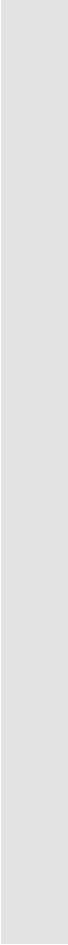
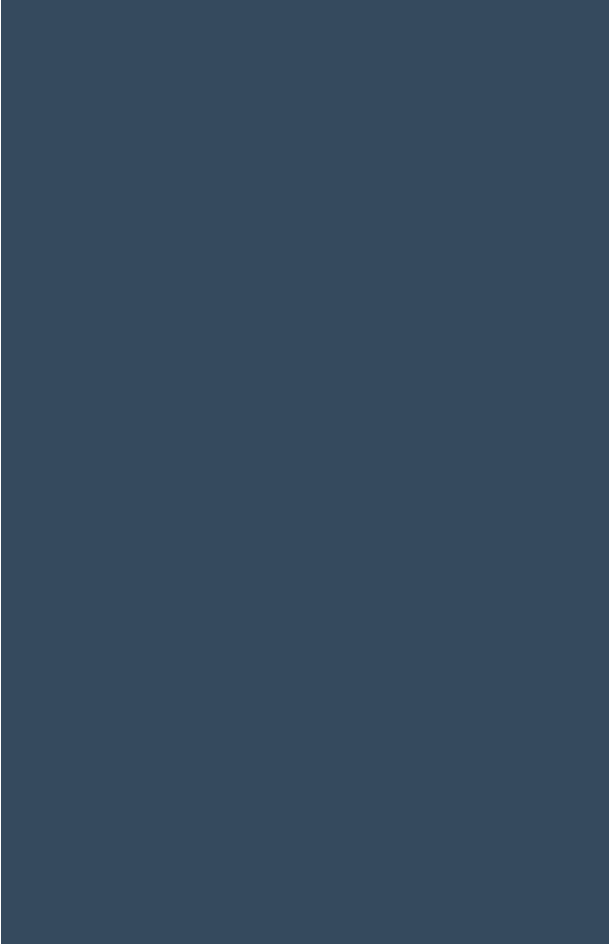


