

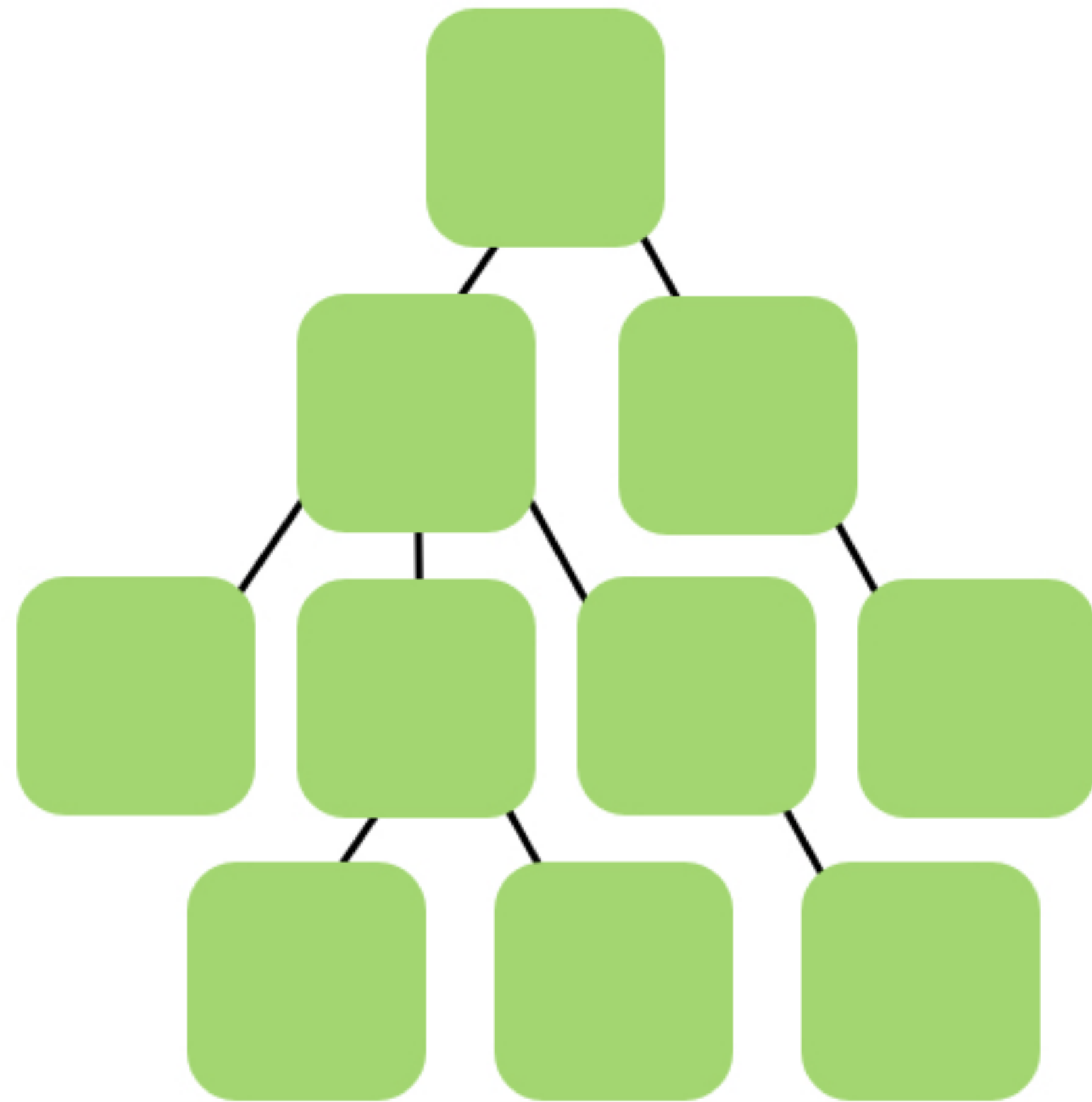


Vuex

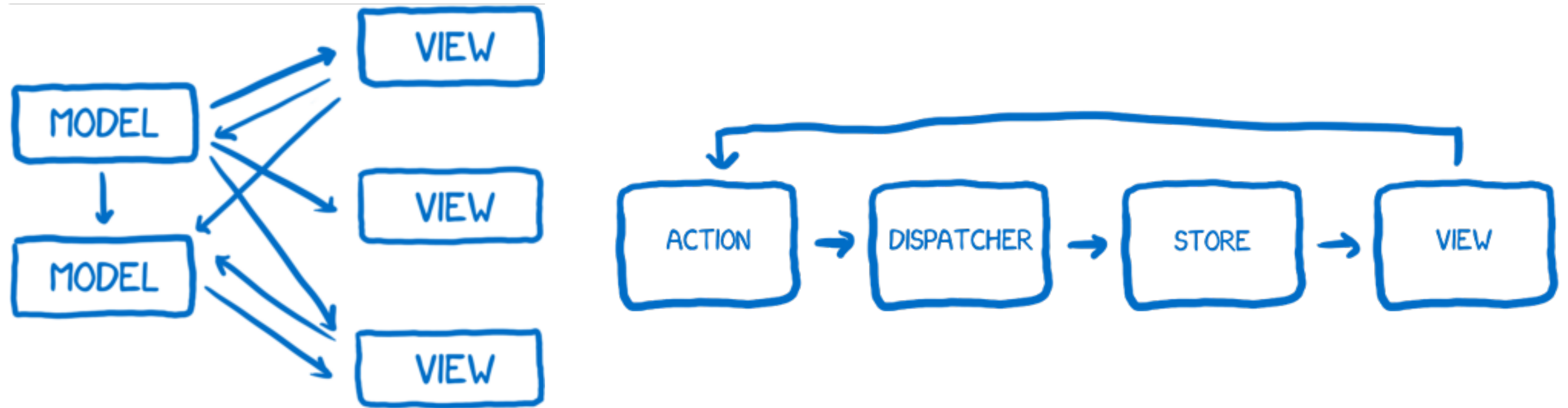


Vuex

バケツリレー・イベントバスの限界



Flux 一方向のデータフロー

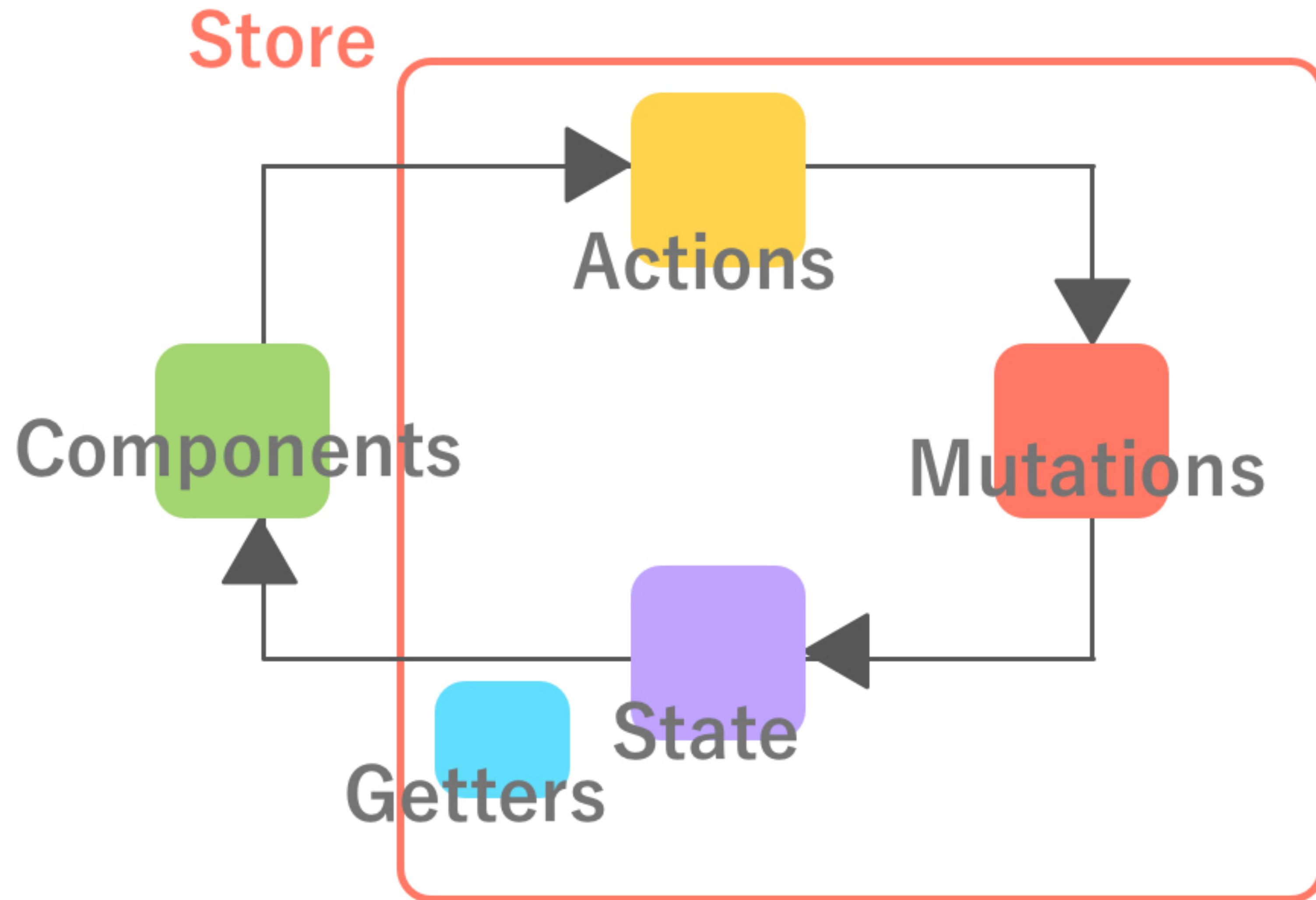


漫画で説明するFlux

<https://medium.com/samyamashita/>

%E6%BC%AB%E7%94%BB%E3%81%A7%E8%AA%AC%E6%98%8E%E3%81%99%E3%82%8B-flux-1a219e50232b


Vuex 簡易図



Vuex OptionsAPIとの対応表

Options API	Vuex	Memo
data	state	
computed(get)	getters	プロパティなら キャッシュあり
methods	mutations	同期 履歴残る
methods	actions	非同期


Vuex の注意



乱用は禁物(かえってわかりづらくなる)

