤4:

- Мутації більше не існують.
- Немає потреби створювати власні складні оболонки для підтримки TypeScript.
- Немає більше магічних рядків для введення, імпортуйте функції, викликайте їх, насолоджуйтесь автозавершенням!
- Не потрібно динамічно додавати сховища, вони всі динамічні за замовчуванням
- Більше жодного вкладеного структурування модулів. Pinia пропонує плоску структуру за дизайном, у той же час дозволяючи способи перехресної композиції між сховищами. Ви навіть можете мати циклічні залежності сховищ.
- Немає модулів із простором імен



Pinia

https://pinia-ua.netlify.app/getting-started.html

Встановлюємо	`yarn add pinia` `# або з npm` `npm install pinia`
Додаємо до проєкту як плагін	`import { createApp } from 'vue'` ***`import { createPinia } from 'pinia'`*** `import App from './App.vue'` ***`const pinia = createPinia()`*** `const app = createApp(App)` ***`app.use(pinia)`*** `app.mount('#app')`

Визначення сховища

Створення сховища Рекомендовано, щоб ідентифікатор (є обов'язковим) експортованого об'єкта сховища містила слова `use` та `Store` (наприклад, `use***User***Store`, `use***Cart***Store`, `use***Product***Store`)	import { *defineStore* } from 'pinia' export const use***Назва***Store = *defineStore*(' ***назва_сховища*** ', { // функція налаштувань або об'єкт опцій })	import { *defineStore* } from 'pinia' export const useCounterStore = *defineStore*('counter', { . . . . . . }) export const useUsersStore = *defineStore*('users', { . . . . . . })

Опційні сховища

Загальна форма	Приклад
import { defineStore } from 'pinia' export const useНазваStore = defineStore('назва_сховища', { })	import { defineStore } from 'pinia' export const useCounterStore = defineStore('counter', { })

Опційні сховища

Загальна форма

```
import { defineStore } from 'pinia'

export const useНазваStore = defineStore(
  'назва_сховища',
  {
    //Дані сховища (аналог data())
    state: () => (
           {
             властивіть1 : поч.знач.1,
             властивіть2 : поч.знач.2,
             . . . . . .
           }
         ),
  }
)
```

Приклад

```
import { defineStore } from 'pinia'

export const useCounterStore = defineStore(
  'counter',
  {
    state: () => (
                {
                  count: 0,
                  name: 'Дмитро'
                }
              ),
  }
)
```

Опційні сховища

Загальна форма	Приклад
import { defineStore } from 'pinia' export const useНазваStore = defineStore('назва_сховища', {	import { defineStore } from 'pinia' export const useCounterStore = defineStore('counter', {
//Дані сховища (аналог ***data()***) ***state***: () => ({ властивіть1 : поч.знач.1, властивіть2 : поч.знач.2, ... }),	***state***: () => ({ count: 0, name: 'Дмитро' }),
//Функції, що дозволяють **отримати обч. значення** з state ***getters***: { //-- аналог обчислювальних властивостей ***computed*** -- назва_геттера1: (state) => обч._значення_зі_state, ... },	***getters***: { doubleCount: (state) => state.count * 2, halfCount: (state) => state.count / 2, },
})	})

Опційні сховища

Загальна форма

```
import { defineStore } from 'pinia'

export const useНазваStore = defineStore(
  'назва_сховища',
  {

    //Дані сховища (аналог data())
    state: () => (
           {
             властивіть1 : поч.знач.1,
             властивіть2 : поч.знач.2,
             . . . . . .
           }
         ),
    //Функції, що дозволяють отримати обч. значення з state
    getters: {
      //-- аналог обчислювальних властивостей computed --
      назва_геттера1: (state) => обч._значення_зі_state,
      . . . . . .
    },
    //Функції, що дозволяють робити зміни у state
    actions: {
      //-- аналог methods --
      назва_функції_action ( параметри ) {
        ... зміна значень у state через this...
      },
      . . . . . .
    },

  }
)
```