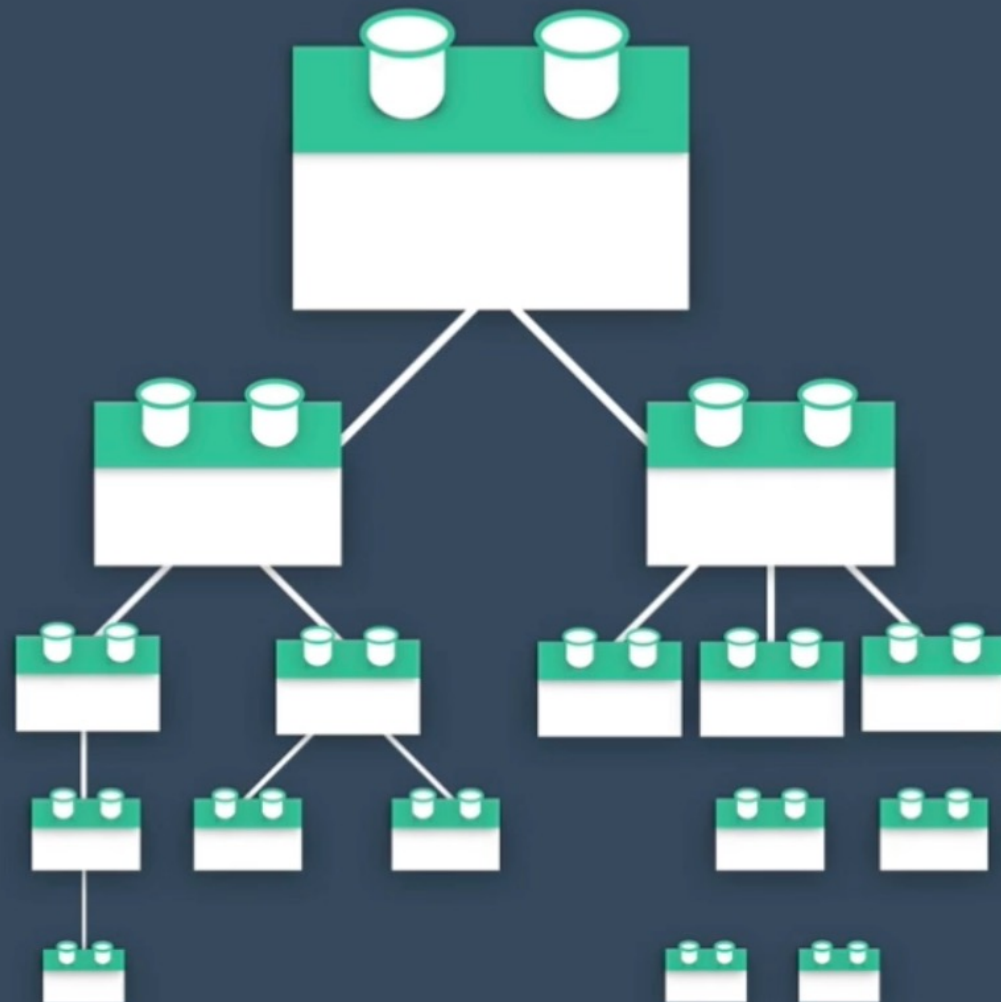
# State Management

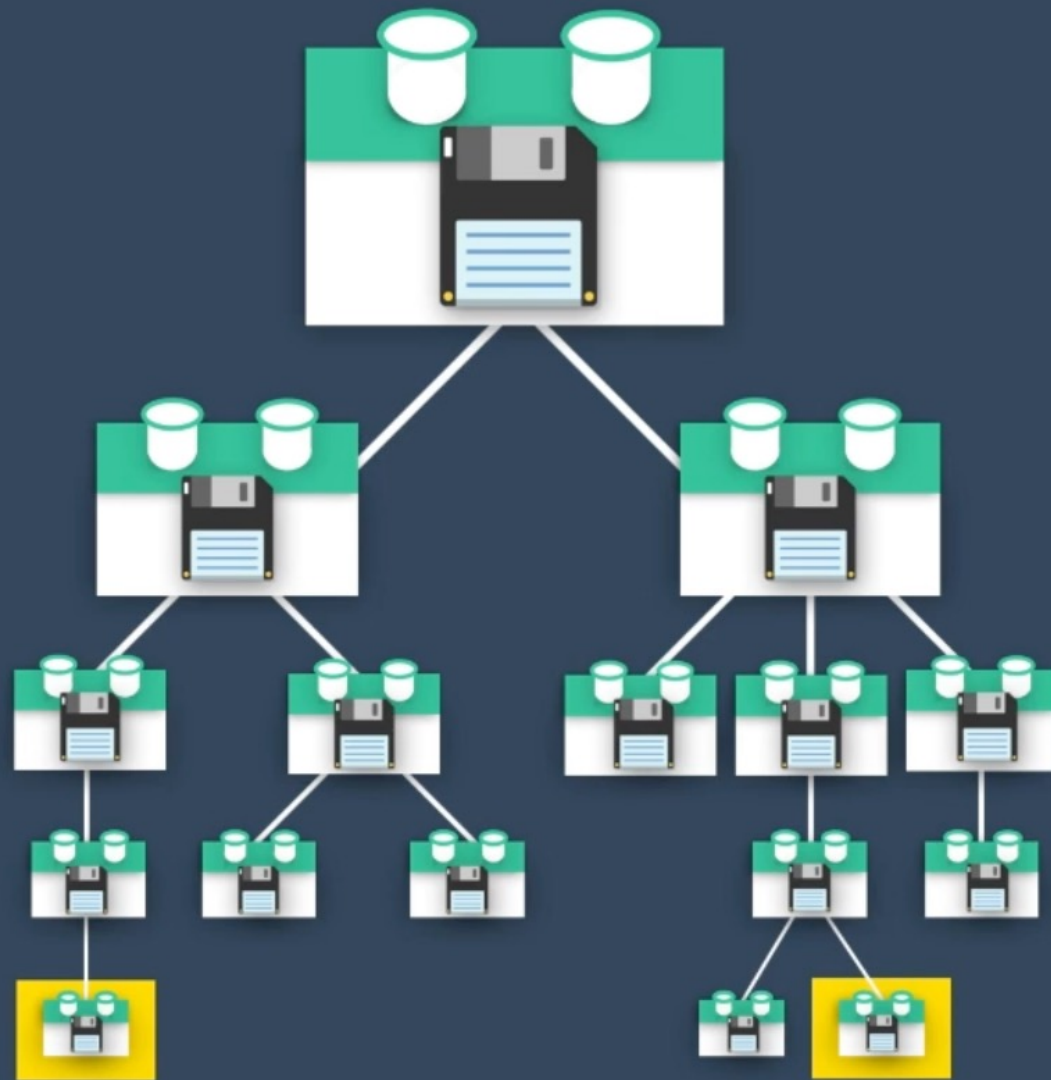
VueJs



Imagine you have a **dozen of components** inside your VueJs project.

