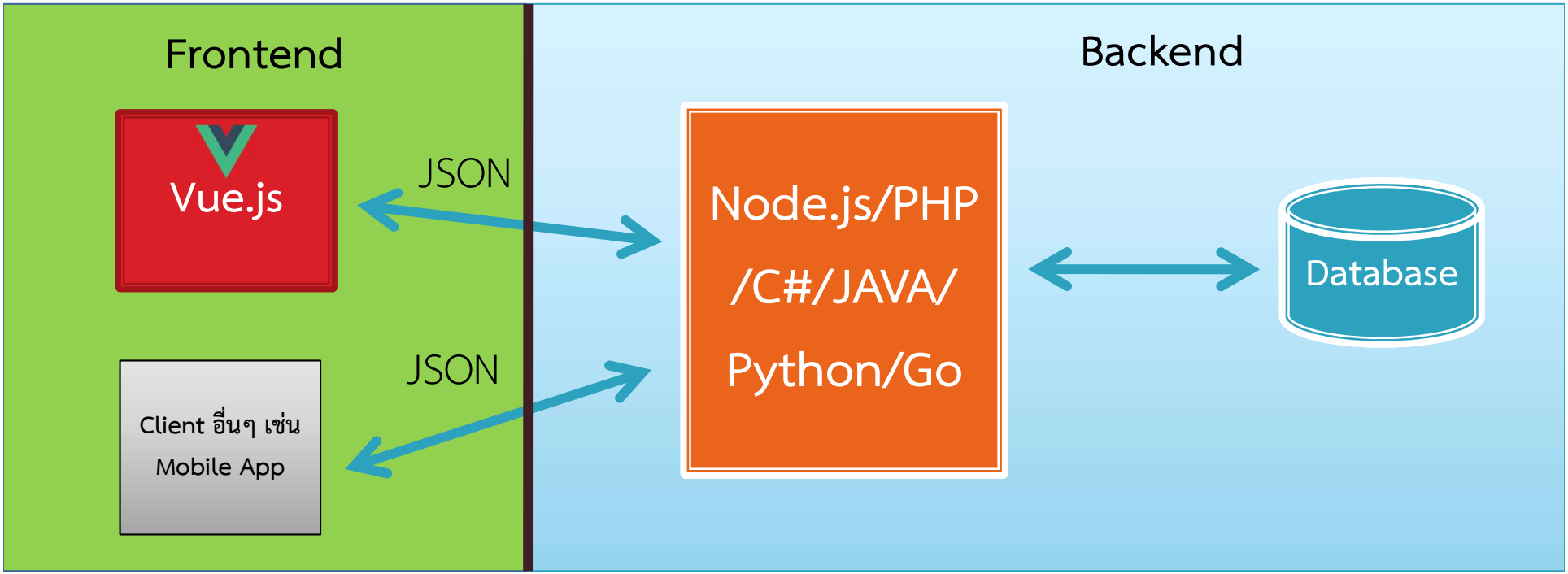
หลักสูตรออนไลน์ "พัฒนา Modern Web App ด้วย Vue.js 3 และ Composition API"

โค้ชเอก

codingthailand.com

ภาพใหญ่ของคอร์สนี้

- เขียน **Frontend** เป็น Web Application ด้วย **Vue.js**



การติดต่อกับ Backend

- https://github.com/axios/axios

npm install axios

State Management

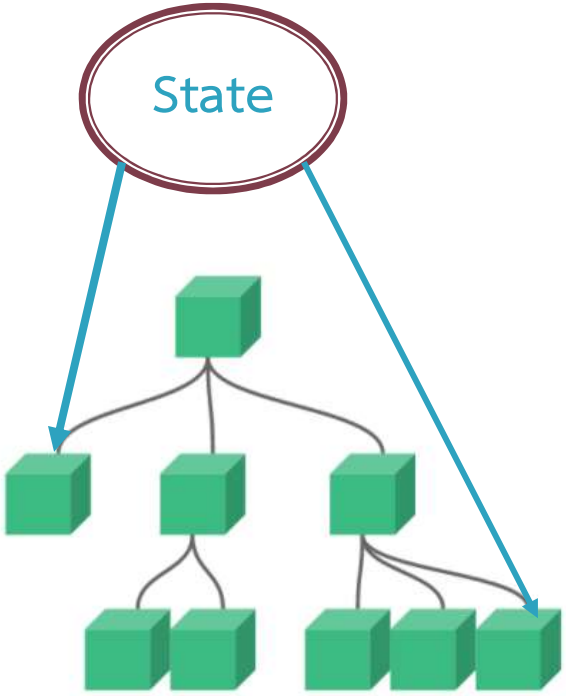
Vuex

- Vuex is a **state management pattern + library** for Vue.js applications. It serves as a centralized store for all the components in an application.

