

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%2FuvB6xeObn6ITYLy2IRJWZxSqnIbB6P41f83y1IKczNxc5%2FXIk%2B%2B0BZ4zKpPSZAJVveEHzpROesCJ%2BJn%2FDw3tabttXgwwFVzWJC5yAmstJJE%2FNNULCJEmi1%2BOzPW7eyrok0Cg4HIqMhJoGVNp2DiUPZnqoYK71ugiNzMXos5bTk4%2FOWipsxuJeSVOeaD5N2CfDLQAOGNM0dVKCxQseiNUaYaKNe0bbAva9zmlbTlxVYNZlGPw3R0i0gs5WBAQiFHn0OlcQxcFP3F%2BKksqg3%2BHmlORkGDjnbjEdjX9pnZNh%2BqaVgg2p3ICXvQg%3D%3D>

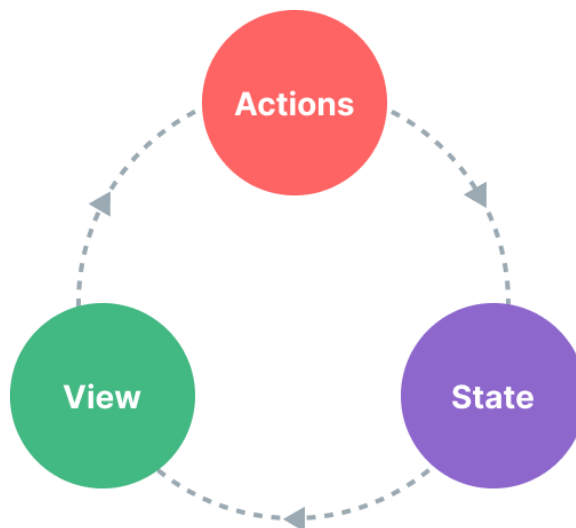
Що таке сховище?

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.

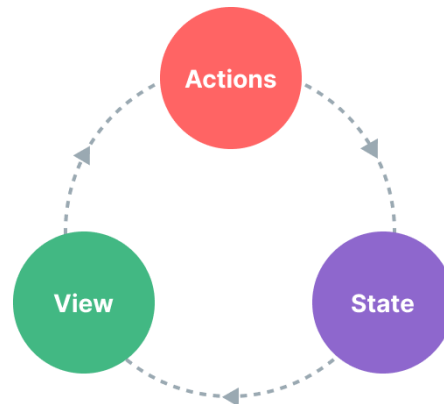


Переваги від використання Pinia:

- 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	<pre>yarn add pinia # або з npm npm install pinia</pre>
Додаємо до проекту як плагін	<pre>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')</pre>

Визначення сховища

<p>Створення сховища</p> <p>Рекомендовано, щоб ідентифікатор (є обов'язковим) експортованого об'єкта сховища містила слова `use` та `Store` (наприклад, `useUserStore`, `useCartStore`, `useProductStore`)</p>	<pre>import { defineStore } from 'pinia' export const useНазваStore = defineStore('назва_сховища', { // функція налаштувань або об'єкт опцій })</pre>	<pre>import { defineStore } from 'pinia' export const useCounterStore = defineStore('counter', { }) export const useUsersStore = defineStore('users', { })</pre>
--	--	--

Опційні сховища

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

Опційні сховища

Загальна форма	Приклад
<pre>import { <i>defineStore</i> } from 'pinia' export const use<i>Назва</i>Store = <i>defineStore</i>('<i>назва_сховища</i>', { //Дані сховища (аналог <i>data()</i>) <u><i>state</i></u>: () => ({ <i>властивість1</i> : поч.знач.1, <i>властивість2</i> : поч.знач.2, }), })</pre>	<pre>import { <i>defineStore</i> } from 'pinia' export const use<i>Counter</i>Store = <i>defineStore</i>('<i>counter</i>', { <u><i>state</i></u>: () => ({ <i>count</i>: 0, <i>name</i>: 'Дмитро' }), })</pre>