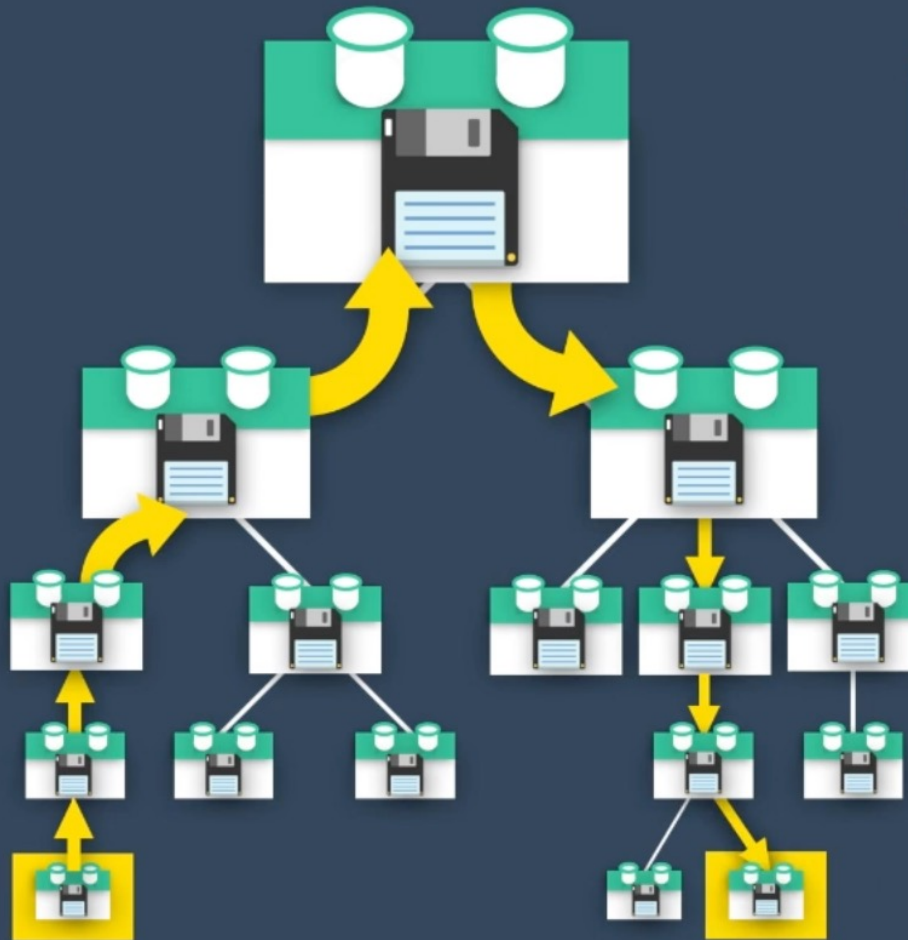
You want some of these components to manage **a set of information**.  
How can you do it?





