Vuex. Modules

Для чого потрібні модулі?

У випадку, коли у додатку використовуються різні сутності.

Наприклад, зберігаються дані:

- про користувачів,
- про товари
- про виробників
- про постачальників
- про замовлення
- . . . . . .

Для чого потрібні модулі?

У випадку, коли у додатку використовуються різні сутності то :

1)у state доведеть зберігати дані різних сутностей

2)у getters доводиться зберіагти геттери для роботи з різними сутностями

3)у mutations мутації для роботи з різними сутностями

. . . .

Для чого потрібні модулі?

У випадку, коли у додатку використовуються різні сутності то :

1)у state доведеть зберігати дані різних сутностей

2)у getters доводиться зберіагти геттери для роботи з різними сутностями

3)у mutations мутації для роботи з різними сутностями

. . . .

Легко заплутатись до яких сутностей відносяться методи для роботи з даними

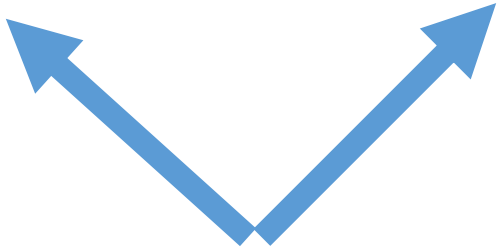
```
const store = createStore({
    state: () => {
        return {
            products: [ ···
            ],
            users: [ ···
            ],
        }
    },
    getters:{
        products: ({ products }) => products,
        users: ({ users }) => users,
        getUserProductsData: (state) => (userId) => {
        },
    },
    mutations : {
        editProduct(state, editedProduct) {···
        },
        deleteProduct(state, productId) {···
        },
        addUser(state, user) {···
        },
        editUser(state, editedUser) {···
        },
        deleteUser(state, userId) {···
        },
    },
```

Модуль 1

```
const moduleA = {
  state: () => ({ ... }),
  mutations: { ... },
  actions: { ... },
  getters: { ... }
}
```

Модуль 2

```
const moduleB = {
  state: () => ({ ... }),
  mutations: { ... },
  actions: { ... }
}
```



1) Обробку даних окремих сутностей виносимо в окремі модулі

Модуль 1

```
const moduleA = {
  state: () => ({ ... }),
  mutations: { ... },
  actions: { ... },
  getters: { ... }
}
```

Модуль 2

```
const moduleB = {
  state: () => ({ ... }),
  mutations: { ... },
  actions: { ... }
}
```

1) Обробку даних окремих сутностей виносимо в окремі модулі

2)Створення кореневого сховища, яке об'єднує модулі

```
const store = createStore({
  state() { . . . . . },
  getters: { . . . . . },
  mutations: { . . . . . },
  actions: { . . . . . },

  //---підключення модулів ---
  modules:{
      модуль1,
      модуль2,
       . . . . .
      нова_назва_модуля : модуль
       . . . . .
   }
})
```

```
const store = createStore({
  modules: {
    a: moduleA,
    b: moduleB
  }
})
```

```
store.state.a // -> `moduleA`'s state
store.state.b // -> `moduleB`'s state
```

products.js store U

src > store > products.js > getters > products

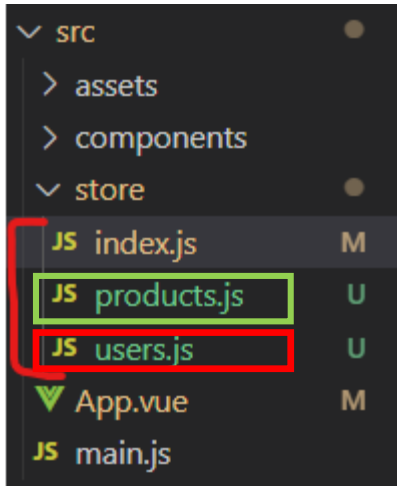
```
const state = {
    products: [
        { id: 1, title: 'Продукт 1', price: 100 },
        { id: 2, title: 'Продукт 2', price: 200 },
        { id: 3, title: 'Продукт 3', price: 150 },
        { id: 4, title: 'Продукт 4', price: 120 },
        { id: 5, title: 'Продукт 5', price: 180 },
    ],
}
const getters = {
    products: ({ products }) => products,
}
const mutations = {…
}

const actions = {…
}

export default {
    namespaced: true,
    state,
    getters,
    mutations,
    actions,
}
```