Приклад

```
import { defineStore } from 'pinia'

export const useCounterStore = defineStore(
  'counter',
  {

    state: () => (
               {
                 count: 0,
                 name: 'Дмитро'
               }
             ),
    getters: {
       doubleCount: (state) => state.count * 2,
       halfCount: (state) => state.count / 2,
    },
    actions: {
      increment() {
        this.count++
      },
      decrement() {
        this.count--
      },
    },
  }
)
```

Для доступу до елеметів сховища використовуємо ***this***

Доступ до значень та геттерів сховища у компонентах

Компонент

```vue
<template>
    <div class="home">




    </div>
</template>

<script>
import { mapState } from 'pinia'


export default {
    name: 'HomeView',




}
</script>
```

1. Імпортуємо mapState

Сховище

```js
import { defineStore } from 'pinia'

export const useCounterStore = defineStore('counter', {
    state: () => ({
        count: 7,
        name: 'Eduardo',
    }),
    getters: {
        doubleCount: (state) => state.count * 2,
    },
    actions: {
        increment() {
            this.count++
        },
    },
})
```

Доступ до значень та геттерів сховища

Компонент

```vue
<template>
    <div class="home">




    </div>
</template>

<script>
import { mapState } from 'pinia'
import { useCounterStore } from '../store/counter'

export default {
    name: 'HomeView',




}
</script>
```

1. Імпортуємо mapState

2. Імпортуємо сховище

Сховище

```js
import { defineStore } from 'pinia'

export const useCounterStore = defineStore('counter', {
    state: () => ({
        count: 7,
        name: 'Eduardo',
    }),
    getters: {
        doubleCount: (state) => state.count * 2,
    },
    actions: {
        increment() {
            this.count++
        },
    },
})
```

Доступ до значень та геттерів сховища

Компонент

```vue
<template>
    <div class="home">

    </div>
</template>

<script>
import { mapState } from 'pinia'
import { useCounterStore } from '../store/counter'

export default {
    name: 'HomeView',

    computed: {
      ...mapState(useCounterStore, ['count', 'doubleCount']),
    },
}
</script>
```

1. Імпортуємо mapState

2. Імпортуємо сховище

3. Імпортуємо властивості та геттери

Сховище

```js
import { defineStore } from 'pinia'

export const useCounterStore = defineStore('counter', {
    state: () => ({
        count: 7,
        name: 'Eduardo',
    }),
    getters: {
        doubleCount: (state) => state.count * 2,
    },
    actions: {
        increment() {
            this.count++
        },
    },
})
```

Доступ до значень та геттерів сховища

Компонент

```vue
<template>
    <div class="home">
        <div>{{ count }}</div>
        <div>{{ doubleCount }}</div>

                4. Використовуємо властвості та геттери


    </div>
</template>

                1. Імпортуємо  mapState
<script>
import { mapState } from 'pinia'
import { useCounterStore } from '../store/counter'

export default {                2. Імпортуємо сховище
    name: 'HomeView',

                3. Імпортуємо властвості та геттери
    computed: {
      ...mapState(useCounterStore, ['count', 'doubleCount']),
    },
}
</script>
```

Сховище

```vue
import { defineStore } from 'pinia'

export const useCounterStore = defineStore('counter', {
    state: () => ({
        count: 7,
        name: 'Eduardo',
    }),
    getters: {
        doubleCount: (state) => state.count * 2,
    },
    actions: {
        increment() {
            this.count++
        },
    },
})
```

Використання action-методів сховища

Компонент

```vue
<template>
    <div class="home">




    </div>
</template>

<script>
import { mapState, mapActions } from 'pinia'


export default {
    name: 'HomeView',

    computed: {
        ...mapState(useCounterStore, ['count',
'doubleCount']),
    },




        reset() {
            const counter = useCounterStore()
            counter.$reset()
        },
    },
}
</script>
```

1. Імпортуємо mapActions

Сховище

```js
import { defineStore } from 'pinia'

export const useCounterStore = defineStore('counter', {
    state: () => ({
        count: 7,
        name: 'Eduardo',
    }),
    getters: {
        doubleCount: (state) => state.count * 2,
    },
    actions: {
        increment() {
            this.count++
        },
    },
})
```

