Axios

Prototype을 이용하여 공통적으로 사용될 함수 묶음에 Axios function을 정의해 놓았으므로, 만들어진 함수만 호출하여 사용하면 됩니다.

폴더 구조

Module별로 그룹지어 js 파일을 나누었습니다.

```
├─api
|      index.js
|      module1.js
|      module2.js
```

module1.js

```
import axios from 'axios';

const module1 = {};

module1.getModule = (url, data) => {
  try {
    return axios({
      method: 'get',
      baseURL: process.env.VUE_APP_BASEURL,
      url,
      params: { ...data },
    });
  } catch (error) {
    return console.error(`[API][module1] getModule is failed url: ${url}, error: ${error}`);
  }
};

module1.postModule = (url, data) => {
  try {
    return axios({
      method: 'post',
      baseURL: process.env.VUE_APP_BASEURL,
      url,
      data,
    });
  } catch (error) {
    return console.error(`[API][module1] postModule is failed url: ${url}, error: ${error}`);
  }
};

module1.putModule = (url, data) => {
```

```
  try {
    return axios({
      method: 'put',
      baseURL: process.env.VUE_APP_BASEURL,
      url,
      data,
    });
  } catch (error) {
    return console.error(`[API][module1] putModule is failed url: ${url}, error: ${error}`);
  }
};

module1.patchModule = (url, data) => {
  try {
    return axios({
      method: 'patch',
      baseURL: process.env.VUE_APP_BASEURL,
      url,
      data,
    });
  } catch (error) {
    return console.error(`[API][module1] patchModule is failed url: ${url}, error: ${error}`);
  }
};

module1.deleteModule = (url) => {
  try {
    return axios({
      method: 'delete',
      baseURL: process.env.VUE_APP_BASEURL,
      url,
    });
  } catch (error) {
    return console.error(`[API][module1] deleteModule is failed url: ${url}, error: ${error}`);
  }
};

export default module1;
```

process.env.VUE_APP_BASEURL은 env 환경변수에서 선언하여 기본적인 API Base Url을 설정 합니다.

.env

.env 파일은 프로젝트 Root 경로에 생성하여 사용하며, 'VUE_APP'으로 이름을 시작하게 되면, Vue CLI으로 인하여 자동으로 process.env에 주입됩니다. 그 외의 이름을 사용하려 하면, vue.config.js에서 Webpack 설정을 해주어야 사용할 수 있습니다.

```
VUE_APP_BASEURL=https://jsonplaceholder.typicode.com
```

index.js

```
import module1 from './module1';
import module2 from './module2';

export { module1, module2 };
```

globalFunction.js

```
import * as apiModule from '@/api/index';

let globalFunction = {};

/**
 * @description axios function
 */
globalFunction = Object.assign({}, apiModule);

const globalFunctions = {
  install(Vue) {
    Vue.prototype.$globalFunctions = globalFunction;
  },
};

if (typeof window !== 'undefined' && window.Vue) {
  window.Vue.use(globalFunctions);
}

export default globalFunctions;
```

Axios 사용법

```
import Vue from 'vue';

export default {
  mounted() {
    const data = {
      userId: 2,
    };
    this.axiosGet('/posts', data);
  },
  methods: {
    async axiosGet(url, data) {
      const get = await this.$globalFunctions.module1.getModule(url, data);
      const get2 = await Vue.prototype.$globalFunctions.module1.getModule(url, data);
      console.log(get.data);
      console.log(get2.data);
    },
  },
};
```

아래의 방법중 'this'로 시작되는 것은 'this'가 생성된 Vue instance를 가르킬때 즉, 'vue' 파일 안에서 사용하면 되며, Vue.prototype의 경우에는 'this'가 Vue instance가 아닌경우에 사용하여야 합니다.

```
this.$globalFunctions.module1.getModule(url, data);
Vue.prototype.$globalFunctions.module1.getModule(url, data);
```

Vue.prototype를 사용할 경우는 아래의 예시를 참고하시기 바랍니다.

store/modules/common/actions.js

```
import Vue from 'vue';
import { commonActionType } from '@/store/actionsType';
import { commonMutationType } from '@/store/mutationsType';

const actions = {
  /**
   * @description action 설명
   * @param commit
   * @param payload 전달 받은 값
   */
  async [commonActionType.ACTION_SAMPLE]({ commit }, payload) {
    const vm = Vue.prototype.$globalFunctions;
    commit(commonMutationType.SET_LOADING_COMPLETE, payload);
    if (payload) {
      const response = await vm.module1.getModule('/posts');
      commit(commonMutationType.SET_POSTS_DATA, response);
    }
  },
};

export default actions;
```