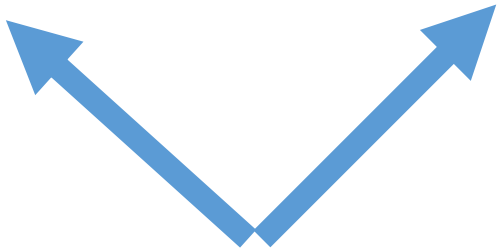
Приклад. Сховище складається з двох частин:
1)для роботи з списком товарів
2)для роботи з списком користувачів

users.js store U

src > store > users.js > getters

```
const state = {
    users: [
        { id: 1, name: 'Користувач 1', productsList: [1, 3] },
        { id: 2, name: 'Користувач 2', productsList: [2, 4] },
        { id: 3, name: 'Користувач 3', productsList: [1, 5] },
        { id: 4, name: 'Користувач 4', productsList: [2, 3, 4] },
        { id: 5, name: 'Користувач 5', productsList: [5] },
    ],
}

const getters = {…
}

const mutations = {…
}

const actions = {…
}

export default {
    namespaced: true,
    state,
    getters,
    mutations,
    actions,
}
```





1) Обробку даних окремих сутностей виносимо в окремі модулі

Приклад. Сховище складається з двох частин:
1)для роботи з списком товарів
2)для роботи з списком користувачів

products.js store U

src > store > products.js > getters > products

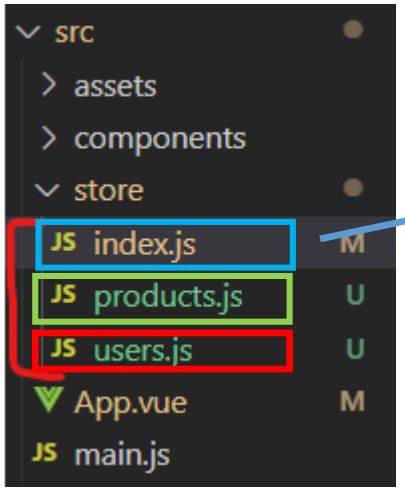
```
const state = {
    products: [
        { id: 1, title: 'Продукт 1', price: 100 },
        { id: 2, title: 'Продукт 2', price: 200 },
        { id: 3, title: 'Продукт 3', price: 150 },
        { id: 4, title: 'Продукт 4', price: 120 },
        { id: 5, title: 'Продукт 5', price: 180 },
    ],
}
const getters = {
    products: ({ products }) => products,
}
const mutations = {…
}

const actions = {…
}

export default {
    namespaced: true,
    state,
    getters,
    mutations,
    actions,
}
```

users.js store U

src > store > users.js > getters

```
const state = {
    users: [
        { id: 1, name: 'Користувач 1', productsList: [1, 3] },
        { id: 2, name: 'Користувач 2', productsList: [2, 4] },
        { id: 3, name: 'Користувач 3', productsList: [1, 5] },
        { id: 4, name: 'Користувач 4', productsList: [2, 3, 4] },
        { id: 5, name: 'Користувач 5', productsList: [5] },
    ],
}

const getters = {…
}

const mutations = {…
}

const actions = {…
}

export default {
    namespaced: true,
    state,
    getters,
    mutations,
    actions,
}
```

1) Обробку даних окремих сутностей виносимо в окремі модулі



index.js store M

src > store > index.js > ...

You, 3 seconds ago | 1 author (You)

```
import { createStore } from 'vuex'
import productsModule from './products'
import usersModule from './users'

// Create a new store instance.
const store = createStore({
    modules: {
        products: productsModule,
        users: usersModule,
    },
})
export default store
```

2)Створення кореневого сховища

Використання ***геттерів з модулів*** у компонентах

Загальна схема

```
<template>




</template>
<script>
//1. Імпортуємо mapGetters
import { mapGetters } from 'vuex'

export default {




}
</script>
```

