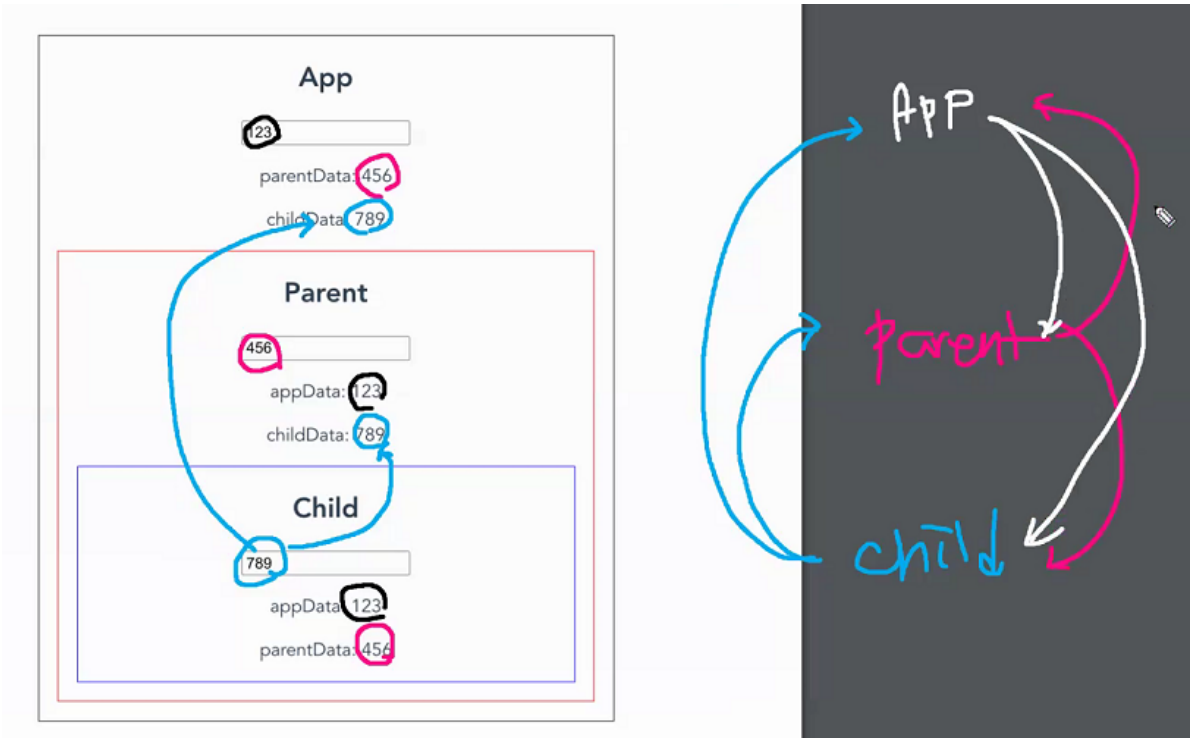
이렇게 구성하기



app.vue, helloworld.vue 지우기

app이 최상위 컴포넌트라서 main과 같은 자리에 둘 것이다.

src로 들어옴

touch App.vue

꼭 이 자리에 있을 필요는 없긴함

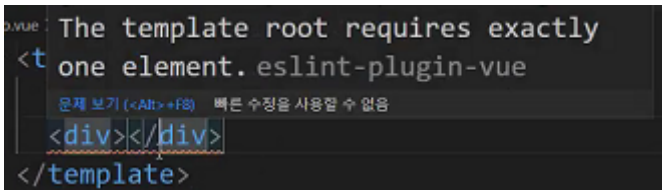
```
eduneo@neo-desktop MINGW64 ~/5ss2_neo_origi
-vue-cli (master)
$ cd src/components/

eduneo@neo-desktop MINGW64 ~/5ss2_neo_origi
-vue-cli/src/components (master)
$ touch Parent.vue Child.vue

eduneo@neo-desktop MINGW64 ~/5ss2_neo_origi
-vue-cli/src/components (master)
$
```

