7/13 ~ 7/17 2주차 - 10

컴포넌트와 Computed,Wathch

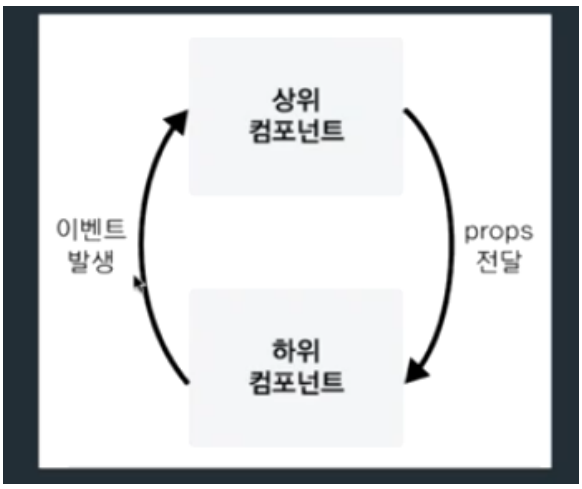
컴포넌트란?

화면의 특정 영역을 재사용 가능한 형태로 구성한 코드

지역, 전역 컴포넌트

```
// 전역 컴포넌트
Vue.component('컴포넌트 이름', 컴포넌트 내용);

//지역 컴포넌트
new Vue({
  components: {
    '컴포넌트 이름' : 컴포넌트 내용
  }
});
```

컴포넌트 통신 방법




```
// 최소한 타입이라도 적기
props: {
  justString : String
```

```
}

// 더 디테일 하면 좋다.
props : {
  justString: {
     type: String,
     required: true,
  }
}
```

Props

```
<div id="app">
  <child-component just-string="hi"></child-component>
</div>

components: {
  'child-component': {
    props: ['justString'],
    template: '<p>{{ justString }}</p>'
  }
}
```

Event Emit

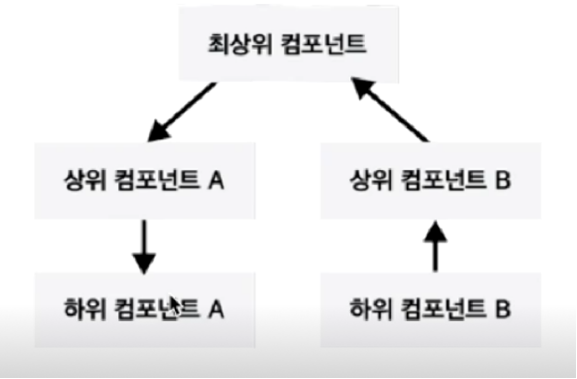
```
<div id="app">
  <child-component v-on:add="addCount"></child-component>
</div>

// ChildComponent - 하위 컴포넌트
this.$emit('add');

// RooComponent - 상위 컴포넌트
methods: {
  addCount: function() { ... }
}
```

Event Bus

컴포넌트 통신의 기본 규칙을 따르지 않고 특정 컴포넌트간에 통신하는 방법

가급적 사용하지 않는 것이 좋다

좋은 예



Event Bus 코드

```
var bus = new Vue();
// 이벤트를 보낼 컴포넌트의 메서드
bus.$emit('show');
// 이벤트를 받는 컴포넌트의 라이프 사이클 훅
bus.$on('show', callbackFunction);
```

Event Bus 사용시 주의할점

쓰고나면 반드시 $off()로 해제 안그러면 적체가 생김

컴포넌트 통신방법 Vs Event Bus

Event Bus가 없어도 서비스는 구현할 수 있습니다.

컴포넌트 레벨이 2단계 이하면 props, event emit을 쓰는 것이 좋다.

토스트 팝업, 프로그레스 바와 같이 on, off성격의 UI컴포넌트에 활용

→ 프로젝트 여건과 리소스에 맞춰 최대한 클린 코드를 지향.

Smart Watch

Watch

공식사이트 기본 문법

데이터가 변할때마다 특정 메서드를 실행한다.

페이지번호가 바껴을 때 데이터를 불러오는 동작 가능.

```
watch: {
  // whenever question changes, this function will run
  question: function (newQuestion, oldQuestion) {
    this.answer = 'Waiting for you to stop typing...'
    this.debouncedGetAnswer()
  }
},
```

실제 기본 문법

```
data: {
  num : 0
}
watch: {
  num: function(newValue, oldValue) {
    // ...
  }
}
```

```
data: {
  num : 0
}
watch: {
  num: {
```

```
    handler: function(newValue, oldValue) {},
    immediate: true, // created 라이프 사이클 훅 대신
    deep: true; // 객체의 속성 레벨이 깊어도 반응
                // 객체가 네스티트 되었을 때도 감지해줌.
  }
}
```

Intuitive Computed

data 속성의 변화에 따라 함께 변하는 속성

```
<!— html —>
<p>{{ doubleNum }}</p>

// js
data: {
  num: 10
}
computed: {
  doubleNum: function() {
    return this.num * 2;
  }
}
```

템플리 코드를 깔끔하게 해주는 핵심 속성

```
<!-- X -->
<div>
  {{ this.alertTitle === 'Delete' ? ' Delete ${this.itemForDelete.name}' :
this.alertTitle }}
</div>

<!-- O -->
<div>
  {{ deleteAlertTitle }}
</div>
```

직관적인 템플릿 코드 작성 가능

```
<li v-bind:class ="{'disabled' : isLastPage}"></li>
```

```
computed: {
  isLastPage() {
    const lastPageCondition =
      this.paginationInfo.current_page >= this.paginationInfo.last_page;
    const nothinFetched = Object.key(this.paginationInfo).length === 0;
    return lastPageCondition || nothingFetched;
  },
}
```

→ 코드해석 : 라스트페이지이면 disable클래스를 추가해라.

Watch Vs Computed

watch 를 사용하기 전에 computed로 해결할 수 있는지 확인

compute로 해결되지 않으면 methods로 해결할 수 있는지 확인

watch는 데이터 호출 로직과 연관된 동작에만 사용하면서 최소한으로 사용

클린 코드를 위한 속성 우선 순위 Computed > Methods > watch

watch를 RowNum 과 연관 시켜서 페이징 처리를 해보자.

watch의 사용은 줄인다. reactivity dependency가 많이 걸려있으면

화면 유아이가 많이 바뀔수록 추적하기가 어렵다

대부분의 경우 computed, methods로 해결이 쉽다.