Приклад

App.vue src M

src > App.vue > {} "App.vue" > script

You, 2 minutes ago | 1 author (You)

```
1   <template>
2   ···
3   ···
4   ···
5   ···
6   ···
7   </template>
8
9   <script>
10  import { mapGetters } from 'vuex'
11
12  export default {
13  ···  name: 'App',
14
15  ···  components: {},
16
17  ···  data() {
18  ···  ···  return {}
19  ···  },
20
21  ···  computed: {
22  ···  ··
23  ···  ··
24  ···  },
25
26  ···  methods: {},
27  }
28  </script>
29
```

Використання ***геттерів з модулів*** у компонентах

Загальна схема

```
<template>

</template>
<script>
//1. Імпортуємо mapGetters
import { mapGetters } from 'vuex'

export default {
     . . . . . .

//2. Підключаємо геттери
 computed: {
    ...mapGetters('назва_модуля',[‘геттер1‘,...]),
 },
     . . . . . .
}
</script>
```

Приклад

App.vue src M

src > App.vue > {} "App.vue" > script

You, 2 minutes ago | 1 author (You)

```
1   <template>
2   ...
3   ...
4   ...
5   ...
6   ...
7   </template>
8
9   <script>
10  import { mapGetters } from 'vuex'
11
12  export default {
13      name: 'App',
14
15      components: {},
16
17      data() {
18          return {}
19      },
20
21      computed: {
22          ...mapGetters('users', ['users', 'getUserProductsData']
23      ...
24      },
25
26      methods: {},
27  }
28  </script>
29
```

Використання ***геттерів з модулів*** у компонентах

Загальна схема

```
<template>
//3. Використовуємо геттери як звичайну обчислювальну властивість
      . . . . . .
    <тег>{{ геттер }}</тег >
      . . . . . .
    <тег :атрибут = “геттер” />
      . . . . . .
</template>
<script>
//1. Імпортуємо mapGetters
import { mapGetters } from 'vuex'

export default {
     . . . . . .

//2. Підключаємо геттери
 computed: {
    ...mapGetters('назва_модуля',[‘геттер1’,...]),
 },
     . . . . . .
}
</script>
```

Приклад

App.vue src M

src > App.vue > {} "App.vue" > script

You, 2 minutes ago | 1 author (You)

```
1   <template>
2       <div>{{ users }}</div>
3       <hr />
4       <div>{{ getUserProductsData(1) }}</div>
5
6
7   </template>
8
9   <script>
10  import { mapGetters } from 'vuex'
11
12  export default {
13      name: 'App',
14
15      components: {},
16
17      data() {
18          return {}
19      },
20
21      computed: {
22          ...mapGetters('users', ['users', 'getUserProductsData'])
23
24      },
25
26      methods: {},
27  }
28  </script>
29
```

Використання ***геттерів з модулів*** у компонентах

App.vue src M

src > App.vue > {} "App.vue" > script

You, 2 minutes ago | 1 author (You)

```vue
<template>
    <div>{{ users }}</div>
    <hr />
    <div>{{ getUserProductsData(1) }}</div>
    <hr />
    <div>{{ products }}</div>
</template>

<script>
import { mapGetters } from 'vuex'

export default {
    name: 'App',

    components: {},

    data() {
        return {}
    },

    computed: {
        ...mapGetters('users', ['users', 'getUserProductsData']),
        ...mapGetters('products', ['products']),
    },

    methods: {},
}
</script>
```

users.js store U

src > store > users.js > ...

```js
const state = {
    users: [ ···
    ],
}

const getters = {
    users: ({ users }) => users,
    getUserProductsData: (state, getters,
    rootState, rootGetters) => (userId) => {
    },
}

const mutations = { ···
}
```

products.js store U

src > store > products.js > mutations

```js
const state = {
    products: [ ···
    ],
}
const getters = {
    products: ({ products }) => products,
}
const mutations = { ···
}

const actions = { ···
}
```

Використання ***геттерів з модулів*** у компонентах

App.vue src M

src > App.vue > {} "App.vue" > script

You, 2 minutes ago | 1 author (You)

```vue
<template>
    <div>{{ users }}</div>
    <hr />
    <div>{{ getUserProductsData(1) }}</div>
    <hr />
    <div>{{ products }}</div>
</template>

<script>
import { mapGetters } from 'vuex'

export default {
    name: 'App',

    components: {},

    data() {
        return {}
    },

    computed: {
        ...mapGetters('users', ['users', 'getUserProductsData']),
        ...mapGetters('products', ['products']),
    },

    methods: {},
}
</script>
```

users.js store U

src > store > users.js > ...

```js
const state = {
    users: [ ···
    ],
}

const getters = {
    users: ({ users }) => users,
    getUserProductsData: (state, getters,
    rootState, rootGetters) => (userId) => {
    },
}

const mutations = { ···
}
```

products.js store U

src > store > products.js > mutations

```js
const state = {
    products: [ ···
    ],
}
const getters = {
    products: ({ products }) => products,
}
const mutations = { ···
}

const actions = { ···
}
```