app.vue에서 해야할 일은

프레임워크이므로 틀에 박힌 일

꼭해야할 일이 3가지 있음

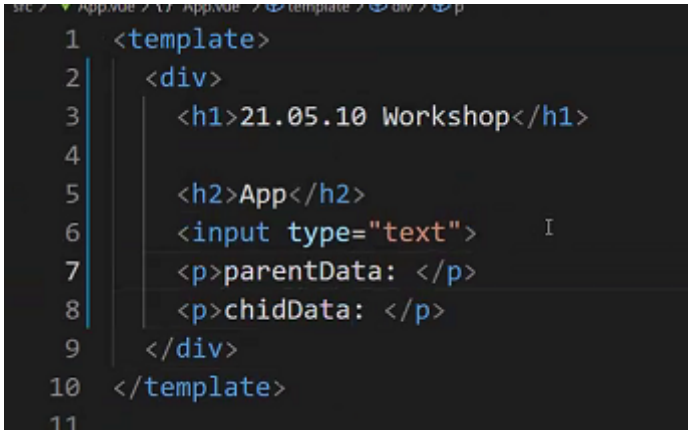
뒤의 { } 는 block이 아니라 object



템플릿에는 딱 하나만 들어갈 수 있음

div가 있으면 그 안에다가 넣을 수 있는 것

여기까지의 작업은 생각을 안하고 하는 작업



parent.vue와 child.vue에 같은 작업반복

vue치고 엔터누르면 자동으로 틀 생성

여기까지가 component를 기획하는 단계(각각을 컴포넌트화)

이걸 하고나야 데이터 모델링할수 있음

1.불러서

```
<script>
import Parent from './components/Parent.vue'

const obj = {
  name: 'App',
}

export default obj
</script>
```

parent는 그냥 변수값같은것(as)

p라고 써도 됨

2.등록

```
<script>
// 1. 불러서
import Parent from './components/Parent.vue'

const obj = {
  name: 'App',
  // 2. 등록하고
  components: {
    'Parent': Parent,
  }
}
```

key는 템플릿에서 사용할 이름

value는 위에서 가져온 것

근데 웬만하면 같은거로 쓰면 좋으니까

```
const obj = {
  name: 'App',
  // 2. 등록하고
  components: {
    Parent,
  }
}
```

3.사용

```vue
<template>
  <div>
    <h1>21.05.10 Workshop</h1>

    <h2>App</h2>
    <input type="text">
    <p>parentData: </p>
    <p>chidData: </p>
    <!-- 3. 사용한다. -->
    <Parent />
  </div>
</template>
```

동일한 작업 해주기

App.vue M | Parent.vue U | Child.vue U

src > components > Parent.vue > {} "Parent.vue" > template > div > Child

```vue
<template>
  <div>
    <h2>Parent</h2>
    <input type="text">
    <p>appData: </p>
    <p>chidData: </p>
    <Child />
  </div>
</template>

<script>
import Child from './Child.vue'

export default {
  name: 'Parent',
  components: {
   Child
  }

}
</script>

<style>

</style>
```

App은 data 필요

data는 함수

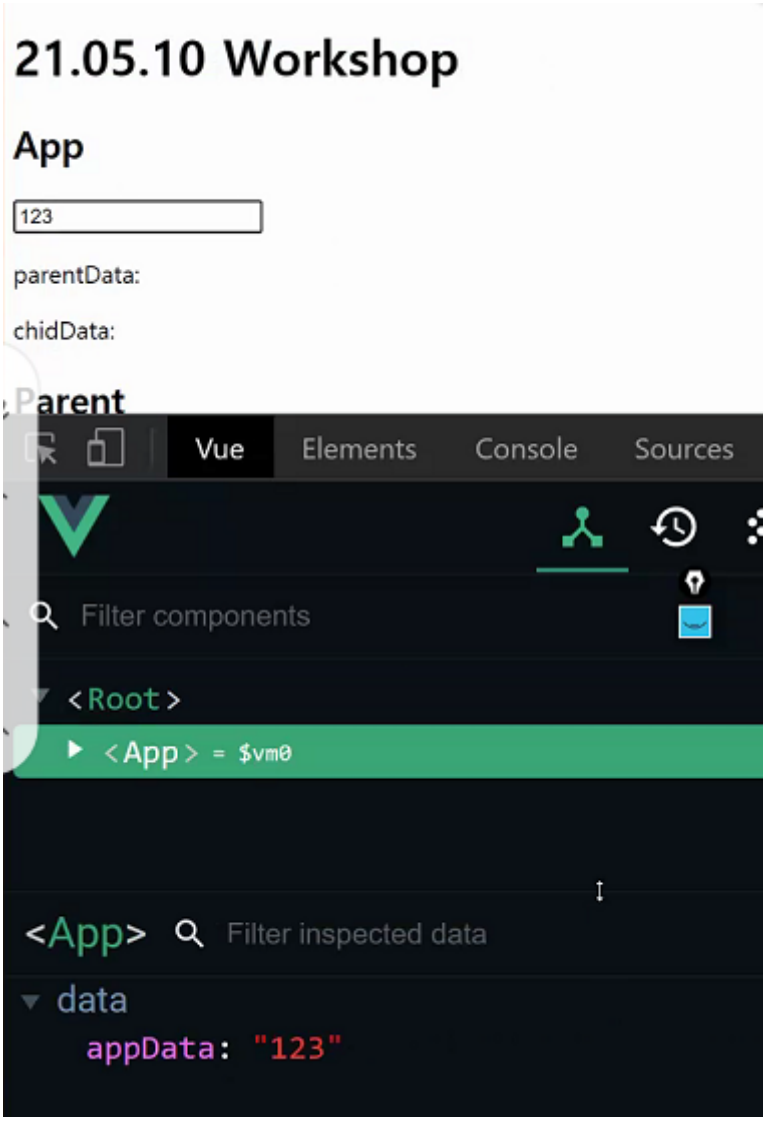
return은 appData

```
const obj = {
  name: 'App',
  // 2. 등록하고
  components: {
    Parent,
  },
  data() {
    return {
      appData: '',
    }
  },
}
```