コンポーネント内 ・ OptionsAPI
親子間のやりとり ・ EventUpPropsDown

ContainerComponent ・ Vuex使用
PresentationalComponent ・ UI



Vuexのインストール

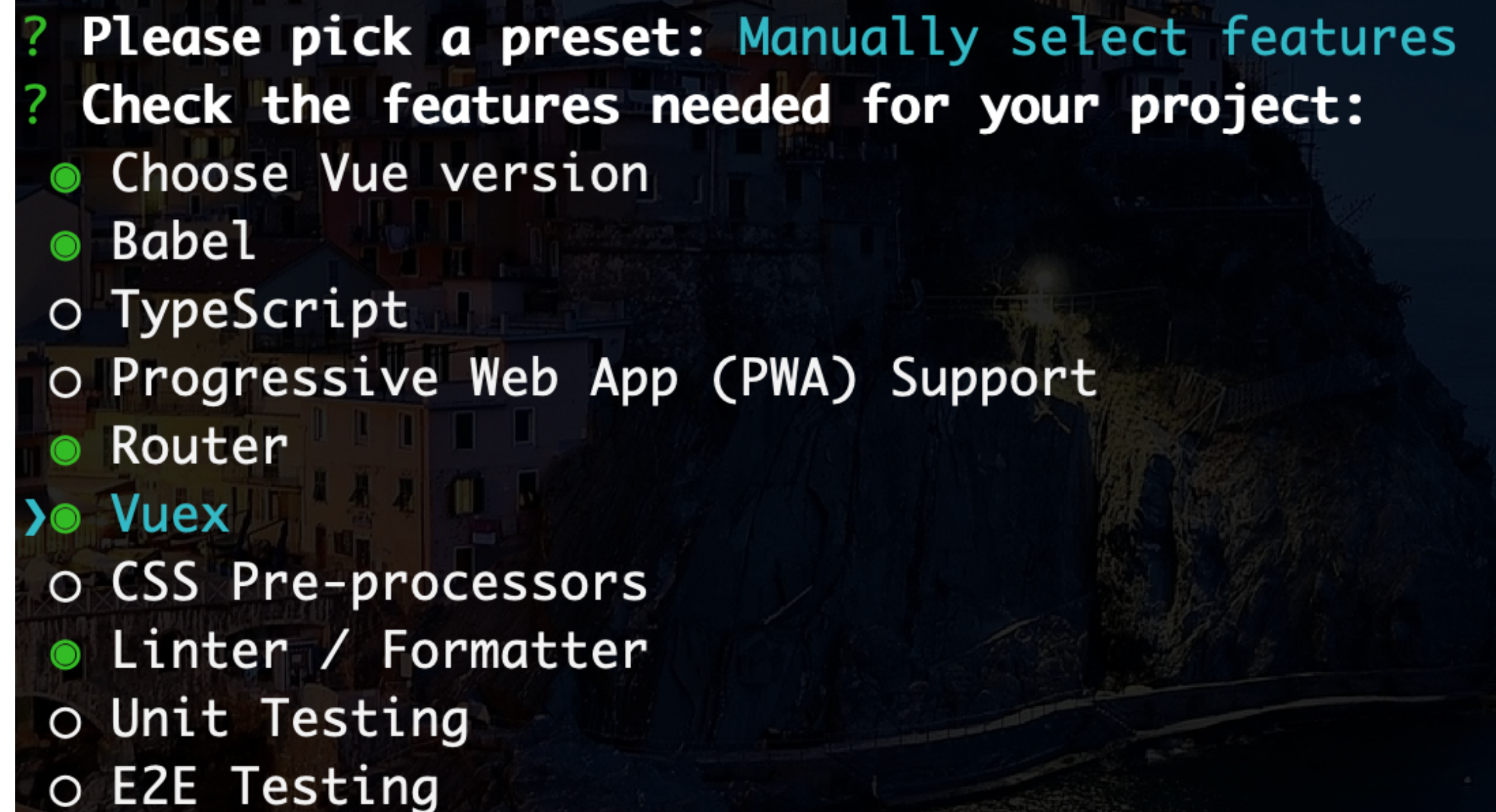
Vuexインストール

npm i vuex —save

またはVueCLIで

vue create xxx

<https://vuex.vuejs.org/ja/>

A screenshot of the Vue CLI create project wizard. The background is a dark, atmospheric image of a coastal town at night. The text is white and green. It prompts the user to pick a preset and then lists features to be selected for the project. The 'Vuex' feature is highlighted with a green checkmark and a green arrow.

```
? Please pick a preset: Manually select features
? Check the features needed for your project:
  ● Choose Vue version
  ● Babel
  ○ TypeScript
  ○ Progressive Web App (PWA) Support
  ● Router
  >● Vuex
  ○ CSS Pre-processors
  ● Linter / Formatter
  ○ Unit Testing
  ○ E2E Testing
```


