7/6 ~ 7/10 1주차 - 6

Vue 디렉티브의 대해서 알아보자.

디렉티브는 지시자로 View에 데이터를 표현하는등의 용도로 사용되는 특별한 녀석.
v-를 접두어로 사용한다.

주요 디렉티브 목록.

1. 텍스트 표현 : v- text, v-html, v-once
2. html 속성 바인딩 : v-bind
3. 양방향 데이터 바인딩 : v- model
4. 제어문 : v-if, v-else, v-else-if, v-show
5. 반복문 : v-for
6. 기타 : v-pre, v-cloak

디렉티브	설명
v-text, {{}}	innerText 속성에 연결되며 문자열 그대로 화면에 표현. Model과의 반응성이 유지됨
v-html	innerHtml 속성에 연결되며 문자열 중 html 태그를 파싱해서 화면에 나타냄. Model과의 반응성이 유지됨
v-once	v-text,v-html에 부가적으로 사용되며 v-once는 한번 연동 후 반응성이 제거됨

```
<ul id="app">
  <li>{{message}}</li>
  <li v-text="message"></li>
  <li v-html="message"></li>
  <li v-once>{{message}}</li>
</ul>

<script>
  let model={
    message:"<i>Hi Vue</i>"
  }

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

- Hi Vue
- Hi Vue
- *Hi Vue*
- Hi Vue

먼저 script 영역을 살펴보면 model이 선언되어있다.

model은 JSON으로 message속성이 <I>Hi Vue로 등록되어있다.

Vue rorcpdpsms el이 #app이므로 html영역에서 id 속성이 app인 화면 요소를 찾는다.

즉 <ul id="app">이 Vue 객체의 스코프이다. 그리고 data에 model이 선언되어있으므로 View에 data를 연결한다.

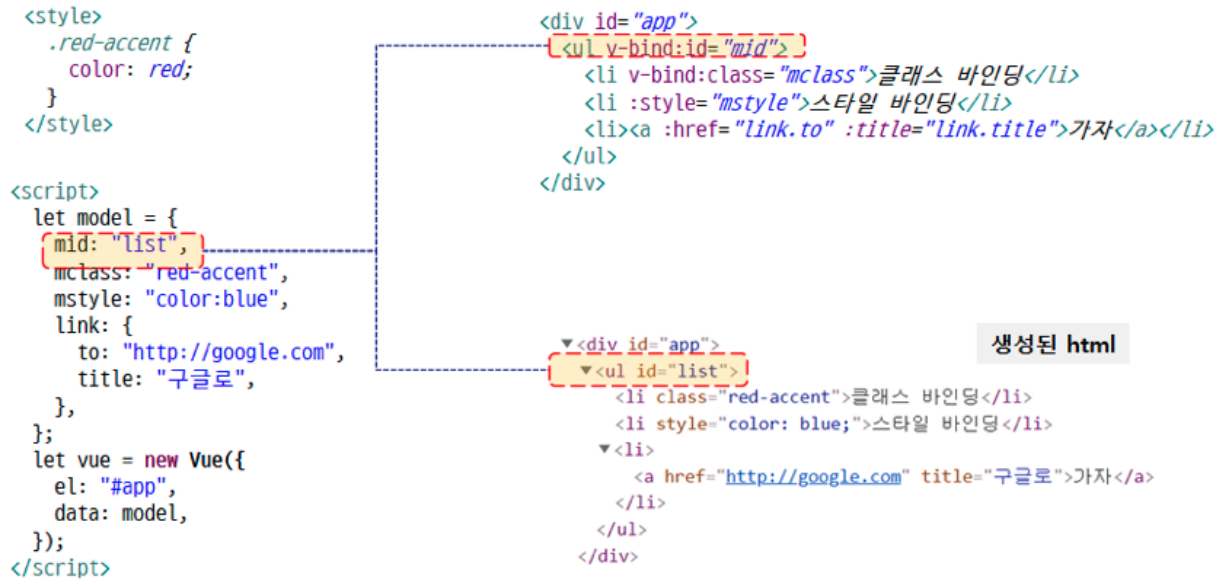
{{}}나 v-text, v-once는 html이 파싱되지 않고 그냥 문자열로 출력된다.

v-html은 html코드가 파싱되서 이탤릭으로 표현된다.

v-html은 model에 악의적인 JavaScript가 심어져 있을 때 XSS 공격의 위험이 있으므로 가급적 v-text의 사용을 권장한다. 화면은 View에게 처리 시킴

v-bind는 html의 속성이 id, class, style에 model 값을 연결할 때 사용한다.

v-bind → :



위 예제는 id, class, style, href, title

실제 화면을 디자인 할 때 미리 다양한 CSS들을 만들어 놓고 동적으로 class를 바꿔가며 스타일을 변경하는 경우가 많은데 이 경우 아주 유용한 기능이다.

html의 class 속성의 경우 여러 값이 설정될 수 있는데 Vue에서 v-bind를 통해서도 당연히 여러 값을 설정할 수 있다. 이 경우 객체 형태로 값을 전달하는데 boolean 타입의 model 값을 이용해서 해당 class를 on/off 처리할 수도 있다.