input과 appData를 양방향 바인딩하기 위해서는 v-model

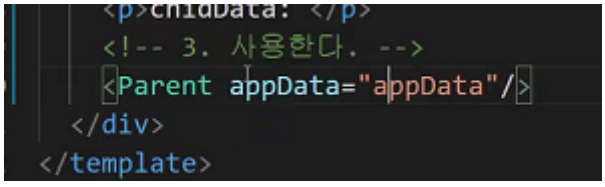
```
<template>
  <div>
    <h1>21.05.10 Workshop</h1>

    <h2>App</h2>
    <input v-model="appData" type="text">
    <p>parentData: </p>
    <p>chidData: </p>
    <!-- 3. 사용한다. -->
    <Parent />
  </div>
</template>
```

check



데이터를 내려줘야함

내려주기 위해서는 props가 필요



parent는 app데이터를 내려받아야하니까

```vue
<script>
import Child from './Child.vue'

export default {
  name: 'Parent',
  components: {
   Child
  },
  props: {
    appData: String,
  },

}
</script>
```

내가 부모한테 상속받을게 있는데 그게 string이야

```vue
<h1>21.05.10 Workshop</h1>

<h2>App</h2>
<input v-model="appData" type="text">
<p>parentData: </p>
<p>chidData: </p>
<!-- 3. 사용한다. -->
<Parent :appData="appData"/>
</div>
</template>
```

