7/13 ~ 7/17 2주차 - 2

Vue의 디렉티브 추가 내용

Vue에서 제어문을 사용을 위한 디렉티브를 다룬다.

조건문

- v-if
- v-else-if
- v-else
- v-show

반복문

- v-for

v-if, v-else-if, v-else, v-show는 조건에 따라 렌더링 여부를 결정하는 기능이다.

```
<div id="info">
  <label>나이를 입력하세요.</label>
  <input type="text" v-model="age" />
  <img v-if="age < 0" src="../images/error.png" />
  <img v-show="age < 0" src="../images/error.png" />
  당신은
  <span v-if="age < 0">입력 오류</span>
  <span v-else-if="age < 8">미취학</span>
  <span v-else-if="age < 19">미성년</span>
  <span v-else>성년</span>
</div>

<script>
  let model = {age:"" }

  let info = new Vue({el : "#info", data : model });
</script>
```

연속해서 등장

페이지를 실행하고 나이에 15를 입력해보면.

나이를 입력하세요. 15

당신은 미성년

```
<hr>
"
                당신은
"
<span>미성년</span>
```

미성년 부분이 출력된다.

v-if와 v-show에 대해서

	v-if	v-show
조건 불일치 시 렌더링 방식	렌더링하지 않음	display=none으로 처리
이에 따른 비용	토글 비용이 큼	초기 렌더링 비용이 큼
유용한 경우	자주 바뀌지 않을 때	자주 바뀔 때

v-if 는 애초에 요소를 그리지 않기 때문에 <!— —>> 으로 처리하고

v-show는 그린 후 display-none으로 안보여 주는 것.

비용등을 고려하는 것이 좋다.

```
<input type="text">
<!---->
<img src="../images/error.png" style="display: none;">
```

```
<input type="text">
<img src="../images/error.png">
<img src="../images/error.png" style>
```

v-for

```
<ul>
  <li v-for="(value[, key[, index]]) in data">{{index}}, {{key}}, {{value}}</li>
</ul>
```

- 구성요소
  - data는 배열 또는 객체이다.
  - value는 data안에 있는 개별 item의 값을 의미한다.
  - key는 data가 객체일 경우는 속성이름, 배열의 경우는 index이다.(기억해두기)
  - index는 data가 객체일 경우는 0부터 시작하는 index이고 배열의 경우는 빈 값이다.
  - value를 묶는 ()는 생략할 수 있다. (,만으로 가능할 듯?)

:key 속성

- vue는 렌더링된 목록을 갱신할 경우 DOM 요소를 이동하는 대신 다시 렌더링 한다.

  → 만약 100개의 아이템이 있고 1번째가 지워지면 99개의 아이템을 다시 렌더링 한다...
- 이때 :key 속성을 이용해 인덱스와 같은 고유ID를 설정하면 DOM요소를 재활용해서 이동할 수 있다. (kanban board에서 활용할 수 있을거 같은데? 👳)

사용 예

```html
<div id="app">
  <h3>객체 데이터 표현</h3>
  <ul>
    <li v-for="(value, key, index) in ironMan" :key="key">{{index}}, {{key}}, {{value}}</li>
  </ul>

  <h3>배열 데이터 표현</h3>
  <ul>
    <li v-for="(value, key) in heroes" v-if="key %2==0">{{key}}, {{value}}</li>
  </ul>
</div>

<script>
  let model = {
    heroes : [ "아이언맨", "헐크", "토르", "캡틴아메리카", "비전" ],
    ironMan : {
      name : "토니스타크",
      nickName : "깡통"
    }
  }

  let app = new Vue({
    el : "#app",
    data : model
  });
</script>
```

유니크한 속성명을 이용해서 :key 설정

당연히 조건문과의 연계도 가능

객체 데이터 표현

- 0, name, 토니스타크
- 1, nickName, 깡통

배열 데이터 표현

- 0, 아이언맨
- 2, 토르
- 4, 비전

<template> 태그

여러 요소를 묶어서 반복 렌더링 할때 요소들을 묶어주는 태그이다.

실제 렌더링 내용에는 포함되지 않고 단지 요소들을 Grouping 한다.

v-pre

해당 태그와 자식 태그들은 vue를 사용하고 있지 않는다고 알려주는 디렉티브이다.

왜 이런게 필요할까? ❓

→ 페이지를 처리하는 과정에서 Vue가 관심 갖지 않고 건너 뛴다. 따라서 그만큼 속도의 향상을 얻을 수 있음.

만약 이 부분에 {{ }} 같은 Vue 관련 디렉티브를 사용한다면 당연히 컴파일 안되고 그대로 출력 !!

v-cloak

UI가 복잡해지고 Vue가 처리하는 내용이 많아지면 렌더링이 느려져 일시적으로 콧수염 표현식이 화면에 나타난다 아주 짧기만 계속해서 새로고침을 하면 뜨는게 보인다.

이를 방지하기 위해 화면 초기에 컴파일 되지 않은 템플릿은 나타나지 않게 v-cloak를 사용합

컴파일이 끝나면 v-cloak속성은 자동으로 삭제된다.

처음에는 css에 의해 display가 none으로 설정된다.

사용자 정의 디렉티브

사용자 정의 디렉티브를 정의할 때는 Vue.directive(id, definition) 형태의 메서드를 사용한다.

```
Vue.directive('my-directive', {
  bind: function () {},
  inserted: function () {},
  update: function () {},
  componentUpdated: function () {},
  unbind: function () {}
})
```

사용할때는 v-my-directive라고 써서 사용 🤝

- definition 부분은 객체 형태로 디렉티브의 라이프 사이클 별로 동작할 콜백 함수를 등록한다. 일반적으로 콜백을 간단히 사용할 때는 아래 처럼 bind와 update에서 동작할 function만 등록하기도 한다.

```
Vue.directive('my-directive', function () {
    // bind와 update 과정에서 호출됨
  }
)
```

라이프 사이클 콜백의 파라미터

라이프 사이클 콜백의 파라미터로는 elm, binding, vnode가 전달된다.

- elm: 디렉티브가 적용되는 대상 엘리먼트
- binding: 전달되는 파라미터 참조에 사용되는 객체
- vnode: Vue compiler에 의해 생성되는 virtual node