Vuex 追記箇所



src/main.js


```
import store from './store'  
new Vue({ store })
```

src/store/index.js ・ ・ Vuex記述ファイル



Vuexを使ってみる

Vuex 使い方



テンプレート側

`$store.state.xxx`

スクリプト側

`this.$store.state.xxx, this.$store.commit('')`

mapヘルパー

`mapState, mapActions` など

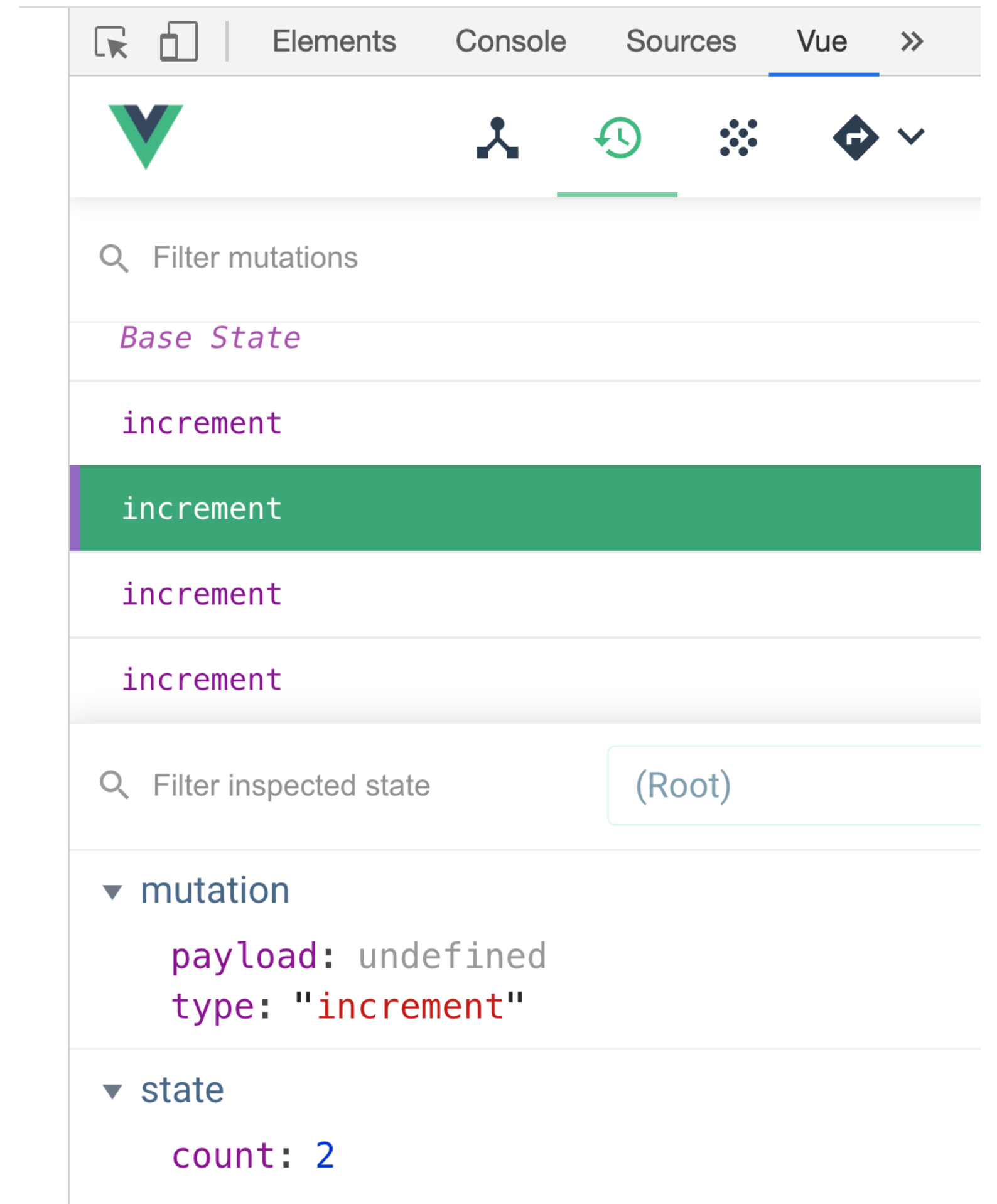
Vuex 使い方・引数

Vuex	呼び出し	引数
state	<code>\$store.state.xxx</code>	
getters	<code>\$store.getters.xxx</code> <code>\$store.getters('xxx')</code>	state, [getters]
mutations	<code>\$store.commit('xxx')</code>	state, {値(payload)}
actions	<code>\$store.dispatch('xxx')</code>	context, {値(payload)} ※{commit} {state}など含む

