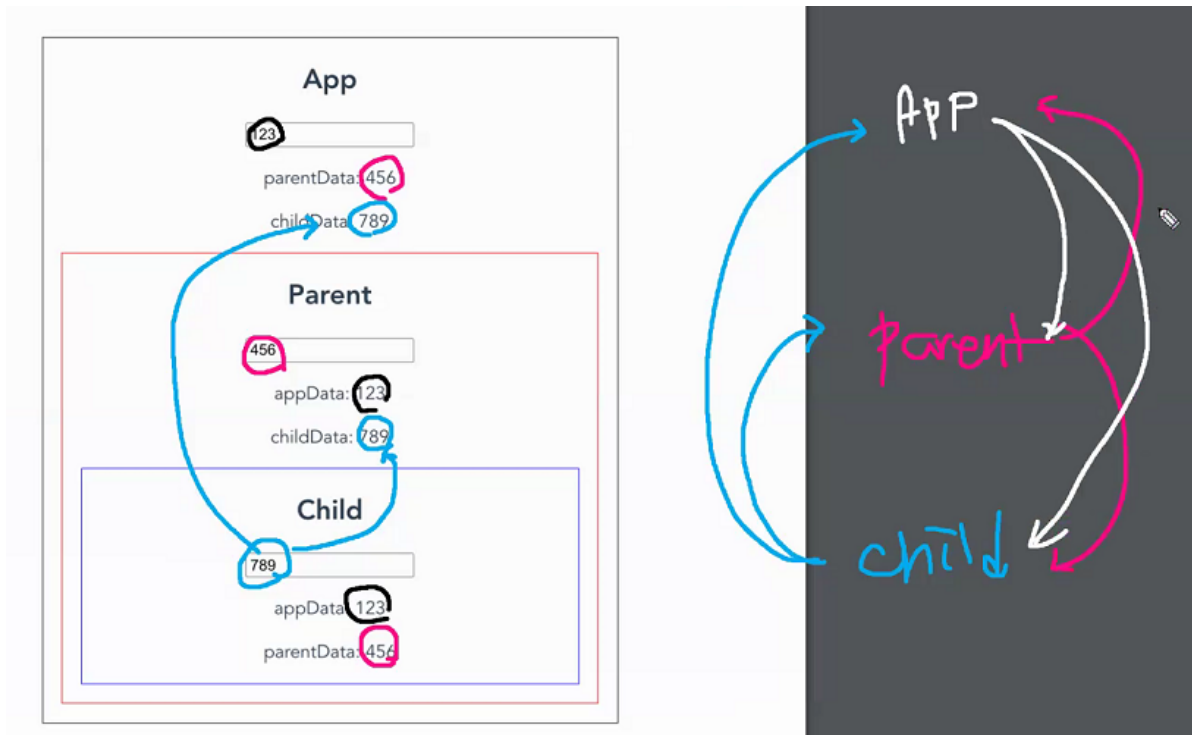


이렇게 구성하기



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_origin
-vue-cli (master)
$ cd src/components/

eduneo@neo-desktop MINGW64 ~/5ss2_neo_origin
-vue-cli/src/components (master)
$ touch Parent.vue Child.vue

eduneo@neo-desktop MINGW64 ~/5ss2_neo_origin
-vue-cli/src/components (master)
$
```

app.vue에서 해야할 일은

프레임워크이므로 틀에 박힌 일

꼭해야할 일이 3가지 있음

```
App.vue M
src > App.vue > style
1 <template>
2 |
3 </template>
4
5 <script>
6
7 </script>
8
9 <style scoped>
10
11 </style>
12
```

```
<script>
export default {}

```

뒤의 {} 는 block이 아니라 object

```
App.vue The template root requires exactly
<t one element. eslint-plugin-vue
문제 보기 (<Alt>+f8) 빠른 수정을 사용할 수 없음
<div></div>
</template>
```

템플릿에는 딱 하나만 들어갈 수 있음

div가 있으면 그 안에다가 넣을 수 있는 것

여기까지의 작업은 생각을 안하고 하는 작업

```
src > App.vue > App.vue > template > div > p
1 <template>
2   <div>
3     <h1>21.05.10 Workshop</h1>
4
5     <h2>App</h2>
6     <input type="text">
7     <p>parentData: </p>
8     <p>chidData: </p>
9   </div>
10 </template>
11
```

parent.vue와 child.vue에 같은 작업반복

vue치고 엔터누르면 자동으로 틀 생성

