npm install vuex --save

**Vuex Computed**

Create you store.js file and export variables

import Vue from 'vue';

import Vuex from 'vuex';

Vue.use(Vuex)

export const store = new Vuex.Store({

  state: {

    Fruits: [

      { name: 'banana', price: "25" },

      { name: 'apple', price: "15" },

      { name: 'pear', price: "18" },

      { name: 'strawberrie', price: "45" }

    ]

  }

})

Import it in main.js

import {store} from '@/store/store'

Vue.config.productionTip = false;

new Vue({

  store,

  router,

  render: h => h(App)

}).$mount("#app");

Now you can use it in any component

computed:{

   Fruits(){

     return this.$store.state.Fruits *//Fruits: array*

   }

  }

**Vuex Getters**

You can make functions on the global states that you made before

Sale Example:

In store.js file

export const store = new Vuex.Store({

  state: {

    Fruits: [

      { name: 'banana', price: "25" },

      { name: 'apple', price: "15" },

      { name: 'pear', price: "18" },

      { name: 'strawberrie', price: "45" }

    ]

  },

  getters:{

    saleFruits: state => {

      let saleFruits = state.Fruits.map(fr =>{

        return {

          name: fr.name + "  SALE!",

          price: fr.price / 2

        }

      })

      return saleFruits

    }

  }

})

Use it in any component

computed: {

    Fruits() {

      return this.$store.state.Fruits; *//Fruits: array*

    },

    saleFruits() {

      return this.$store.getters.saleFruits; *//saleFruits: array*

    },

  }

**Vuex Mutation Handlers**

Mutations is responsible for state changing the data and they can be tracked

With mutations we can see the changes where is coming from, it’s much better for debugging

من الاخر تستخدمها لما تعمل اكشنز علي الداتا اللي في store.js و ال state

First of all if we want to protect store data from any (outside function) to edit

Or if we want to edit it with mutation only we should type strict: true

They will print console error if we edit store state from outside funtion

export const store = new Vuex.Store({

  strict:true, *//to print console error for outside actions functions*

  state: {

    Fruits: [

      { name: 'banana', price: "25" },

      { name: 'apple', price: "15" },

      { name: 'pear', price: "18" },

      { name: 'strawberrie', price: "45" }

    ]

  },

  mutations: {

    reducePrice:state => {

      state.Fruits.forEach(fr => {

        fr.price -= 1

      });

    }

  }

})

Calling the mutations with commit in any component

methods:{

    reducePrice(){

      this.$store.commit('reducePrice');

    }

  }

**Vuex Actions**

Actions use to dispatch the mutations

Its better in (asynchronous code) than use mutations directly

mutations: {

    reducePrice:state => {

      state.Fruits.forEach(fr => {

        fr.price -= 1

      });

    }

  },

  actions:{

    reducePrice: context => {

      setTimeout(function(){

        context.commit('reducePrice')

      },2000)

    }

  }

Usage in component

methods:{

    reducePrice(){

*// this.$store.commit('reducePrice'); //Use Mutations Directly*

      this.$store.dispatch('reducePrice');  *//The same function but with Actions*

    }

  }

Pass Parameter

mutations: {

    reducePrice:(state, payload) => {

      state.Fruits.forEach(fr => {

        fr.price -= payload

      });

    }

  },

  actions:{

    reducePrice: (context, payload) => {

      setTimeout(function(){

        context.commit('reducePrice', payload)

      },2000)

    }

  }

In component

methods:{

    reducePrice(x){

*// this.$store.commit('reducePrice', x); //Use Mutations Directly*

      this.$store.dispatch('reducePrice', x);  *//The same function but with Actions*

    }

  }

**mapState & mapGetters & mapActions**

These are a shortcut use it when you need a lot of **State or Getter or Actions**

first of all you gonna use babel (install it )

then import mapGetters and mapActions to any componant

import {mapGetters, mapState, mapActions} from 'vuex'

 methods:{

*// reducePrice(x){*

*//   // this.$store.commit('reducePrice', x); //Use Mutations Directly*

*//   this.$store.dispatch('reducePrice', x);  //The same function but with Actions*

*// },*

    ...mapActions([

        'reducePrice',

        'Action2',

        'Action3'

      ])

  },

  computed: {

*// Fruits() {*

*//   return this.$store.state.Fruits;*

*// },*

    ...mapState([

      'Fruits',

      'State2',

      'State3'

    ]),

*// saleFruits() {*

*//   return this.$store.getters.saleFruits; //saleFruit: array*

*// },*

     ...mapGetters([

      'saleFruits',

      'Getter2',

      'Getter3'

    ])

  }