Використання ***action-методів з модулів*** у компонентах

Загальна схема

```
<template>









<script>
//1. Імпортуємо mapActions
import { mapActions } from 'vuex'

export default {









}
</script>
```

Приклад

```
<template>
  ..
  ..
  ..
  ..
</template>

<script>
import { mapGetters, mapActions } from 'vuex'

export default {
    name: 'App',

    computed: {
        ...mapGetters('users', ['users',
        'getUserProductsData']),
    },

    methods: {
      ..
    },
}
</script>
```

Використання ***action-методів з модулів*** у компонентах

Загальна схема

```
<template>

</template>

<script>
//1. Імпортуємо mapActions
import { mapActions } from 'vuex'

export default {
      . . . . . .

//2. Підключаємо action-методи
methods: {
...mapActions('назва_модуля',['action-метод']),
},
      . . . . . .
}
</script>
```

Приклад

```
<template>

</template>

<script>
import { mapGetters, mapActions } from 'vuex'

export default {
    name: 'App',

    computed: {
        ...mapGetters('users', ['users',
        'getUserProductsData']),
    },

    methods: {
        ...mapActions('users', ['deleteUser']),
    },
}
</script>
```

Використання ***action-методів з модулів*** у компонентах

Загальна схема

```
<template>
//3. Використовуємо action-методи як звичайні
методи
      . . . . . . . . . . .
    <тег @подія=”action-метод” />
      . . . . . . . . . . .
</template>

<script>
//1. Імпортуємо mapActions
import { mapActions } from 'vuex'

export default {
      . . . . . . . . . . .

//2. Підключаємо action-методи
methods: {
...mapActions(‘назва_модуля’,['action-метод']),
},
      . . . . . . . . . . .
}
</script>
```

Приклад

```
<template>
  <div v-for="user in users" :key="user.id">
    <span>{{ user.name }}</span>
    <button @click="deleteUser(user.id)">Delete</buttor
  </div>
</template>

<script>
import { mapGetters, mapActions } from 'vuex'

export default {
    name: 'App',

    computed: {
        ...mapGetters('users', ['users',
        'getUserProductsData']),
    },

    methods: {
        ...mapActions('users', ['deleteUser']),
    },
}
</script>
```

Використання ***action-методів з модулів*** у компонентах

App.vue src M

src > App.vue > {} "App.vue" > template > div

You, 1 second ago | 1 author (You)

```vue
<template>
  <div v-for="user in users" :key="user.id">
    <span>{{ user.name }}</span>
    <button @click="deleteUser(user.id)">Delete</button>
  </div>
</template>

<script>
import { mapGetters, mapActions } from 'vuex'

export default {
    name: 'App',

    computed: {
        ...mapGetters('users', ['users', 'getUserProductsData']),
    },

    methods: {
        ...mapActions('users', ['deleteUser']),
    },
}
</script>
```