```
▼ App.vue M    ▼ Parent.vue U X
src > components > ▼ Parent.vue > {} "Parent.vue" > template > div > p
1  <template>
2    <div>
3      <h2>Parent</h2>
4      <input type="text">
5      <p>appData: </p>
6      <p>chidData: </p>
7    </div>
8  </template>
9
10 <script>
11 export default {
12   name: 'Parent',
13 }
14 </script>
15
16 <style>
17
18
19 </style>
```

```
▼ App.vue M    ▼ Parent.vue U    ▼ Child.vue U X
src > components > ▼ Child.vue > {} "Child.vue" > script > default > name
1  <template>
2    <div>
3      <h2>Child</h2>
4      <input type="text">
5      <p>appData: </p>
6      <p>parentData: </p>
7    </div>
8  </template>
9
10 <script>
11 export default {
12   name: 'Child',
13 }
14 </script>
15
16 <style>
17
18
19 </style>
```

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

```
4 <script>
5 // 1. 불러서
6 import Parent from './components/Parent.vue'
7
8 const obj = {
9   name: 'App',
10  // 2. 등록하고
11  components: {
12    'Parent': Parent,
13  }
14 }
```

key는 템플릿에서 사용할 이름

value는 위에서 가져온 것

근데 웬만하면 같은거로 쓰면 좋으니까

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

3.사용

```

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

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

```

동일한 작업 해주기

```

App.vue M  Parent.vue U x  Child.vue U
src > components > Parent.vue > {} "Parent.vue" > template > div > Child
1  <template>
2    <div>
3      <h2>Parent</h2>
4      <input type="text">
5      <p>appData: </p>
6      <p>chidData: </p>
7      <Child />
8    </div>
9  </template>
10
11 <script>
12 import Child from './Child.vue'
13
14 export default {
15   name: 'Parent',
16   components: {
17     Child
18   }
19 }
20
21 </script>
22
23 <style>
24
25 </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

21.05.10 Workshop

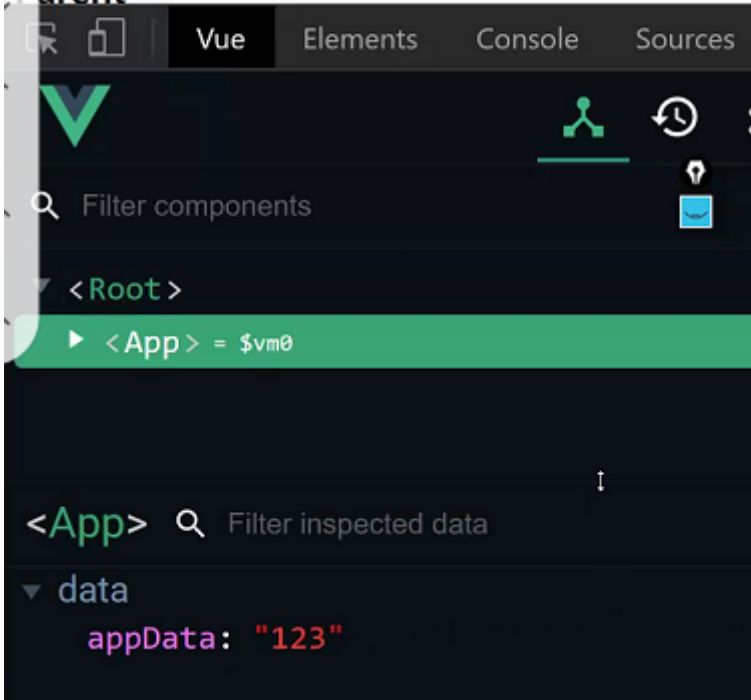
App

123

parentData:

chidData:

Parent



데이터를 내려줘야함

내려주기 위해서는 props가 필요

```
<p>chidData! </p>
<!-- 3. 사용한다. -->
<Parent appData="appData"/>
</div>
</template>
```

parent는 app데이터를 내려받아야하니까

```
11 <script>
12 import Child from './Child.vue'
13
14 export default {
15   name: 'Parent',
16   components: {
17     Child
18   },
19   props: {
20     appData: String,
21   },
22 }
23 </script>
```