:appData

속성값을 보낸다는 뜻

Child도 똑같이 할거임

reactive하게 반응하는지 확인

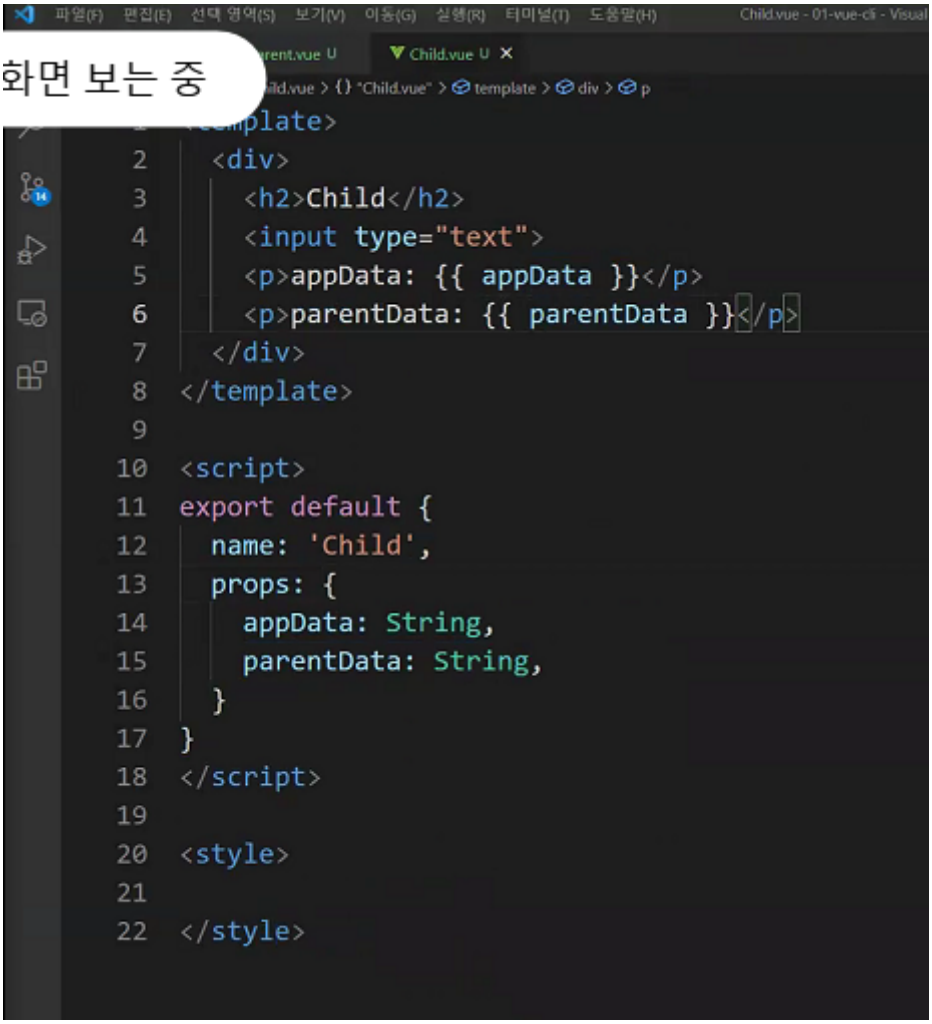
f12 콘솔 vue로

나머지 작업 해주자


App.vue M
Parent.vue U
Child.vue U
src > components > Parent.vue > {} "Parent.vue" > template > div > Child
<template>

<h2>Parent</h2>
<input v-model="parentData" type="text">
appData: {{ appData }}
chidData:
<Child :appData="appData" :parentData="parentData" />

</template>
<script>
import Child from './Child.vue'
export default {
name: 'Parent',
components: {
Child
},
props: {
appData: String,
},
data () {
return {
parentData: '',
}
}
}
</script>
<style>



