**Vuex ------ State Management**

* Vuex is a **state management pattern + library** for Vue.js applications.
* It serves as a centralized store for all the components in an application, with rules ensuring that the state can only be mutated in a predictable fashion.

**What is a "State Management Pattern"?**

Let's start with a simple Vue counter app:

**const Counter = {**

**// state**

**data () {**

**return {**

**count: 0**

**}**

**},**

**// view**

**template: `**

**<div>{{ count }}</div>**

**`,**

**// actions**

**methods: {**

**increment () {**

**this.count++**

**}**

**}**

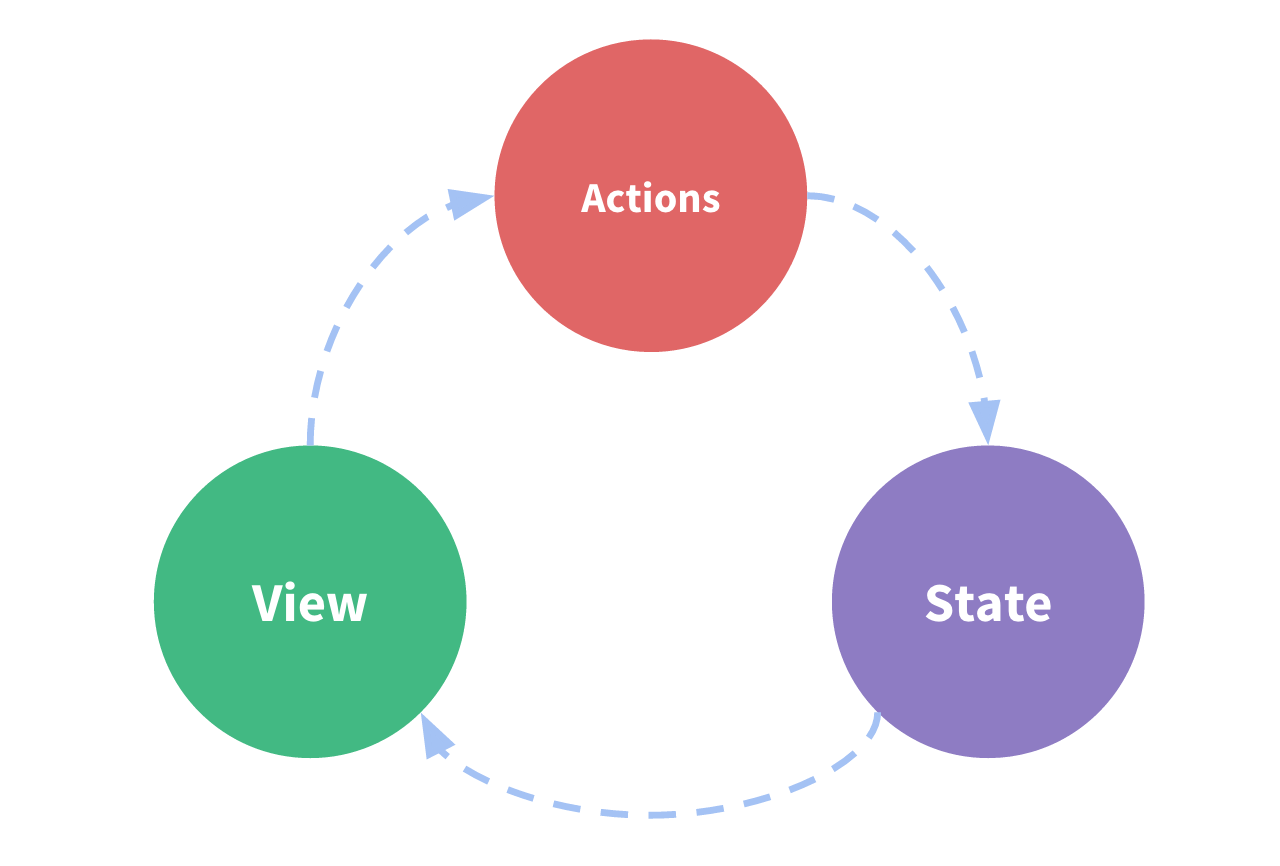
**}**

**createApp(Counter).mount('#app')**

It is a self-contained app with the following parts:

* The **state**, the source of truth that drives our app;
* The **view**, a declarative mapping of the **state**;
* The **actions**, the possible ways the state could change in reaction to user inputs from the **view**.

