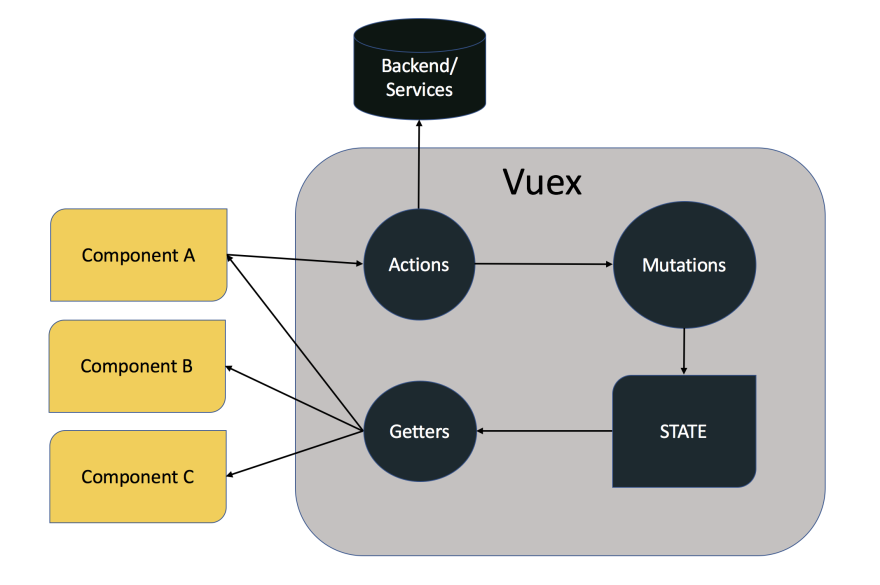
**Vuex**

Vuex dapat digunakan untuk menangani *state* (**state manajemen**), dimana data yang ada dapat didefinisikan dan dipusatkan pada sebuah *file* sehingga bisa digunakan oleh semua *component* yang ada.

Dengan menggunakan Vuex, komunikasi antar *component* menjadi lebih mudah. Sebagaimana yang kita ketahui bahwasanya dalam interaksi antar *component* agar dapat saling bertukar data adalah dengan menggunakan **props**, maka dengan Vuex kamu cukup mengakses **state** yang telah didefinisikan pada ***Vuex Store***.

Store pada Vuex merupakan container yang mengisolasi suatu **state management**. Bisa dikatakan **store** merupakan *inti* dari Vuex. Store terdiri dari lifecycle **state management** seperti *state, mutations, dan actions*. Vuex memberlakukan aturan yang akan menjalankan pola **state management**.



**Lifecycle Vuex Store**

1. **State**

State adalah sebuah object dimana tempat semua data aplikasi berada.

**const store = new Vuex.Store({**

**state: {**

**count: 1,**

**title: 'belajar-vuex'**

**}**

**})**

1. **Mutation**

Mutation adalah satu-satunya cara untuk merubah state. Jadi mutation ini bisa dibilang mirip event yang ada di dalam store.

const store = new Vuex.Store({

state: {

count: 1

},

mutations: {

increment (state) {

// mutate state

state.count++

}

}

})

1. **Action**

Action mirip dengan mutation, bedanya adalah mutation hanya bisa di akses secara synchronous sedangkan action bisa diakses secara asynchronous. Action ini tidak merubah state, tapi action akan memanggil mutation dan mutation yang akan merubah state.Kita butuh action ketika melakukan proses asynchronous dan perlu untuk merubah state. Misalnya saat selesai melakukan request pada rest api.

**const store = new Vuex.Store({**

**state: {**

**number: 0**

**},**

**mutations: {**

**count(state) {**

**state.number++**

**}**

**},**

**actions: {**

**count(newNumber){**

**newNumber.commit('count')**

**}**

**}**

**})**

1. **Getter**

Getter berfungsi untuk mengakses state.Dengan menggunakan Getter kita bisa mengolah terlebih dahulu state yang akan kita ambil. Jadi kita bisa memfilter data state sebelum di panggil.

getters: {

publishTodos(state){

return state.todos.filter(function(item){

return item.publish == true

})

},

},

1. **Modules**

Store juga dapat dimodulasikan melalui modules. Masing-masing modules yang dibuat terdiri dari state, mutation, dan actions nya sendiri. Kemudian setiap modules disatukan pada single tree store.

const module1 = {

state: { ... },

mutations: { ... },

actions: { ... },

getters: { ... }

}

const module2 = {

state: { ... },

mutations: { ... },

actions: { ... }

}

const module3 = {

state: { ... },

mutations: { ... },

actions: { ... }

}

const store = new Vuex.Store({

modules: {

a: module1,

b: module2,

c: module3

}

})

**Kesimpulan**

Vuex membantu mengelola data secara interaktif untuk saling bertukar satu sama lainnya. Atau saling berbagi data pada *component* di setiap aplikasinya.

**-delshady**