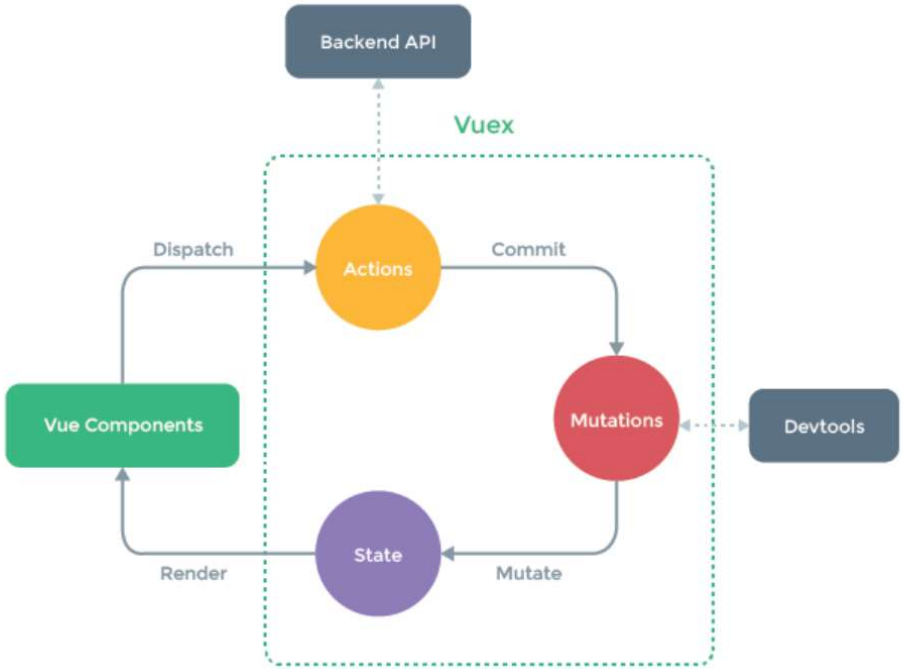
- https://next.vuex.vuejs.org/

- npm install vuex@4

Vuex



ภาพใหญ่ของ Vuex



Core Concepts

Mutations

- The only way to actually change state in a Vuex store is by committing a mutation. (Commit + track state changes)

```js
// ...
mutations: {
  increment (state, n) {
    state.count += n
  }
}
```

```js
store.commit('increment', 10)
```

Getters

- Sometimes we may need to compute derived state based on store state, for example filtering through a list of items and counting them.

```js
const store = createStore({
  state: {
    todos: [
      { id: 1, text: '...', done: true },
      { id: 2, text: '...', done: false }
    ]
  },
  getters: {
    doneTodos (state) {
      return state.todos.filter(todo => todo.done)
    }
  }
})
```

Property-Style Access

The getters will be exposed on the `store.getters` object, and you access values as properties:

```js
store.getters.doneTodos // -> [{ id: 1, text: '...', done: true }]
```

Actions

- Actions are similar to mutations, the differences being that:
- Instead of mutating the state, actions commit mutations.
- Actions can contain arbitrary asynchronous operations.
- สำหรับเรียก commit mutation

```js
const store = createStore({
  state: {
    count: 0
  },
  mutations: {
    increment (state) {
      state.count++
    }
  },
  actions: {
    increment (context) {
      context.commit('increment')
    }
  }
})
```

Modules

- Vuex allows us to divide our store into **modules**. Each module can contain its own state, mutations, actions, getters, and even nested modules.

```js
const moduleA = {
  state: () => ({ ... }),
  mutations: { ... },
  actions: { ... },
  getters: { ... }
}

const moduleB = {
  state: () => ({ ... }),
  mutations: { ... },
  actions: { ... }
}

const store = new Vuex.Store({
  modules: {
    a: moduleA,
    b: moduleB
  }
})

store.state.a // -> `moduleA`'s state
store.state.b // -> `moduleB`'s state
```

การใช้งาน Vuex และ **Composition API**

- https://next.vuex.vuejs.org/guide/composition-api.html

```
import { useStore } from 'vuex'

export default {
  setup () {
    const store = useStore()
  }
}
```

การใช้งาน Vuex และ Composition API

- **Accessing State and Getters**

```
import { computed } from 'vue'
import { useStore } from 'vuex'

export default {
  setup () {
    const store = useStore()

    return {
      // access a state in computed function
      count: computed(() => store.state.count),

      // access a getter in computed function
      double: computed(() => store.getters.double)
    }
  }
}
```

การใช้งาน Vuex และ Composition API

- **Accessing Mutations and Actions**

```
import { useStore } from 'vuex'

export default {
  setup () {
    const store = useStore()

    return {
      // access a mutation
      increment: () => store.commit('increment'),

      // access an action
      asyncIncrement: () => store.dispatch('asyncIncrement')
    }
  }
}
```

Deployment

- https://cli.vuejs.org/guide/deployment.html

The end