내가 부모한테 상속받을게 있는데 그게 string이야

```
App.vue M Parent.vue U Child.vue U
src > App.vue > {} "App.vue" > template > div > Parent
3 <h1>21.05.10 Workshop</h1>
4
5 <h2>App</h2>
6 <input v-model="appData" type="text">
7 <p>parentData: </p>
8 <p>chidData: </p>
9 <!-- 3. 사용한다. -->
10 <Parent :appData="appData"/>
11 </div>
12 </template>
13
```

:appData

속성값을 보낸다는 뜻

Child도 똑같이 할거임

보는 중

```
parent.vue U Child.vue U X
Child.vue > {} "Child.vue" > template > div
1 <template>
2   <div>
3     <h2>Child</h2>
4     <input type="text">
5     <p>appData: {{ appData }}</p>
6     <p>parentData: </p>
7   </div>
8 </template>
9
10 <script>
11 export default {
12   name: 'Child',
13   props: {
14     appData: String,
15   }
16 }
17 </script>
18
19 <style>
20
21 </style>
```

```
App.vue M Parent.vue U Child.vue U X
c > components > Child.vue > {} "Child.vue" > script > default > props
1 <template>
2   <div>
3     <h2>Child</h2>
4     <input type="text">
5     <p>appData: {{ appData }}</p>
6     <p>parentData: </p>
7   </div>
8 </template>
9
10 <script>
11 export default {
12   name: 'Child',
13   props: {
14     appData: String,
15   }
16 }
17 </script>
18
19 <style>
20
21 </style>
```

reactive하게 반응하는지 확인

f12 콘솔 vue로

나머지 작업 해주자

```
▼ App.vue M    ▼ Parent.vue U x    ▼ Child.vue U
src > components > ▼ Parent.vue > {} "Parent.vue" > template > div > Child
1  <template>
2    <div>
3      <h2>Parent</h2>
4      <input v-model="parentData" type="text">
5      <p>appData: {{ appData }}</p>
6      <p>chidData: </p>
7      <Child :appData="appData" :parentData="parentData" />
8    </div>
9  </template>
10
11 <script>
12 import Child from './Child.vue'
13
14 export default {
15   name: 'Parent',
16   components: {
17     Child
18   },
19   props: {
20     appData: String,
21   },
22   data () {
23     return {
24       parentData: '',
25     }
26   }
27 }
28 }
29 </script>
30
31 <style>
```

Child.vue - 01-vue-cli - Visual Studio Code

Child.vue U x

Child.vue > {} "Child.vue" > template > div > p

Child.vue

```
1 <template>
2   <div>
3     <h2>Child</h2>
4     <input type="text">
5     <p>appData: {{ appData }}</p>
6     <p>parentData: {{ parentData }}</p>
7   </div>
8 </template>
9
10 <script>
11 export default {
12   name: 'Child',
13   props: {
14     appData: String,
15     parentData: String,
16   }
17 }
18 </script>
19
20 <style>
21
22 </style>
```

여기까지가 데이터를 내리는 과정

근데 올리는건 약간 어려움

이벤트(시그널)이 있어야함

```
▼ App.vue M    ▼ Parent.vue U    ▼ Child.vue U X
src > components > ▼ Child.vue > {} "Child.vue" > template > div > input
1  <template>
2    <div>
3      <h2>Child</h2>
4      <input v-model="childData" type="text">
5      <p>appData: {{ appData }}</p>
6      <p>parentData: {{ parentData }}</p>
7    </div>
8  </template>
9
10 <script>
11   export default {
12     name: 'Child',
13     props: {
14       appData: String,
15       parentData: String,
16     },
17     data() {
18       return {
19         childData: '',
20       }
21     },
22   }
23 </script>
24
25 <style>
26
27 </style>
```

child에서는 parent한테 직접적으로 무언갈 할수 없으니

살려달라고 소리를 지르자(시그널을 던지자 = 이벤트를 발생시키자)

```
App.vue M Parent.vue U Child.vue U
src > components > Child.vue > {} "Child.vue" > template > div > input
2 <div>
3   <h2>Child</h2>
4   <input
5     v-model="childData"
6     @input="onInputChange" >
7   <p>appData: {{ appData }}</p>
8   <p>parentData: {{ parentData }}</p>
9 </div>
10 </template>
11
12 <script>
13 export default {
14   name: 'Child',
15   props: {
16     appData: String,
17     parentData: String,
18   },
19   data() {
20     return {
21       childData: '',
22     }
23   },
24   methods: {
25     onInputChange() {
26       console.log('으악')
27     },
28   }
29 }
30 </script>
31
```