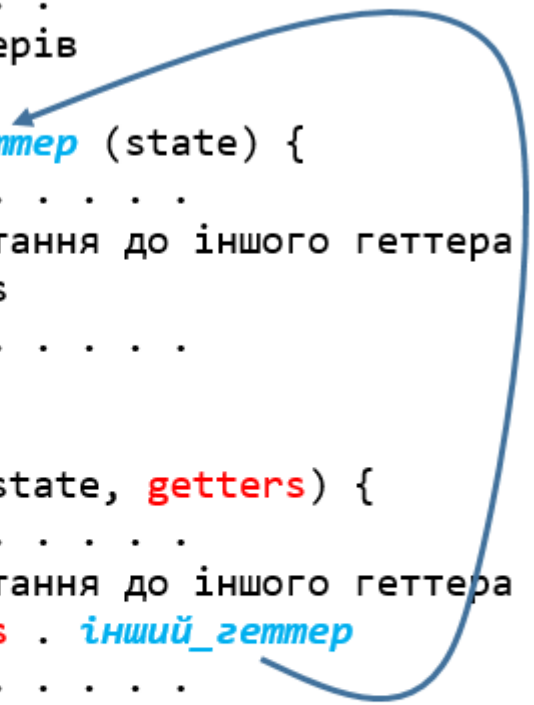
users.js store U

src > store > users.js > mutations

```js
const state = {
    users: [ ···
    ],
}

const getters = {
    users: ({ users }) => users,
    getUserProductsData: (state, getters, rootState, rootGetters) => (userId) => { ···
    },
}

const mutations = { ···
}

const actions = {
    addUser(context, user) { ···
    },
    editUser(context, editedUser) { ···
    },
    deleteUser(context, userId) { ···
    },
}
```

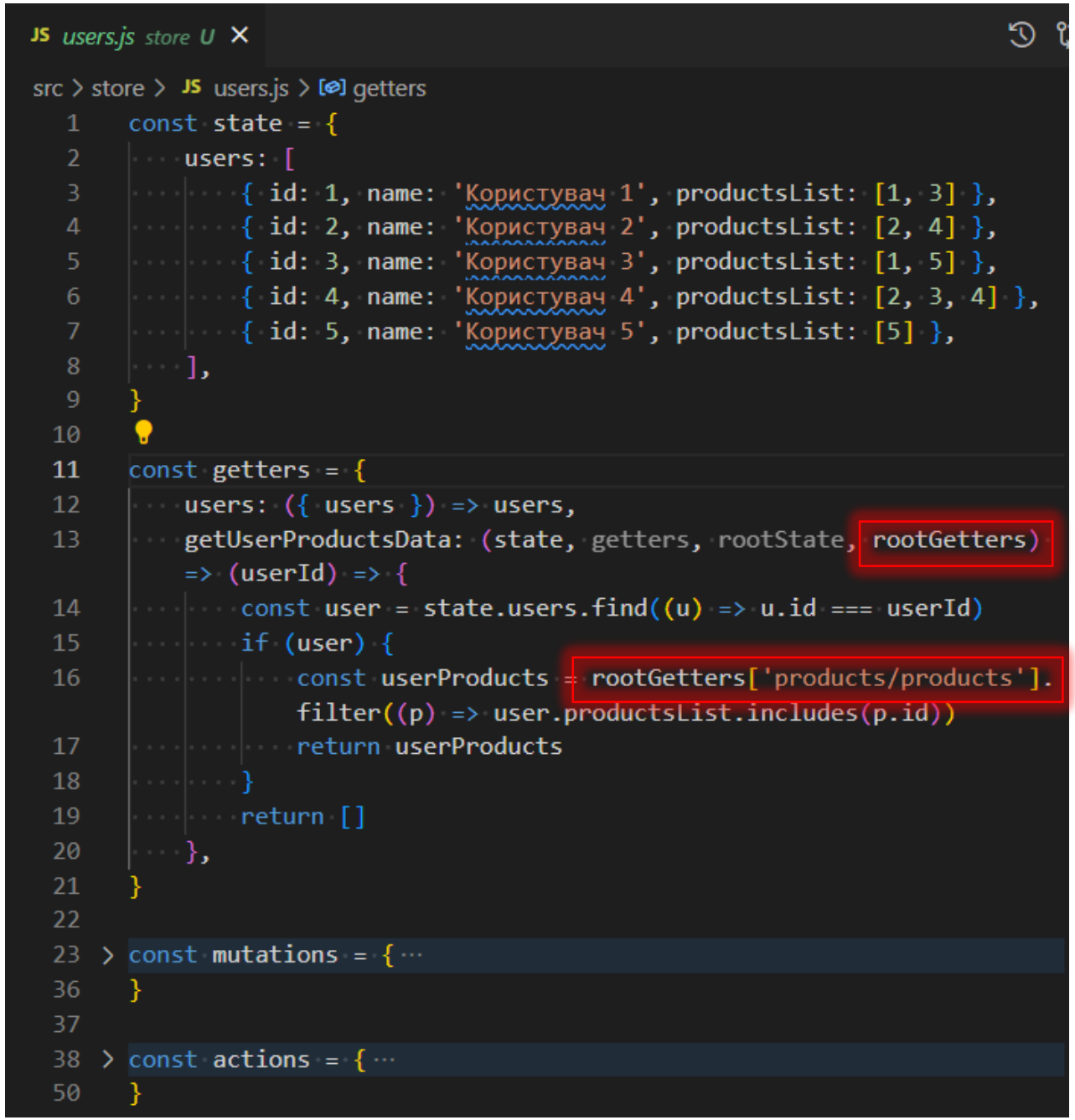
Як звернутись до ***<u>іншого геттера в межах одного локального модуля</u>***

