Rakamin Academy

Core Initiative

Project Based Internship

Javascript State and Event

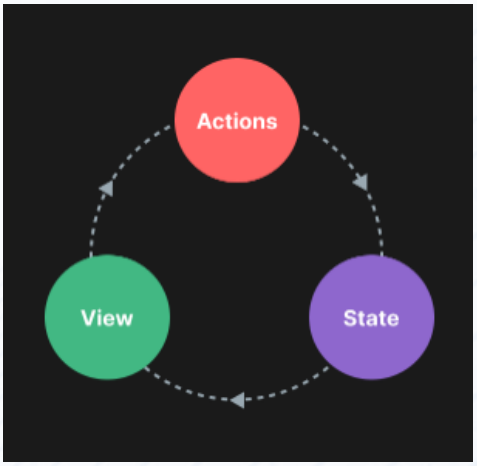
Pengenalan mengenai State dan Event pada Vue Js

Daftar Isi

A. Apa itu State?

Ketika anda sudah membangun sebuah program, anda akan menyadari akan ada beberapa component yang bergantung pada sebuah props tertentu. Akan rumit jika kita harus mengirimkan props melalui cara child to parent apabila props yang digunakan oleh salah satu component tidak memiliki hubungan langsung dengan component penyedia props tersebut / bukan "one way data flow" oleh karena itu masalah ini sering disebut juga dengan pengeboran props (drilling props) dimana anda akan mencari sumber props yang akan anda gunakan ditengah hirarki component anda.

Untuk sebuah program sederhana biasanya untuk melakukan penggunaan state akan membutuhkan 3 bagian :



Gambar , Reprsentasi One Way Data Flow

- State : sumber data.
- View : menampilkan state.
- Action : melakukan perubahan pada state.

a. State Management

Konsep state management lahir untuk menjadi solusi dari permasalahan tadi. Dengan cara ini, state anda akan disimpan melalui store (file terpisah) dan ketika anda ingin menggunakan state tersebut, anda dapat mengimport store pada component yang akan menggunakan state. Begitu pula anda dapat menyimpan function/method ke dalam store anda.

Anda dapat membuat store sebagai file terpisah dengan ekstensi javascript (.js), untuk penamaan pada dasarnya anda dibebaskan untuk memilih apapun, namun akan lebih mudah dipahami apabila anda memberi nama file store anda dengan nama "store".

File store tersebut nantinya akan mengimport "reactive" yaitu sebuah fitur yang disediakan oleh vue untuk melakukan state management sederhana. Lalu anda dapat mengeksport data pada store tersebut untuk dapat digunakan pada component lain ketika anda melakukan import.

```
import { reactive } from 'vue';

export default reactive({
    message: "Ini Store",
})
```

```
<template>
  <h1>{{store.message}}</h1>
</template>
```

```
<script>
import store from './store/store'

export default {
  name: "App",
  data(){
    return{
      store
    }
  }
};
</script>
```

Anda dapat menyimpan banyak hal didalam store, salah satunya adalah function/method. Contoh sederhana dari penyimpanan function/method pada store sebagai berikut.

```
import { reactive } from 'vue';

export default reactive({
    count: 0,
    addCount() {
        this.count++
    }
})
```

```
<template>
  <h1>{{store.count}}</h1>
  <button @click="store.addCount">+</button>
</template>
```

b. Library Vuex State Management

Vuex adalah pola manajemen status + perpustakaan untuk aplikasi Vue.js. Ini berfungsi sebagai penyimpanan terpusat untuk semua komponen dalam aplikasi, dengan aturan yang memastikan bahwa status hanya dapat dimutasi dengan cara yang dapat diprediksi. Karena vuex merupakan library eksternal, maka anda perlu menginstall terlebih dahulu dengan menggunakan npm.

npm install vuex@next –save

Yang membedakan vuex dengan reactive (fitur state management bawaan vue) adalah anda tidak harus mengimport store setiap kali anda ingin menggunakan state, namun anda perlu mengaitkan store pada main.js.

```
import { createStore } from 'vuex'

export default createStore({
    state: {
        number: 0
    }
})
```

```
import { createApp } from 'vue'
import App from './App.vue'
import store from './store/store'

createApp(App).use(store).mount('#app')
```

a. Mutations

State yang di store melalui vuex tidak akan bisa diubah kecuali melalui mutations. Hal ini untuk melindungi state dari perubahan yang tidak diinginkan yang nantinya akan mempengaruhi ketika state tersebut akan digunakan. Untuk menggunakan mutations, anda harus melakukan dispacth untuk menggantikan fungsi dari emit.

```
export default createStore({
    state: {
        number: 0
    },
    mutations: {
        count(state) {
            state.number++
        }
    }
})
```

```
<template>
  <button @click="$store.dispatch('addNumber',$store.state)">+</button>
</template>
```

b. Action

Action dan mutations akan selalu terkait, karna state tetap tidak akan dapat berubah sebelum anda melakukan commit mutations anda pada pada action. Dan yang harus anda perhatikan bahwa yang nantinya akan didispatch adalah action dan bukan mutation.

