Vue.js (data)

Vue.js (data)
1. store에 있는 data로 update
2. watch (object, 중첩 데이터의 value 변화 감지)
3. 변수로 :id 주기, 이미지(require) 넣기
4. axios.all

1. store에 있는 data로 update

1. created() 에서 store쪽으로 action => mutation => state 갱신 => data에 넣기 X
2. store에 있는 state 는 컴포넌트에서 computed에서 가져와야 함
   - https://vuex.vuejs.org/kr/guide/state.html
   - map헬퍼 사용 혹은 변수 지정 후 return this.$store.state.??? 처럼 사용
3. jubging 사용

```
// src/components/mission/MissionActive.vue

export default {
  data() {
    return {
        badges: [
          {
            name: '좋아요 뱃지',
            imgFolder_name: 'like',
            description: '좋아요를 눌러 뱃지를 획득하세요!',
            // current에 store state에 있는 missions객체의 like 키 값을 넣고 싶
음
            current: '',
            bronze: 10,
            silver: 50,
            gold: 100,
            bg_image: 'like.jpg',
          },
       ]
    }
  },
  computed: {
    // store state의 missions
    ...mapState(['missions'])
  },
  watch: {
    // store state의 missions 객체가 바뀔 때 함수 실행
    // 처음 상위 컴포넌트(src/components/mission/Mission.vue)의 created에서 한
번 missions를 axios get하는데, 하위 컴포넌트(여기)가 만들어지기 전 axios요청이 완료
되지 않아서 missions에 데이터가 없다. 그래서 watch로 missions를 지켜보다가 axios요
청이 완료돼서 data가 들어가면 함수를 실행시켜 새로 this.badges를 갱신해준다.
    missions() {
      this.badges[0].current = parseInt(this.missions.like)
    }
  }
```

2. watch (object, 중첩 데이터의 value 변화 감지)

1. 참고 자료
    - vue watch 속성에서 Object와 같이 중첩된 값의 변경을 감지하자 https://blog.woolta.com/categories/10/posts/196
    - Vue에서 중첩 데이터를 감시하는 법 https://ui.toast.com/weekly-pick/ko_20190307
2. 줍깅 예시

```
// src/views/user/SignUp.vue

data() {
    return {
        // object형식의 중첩된 데이터 credentials
        credentials: {
            email: '',
            password: '',
            passwordConfirmation: '',
            nickname: '',
            profilePath: '~~~',
        },
    }
},
watch: {
    // credentials가 변화될 때마다 this.checkForm() 함수를 실행
    // 그냥 credentials() {this.checkForm()} 이렇게 쓰면 실행되지 않는다.
    credentials: {
        deep: true,
        handler() {
            this.checkForm()
        }
    },
},
```

3. 변수로 :id 주기, 이미지(require) 넣기

1. :id로 변수형 id 설정 가능

```
// src/components/mission/MissionActive.vue
// :id로 변수형 id 설정 가능

<div class="current_badge" :id="`current-badge-${badge.imgFolder_name}`">
  <span>{{ badge.current }}</span>
  <span><font-awesome-icon icon="sort-down" class="icon"/></span>
</div>
```

2. 이미지 변수로 사용하기 (require) : 줍깅 예시
    - vue 이미지 변수로 연결하기 https://ordinary-code.tistory.com/85

```
// src/components/mission/MissionActive.vue
// :style로 넣기
<div class="..." :style="{ backgroundImage: 'url(' +
require(`@/assets/bg/${badge.bg_image}`) + ')'}">
    <h2 class="title">{{ badge.name }}</h2>
    <span class="sub">{{ badge.description }}</span>
</div>

// src/components/mission/MissionJubging.vue
// :src로 넣기
<div class="badge-group">
    <img class="badge"
:src="require(`@/assets/badge/${badge.imgFolder_name}/bronze.jpg`)" alt="">
    <div class="badge-bridge-bronze"></div>
</div>
```

4. axios.all

1. axios 멀티 요청 https://xn--xy1bk56a.run/axios/guide/usage.html#post-%EC%9A%94%EC%B2%AD
    - 여러 개의 요청을 동시 수행할 경우 `axios.all()` 메서드를 사용합니다.
    - 잘 짜서 await/async나 .then으로 동기처리(순차적) 해도 되지만 말그대로 "동시에" 요청을 보내고 결과를 사용해야 할 때도 있다. 그럴 때 쓰면 유용함.
    - 그런 면에서, 아래 signup는 all을 사용했지만 꼭 all을 사용해야만 하는 건 아님. + await/async을 안 써봐서 + .then으로 쓰면 길어져서 all을 사용
2. jubging 내 사용

```
// views/user/SignUp.vue
import { HTTP } from '@/util/http-common' // baseurl export

axios.all([HTTP.post('user/emailck', this.credentials),
HTTP.post('user/nicknameck', this.credentials)])
  .then(axios.spread((res1, res2) => {
    ...
    // user/emailck의 post 요청의 결과 res1
    // user/nicknameck의 post 요청 결과 res2
}))
```