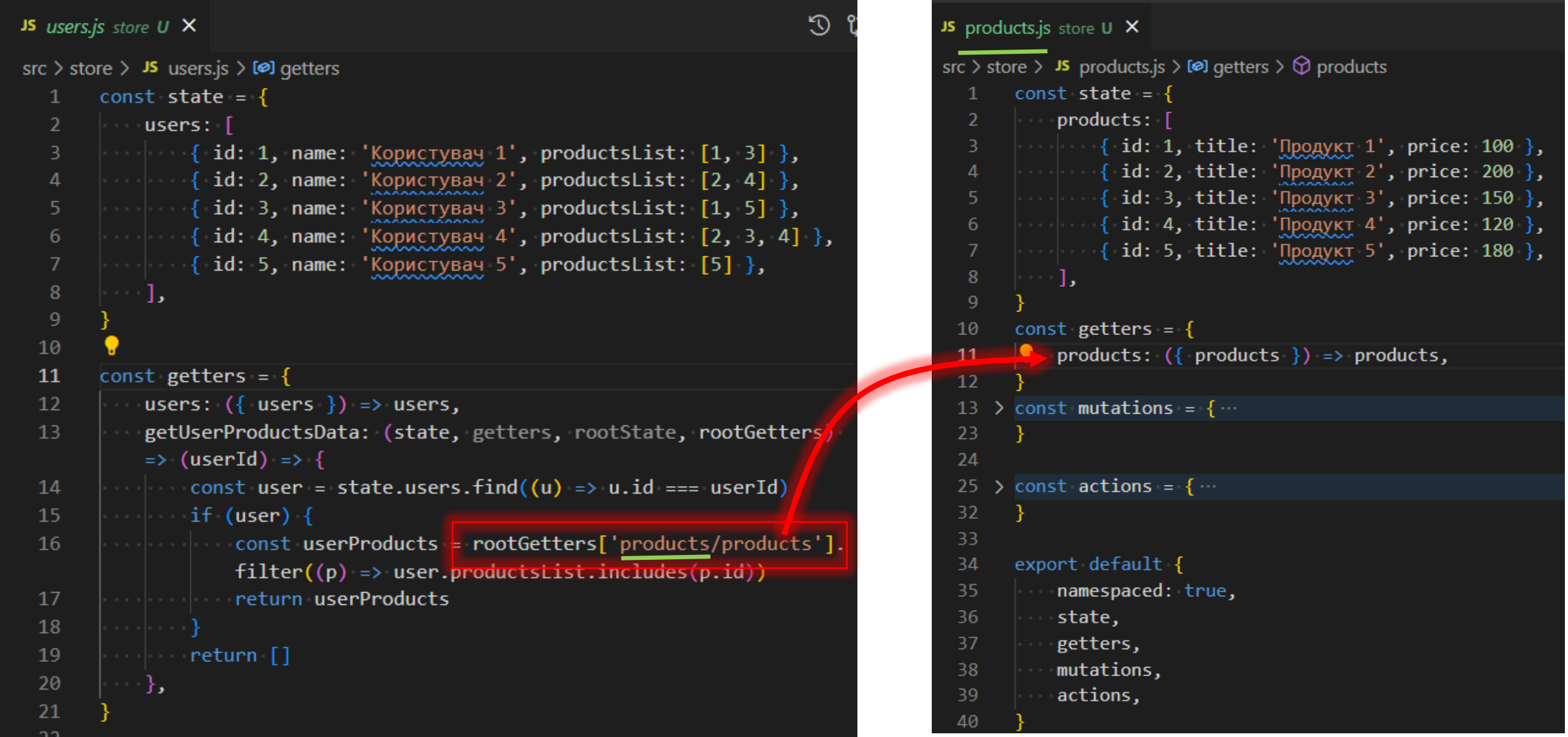
```
import { createStore } from 'vuex'

const store = createStore({
 . . . . . .
  //опис геттерів
  getters: {
     інший_геттер (state) {
       . . . . . .
       //звертання до іншого геттера
       getters
       . . . . . .
     },

     геттер1(state, getters) {
       . . . . . .
       //звертання до іншого геттера
       getters . інший_геттер
       . . . . . .
     },

   . . . . . .

  },
 . . . . . .

})
```