This is a simple representation of the concept of "one-way data flow":



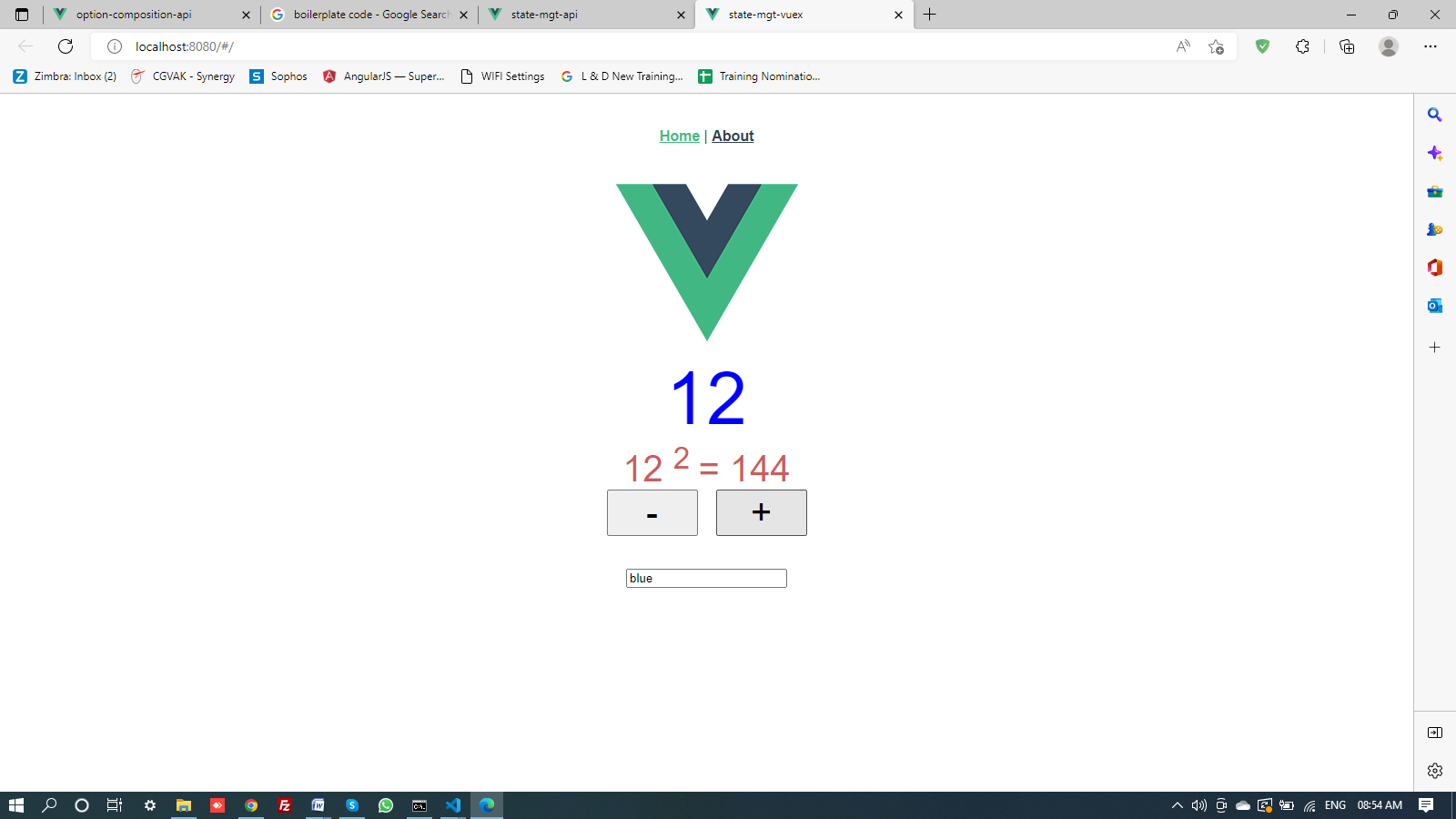
However, the simplicity quickly breaks down when we have **multiple components that share a common state**:

* Multiple views may depend on the same piece of state.
* Actions from different views may need to mutate the same piece of state.

By defining and separating the concepts involved in state management and enforcing rules that maintain independence between views and states, we give our code more structure and maintainability.

This is the basic idea behind Vuex, inspired by [Flux](https://facebook.github.io/flux/docs/overview), [Redux](http://redux.js.org/" \t "_blank) and [The Elm Architecture](https://guide.elm-lang.org/architecture/). Unlike the other patterns, Vuex is also a library implementation tailored specifically for Vue.js to take advantage of its granular reactivity system for efficient updates.