Використання action-методів сховища

Компонент

```vue
<template>
    <div class="home">

    </div>
</template>

<script>
import { mapState, mapActions } from 'pinia'
import { useCounterStore } from '../store/counter'

export default {
    name: 'HomeView',

    computed: {
        ...mapState(useCounterStore, ['count',
'doubleCount']),
    },

        reset() {
            const counter = useCounterStore()
            counter.$reset()
        },
    },
}
</script>
```

1. Імпортуємо mapActions

2. Імпортуємо сховище

Сховище

```vue
import { defineStore } from 'pinia'

export const useCounterStore = defineStore('counter', {
    state: () => ({
        count: 7,
        name: 'Eduardo',
    }),
    getters: {
        doubleCount: (state) => state.count * 2,
    },
    actions: {
        increment() {
            this.count++
        },
    },
})
```

Використання action-методів сховища

Компонент

```vue
<template>
    <div class="home">




    </div>
</template>

<script>
import { mapState, mapActions } from 'pinia'
import { useCounterStore } from '../store/counter'

export default {
    name: 'HomeView',

    computed: {
        ...mapState(useCounterStore, ['count',
'doubleCount']),
    },

    methods: {
        ...mapActions(useCounterStore, ['increment']),
        reset() {
            const counter = useCounterStore()
            counter.$reset()
        },
    },
}
</script>
```

1. Імпортуємо mapActions

2. Імпортуємо сховище

3. Імпортуємо action-методи

Сховище

```js
import { defineStore } from 'pinia'

export const useCounterStore = defineStore('counter', {
    state: () => ({
        count: 7,
        name: 'Eduardo',
    }),
    getters: {
        doubleCount: (state) => state.count * 2,
    },
    actions: {
        increment() {
            this.count++
        },
    },
})
```

Використання action-методів сховища

Компонент

```vue
<template>
    <div class="home">
        <div>{{ count }}</div>
        <div>{{ doubleCount }}</div>
        <div>
            <button @click="increment">Add</button>
        </div>
    </div>
</template>

<script>
import { mapState, mapActions } from 'pinia'
import { useCounterStore } from '../store/counter'

export default {
    name: 'HomeView',

    computed: {
        ...mapState(useCounterStore, ['count',
'doubleCount']),
    },

    methods: {
        ...mapActions(useCounterStore, ['increment']),
        reset() {
            const counter = useCounterStore()
            counter.$reset()
        },
    },
}
</script>
```

4. Використовуємо методи

1. Імпортуємо mapActions

2. Імпортуємо сховище

3. Імпортуємо action-методи

Сховище

```js
import { defineStore } from 'pinia'

export const useCounterStore = defineStore('counter', {
    state: () => ({
        count: 7,
        name: 'Eduardo',
    }),
    getters: {
        doubleCount: (state) => state.count * 2,
    },
    actions: {
        increment() {
            this.count++
        },
    },
})
```

Використання сховищ у інших сховищах

- - - - - - otherCounter.js - - - - - -

```
import { defineStore } from 'pinia'

export const useOtherCounterStore = defineStore('otherCounter',
{
    state: () => ({
        count: 10,
        testName: 'Ivan',
    }),
    getters: {
        doubleCount: (state) => state.count * 2,
    },
    actions: {
        increment2() {
            this.count++
        },
    },
})
```

- - - - - - counter.js - - - - - -

```
import { defineStore } from 'pinia'
//----- Підключаємо інше сховище -------
import { useOtherCounterStore } from './otherCounter'

export const useCounterStore = defineStore('counter', {
    state: () => ({
        count: 7,
        name: 'Eduardo',
    }),
    getters: {
        doubleCount: (state) => state.count * 2,
    },
    actions: {
        increment() {
            this.count++
            //--- використовуємо ---
            const otherCounter = useOtherCounterStore()
            otherCounter.increment2()
        },
    },
})
```