여기까지가 데이터를 내리는 과정

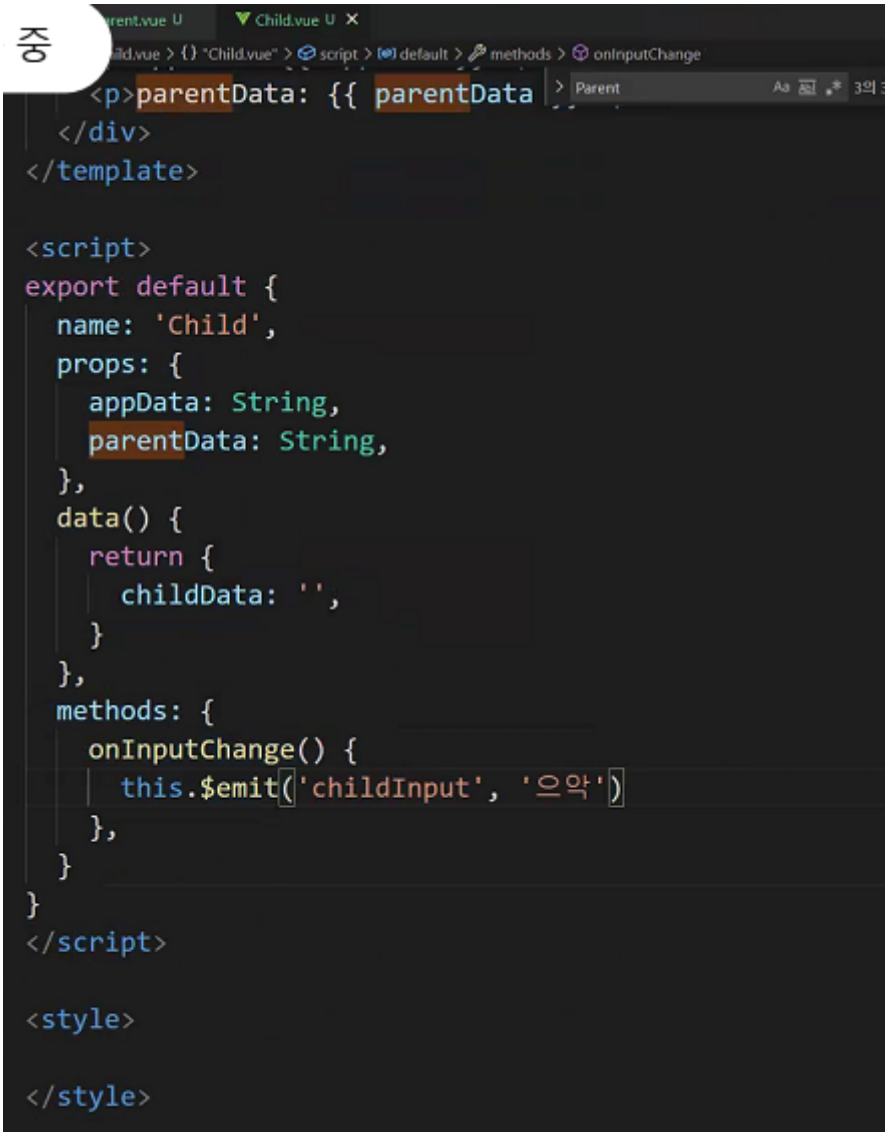
근데 올리는건 약간 어려움

이벤트(시그널)이 있어야함

child에서는 parent한테 직접적으로 무언갈 할수 없으니

살려달라고 소리를 지르자(시그널을 던지자 = 이벤트를 발생시키자)

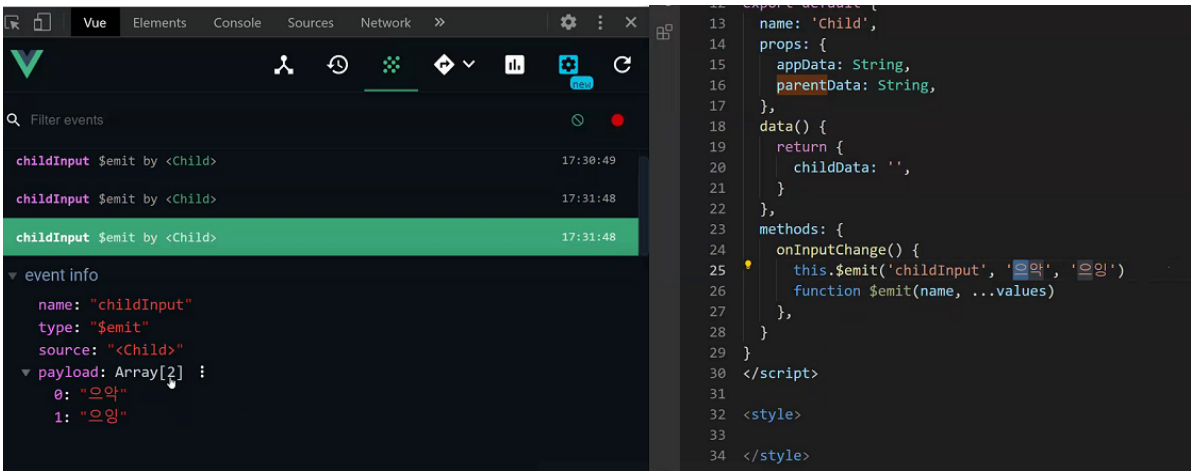
@input으로 엮어주자(change는 엔터칠때 input은 칠때마다)