mutations -> state

mutationを使う場合は
`store.commit('xxx')`。

mutationを実行する
タイミングで
履歴を残す事ができる。



mutationsの補足

第2引数で値を渡すことができる
payload ・ ・ データ本体
オブジェクトで渡すのが一般的

```
src/store/index.js
mutations:{
  addCount( state, payload){
    state.count += payload.value
  }
}
```

component側

```
methods:{
  addCount(){
    this.$store.commit('increment', {
      value: 10
    })
  }
}
```

actions->mutations->state



同期・非同期で actions/mutations を
使い分けると混乱の元。

同期処理でも、
actions->mutations->state の
流れで書くようにする。

getters (プロパティスタイル)

`visibleUsers: state => state.users.filter(user => user.isVisible)`

第2引数で他のgettersも呼び出せる

computed同様 キャッシュが残る
computed内で呼び出す

`store.getters.visibleUsers`

getters (メソッドスタイル)

```
getUserById: state => id => {  
  return state.users.find(user => user.id === id )  
}
```

```
store.getters.getUserById(2)
```

メソッドスタイル(引数あり)の場合は
キャッシュが残らない



Mapヘルパー

Mapヘルパー



アクションするだけのメソッドを
アクションと同じmethodsとして展開する
(繰り返し書かなくていいように)

```
import { mapActions } from 'vuex'
```

スプレッド構文

```
...mapActions(['xxx', 'yyy'])
```


Mapヘルパー 使うのは2つ

Vuex	Mapヘルパー
state	△ mapState
getters	○ mapGetters
mutations	△ mapMutations
actions	○ mapActions