```
<div id="app">
  <div :class="{set1: s1, set2: s2, set3: s3}">
    여기에 머 쓰면 스타일이 막 바뀐다며?
  </div>
</div>


<script>
  let vm = new Vue({
    el : "#app",
    data : {
      s1 : true,
      s2 : false,
      s3 : true,
    },
  });
</script>
```

```
<style>
  .set1 {
    background: aqua;
    color: purple;
  }

  .set2 {
    text-decoration: red;
    font-style: italic;
  }

  .set3 {
    border: 1px solid blue;
   }
</style>
```

위 예는 app 내부의 div 클래스 속성에 각각 ser1, set2, set3를 바인딩 시키는데 s1, s2, s3 값을 사용한다.

> 이 값들은 model에서 각각 true, false, true로 되어있으므로 실제 생성된 html 코드에는 set1과 set2만 적용된다.

v-model

View와 Model 사이의 양방향 데이터 바인딩에 사용되는 디렉티브이다.

일반적으로 사용자로부터 값을 입력받는 <input>, <select>, <textarea> 등에서 사용되며 사용자가 입력한 값을 모델에 반영시켜준다. 당연히 모델이 변경되면 입력요소의 값도 변경된다.

다중 선택이 지원되는 <select>나 checkbos 의 경우의 값은 배열 객체와 연결되며 나머지 요소들은 단일 값과 연결된다.

수식어

v-model 디렉티브에는 특별한 기능을 추가하기 위해 수식어를 사용할 수 있다. 다음은 주요 수식어들이다.

수식어	설명
lazy	입력 폼에서 change 이벤트가 발생하면 데이터 동기화
number	입력 값을 parseInt 또는 parseDouble를 이용해 number 타입으로 저장
trim	입력 문자열의 앞 뒤 공백을 자동으로 제거

이런 수식어는 '.'를 이용해서 설정하며 여러 속성을 chaining 할 수도 있다. 다음 코드는 첫 번째 input 요소에 favorite라는 model 속성을 연결하는데 **trim**에 의해 입력 값에서 앞, 뒤 공백을 제거하는 것이다. 두 번째 input 요소에는 year라는 속성이 연결되고 **number**이므로 숫자로 parsing 되며 **lazy**에 의해 change 이벤트가 발생하면 model에 반영된다. 기본 속성은 key 이벤트가 발생할 때마다 반영된다.

```
<input type="text" v-model.trim="favorite">
<input type="text" v-model.lazy.number="year">
```

수식어

v-model 디렉티브에는 특별한 기능을 추가하기 위해 수식어를 사용할 수 있다. 다음은 주요 수식어들이다.

수식어	설명
lazy	입력 폼에서 change 이벤트가 발생하면 데이터 동기화
number	입력 값을 parseInt 또는 parseDouble를 이용해 number 타입으로 저장
trim	입력 문자열의 앞 뒤 공백을 자동으로 제거

이런 수식어는 '.'를 이용해서 설정하며 여러 속성을 chaining 할 수도 있다. 다음 코드는 첫 번째 input 요소에 favorite라는 model 속성을 연결하는데 **trim**에 의해 입력 값에서 앞, 뒤 공백을 제거하는 것이다. 두 번재 input 요소에는 year라는 속성이 연결되고 **number**이므로 숫자로 parsing 되며 **lazy**에 의해 change 이벤트가 발생하면 model에 반영된다. 기본 속성은 key 이벤트가 발생할 때마다 반영된다.

```
<input type="text" v-model.trim="favorite">
<input type="text" v-model.lazy.number="year">
```