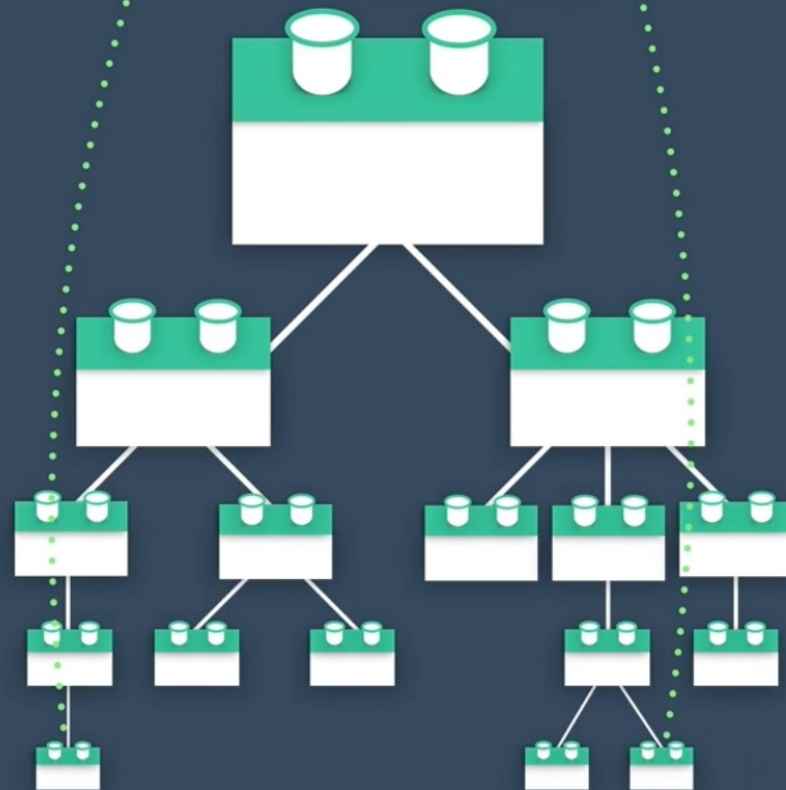
Props


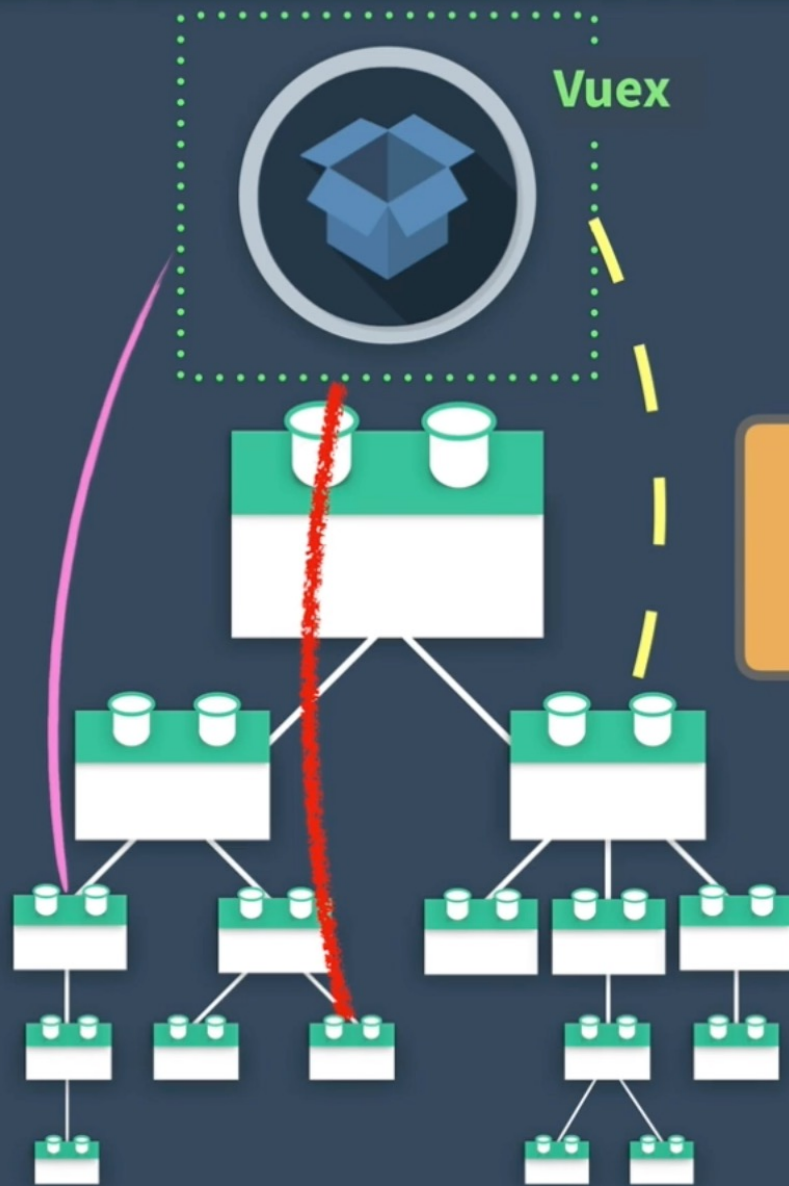
Events





Vuex





**State changes could be  
unpredictable & untraceable**

# VUEX Provides

- Single source of truth
- Useful library of Actions, Mutations, Getters
- Well structure state pattern



## Installation for Vuex 4 (for Vuejs 3)

```
npm install vuex@next --save
```

sh

## Installation for Vuex 3 (for Vuejs 2)

```
npm install vuex --save
```

sh