****

**State management using Vuex**

**Code in > src > strore > index.js**

import { createStore } from 'vuex'

//import {axios} from 'axios'

export default createStore({

state: { /// contails the ppty/variable which change the state

counter: 10,

colorCode: 'blue'

},

getters: {

counterSquare(state){

return state.counter \* state.counter

}

},

mutations: { /// mutation functions that make the changes

increaseCounter(state){

state.counter++

},

//increaseCounter(state, randomNumber){

// //console.log('randomNumber: ',randomNumber)

// state.counter += randomNumber

// },

decreaseCounter(state){

if (state.counter<= 0)

state.counter=0

else

state.counter--;

},

setColorCode(state, newVal){

state.colorCode = newVal

}

},

actions: { // contains asynchronous code

// increaseCounter(){ // install in command line --- npm install axiox

// state.counter++ // Google -- Random number api & copy the link and paste

// console.log('+')

// },

// increaseCounter({commit}){

// axios('https://www.random.org/integers/?num=1&min=1&max=6&col=1&base=10&format=plain&rnd=new')

// .then(response => {

// commit('increaseCounter', response.data)

// })

// }

setColorCode( {commit }, newVal){

commit('setColorCode',newVal)

}

},

modules: {

}

})

**Code in > src > views > HomeView.vue**

<template>

<div class="home">

<img alt="Vue logo" src="../assets/logo.png">

<div class="counter"

:style="{ color: $store.state.colorCode }">

{{$store.state.counter}}

</div>

<!-- <counter /> -->

<div class="counter-squared">

{{$store.state.counter}}

<sup> 2 </sup> =

{{$store.getters.counterSquare}}

</div>

<!-- <counterSquare /> -->

<div class="buttons">

<button @click="$store.commit('decreaseCounter')"> - </button>

<!-- commit is used to call mutation functions-->

<button @click="$store.commit('increaseCounter')"> + </button>

<!--<button @click="$store.dispatch('increaseCounter')"> + </button> -->

<!-- dispatch() method is used instead of commit method -->

</div>

<br /> <br />

<div>

<input type="text" placaholder="Enter color code"

v-model="colorCode" />

<!-- v-model="$store.state.colorCode" /> -->

</div>

</div>

</template>

<script>

import Counter from "../components/Counter.vue";

import CounterSquare from "../components/CounterSquare.vue";

export default {

name: 'HomeView' ,

components:{

"counter": require("@/components/Counter.vue").default,

"counterSquare": require("@/components/CounterSquare.vue").default,

CounterSquare

},

// data(){

// return{

// counter: 0

// }

// },

// methods:{

// increaseCounter(){

// this.counter++

// },

// decreaseCounter(){

// if (this.counter<= 0)

// this.counter=0

// else

// this.counter--;

// }

// }

computed:{

colorCode: {

get(){

return this.$store.state.colorCode

},

set(newVal){

// this.$store.commit('setColorCode',newVal)

this.$store.dispatch('setColorCode', newVal)

}

}

}

}

</script>

<style>

.div{

margin-bottom: 10px;

}

.counter{

font-size: 80px;

}

.buttons button{

font-size: 40px;

width: 100px;

margin: 0 10px;

}

.counter-squared{

font-size: 40px;

color:indianred } </style>

**Code in > src > views > AboutView.vue**

<template>

<div class="counter-squared">

Square Value

{{$store.state.counter}}

<sup> 2 </sup> =

{{$store.getters.counterSquare}}

</div>

</template>

**Code in > src > components > Counter.vue**

<template>

<div class="counter"

:style="{ color: $store.state.colorCode }">

{{$store.state.counter}}

</div>

</template>

**Code in > src > components > CounterSquare.vue**

<template>

<div class="counter-squared">

{{$store.state.counter}}

<sup> 2 </sup> =

{{$store.getters.counterSquare}}

</div>

</template>

**Code in > router > index.js**

import { createRouter, createWebHashHistory } from 'vue-router'

import HomeView from '../views/HomeView.vue'

const routes = [

{

path: '/',

name: 'home',

component: HomeView

},

{

path: '/about',

name: 'about',

// route level code-splitting

// this generates a separate chunk (about.[hash].js) for this route

// which is lazy-loaded when the route is visited.

component: () => import(/\* webpackChunkName: "about" \*/ '../views/AboutView.vue')