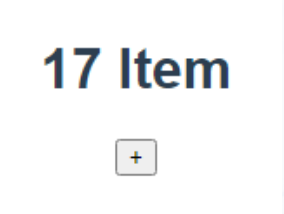
```
export default createStore({
    state: {
        number: 0
    },
    mutations: {
        count(state) {
            state.number++
        }
    },
    actions: {
        addNumber(newNumber) {
            newNumber.commit('count')
        }
    }
})
```

c. Getter

Selain anda dapat menyimpan data pada state, anda juga dapat menyimpan function untuk mengolah state. Didalam gatter anda dapat melakukan banyak hal seperti arrow function, traditional function, dan juga mengakses getter lain.

```
export default createStore({
    state: {
        number: 0
    },
    getters: {
        setView(state) {
            return state.number + " Item"
        }
    }
})
```

```
<template>
  <h1>{{$store.getters.setView}}</h1>
  <ComponentSatu/>
</template>
```



d. Modules

Anda apat membuat module-module untuk mengelompokkan kategori tertentu dari state, mutations, getters dan actions.

```
const moduleA = {
    state: {
        number: 0
    }
}

const moduleB = {
    state: {
        message: "module B"
    }
}

export default createStore({
    modules:{
        a: moduleA,
        b: moduleB
    }
})
```

B. Apa itu Event?

Event handling merupakan cara untuk anda merespon interkasi dari pengguna ketika menggunakan web. Pada beberapa materi sebelumnya kita sudah banyak menggunakan event handling hal ini menunjukkan event handling akan sering digunakan ketika anda membuat program vue. Untuk menginisiasi event handling anda dapat menggunakan "v-on" atau seperti yang sudah kita lakukan sebelumnya dapat menggunakan shortcut "@".

```
<template>
  <button @click="handleClick">+</button>
</template>
```

```
<template>
  <button v-on:click="handleClick">+</button>
</template>
```

Ada banyak jenis event yang dapat anda gunakan selain "click", antara lain :

- click
- scroll
- submit
- keyup
- Dll.

a. Event Modifier

Merupakan kebutuhan yang sangat umum untuk memanggil event.preventDefault() atau event.stopPropagation() di dalam event handler. Meskipun kita dapat melakukan ini dengan mudah di dalam metode, akan lebih baik jika metodenya murni tentang logika data daripada harus berurusan dengan detail acara DOM.

```
<template>
  <button v-on:submit.prevent="handleSubmit">+</button>
</template>
```

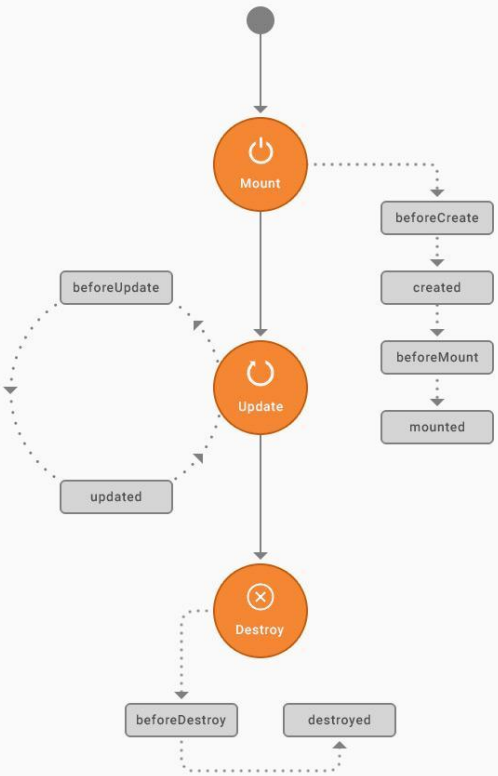
Ada beberapa modifier yang dapat anda gunakan untuk memodifikasi event seperti :

- .stop
- .prevent
- .self
- .capture
- .once
- .passiv

C. Lifecycle Hooks

Dalam menjalankan programnya, vue memiliki alur dalam menjalankan programnya. Anda dapat mengelompokkan program mana yang ingin anda jalankan dan kapan anda ingin menjalankan. Lifecycle ini dapat anda tambahkan pada properti ketika anda melakukan export. Ada 8 tahapan yang dapat anda gunakan dan anda dapat menjalankan program tertentu didalamnya.

- beforeCreate()
- created()
- beforeMount()
- mounted
- beforeUpdate()
- updated()
- beforeDestroy()
- destroyed()




```
<script>
export default {
  name: "ComponentSatu",
  props: {
  },
  beforeMount(){
    // do something
  }
};
</script>
```

References

https://vuejs.org/guide/scaling-up/state-management.html

https://docs.vuejs.id/v2/guide/state-management.html

https://medium.com/cetakk-id/memahami-konsep-state-management-pada-vue-js-e5cb126c06e9

https://www.codingame.com/playgrounds/6661/vuex-tutorial

https://blog.logrocket.com/introduction-to-vue-lifecycle-hooks/

https://vuex.vuejs.org/

https://www.digitalocean.com/community/tutorials/how-to-manage-state-in-a-vue-js-application-with-vuex

https://flaviocopes.com/vuex/

https://vueschool.io/articles/vuejs-tutorials/vuex-the-official-vuejs-store/