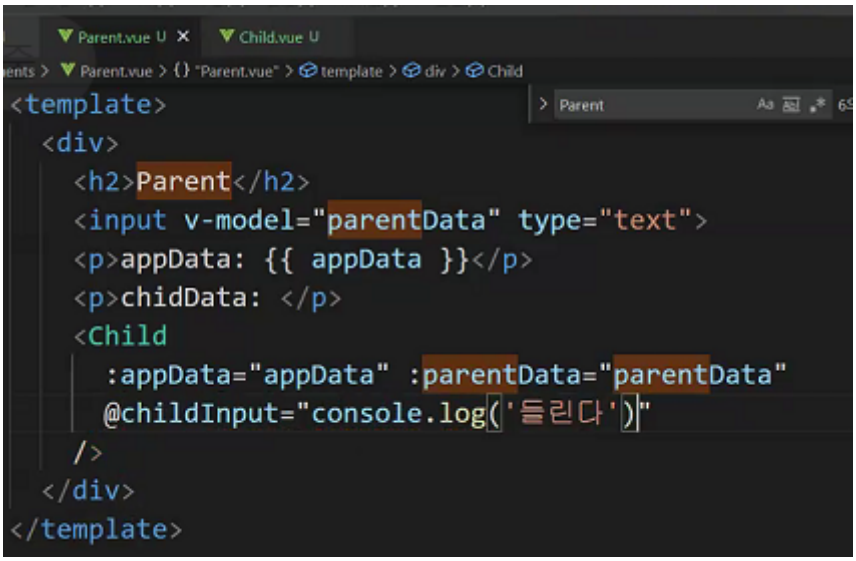
this는 뷰 인스턴스> child 컴포넌트 전체

emit: 방출하다

```
this.$emit('childInput', '으악')

차일드가.방출합니다(key(이름), value)
```



Parent도 들을 준비를 해야지



함수의 인자는 event로 하려고 했는데

그건 event 전체의 결과가 오는거니까

parent도 app에 소리쳐야하므로

data로 승격시켜야함

App.vue M Parent.vue U Child.vue U

src > components > Parent.vue > {} "Parent.vue" > script > default > methods > onChildInput

```
    Child
  },
  props: {
    appData: String,
  },
  data () {
    return {
      parentData: '',
      childData: '',
    }
  },
  methods: {
    onChildInput(childData) {
      this.childData = childData

    },
  }

}
</script>
```

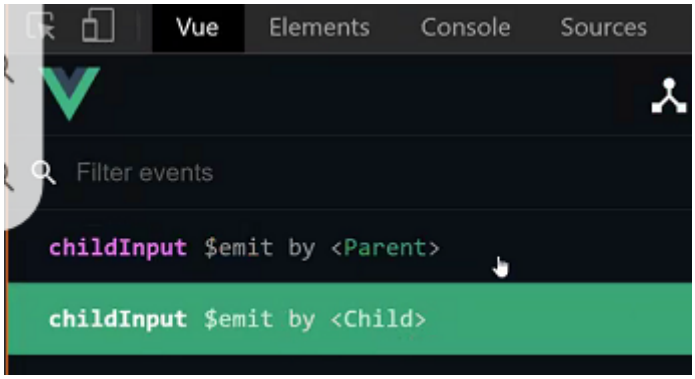
방출

보는 중

Parent.vue U Child.vue U

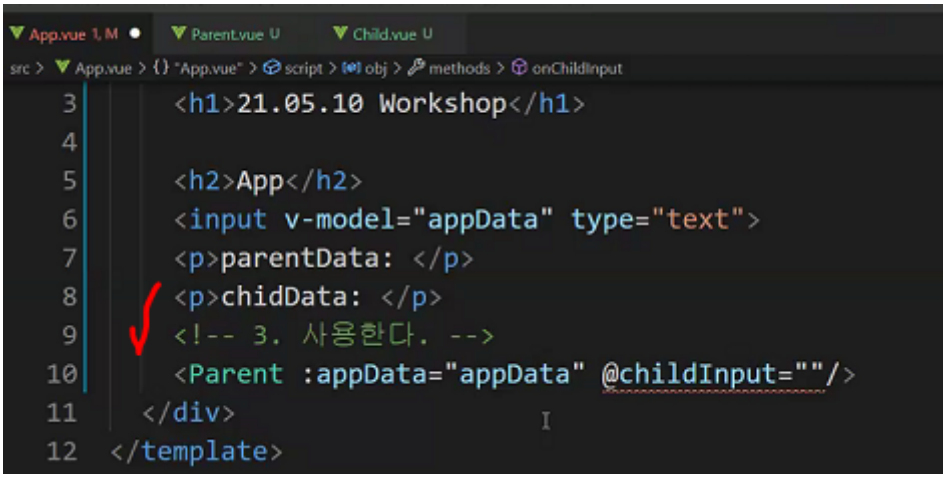
Parent.vue > {} "Parent.vue" > script > default > methods > onChildInput