@input으로 엮어주자(change는 엔터칠때 input은 칠때마다)

중

```
parent.vue U Child.vue U X
Child.vue > {} "Child.vue" > script > default > methods > onInputChange
parentData: {{ parentData
</div>
</template>

<script>
export default {
  name: 'Child',
  props: {
    appData: String,
    parentData: String,
  },
  data() {
    return {
      childData: '',
    }
  },
  methods: {
    onInputChange() {
      this.$emit('childInput', '으악')
    },
  }
}
</script>

<style>
</style>
```

this는 뷰 인스턴스> child 컴포넌트 전체

emit: 방출하다

```
this.$emit('childInput', '으악')
```

차일드가. 방출합니다(key(이름), value)

The screenshot displays the Vue DevTools interface. On the left, the 'Console' tab shows three log entries for the 'childInput' event emitted by the 'Child' component. The third entry is selected, showing the event info: name 'childInput', type '\$emit', source '<Child>', and a payload array containing '으악' and '으잉'. On the right, the 'Child.vue' component code is visible, showing the 'onInputChange' method that calls 'this.\$emit('childInput', '으악', '으잉')'.

```

7 |   <p>parentData: {{ parentData }}</p>
8 | </div>
9 | </template>
10 |
11 | <script>
12 | export default {
13 |   name: 'Child',
14 |   props: {
15 |     appData: String,
16 |     parentData: String,
17 |   },
18 |   data() {
19 |     return {
20 |       childData: '',
21 |     }
22 |   },
23 |   methods: {
24 |     onChange() {
25 |       this.$emit('childInput', this.childData)
26 |     },
27 |   },
28 | }
29 | }
30 | </script>
31 |
32 | <style>
33 |
34 | </style>

```

Parent도 들을 준비를 해야지

```

<template>
  <div>
    <h2>Parent</h2>
    <input v-model="parentData" type="text">
    <p>appData: {{ appData }}</p>
    <p>childData: </p>
    <Child
      :appData="appData" :parentData="parentData"
      @childInput="console.log('들린다')"
    />
  </div>
</template>

```

```
Parent.vue U x Child.vue U
components > ▼ Parent.vue > {} "Parent.vue" > script > default > methods > a
  <Child
    :appData="appData" :parentData="parentData"
    @childInput="a"
  />
</div>
</template>

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

export default {
  name: 'Parent',
  components: {
    Child
  },
  props: {
    appData: String,
  },
  data () {
    return {
      parentData: '',
    }
  },
  methods: {
    a(event) {
      console.log(event)
    },
  }
}
```

함수의 인자는 event로 하려고 했는데

그건 event 전체의 결과가 오는거니까


```
App.vue M Parent.vue U Child.vue U
src > components > Parent.vue > {} "Parent.vue" > script > default > methods > onChildInput
9 | @childInput="a"
10 | />
11 | </div>
12 | </template>
13 |
14 | <script>
15 | import Child from './Child.vue'
16 |
17 | export default {
18 |   name: 'Parent',
19 |   components: {
20 |     Child
21 |   },
22 |   props: {
23 |     appData: String,
24 |   },
25 |   data () {
26 |     return {
27 |       parentData: '',
28 |     }
29 |   },
30 |   methods: {
31 |     onChildInput(childData) {
32 |       console.log(childData)
33 |     },
34 |   }
35 | }
36 | </script>
37 |
38 |
39 | <style>
```

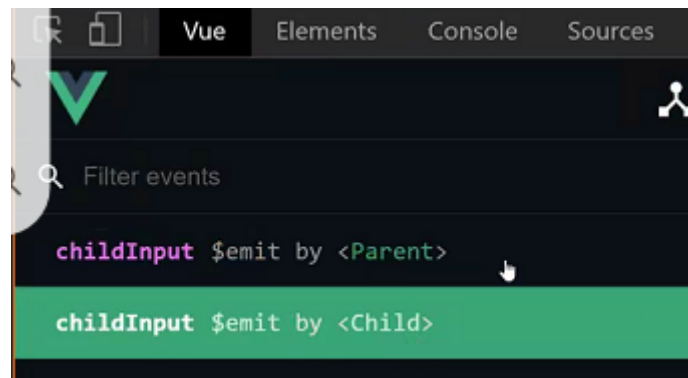
parent도 app에 소리쳐야하므로

data로 승격시켜야함

```
▼ App.vue M    ▼ Parent.vue U •    ▼ Child.vue U
src > components > ▼ Parent.vue > {} "Parent.vue" > script > default > methods > onChildInput
20   Child
21 },
22 props: {
23   appData: String,
24 },
25 data () {
26   return {
27     parentData: '',
28     childData: '',
29   }
30 },
31 methods: {
32   onChildInput(childData) {
33     this.childData = childData
34   }
35 },
36 }
37
38 }
39 </script>
```