Використання ***геттерів з інших модулів***

```
import { createStore } from 'vuex'

const store = createStore({
  . . . . . .
  //опис обчисл.власт.(аналог computed)
  getters: {
    обч.геттер1(state, getters, rootState, rootGetters) {
      . . . . . .
      rootGetters('назва_модуля / геттер ')
    },
      . . . . . .
  },
  . . . . . .

})
```





users.js store U

src > store > users.js > getters

```
const state = {
    users: [
        { id: 1, name: 'Користувач 1', productsList: [1, 3] },
        { id: 2, name: 'Користувач 2', productsList: [2, 4] },
        { id: 3, name: 'Користувач 3', productsList: [1, 5] },
        { id: 4, name: 'Користувач 4', productsList: [2, 3, 4] },
        { id: 5, name: 'Користувач 5', productsList: [5] },
    ],
}

const getters = {
    users: ({ users }) => users,
    getUserProductsData: (state, getters, rootState, rootGetters)
    => (userId) => {
        const user = state.users.find((u) => u.id === userId)
        if (user) {
            const userProducts = rootGetters['products/products'].
            filter((p) => user.productsList.includes(p.id))
            return userProducts
        }
        return []
    },
}

const mutations = {…
}

const actions = {…
}
```

products.js store U

src > store > products.js > getters > products

```
const state = {
    products: [
        { id: 1, title: 'Продукт 1', price: 100 },
        { id: 2, title: 'Продукт 2', price: 200 },
        { id: 3, title: 'Продукт 3', price: 150 },
        { id: 4, title: 'Продукт 4', price: 120 },
        { id: 5, title: 'Продукт 5', price: 180 },
    ],
}
const getters = {
    products: ({ products }) => products,
}
const mutations = {…
}

const actions = {…
}

export default {
    namespaced: true,
    state,
    getters,
    mutations,
    actions,
}
```

```
import { createStore } from 'vuex'

const store = createStore({
 . . . . . .
  //опис методів
  actions: {

    інший_action_метод (. . . . .) {
      . . . . . .
    },

    action_метод({ dispatch, commit }) {
      dispatch(
        'інший_action_метод',
        параметри
      )
    }
   . . . . . .

  },
 . . . . . .

})
```

Використання ***action-методів з інших модулів***

```
import { createStore } from 'vuex'

const store = createStore({
 . . . . . .
  //опис методів
  actions: {
    action_метод({ dispatch, commit, getters, rootGetters }) {
      dispatch(
       ‘назва_модуля / назва_action_метода’,
       параметри ,
       { root: true }
      )
    }
   . . . . . .

  },
 . . . . . .

})
```

```
const state = {
    products: [ ···
    ],
}
const getters = { ···
}
const mutations = { ···
}

const actions = {
    editProduct(context, editedProduct) { ···
    },
    deleteProduct({ commit, dispatch }, productId) {
        commit('deleteProduct', productId)
        dispatch(
            'users/deleteProductFromUsers',
            productId,
            { root: true }
        )
    },
}
```

Використання ***action-методів з інших модулів***

products.js store U

src > store > products.js > actions > deleteProduct

```
1   const state = {
2   >   products: [ ···
8       ],
9   }
10  > const getters = { ···
12  }
13  > const mutations = { ···
23  }
24
25  const actions = {
26  >   editProduct(context, editedProduct) { ···
28      },
29      deleteProduct({ commit, dispatch }, productId) {
30          commit('deleteProduct', productId)
31          dispatch(
32              'users/deleteProductFromUsers',
33              productId,
34              { root: true }
35          )
36      },
37  }
38
```