Gettersで
Stateの値を監視算出するなら、
mapStateは不要


必ずActionを通るなら、
mapMutationsは不要

Mapヘルパーの書き方

```
import { mapActions } from 'vuex'
```

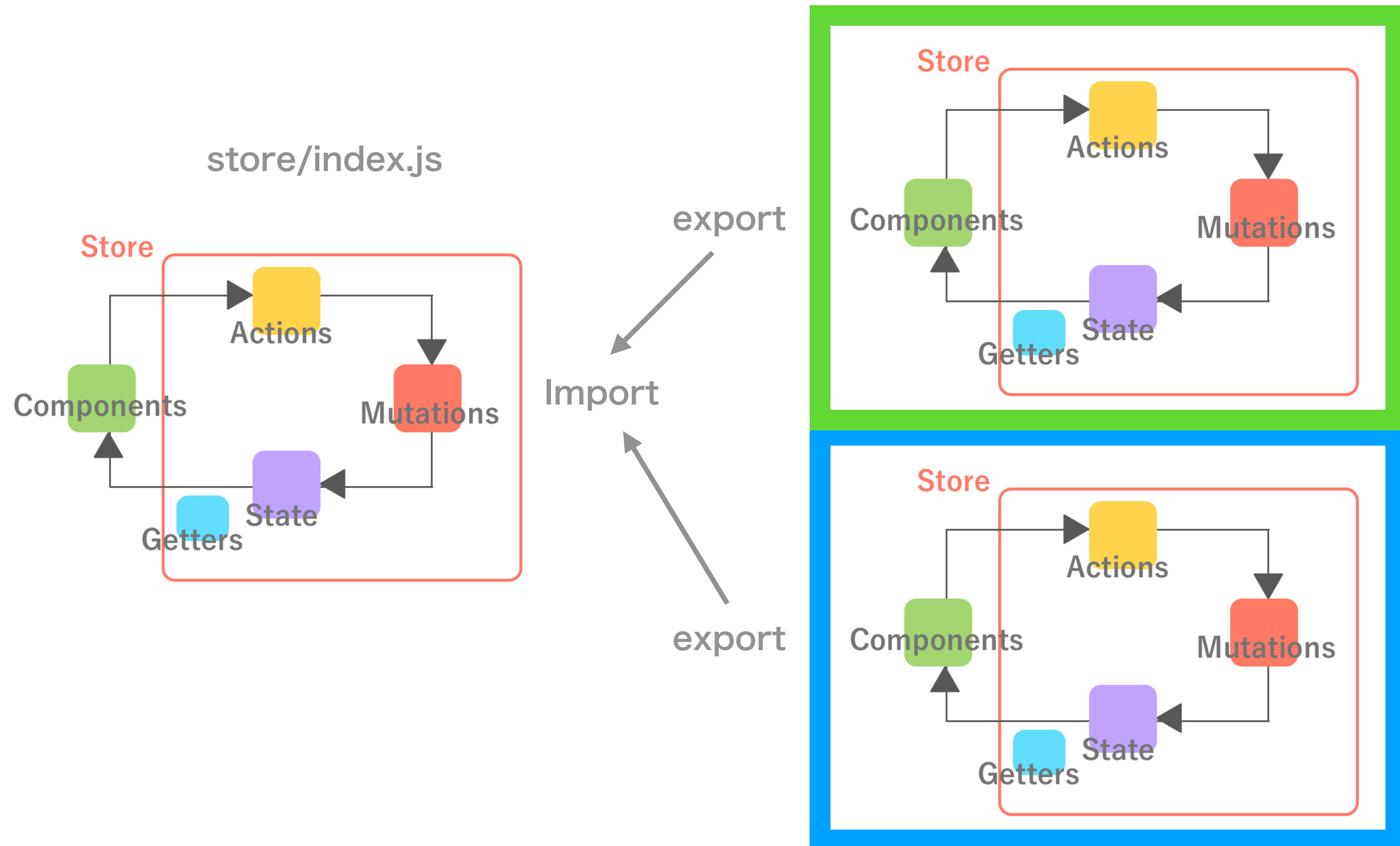
```
methods:{  
  ...mapActions(['incrementAction']) //配列で複数書ける  
  
  incrementAction(){ // このメソッドと同じになる  
    this.$store.dispatch('incrementAction')  
  }  
}
```

引数ありなら別メソッドから
this.incrementAction({引数}) でOK



モジュール分割と 名前空間

モジュール化・名前空間 イメージ図



モジュールの読み込み方法



Store/index.js

```
import auth from './auth'
```

```
modules : { auth }
```

store/auth/index.js

```
const state = {} // mutations, actions, gettersもconstで  
export default {  
  namespace: true,  
  state, // mutations, actions, gettersも  
}
```

モジュールの指定方法



...mapActions('モジュール名', ['アクション名'])

他

rootState ・ ・ ルートのstate

モジュール間のやりとりは基本NG
できるだけ名前空間内で完結させる



Vuexの使い所

モジュール結合度

数字が低いほうが望ましい 数字が高いほど密結合

	モジュール結合度	説明	Vueの機能
1	データ結合	引数で単純なデータを渡す	props
2	スタンプ結合	引数でオブジェクトを渡す	props
3	制御結合	引数でメソッド内の処理が変わる	
4	外部結合	単一のグローバルデータを参照	
5	共通結合	複数のグローバルデータを参照	state
6	内容結合	他のオブジェクトの内部を参照	getters, actions

<https://user-first.ikyu.co.jp/entry/design-of-vue-and-vuex>

Vuexやりとりと表示を別で

ContainerComponent・・・Vuex使用

PresentationalComponent・・・UI

ファイル名にContainerとつけたらVuex使用

