

## Pinia

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

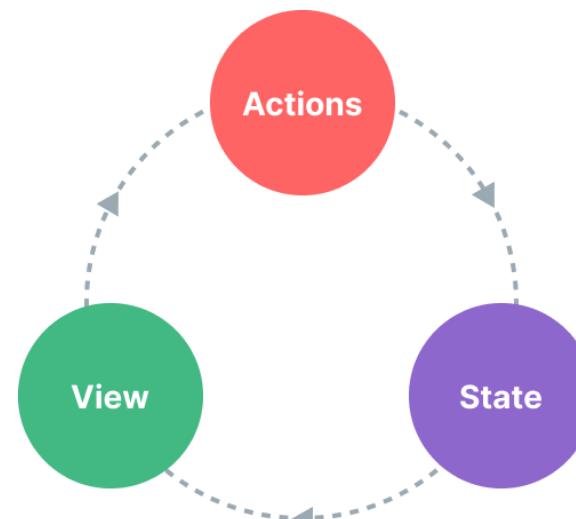
https://storage.googleapis.com/vue-mastery.appspot.com/flamalink/media/Pinia-Cheat-Sheet.pdf?GoogleAccessId.firebaseio-adminsdk-jyioc%40vue-mastery.iam.gserviceaccount.com&Expires=16725225600&Signature=kyMynndU2WMV1ie1HcAdGZkajToy0%2FuvB6xeObn6lTYLy2IRJWZxSqnlbB6P41f83y1IKczNxc5%2FXIk%2B%2B0BZ4zKpPSZAJVveEHzpROesCJ%2BJn%2FDw3tabttXgwwFVzWJC5yAmstJJ%2FNNULCJEmi1%2BOzPW7eyrok0Cg4HIqMhJoGVNp2DiUPZnqoYK71ugiNzMXos5bTk4%2FOwipsxuJeSVOeaD5N2CfDLQAOGNM0dVKCxQseiNUaYaKNe0bbAva9zmlbTlxVYNZIGPw3R0i0gs5WBAQiFHn0OlcQxcFP3F%2BKksqg3%2BHmlORKGDjnbEdjX9pnZNh%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.

