Mixins VS HOC(High Order Component)

1부

둘 다 나온 배경

• 여러 컴포넌트 간에 공통으로 사용하고 있는 로직, 기능들을 재사용하겠다!

예를 들어.. NewsView, JobsView, AskView 컴포넌트가 존재한다고 가정

News | Ask | Jobs Apple buys autonomous driving company Drive.ai 199 by Deimorz Introducing people.kernel.org 15 by Tomte BTrDB: Berkeley Tree Database 149 by tectonic NewslAsk IJobs Ask HN: Can we create a new internet where search engines are irrelevant? by subhrm Ask HN: What do you do with your Raspberry Pi? 1603 by xylo

News를 가져올때와 Ask, Jobs를 가져올 때 로직이 똑같다.

134

by pierlu

Oracle Dyn DNS Services Shutting Down in 2020

이런 로직이 3개의 컴포넌트에 똑같이 존재 => 불편러 소환됨

```
import ListItem from '../components/ListItem.vue';
import bus from '../utils/bus.js';
export default {
    created(){
        bus.$emit('start:spinner');
        this.$store.dispatch('FETCH_ASK') /* NEWS, JOBS *,
                     .then(()=>{
                         console.log('fetched!');
                         bus.$emit('end:spinner');
                     })
                     .catch(err=>{
                         console.log(err);
                     });
    },
    components: {
        ListItem,
    },
}
```

HOC(High Order Component) 란?

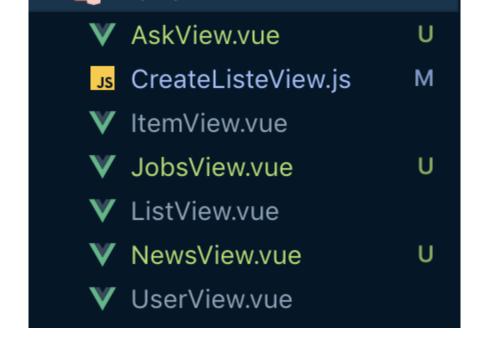
리액트 진영의 함수형 프로그래밍에서 기반한 컴포넌트 개발 패턴

- 컴포넌트의 코드를 재사용하기 위한 방법
- 캡슐화(encapsulation)와 컴포넌트 추상화를 구현하는 방법
- 컴포넌트의 로직을 훼손하지 않고 재사용성을 최대한 끌어올리겠다는 전략

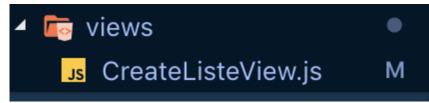
```
import ListView from './ListView.vue'; // 공통 컴포넌트
export default function createListView(name) {
    return {
       // 재사용할 인스턴스(컴포넌트) 옵션 & 로직들이 들어갈 자리
       name, // === name: name
       created() { // News, Ask, Jobs 컴포넌트에 있었던 로직
           this.$store.dispatch('FETCH_LIST',this.$route
                    .then(() => {
                       console.log('fetched!');
                   })
                   .catch(err => {
                       console.log(err);
                   });
       },
        render(createElement) {
           return createElement(ListView);
```

*router.js

```
path: '/news',
    name: 'news',
    // component:NewsView
    component: createListeView('NewsView')
},
{
    path: '/jobs',
    name: 'jobs',
    // component:JobsView
    component: createListeView('JobsView')
},
{
    path: '/ask',
    name: 'ask',
    // component:AskView
    component: createListeView('AskView')
},
```



• 후



Mixins

- Mixins의 존재 이유도 똑같다.
- 여러 컴포넌트 간에 공통으로 사용하고 있는 로직, 기능들을 재사용위함.

*ListMixins.js

```
import bus from '../utils/bus.js'
export default {
    // 재사용할 컴포넌트 옵션 & 로직
    created(){
        bus.$emit('start:spinner');
        this.$store.dispatch('FETCH_LIST', this.$route.name
                    .then(()=>{
                        console.log('fetched!');
                        bus.$emit('end:spinner');
                    })
                    .catch(err=>{
                        console.log(err);
                    });
```

*Jobs, Ask, NewsViesw.vue

```
import ListItem from '../components/ListItem.vue';
import bus from '../utils/bus.js';
import ListMixin from '../mixins/ListMixin.js'

export default {
    components: {
        ListItem,
    },
    mixins:[ListMixin]
}
```

컴포넌트의 갯수를 줄이진 않지만, 각 컴포넌트의 로직 코드량이 줄어든다.

둘의 차이점은..?

HOC

• 부모와 자식 사이에 컴포넌트가 하나 더 생김

Mixins

• 컴포넌트가 생기지 않음

각 장단점은..? 다음 2부에서~