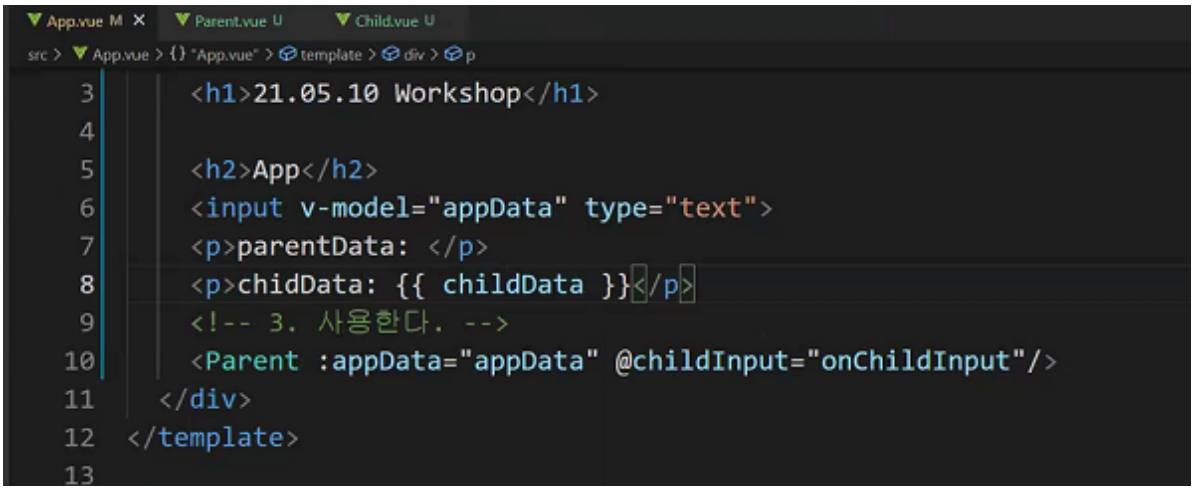
```
<script>
import Child from './Child.vue'

export default {
  name: 'Parent',
  components: {
   Child
  },
  props: {
    appData: String,
  },
  data () {
    return {
      parentData: '',
      childData: '',
    }
  },
  methods: {
    onChildInput(childData) {
      this.childData = childData
      this.$emit('childInput', childData)
    },
  }

}
```



이름 겹치긴 하는데 상관없음

app에 child를 듣는 작업해주자





App.vue M | Parent.vue U | Child.vue U

src > App.vue > {} "App.vue" > template > div > Parent

```vue
    <h1>21.05.10 Workshop</h1>

    <h2>App</h2>
    <input v-model="appData" type="text">
    <p>parentData: </p>
    <p>chidData: </p>
    <!-- 3. 사용한다. -->
    <Parent :appData="appData" @childInput="onChildInput"/>
  </div>
</template>

<script>
// 1. 불러서
import Parent from './components/Parent.vue'

const obj = {
  name: 'App',
  // 2. 등록하고
  components: {
    Parent,
  },
  data() {
    return {
      appData: '',
    }
  },
  methods: {
    onChildInput(childData) {
      console.log(childData)
    }
  }
```

App.vue | Parent.vue M | Child.vue

src > components > Parent.vue > {} "Parent.vue" > template > div > input

```vue
<template>
  <div>
    <h2>Parent</h2>
    <input v-model="parentData" @input="onParentInput">
    <p>appData: {{ appData }}</p>
    <p>chidData: {{ childData }}</p>
    <Child
      :appData="appData" :parentData="parentData"
      @child-input="onChildInput"
    />
  </div>
```

```vue
export default {
  name: 'Parent',
  components: {
   Child
  },
  props: {
    appData: String,
  },
  data () {
    return {
      parentData: '',
      childData: '',
    }
  },
  methods: {
    onChildInput(childData) {  // 아래에서 발생한 event => 데이터가 함께 오기에 인자
      this.childData = childData
      this.$emit('child-input', childData)
    },

    onParentInput() {  // 여기서 발생한 event => 위로 올리기만 하면 되기에 인자 x
      this.$emit('parent-input', this.parentData)
    }
  }

}
</script>
```

```vue
<template>
  <div>
    <h1>21.05.10 Workshop</h1>

    <h2>App</h2>
    <input v-model="appData" type="text">
    <p>parentData: {{ parentData }}</p>
    <p>chidData: {{ childData }}</p>
    <!-- 3. 사용한다. -->
    <Parent :appData="appData" @child-input="onChildInput" @parent-input="onParentInput"/>
  </div>
</template>
```

아래 메소드 설정도 하기