Опційні сховища

Загальна форма	Приклад
<pre>import { <i>defineStore</i> } from 'pinia' export const use<i>Назва</i>Store = <i>defineStore</i>('<i>назва_сховища</i>', { //Дані сховища (аналог <i>data()</i>) <u>state</u>: () => ({ <i>властивість1</i> : поч.знач.1, <i>властивість2</i> : поч.знач.2, }), //Функції, що дозволяють <u>отримати обч. значення</u> з <i>state</i> <u>getters</u>: { //-- аналог обчислювальних властивостей <i>computed</i> -- <i>назва_геттера1</i>: (<i>state</i>) => обч._значення_зі_state, }, })</pre>	<pre>import { <i>defineStore</i> } from 'pinia' export const useCounterStore = <i>defineStore</i>('<i>counter</i>', { <u>state</u>: () => ({ <i>count</i>: 0, <i>name</i>: 'Дмитро' }), <u>getters</u>: { <i>doubleCount</i>: (<i>state</i>) => <i>state.count</i> * 2, <i>halfCount</i>: (<i>state</i>) => <i>state.count</i> / 2, }, })</pre>

Опційні сховища

Загальна форма	Приклад
<pre>import { <i>defineStore</i> } from 'pinia' export const use<i>Назва</i>Store = <i>defineStore</i>('<i>назва_сховища</i>', { //Дані сховища (аналог <i>data()</i>) <u>state</u>: () => ({ <i>властивість1</i> : поч.знач.1, <i>властивість2</i> : поч.знач.2, }), //Функції, що дозволяють <u>отримати</u> обч. значення з <i>state</i> <u>getters</u>: { //-- аналог обчислювальних властивостей <i>computed</i> -- <i>назва_геттера1</i>: (<i>state</i>) => обч._значення_зі_state, }, //Функції, що дозволяють <u>робити зміни</u> у <i>state</i> <u>actions</u>: { //-- аналог <i>methods</i> -- <i>назва_функції_action</i> (параметри) { ... зміна значень у <i>state</i> через <i>this</i>... }, }, })</pre>	<pre>import { <i>defineStore</i> } from 'pinia' export const useCounterStore = <i>defineStore</i>('counter', { <u>state</u>: () => ({ count: 0, name: 'Дмитро' }), <u>getters</u>: { doubleCount: (<i>state</i>) => <i>state</i>.count * 2, halfCount: (<i>state</i>) => <i>state</i>.count / 2, }, <u>actions</u>: { increment() { <i>this</i>.count++ }, decrement() { <i>this</i>.count-- }, }, })</pre>

Для доступу до елементів сховища використовуємо **this**

Доступ до значень та геттерів сховища у компонентах

Компонент

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

    </div>
</template>
```

1. Імпортуємо mapState

```
<script>
import { mapState } from 'pinia'
```

```
export default {
  name: 'HomeView',
```

```
}
</script>
```

Сховище

```
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

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

    </div>
</template>
```

1. Імпортуємо mapState

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

2. Імпортуємо сховище

```
export default {
  name: 'HomeView',

}
</script>
```

Сховище

```
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

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

    </div>
</template>
```

1. Імпортуємо mapState

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

2. Імпортуємо сховище

```
export default {
  name: 'HomeView',
```

3. Імпортуємо властивості та геттери

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

Сховище

```
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

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

4. Використовуємо властивості та геттери

```
</div>
</template>
```

1. Імпортуємо mapState

```
<script>
```

```
import { mapState } from 'pinia'
import { useCounterStore } from '../store/counter'
```

2. Імпортуємо сховище

```
export default {
  name: 'HomeView',
```

3. Імпортуємо властивості та геттери

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

Сховище

```
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

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

    </div>
</template>

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

export default {
  name: 'HomeView',

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

  methods: {
    reset() {
      const counter = useCounterStore()
      counter.$reset()
    },
  },
}
```

1. Імпортуємо mapActions

Сховище

```
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

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

    </div>
</template>

1. Імпортуємо mapActions

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

export default {
  name: 'HomeView',

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

  methods: {
    reset() {
      const counter = useCounterStore()
      counter.$reset()
    },
  },
}
```

Сховище

```
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

```
<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

```
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

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

4. Використовуємо методи

1. Імпортуємо mapActions

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

2. Імпортуємо сховище

```
export default {
  name: 'HomeView',

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

3. Імпортуємо action-методи

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

Сховище

```
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

```
<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

```
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()
    },
  },
})
```

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 })
```