Використання декількох сховищ

```vue
<template>
    <div class="home">
        <div>{{ count }}</div>
        <div>{{ doubleCount }}</div>
        <div>
            <button @click="increment">Add</button>
        </div>

        <div>{{ count2 }}</div>
        <div>{{ doubleCount2 }}</div>
    </div>
</template>
<script>
import { mapState, mapActions } from 'pinia'
import { useCounterStore } from '../store/counter'
import { useOtherCounterStore } from '../store/otherCounter'
export default {
    name: 'HomeView',

    computed: {
        ...mapState(useCounterStore, ['count', 'doubleCount']),
        ...mapState(useOtherCounterStore, { count2: 'count',
        doubleCount2: 'doubleCount' }),
    },
    methods: {
        ...mapActions(useCounterStore, ['increment']),
    },
}
</script>
```

Назви співпадають, тому задаємо нову назву

otherCounter.js

```js
import { defineStore } from 'pinia'

export const useOtherCounterStore = defineStore('otherCounter',
{
    state: () => ({
        count: 10,
        testName: 'Ivan',
    }),
    getters: {
        doubleCount: (state) => state.count * 2,
    },
    actions: {
        increment2() {
            this.count++
        },
    },
})
```

counter.js

```js
import { defineStore } from 'pinia'
//----- Підключаємо інше сховище -------
import { useOtherCounterStore } from './otherCounter'

export const useCounterStore = defineStore('counter', {
    state: () => ({
        count: 7,
        name: 'Eduardo',
    }),
    getters: {
        doubleCount: (state) => state.count * 2,
    },
    actions: {
        increment() {
            this.count++
            //--- використовуємо ---
            const otherCounter = useOtherCounterStore()
            otherCounter.increment2()
        },
    },
})
```

Скидання стану

```
store.$reset()
```

Зміна стану

```
store.$patch({
  count: store.count + 1,
  age: 120,
  name: 'DIO',
})
```

```
// це насправді не замінює `$state`
store.$state = { count: 24 }
// а внутрішньо викликає `$patch()`:
store.$patch({ count: 24 })
```

Зміна стану

Зміна стану з використанням об’єкта

```
store.$patch({
  count: store.count + 1,
  age: 120,
  name: 'DIO',
})
```

Зміна стану з використанням функції (якщо треба змінити якісь окремі властивості об’єктів або додати елемент до масиву, ….)

```
store.$patch((state) => {
  state.items.push({ name: 'shoes', quantity: 1 })
  state.hasChanged = true
})
```

Заміна стану

Ви не можете саме замінити стан сховища, оскільки це порушить реактивність. Однак ви можете його підправити:

```
// це насправді не замінює `$state`
store.$state = { count: 24 }
// а внутрішньо викликає `$patch()`:
store.$patch({ count: 24 })
```

Опційні сховища

```
import { defineStore } from 'pinia'

export const useНазваStore = defineStore(
  'назва_сховища',
  {
    //Дані сховища (аналог data())
    state: () => (
          {
            властивість1 : поч.знач.1,
            властивість2 : поч.знач.2,
            . . . . . .
          }
        ),
    //Функції, що дозволяють отримати обч. значення з state
    getters: {
      //-- аналог обчислювальних властивостей computed --
      назва_геттера1: (state) => обч._значення_зі_state,
      . . . . . .
    },
    //Функції, що дозволяють робити зміни у state
    actions: {
      //-- аналог methods --
      назва_функції_action ( параметри ) {
        ... зміна значень у state через this...
      },
      . . . . . .
    },
 }
)
```

Setup сховища

```
import { defineStore } from 'pinia'

export const useНазваStore = defineStore(
  'назва_сховища',
  () =>  {
//Дані сховища
    const властивість1 = ref (поч.знач.1)
    const властивість2 = ref (поч.знач.2)
     . . . . . .
//Функції, що дозволяють отримати обчислене значення
   const назва_геттера1 = computed(() => значення)
   const назва_геттера2 = computed(() => значення)
     . . . . . .
//Функції, що дозволяють робити зміни
    function назва_функції_action1() {
      ... зміни значень value властивостей ...
    }
     . . . . . .
//-- експортуємо властивості, геттери, функції_action
    return {
властивість1,…,геттер1,…,функція_action1,…}
 }
)
```