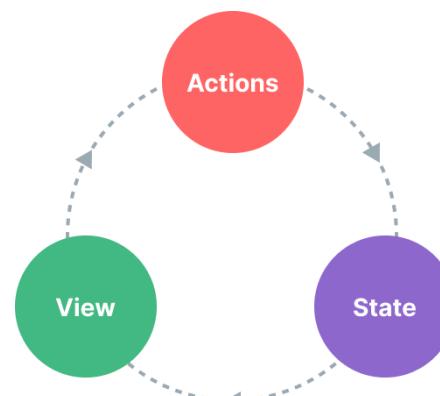


## Переваги від використання Pinia:

- 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: розширяйте 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 проекту 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, 'useUserStore', 'useCartStore', 'useProductStore')

```
import { defineStore } from 'pinia'

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

## Опційні сховища

Загальна форма	Приклад
<pre>import { defineStore } from 'pinia'  export const useНазваStore = defineStore(   'назва_сховища',   {      //Дані сховища (аналог data())     state: () =&gt; (       {         властивіть1 : поч.знач.1,         властивіть2 : поч.знач.2,         . . . . .       }     ),   } )</pre>	<pre>import { defineStore } from 'pinia'  export const useCounterStore = defineStore(   'counter',   {      state: () =&gt; (       {         count: 0,         name: 'Дмитро'       }     ),   } )</pre>

# Опційні сховища

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

# Опційні сховища

Загальна форма	Приклад
<pre>import { defineStore } from 'pinia'  export const useНазваStore = defineStore(   'назва_сховища',   {      //Дані сховища (аналог data())     state: () =&gt; (       {         властивіть1 : поч.знач.1,         властивіть2 : поч.знач.2,         . . . . .       }     ),     //Функції, що дозволяють <u>отримати</u> обч. значення з state     getters: {       //-- аналог обчислювальних властивостей computed --       назва_геттера1: (state) =&gt; обч._значення_зі_state,       . . . . .     },     //Функції, що дозволяють <u>робити зміни</u> у state     actions: {       //-- аналог methods --       назва_функції_action ( параметри ) {         ... зміна значень у state через this...       },       . . . . .     },   } )</pre>	<pre>import { defineStore } from 'pinia'  export const useCounterStore = defineStore(   'counter',   {      state: () =&gt; (       {         count: 0,         name: 'Дмитро'       }     ),     getters: {       doubleCount: (state) =&gt; state.count * 2,       halfCount: (state) =&gt; state.count / 2,     },     actions: {       increment() {         this.count++       },       decrement() {         this.count--       },     },   } )</pre>

Для доступу до елементів  
сховища використовуємо this

## Доступ до значень та геттерів сховища у компонентах

Компонент	Сховище
<pre>&lt;template&gt;   &lt;div class="home"&gt;    &lt;/div&gt; &lt;/template&gt;  1. Імпортуємо mapState &lt;script&gt; import { mapState } from 'pinia'  export default {   name: 'HomeView', }  &lt;/script&gt;</pre>	<pre>import { defineStore } from 'pinia'  export const useCounterStore = defineStore('counter', {   state: () =&gt; ({     count: 7,     name: 'Eduardo',   }),   getters: {     doubleCount: (state) =&gt; state.count * 2,   },   actions: {     increment() {       this.count++     },   } })</pre>

## Доступ до значень та геттерів сховища

Компонент	Сховище
<pre>&lt;template&gt;   &lt;div class="home"&gt;    &lt;/div&gt; &lt;/template&gt;  1. Імпортуємо mapState &lt;script&gt; import { mapState } from 'pinia' import { useCounterStore } from '../store/counter'  export default {   name: 'HomeView', }  &lt;/script&gt;</pre>	<pre>import { defineStore } from 'pinia'  export const useCounterStore = defineStore('counter', {   state: () =&gt; ({     count: 7,     name: 'Eduardo',   }),   getters: {     doubleCount: (state) =&gt; state.count * 2,   },   actions: {     increment() {       this.count++     },   } })</pre>

## Доступ до значень та геттерів сховища

Компонент	Сховище
<pre>&lt;template&gt;   &lt;div class="home"&gt;    &lt;/div&gt; &lt;/template&gt;  1. Імпортуємо mapState  &lt;script&gt; import { mapState } from 'pinia' import { useCounterStore } from '../store/counter'  export default {   name: 'HomeView',   computed: {     ...mapState(useCounterStore, ['count', 'doubleCount']),   }, }&lt;/script&gt;</pre>	<pre>import { defineStore } from 'pinia'  export const useCounterStore = defineStore('counter', {   state: () =&gt; ({     count: 7,     name: 'Eduardo',   }),   getters: {     doubleCount: (state) =&gt; state.count * 2,   },   actions: {     increment() {       this.count++     },   } })</pre>

## Доступ до значень та геттерів сховища

Компонент	Сховище
<pre>&lt;template&gt;   &lt;div class="home"&gt;     &lt;div&gt;{{ count }}&lt;/div&gt;     &lt;div&gt;{{ doubleCount }}&lt;/div&gt;      4. Використовуємо властивості та геттери    &lt;/div&gt; &lt;/template&gt;    1. Імпортуємо mapState &lt;script&gt; import { mapState } from 'pinia' import { useCounterStore } from '../store/counter'    2. Імпортуємо сховище export default {   name: 'HomeView',   computed: {     ...mapState(useCounterStore, ['count', 'doubleCount']),   }, }&lt;/script&gt;</pre>	<pre>import { defineStore } from 'pinia'  export const useCounterStore = defineStore('counter', {   state: () =&gt; ({     count: 7,     name: 'Eduardo',   }),   getters: {     doubleCount: (state) =&gt; state.count * 2,   },   actions: {     increment() {       this.count++     },   } })</pre>

## Використання action-методів сховища

Компонент	Сховище
<pre>&lt;template&gt;   &lt;div class="home"&gt;    &lt;/div&gt; &lt;/template&gt; 1. Імпортуємо mapActions  &lt;script&gt; import { mapState, mapActions } from 'pinia'  export default {   name: 'HomeView',    computed: {     ...mapState(useCounterStore, ['count', 'doubleCount']),   }    methods: {     const counter = useCounterStore()     counter.\$reset()   }, } &lt;/script&gt;</pre>	<pre>import { defineStore } from 'pinia'  export const useCounterStore = defineStore('counter', {   state: () =&gt; ({     count: 7,     name: 'Eduardo',   }),   getters: {     doubleCount: (state) =&gt; state.count * 2,   },   actions: {     increment() {       this.count++     },   } })</pre>

## 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()
    },
  }
}
</script>
```

### 2. Імпортуємо сховище

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

### 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>
```

#### 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' }), Назви співпадають,
      тому задаємо нову назву
  },
  methods: {
    ...mapActions(useCounterStore, ['increment']),
  },
}
</script>
```

## 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'
//---- Підключасмо 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()
    },
  },
})
```

Назви співпадають,  
тому задаємо нову назву

## 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 })
```