users.js store U

src > store > users.js > default

```
1   const state = {
2   >   users: [ ···
8       ],
9   }
10
11  > const getters = { ···
21  }
22
23  > const mutations = { ···
36  }
37
38  const actions = {
39  >   addUser(context, user) { ···
43      },
44  >   editUser(context, editedUser) { ···
46      },
47      deleteUser(context, userId) { ···
49      },
50      deleteProductFromUsers({ commit }, productId) {
51          commit('removeProductFromUsers', productId)
52      },
53  }
```

index.js store M

src > store > index.js > store > mutations

You, 15 minutes ago | 1 author (You)

```
import { createStore } from 'vuex'
import productsModule from './products'
import usersModule from './users'

// Create a new store instance.
const store = createStore({
    state: () => {
        return {
            globalCounter: 99,
        }
    },
    getters: {
        getGlobalCounter: (state) => state.globalCounter,
    },
    mutations: {
        setGlobalCounter(state) {
            state.getGlobalCounter++
        },
    },
    actions: {
        incGlobalCounter({ commit }) {
            commit('setGlobalCounter')
        },
    },
    modules: {
        products: productsModule,
        users: usersModule,
    },
})
```

products.js store U

src > store > products.js > mutations > editProduct >

```
const state = {
    products: [ ···
    ],
}
const getters = {
    products: ({ products }) => products,
}
```

users.js store U

src > store > users.js > getters

```
const state = {
    users: [ ···
    ],
}

const getters = {
    getUsers: ({ users }) => users,

    users: ({ users }) => users,
    getUserProductsData: (state, getters, rootState, rootGetters) => (userId) => {
    },
}

const mutations = { ···
}

const actions = {
    addUser(context, user) { ···
    },
    editUser(context, editedUser) { ···
    },
    deleteUser(context, userId) { ···
    },
    deleteProductFromUsers(cox, productId) {
    },
}
```

rootState

```
▼ Proxy(Object) {globalCounter: 99, products
  ▶ [[Handler]]: MutableReactiveHandler
  ▼ [[Target]]: Object
      globalCounter: 99
    ▼ products:
      ▶ products: (4) [Proxy(Object), Proxy(O
      ▶ [[Prototype]]: Object
    ▼ users:
      ▶ users: (5) [{…}, {…}, {…}, {…}, {…}]
      ▶ [[Prototype]]: Object
```

index.js store M

src > store > index.js > store > mutations

You, 15 minutes ago | 1 author (You)

```
import { createStore } from 'vuex'
import productsModule from './products'
import usersModule from './users'

// Create a new store instance.
const store = createStore({
    state: () => {
        return {
            globalCounter: 99,
        }
    },
    getters: {
        getGlobalCounter: (state) => state.globalCounter,
    },
    mutations: {
        setGlobalCounter(state) {
            state.getGlobalCounter++
        },
    },
    actions: {
        incGlobalCounter({ commit }) {
            commit('setGlobalCounter')
        },
    },
    modules: {
        products: productsModule,
        users: usersModule,
    },
})
```

products.js store U

src > store > products.js > mutations > editProduct >

```
const state = {
    products: [ ···
    ],
}
const getters = {
    products: ({ products }) => products,
}
const mutations = {
    editProduct(state, editedProduct) {
        const index = state.products.findIndex((p) => p.id === editedProduct.id)
        if (index !== -1) {
            state.products[index] = editedProduct
        }
    },
    deleteProduct(state, productId) {
        state.products = state.products.filter((p) => p.id !== productId)
    },
}

const actions = {
    editProduct(context, editedProduct) { ···
    },
    deleteProduct({ commit, dispatch, rootState, rootGetters }, productId) {
    },
}
```

rootGetters

```
▼ {} i
    getGlobalCounter: (...)
    products/products: (...)
    users/getUserProductsData: (...)
    users/getUsers: (...)
    users/users: (...)
```

users.js store U

src > store > users.js > getters

```
const state = {
    users: [ ···
    ],
}

const getters = {
    getUsers: ({ users }) => users,

    users: ({ users }) => users,
    getUserProductsData: (state, getters, rootState, rootGetters) => (userId) => {
    },
}

const mutations = { ···
}

const actions = {
    addUser(context, user) { ···
    },
    editUser(context, editedUser) { ···
    },
    deleteUser(context, userId) { ···
    },
    deleteProductFromUsers(cox, productId) {
    },
}
```