}

]

const router = createRouter({

history: createWebHashHistory(),

routes

})

export default router

**Code in app.vue**

<template>

<nav>

<router-link to="/">Home</router-link> |

<router-link to="/about">About</router-link>

</nav>

<router-view/>

</template>

<style>

#app {

font-family: Avenir, Helvetica, Arial, sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

text-align: center;

color: #2c3e50;

}

nav {

padding: 30px;

}

nav a {

font-weight: bold;

color: #2c3e50;

}

nav a.router-link-exact-active {

color: #42b983;

}

</style>

--- ---- ----- ------ ------ ------ ------- ---- ----- ------

**State management using Composition API**

**Code in > src > store > index.js**

import { reactive } from "vue"

//import { reactive, readonly } from 'vue'

const state = reactive ({

counter : 0,

colorCode: 'blue'

})

const methods = {

decreaseCounter(){

if (state.counter <= 0 )

state.counter=0

else

state.counter--

},

increaseCounter(){

state.counter++

},

setColorCode(val){

state.colorCode=val

}

}

const getters = {

counterSquare(){

return state.counter \* state.counter

}

}

export default {

//state: readonly(state),

state,

methods,

getters

}

**Code in > src > views > HomeView.vue**

<template>

<div class="home">

<img alt="Vue logo" src="../assets/logo.png">

<div class="container" :style="{ color: store.state.colorCode}">

<!-- {{counter}} //No need of local variable -->

{{ store.state.counter}}

</div>

<div class="container">

{{store.state.counter}} <sup>2</sup> =

{{store.getters.counterSquare() }}

</div>

<div class="container">

<!-- <button btn-primary @click="decreaseCounter">-</button>

<button btn-primary @click="increaseCounter">+</button> -->

<button btn-primary @click="store.methods.decreaseCounter">-</button>

<button btn-primary @click="store.methods.increaseCounter">+</button>

</div>

<div class="container">

<!-- <input type="text" v-model="store.state.colorCode" placeholder="Enetr color code" /> Readonly ppty-->

<input type="text" v-model="colorCode" placeholder="Enetr color code" />

</div>

</div>

</template>

<script>

// @ is an alias to /src

//import { ref } from 'vue' -- No need of ref

import {inject , computed } from 'vue'

export default {

name: 'HomeView',

setup(){

const store = inject ('store') // inject the global store object to access globally

//let counter = ref(0) --- No need of local ref-counter value

// const decreaseCounter = function(){

// counter.value--

// }

// const increaseCounter = function(){

// counter.value++

// }

// return { counter,decreaseCounter,increaseCounter}

//return { store,decreaseCounter,increaseCounter} //No nned of local counter

const colorCode = computed(

{

get(){

return store.state.colorCode

},

set(val){

store.methods.setColorCode(val)

}

}

)

return { store, colorCode}

}

}

</script>

**Code in > src > views > AboutView.vue**

<template>

<div class="about">

<h1>Counter works everywhere</h1>

<h3> {{ store.state.counter}}</h3>

</div>

</template>

<script>

import { inject } from 'vue'

export default{

setup(){

const store = inject('store')

return { store } } } </script>

**Code in src > router > index.js**

import { createRouter, createWebHashHistory } from 'vue-router'

import HomeView from '../views/HomeView.vue'

const routes = [

{

path: '/',

name: 'home',

component: HomeView

},

{

path: '/about',

name: 'about',

// route level code-splitting

// this generates a separate chunk (about.[hash].js) for this route

// which is lazy-loaded when the route is visited.

component: () => import(/\* webpackChunkName: "about" \*/ '../views/AboutView.vue')

}

]

const router = createRouter({

history: createWebHashHistory(),

routes

})

export default router

**Code in src > App.vue**

<template>

<nav>

<router-link to="/">Home</router-link> |

<router-link to="/about">About</router-link>

</nav>

<router-view/>

</template>

<script>

import { provide } from 'vue'

import store from '@/store'

export default {

setup() {

provide ('store', store)

} } </script>

<style>

#app {

font-family: Avenir, Helvetica, Arial, sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

text-align: center;

color: #2c3e50;

}

nav {

padding: 30px;

}

nav a {

font-weight: bold;

color: #2c3e50;

}

nav a.router-link-exact-active {

color: #42b983;

}

</style>

**Code in main.js**

import { createApp } from 'vue'

import App from './App.vue'

import router from './router'

createApp(App).use(router).mount('#app')