## And finally, you can import to use it

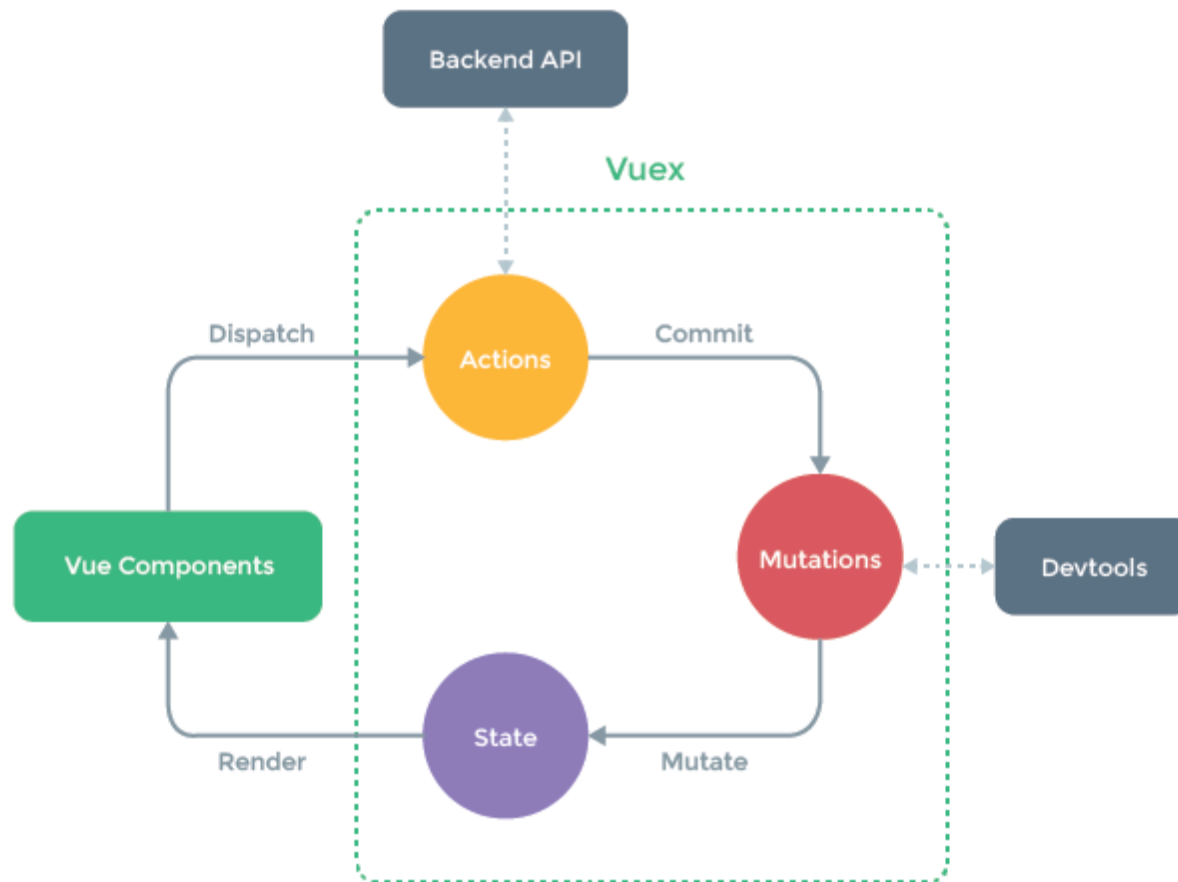
```
js
import { createApp } from 'vue'
import { createStore } from 'vuex'

// Create a new store instance.
const store = createStore({
  state () {
    return {
      count: 0
    }
  },
  mutations: {
    increment (state) {
      state.count++
    }
  }
})

const app = createApp({ /* your root component */ })

// Install the store instance as a plugin
app.use(store)
```

