



7/20 ~ 7/24 3주차 - 5

Vue 코드를 짜다가 막힌 것.

Props

Prop 대소문자 구분(camelCase vs kebab-case)

HTML 어트리뷰트는 대소문자 구분이 없기 때문에 브라우저는 대문자를 소문자로 변경하여 읽는다.

그렇기 때문에 카멜 케이스(대소문자 혼용)로 prop의 이름을 정한 경우에 DOM 템플릿 안에서 케밥 케이스(하이픈으로-연결된-구조)를 사용하여야 올바르게 동작한다.

```
Vue.component('blog-post', {  
  // JavaScript에서의 camelCase  
  props: ['postTitle'],  
  template: '<h3>{{ postTitle }}</h3>'  
})
```

```
<!-- HTML에서의 kebab-case -->  
<blog-post post-title="hello!"></blog-post>
```

아래와 같이 prop을 속성 이름과 타입을 포함하는 오브젝트로 선언함으로써 타입이 지정된 prop의 리스트를 구현할 수 있다.

```

props: {
  title: String,
  likes: Number,
  isPublished: Boolean,
  commentIds: Array,
  author: Object,
  callback: Function,
  contactsPromise: Promise // or any other constructor
}

```

컴포넌트를 읽기 좋게 문서화할 뿐 아니라 브라우저의 자바스크립트 콘솔에서도 잘못된 타입이 전달된 경우 경고를 띄워줄 수 있도록 해준다.

타입 체크와 prop 유효성 검사에서 확인 가능하다.

정적 prop

```
<blog-post title="My journey with Vue"></blog-post>
```

동적 prop

```

<!-- 변수에 담긴 값을 동적으로 할당 -->
<blog-post v-bind:title="post.title"></blog-post>

<!-- 복잡한 표현식의 값을 동적으로 할당 -->
<blog-post
  v-bind:title="post.title + ' by ' + post.author.name"
></blog-post>


```

문자열, 숫자형, boolean, 배열, Object, Object의 속성 전달이 가능하다.

Props - Vue.js

이 페이지는 여러분이 이미 컴포넌트 기초를 읽었다고 가정하고 쓴 내용입니다. 컴포넌트가 처음이라면 기초 문서를 먼저 읽으시기 바랍니다.

HTML 어트리뷰트는 대소문자 구분이 없기 때문에 브라우저는 대문자를

 <https://kr.vuejs.org/v2/guide/components-props.html#%EC%A0%95%EC%A0%81-amp-%EB%8F%99%EC%A0%81-prop-%EC%A0%84%EB%8B%AC%ED%95%98%EA%B8%B0>

예제 코드들이 있다.

단방향 데이터 흐름

- 모든 prop들은 부모와 자식 사이에 단방향으로 내려가는 바인딩 형태이다.
- 부모의 속성이 변경되면 자식 속성에게 전달되지만, 반대 방향으로 전달되지 않는다.
- 자식의 데이터가 부모에게 전달되는 것을 막는 것은 자식 요소가 의도치 않게 부모 요소의 상태를 변경함으로써 앱의 데이터 흐름을 이해하기 어렵게 만드는 일을 막기 위해서이다.
- 부모 컴포넌트가 업데이트될 때 마다 자식 요소의 모든 prop들이 최신 값으로 새로고침된다.
- 곧 사용자가 prop을 자식 컴포넌트 안에서 수정해서는 안 된다는 것을 의미한다. 만약 수정을 시도하는 경우, Vue는 콘솔에 경고가 나온다.

→ 그래도 수정을 해야한다면???? ?

prop은 초기값만 전달하고, 자식 컴포넌트는 그 초기값을 로컬 데이터 속성으로 활용하고 싶은 경우

```
props: ['initialCounter'],
data: function () {
  return {
    counter: this.initialCounter
  }
}
// counter는 변경해서 사용가능.
```

전달된 prop의 형태를 바꾸어야 하는 경우

해당 경우에는 computed 속성을 사용하는 것이 가장 좋다고한다.

```
props: ['size'],
computed: {
  normalizedSize: function () {
    return this.size.trim().toLowerCase()
  }
}
```

여기서 주의사항

- 자바스크립트 오브젝트나 배열을 prop으로 전달하는 경우, 객체를 복사하는 것이 아니라 참조하게 된다.
- 즉, 전달받은 오브젝트나 배열을 수정하게 되는 경우, 자식 요소가 부모 요소의 상태를 변경하게 된다..