방출

```
보는 중
Parent.vue U x    ▼ Child.vue U
Parent.vue > {} "Parent.vue" > script > default > methods > onChildInput
15 import Child from './Child.vue'
16
17 export default {
18   name: 'Parent',
19   components: {
20     Child
21   },
22   props: {
23     appData: String,
24   },
25   data () {
26     return {
27       parentData: '',
28       childData: '',
29     }
30   },
31   methods: {
32     onChildInput(childData) {
33       this.childData = childData
34       this.$emit('childInput', childData)
35     },
36   }
37 }
38 }
```



이름 겹치긴 하는데 상관없음

app에 child를 듣는 작업해주자

```

src > App.vue > {} "App.vue" > script > obj > methods > onChildInput
3 | <h1>21.05.10 Workshop</h1>
4 |
5 | <h2>App</h2>
6 | <input v-model="appData" type="text">
7 | <p>parentData: </p>
8 | <p>chidData: </p>
9 | <!-- 3. 사용한다. -->
10 | <Parent :appData="appData" @childInput=""/>
11 | </div>
12 </template>

```

```

src > App.vue > {} "App.vue" > template > div > p
3 | <h1>21.05.10 Workshop</h1>
4 |
5 | <h2>App</h2>
6 | <input v-model="appData" type="text">
7 | <p>parentData: </p>
8 | <p>chidData: {{ childData }}</p>
9 | <!-- 3. 사용한다. -->
10 | <Parent :appData="appData" @childInput="onChildInput"/>
11 | </div>
12 </template>
13

```

```
App.vue M x Parent.vue U Child.vue U
src > App.vue > {} "App.vue" > template > div > Parent
3 | <h1>21.05.10 Workshop</h1>
4 |
5 | <h2>App</h2>
6 | <input v-model="appData" type="text">
7 | <p>parentData: </p>
8 | <p>childData: </p>
9 | <!-- 3. 사용한다. -->
10 | <Parent :appData="appData" @childInput="onChildInput"/>
11 | </div>
12 | </template>
13 |
14 | <script>
15 | // 1. 불러서
16 | import Parent from './components/Parent.vue'
17 |
18 | const obj = {
19 |   name: 'App',
20 |   // 2. 등록하고
21 |   components: {
22 |     Parent,
23 |   },
24 |   data() {
25 |     return {
26 |       appData: '',
27 |     }
28 |   },
29 |   methods: {
30 |     onChildInput(childData) {
31 |       console.log(childData)
32 |     }
33 |   }
34 | }
```

```
App.vue Parent.vue M x Child.vue
src > components > Parent.vue > {} "Parent.vue" > template > div > input
1 | <template>
2 | <div>
3 | <h2>Parent</h2>
4 | <input v-model="parentData" @input="onParentInput">
5 | <p>appData: {{ appData }}</p>
6 | <p>childData: {{ childData }}</p>
7 | <Child
8 |   :appData="appData" :parentData="parentData"
9 |   @child-input="onChildInput"
10 | />
11 | </div>
12 | </template>
```

```

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.$emit('child-input', childData)
    },
    onParentInput() { // 여기서 발생한 event => 쿼로 올리지만 하면 되기에 인자 x
      this.$emit('parent-input', this.parentData)
    }
  }
}
</script>

```

```

1 <template>
2 <div>
3   <h1>21.05.10 Workshop</h1>
4
5   <h2>App</h2>
6   <input v-model="appData" type="text">
7   <p>parentData: {{ parentData }}</p>
8   <p>childData: {{ childData }}</p>
9   <!-- 3. 사용한다. -->
10  <Parent :appData="appData" @child-input="onChildInput" @parent-input="onParentInput"/>
11 </div>
12 </template>

```

아래 메소드 설정도 하기