## The basics of Vuex:



Vuex is not always recommended, you can always use event/props to pass data around and it is actually recommended if it is not too complicated because we want each component to be independent.

Anyway, medium to big SPA can't be handled with just this so Vuex come to play!

To access to Vuex store, you can do as so

```
store.commit('increment')  
  
console.log(store.state.count) // -> 1
```

js

If you are accessing Vuex store from the components under the registered Vue instance, you can use **this.\$store**

```
methods: {  
  increment() {  
    this.$store.commit('increment')  
    console.log(this.$store.state.count)  
  }  
}
```

js

The state of Vuex is reactive so to get advantage of this, use Vuex state in **computed** property.

```
// let's create a Counter component  
const Counter = {  
  template: `<div>{{ count }}</div>`,  
  computed: {  
    count () {  
      return store.state.count  
    }  
  }  
}
```

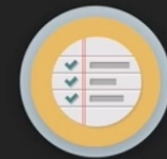
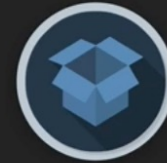
js



So how do you manage this state pattern?

```
const app = new Vue({  
  data: {  
    ...  
  },  
  methods: {  
    ...  
  },  
  computed: {  
    ...  
  }  
})
```

```
const store = new Vuex.Store({  
  state: {  
    ...  
  },  
  mutations: {  
    ...  
  },  
  actions: {  
    ...  
  },  
  getters: {  
    ...  
  }  
})
```