Setup сховища

```
import { defineStore } from 'pinia'

export const useНазваStore = defineStore(
  'назва_сховища',
  () =>  {
    //Дані сховища
    const властивіть1 = ref (поч.знач.1)
    const властивіть2 = ref (поч.знач.2)
     . . . . . .
    //Функції, що дозволяють отримати обчислене значення
   const назва_геттера1 = computed(() => значення)
   const назва_геттера2 = computed(() => значення)
     . . . . . .
    //Функції, що дозволяють робити зміни
    function назва_функції_action1() {
      ... зміни значень value властивостей ...
    }
     . . . . . .
    //-- експортуємо властивості, геттери, функції_action
    return { властивість1,…,геттер1,…,функція_action1,…}
 }
)
```

```
import { defineStore } from 'pinia'

export const useCounterStore = defineStore(
  'counter',
  () =>  {
  //-- Стають властивостями стану --
  const count = ref(0)
  const name = ref('Дмитро')
  //-- Стають гетерами
  const doubleCount = computed(() => count.value * 2)
  //-- Стають action-функціями
  function increment() {
    count.value++
  }
  function decrement() {
    count.value--
  }
  return { count, name, doubleCount, increment }
 }
)
```

Опційні сховища	Setup сховища
import { defineStore } from 'pinia' export const useCounterStore = defineStore('counter', {	import { defineStore } from 'pinia' export const useCounterStore = defineStore('counter', () => {
state: () => ({ count: 0, name: 'Дмитро' }),	//-- Стають властивостями стану -- const count = ref(0) const name = ref('Дмитро')
getters: { doubleCount: (state) => state.count * 2, },	//-- Стають гетерами const doubleCount = computed(() => count.value * 2)
actions: { increment() { this.count++ }, decrement() { this.count-- }, },	//-- Стають action-функціями function increment() { count.value++ } function decrement() { count.value-- }
})	return { count, name, doubleCount, increment } })

Використання

Загальна схема використання

```
<script setup>
import { useСховищеStore } from '…шлях…'

const сховище = useСховищеStore

</script>

<template>
  <div>
    <button @click=" сховище.метод ">...</button>
    <div>{{ сховище.властивість1 }}</div>
    <div>{{ сховище.властивість1 }}</div>
    . . . . . .
  </div>
</template>
```

Опис сховища

```
import { defineStore } from 'pinia'

export const useНазваStore = defineStore(
  'назва_сховища',
  () =>  {

    //Дані сховища
    const властивіть1 = ref (поч.знач.1)
    const властивіть2 = ref (поч.знач.2)
     . . . . . .

    //Функції, що дозволяють отримати обчислене значення
   const назва_геттера1 = computed(() => значення)
   const назва_геттера2 = computed(() => значення)
     . . . . . .

    //Функції, що дозволяють робити зміни
    function назва_функції_action1() {
      ... зміни значень value властивостей ...
    }
     . . . . . .


    //-- експортуємо властивості, геттери, функції_action
    return { властивість1,…,геттер1,…,функція_action1,…}
 }
)
```

Використання

Загальна форма	*Приклад*
<script setup> import { useСховищеStore } from '…шлях…' const *сховище* = useСховищеStore() </script> <template> <button @click=" *сховище.метод* ">...</button> {{ *сховище.**властивість1*** }} {{ *сховище.**властивість2*** }} . . . . . . </template>	<script setup> import { useCounterStore } from '@/stores/counter' const counterStore = useCounterStore() </script> <template> <button @click="counterStore.increment1">Do increment1</button> {{ counterStore.count1 }} {{ counterStore.count2 }} {{ counterStore.doubleCount1 }} <button @click="counterStore.increment2">Do increment2</button> </template>

Деструктурування з Store. storeToRefs

Загальна форма без деструктуризації	*Загальна форма з деструктуризацією*
<script setup> import { useСховищеStore } from '…шлях…' const *сховище* = useСховищеStore() </script> <template> <button @click=" *сховище.метод1* ">...</button> {{ *сховище.властивість1* }} {{ *сховище.властивість2* }} . . . . . . </template>	<script setup> import { useСховищеStore } from '…шлях…' import { ***storeToRefs*** } from 'pinia' const *сховище* = useСховищеStore //--- імпорт реактивних даних (посилань), геттерів --- const{ ***властивість1, властивість2, …*** }=***storeToRefs***(*сховище*) //--- імпорт action-методів(не потрібно ***storeToRefs***) --- const { *метод1, метод2, …*} = *сховище* </script> <template> <button @click=" *метод* ">...</button> {{ ***властивість1*** }} {{ ***властивість2*** }} . . . . . . </template>

