

컴포넌트간의 통신

props

모든 Vue 인스턴스는 격리된 Scope를 가집니다. 즉, 하위 컴포넌트(자식)가 상위 컴포넌트(부모)의 data를 직접 참조할 수 없습니다. 또한 형제끼리의 data 직접 참조도 불가능합니다. 상위 컴포넌트의 data를 참조하고 싶다면 `props` 옵션을 사용해야 합니다.

`props` 는 상위 요소의 data를 참조하는 옵션입니다.

쉽게 비유하자면, 부모가 가지고 있는 사탕의 개수를 세고있는 자식의 모습을 상상하시면 됩니다. 사탕이 하나 줄어들면, 그걸 보고 있는 자식도 알게되겠죠.

`props` 를 사용하는 방법은 쉽습니다. 부모요소에 data를 정의하고, 자식 요소에서 `props` 를 정의하면 됩니다. 그리고 해당 `v-bind` 를 통해 해당 데이터를 바인딩해주면 됩니다. 전역 컴포넌트로 예를 들자면 아래와 같습니다.

```
Vue.component('print-tag', {
  props: ['candy'],
  template: `
    <p>부모의 사탕개수 : {{candy}}</p>
  `
})

new Vue({
  el: '#app',
  data: {
    candy: 10
  }
});
```

```
<div id="app">
  <print-tag :candy="candy"></print-tag>
</div>
<script src="https://unpkg.com/vue"></script>
```

prop

데이터는 `props` 옵션을 사용하여 하위 컴포넌트로 전달 될 수 있습니다.

prop는 상위 컴포넌트의 정보를 전달하기위한 사용자 지정 특성입니다. 하위 컴포넌트는 `props` 옵션을 사용하여 수신 할 것으로 기대되는 props를 명시적으로 선언해야 합니다

- 부모에있는 변수값 이용해서, 컴포넌트로 전달 한다

```
<apex-page v-bind:search-data="searchData">
```

```
<apex-page :search-data="searchData">
```

- 리터럴 prop (문자열로 전달)

```
<apex-page url="/test/url">
```

- 숫자 자체를 전달
- 실제 JavaScript 숫자를 전달하려면 값이 JavaScript 표현식으로 평가되도록 `v-bind`를 사용해야 합니다.

```
<apex-page v-bind:max-size="2">
```

\$emit

이제 `$emit`에 대해 알아보시다.

Vue는 자식요소에서 발생한 이벤트를 감지하지 못합니다.

처음에 말했듯이 각각의 인스턴스가 독립된 Scope를 가지기 때문입니다.

이제 이벤트를 발생시킬 컴포넌트를 하나 더 생성해봅시다. 버튼을 하나 누르면 사탕의 개수가 줄어들고, 다른 버튼을 누르면 사탕 개수가 늘어나는 이벤트가 발생합니다. 우선 이벤트가 발생할 컴포넌트에 해당 이벤트를 정의해줍니다. 이때 해당 이벤트를 직접 실행하는 대신 `$emit` 메서드를 사용하여, `event명`과 `data`를 부모에게 넘깁니다. 일종의 `key: value` 데이터라고 생각하면 될 것 같습니다.

`$emit`을 통해 부모에게 이벤트 발생 사실을 알리게 됩니다.

1.컴포넌트 에서 정의

- 매핑 해준다

```
<project-search @update="projectEvent($event)"></project-search>
```

2.\$emit을 통해 부모에게 이벤트 발생 사실을 알린다

- 이벤트 명 : update
- 부모에게 넘길 data : this.results

```
this.$emit('update', this.results);
```

3.부모

```
new Vue({
  el: '#app',
  data: {
    projectResults: {}
  }
  methods: {
    projectEvent($event) {
      this.projectResults= $event
    }
  }
})
```