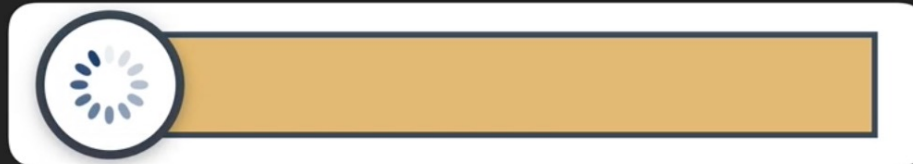
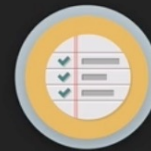
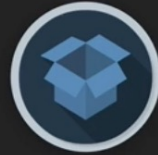
Example



```
const store = new Vuex.Store({
  state: {
    loadingStatus: 'notLoading',
    todos: []
  },
  mutations: {
    SET_LOADING_STATUS(state, status) {
      state.loadingStatus = status
    },
    SET_TODOS(state, todos) {
      state.todos = todos
    }
  },
  actions: {
    fetchTodos(context) {
      context.commit('SET_LOADING_STATUS', 'loading')
      axios.get('/api/todos').then(response => {
        context.commit('SET_LOADING_STATUS', 'notLoading')
        context.commit('SET_TODOS', response.data.todos)
      })
    }
  }
})
```



```
const store = new Vuex.Store({
  state: {
    loadingStatus: 'loading',
    todos: []
  },
  mutations: {
    SET_LOADING_STATUS(state, status) {
      state.loadingStatus = status
    },
    SET_TODOS(state, todos) {
      state.todos = todos
    }
  },
  actions: {
    fetchTodos(context) {
      context.commit('SET_LOADING_STATUS', 'loading')
      axios.get('/api/todos').then(response => {
        context.commit('SET_LOADING_STATUS', 'notLoading')
        context.commit('SET_TODOS', response.data.todos)
      })
    }
  }
})
```

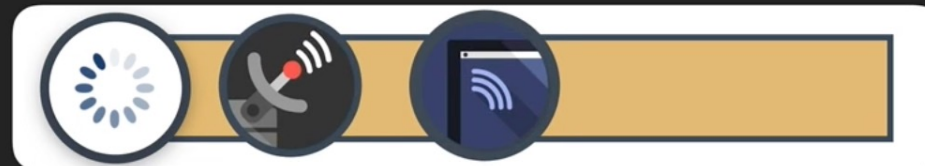
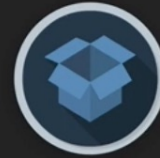




```

const store = new Vuex.Store({
  state: {
    loadingStatus: 'loading',
    todos: []
  },
  mutations: {
    SET_LOADING_STATUS(state, status) {
      state.loadingStatus = status
    },
    SET_TODOS(state, todos) {
      state.todos = todos
    }
  },
  actions: {
    fetchTodos(context) {
      context.commit('SET_LOADING_STATUS', 'loading')
      → axios.get('/api/todos').then(response => {
        context.commit('SET_LOADING_STATUS', 'notLoading')
        context.commit('SET_TODOS', response.data.todos)
      })
    }
  }
})

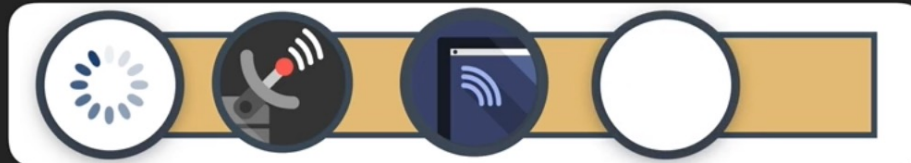
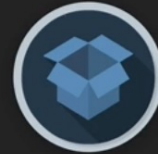
```



```

const store = new Vuex.Store({
  state: {
    loadingStatus: 'notLoading',
    todos: []
  },
  mutations: {
    SET_LOADING_STATUS(state, status) {
      state.loadingStatus = status
    },
    SET_TODOS(state, todos) {
      state.todos = todos
    }
  },
  actions: {
    fetchTodos(context) {
      context.commit('SET_LOADING_STATUS', 'loading')
      axios.get('/api/todos').then(response => {
        context.commit('SET_LOADING_STATUS', 'notLoading')
        context.commit('SET_TODOS', response.data.todos)
      })
    }
  }
})

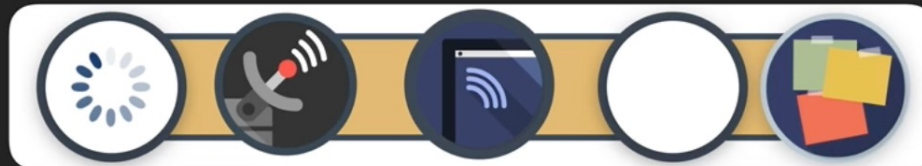
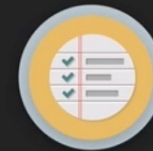
```