Без деструктуризації

```
import { useCounterStore } from '@/stores/counter'




const counterStore = useCounterStore()









</script>

<template>
  <div>
    <button @click="counterStore.increment1">
      Do increment1
    </button>
    <div>{{ counterStore.count1 }}</div>
    <div>{{ counterStore.count2 }}</div>
    <div>{{ counterStore.doubleCount1 }}</div>
    <button @click="counterStore.increment2">
       Do increment2
    </button>
  </div>
</template>
```

Без деструктуризації

```
import { useCounterStore } from '@/stores/counter'



const counterStore = useCounterStore()









</script>

<template>
  <div>
    <button @click="counterStore.increment1">
      Do increment1
    </button>
    <div>{{ counterStore.count1 }}</div>
    <div>{{ counterStore.count2 }}</div>
    <div>{{ counterStore.doubleCount1 }}</div>
    <button @click="counterStore.increment2">
       Do increment2
    </button>
  </div>
</template>
```

Звикористанням деструктуризації

```
<script setup>
import { useCounterStore } from '@/stores/counter'
import { storeToRefs } from 'pinia'

const counterStore = useCounterStore()
//---- імпорту властивостей і геттерів ---
const { count1, count2, doubleCount1 } = storeToRefs(counterStore)

//---- імпорту action-методів ---
const { increment1, increment2 } = counterStore
</script>

<template>
  <div>
    <button @click="increment1">
       Do increment1
    </button>
    <div>{{ count1 }}</div>
    <div>{{ count2 }}</div>
    <div>{{ doubleCount1 }}</div>
    <button @click="increment2">
      Do increment2
    </button>
  </div>
</template>
```

Підписка на стан

Ви можете спостерігати за станом та його змінами за допомогою методу сховища $subscribe()

```
cartStore.$subscribe((mutation, state) => {
  // import { MutationType } from 'pinia'
  mutation.type // 'прямі зміни' | 'об'єкт з оновленими значеннями' | 'функці
  // теж саме, що й cartStore.$id
  mutation.storeId // 'cart'
  // доступний лише з mutation.type === 'об'єкт з оновленими значеннями'
  mutation.payload // об'єкт з оновленими значеннями передається в cartStore.

  // зберігати весь стан у локальному сховищі щоразу, коли він змінюється
  localStorage.setItem('cart', JSON.stringify(state))
})
```

Підписка на стан

Ви можете спостерігати за станом та його змінами за допомогою методу сховища $subscribe()

```
cartStore.$subscribe((mutation, state) => {
  // import { MutationType } from 'pinia'
  mutation.type // 'прямі зміни' | 'об'єкт з оновленими значеннями' | 'функці
  // теж саме, що й cartStore.$id
  mutation.storeId // 'cart'
  // доступний лише з mutation.type === 'об'єкт з оновленими значеннями'
  mutation.payload // об'єкт з оновленими значеннями передається в cartStore.

  // зберігати весь стан у локальному сховищі щоразу, коли він змінюється
  localStorage.setItem('cart', JSON.stringify(state))
})
```

Інший підхід : задаємо watch на об’єкт стану (pinia.state)

```
watch(
  pinia.state,
  (state) => {
    // зберігати весь стан у локальному сховищі щоразу, коли він змінюється
    localStorage.setItem('piniaState', JSON.stringify(state))
  },
  { deep: true }
)
```

Якщо ви також хочете зберегти їх після демонтажу компонента, передайте { detached: true } як другий аргумент, щоб *відокремити підписку на стан* від поточного компонента

```
<script setup>
const someStore = useSomeStore()


// ця підписка буде збережена навіть після того, як компонент буде демонтовано
someStore.$subscribe(callback, { detached: true })
</script>
```