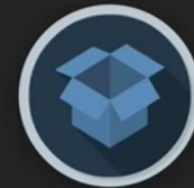
```

const store = new Vuex.Store({
  state: {
    loadingStatus: 'notLoading',
    todos: [{...}, {...}, {...}]
  },
  mutations: {
    SET_LOADING_STATUS(state, status) {
      state.loadingStatus = status
    },
    SET_TODOS(state, todos) {
      state.todos = todos
    }
  },
  actions: {
    fetchTodos(context) {
      context.commit('SET_LOADING_STATUS', 'loading')
      axios.get('/api/todos').then(response => {
        context.commit('SET_LOADING_STATUS', 'notLoading')
        context.commit('SET_TODOS', response.data.todos)
      })
    }
  }
})

```

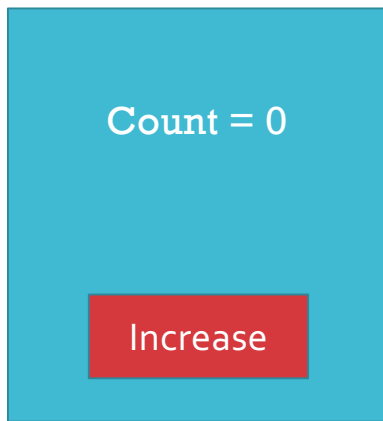


```
const store = new Vuex.Store({
  state: {
    loadingStatus: 'notLoading',
    todos: [
      { id: 1, text: '...', done: false },
      { id: 2, text: '...', done: true },
      { id: 3, text: '...', done: true }
    ],
    getters: {
      doneTodos(state) {
        return state.todos.filter(todo => todo.done)
      }
    }
  })
```

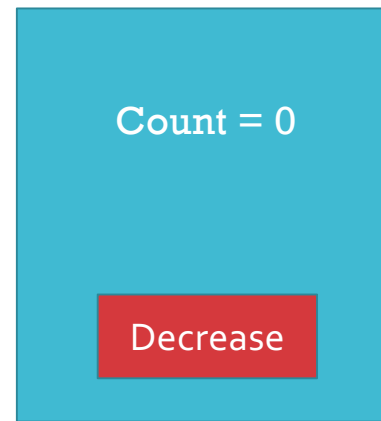


Exercise 1:

Create 2 components: Component A and Component B.



Component A



Component B

By using vuex, when user click on increase in component A, the count data in component B will also increase. The same thing for user click on button decrease in component B.

# The mapState Helper

It is getting repetitive if you are trying to access many Vuex state and need to rewrite it in the **computed** property.

**mapState** is another option

js

```
// in full builds helpers are exposed as Vuex.mapState
import { mapState } from 'vuex'

export default {
  // ...
  computed: mapState({
    // arrow functions can make the code very succinct!
    count: state => state.count,

    // passing the string value 'count' is same as `state => state.count`
    countAlias: 'count',

    // to access local state with `this`, a normal function must be used
    countPlusLocalState (state) {
      return state.count + this.localCount
    }
  })
}
```

# The mapGetters, mapMutations and mapActions Helper

Similar to mapState for accessing state helpers, Getters also has its own Helper.

**mapGetters** is another convenient way of accessing the getters

**mapMutations** apply for the **Mutations** and **mapActions** apply for the **Actions**.



js

```
import { mapGetters } from 'vuex'

export default {
  // ...
  computed: {
    // mix the getters into computed with object spread operator
    ...mapGetters([
      'doneTodosCount',
      'anotherGetter',
      // ...
    ])
  }
}
```