컴포넌트 기초

- 개요
- 컴포넌트 조합
- 컴포넌트의 작성
- DOM 템플릿 구문 작성시 주의 사항
- 컴포넌트에서의 data 옵션
- props와 event
- 이벤트 버스 객체를 이용한 통신

개요

컴포넌트의 사용의 장점

- 뛰어난 재사용성
- 용이한 테스트
- 편리한 디버깅

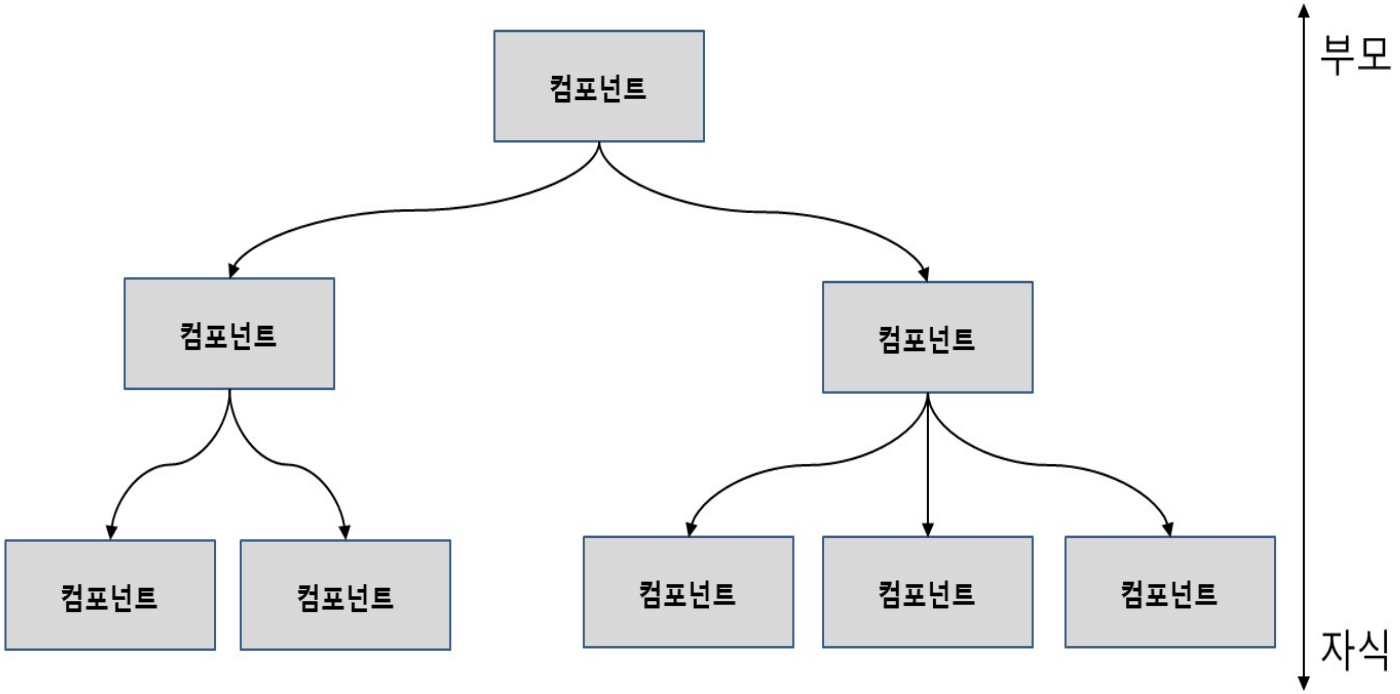
단일 파일 컴포넌트

- Single File Component
- .vue 파일
- webpack, ES6를 학습한 후에 9장에서 더 자세하게 다룸

컴포넌트 조합

부모-자식 관계로 트리 구조 형성

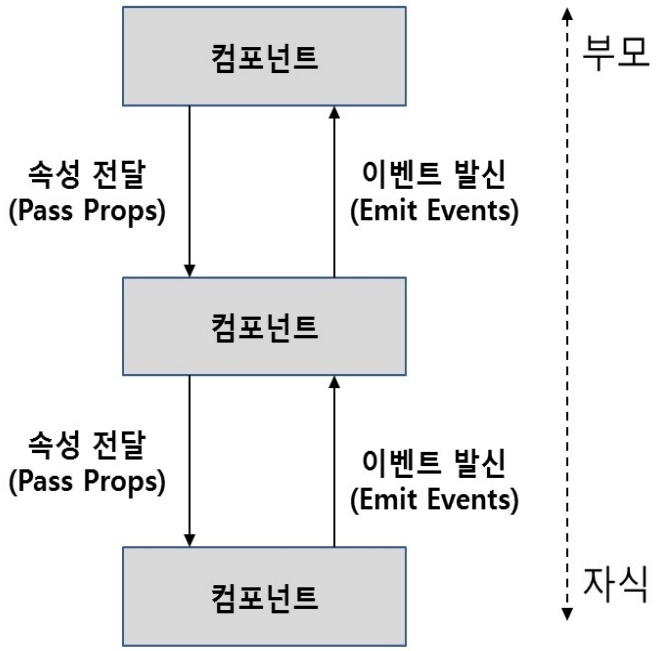
- 부모가 자식을 포함하는 형태



컴포넌트 조합

부모-자식 컴포넌트간의 정보 전달

- 부모 -> 자식
  - 속성(Props) 전달
- 자식 -> 부모
  - 이벤트(event) 발신



대부분의 옵션 사용 가능

- data
- methods
- watch
- computed
- data 옵션은 컴포넌트의 로컬 상태 관리 목적으로만 사용함

컴포넌트의 작성

컴포넌트 작성 메서드

- Vue.component(tagname, options)
  - tagname : 컴포넌트를 사용할 태그명입니다.
  - options : 컴포넌트에서 렌더링할 templet 등을 지정합니다.
- 태그명은 kebob casing 규칙 적용
  - 태그명은 대소문자를 구별하지 않기 때문에...
- 예제 06-02~03

```
<script type="text/javascript">
Vue.component('hello-component', {
    template : '<div>hello world</div>'
})
</script>
```

```
<body>
  <div id="app">
    <hello-component></hello-component>
    <hello-component></hello-component>
    <hello-component></hello-component>
  </div>
</body>
<script type="text/javascript">
Vue.config.devtools = true;
var v = new Vue({
   el : '#app'
})
</script>
```

컴포넌트의 작성

예제 06-02의 코드에서 template을 분리

- <template> 또는 <script type="text/x-template"> 을 이용함

```
<script type="text/javascript">
Vue.component('hello-component', {
   template : '<div>hello world!!!</div>'
})
</script>
```

```
<template id="helloTemplate">
   <div>hello world!!!</div>
</template>
<script type="text/javascript">
Vue.component('hello-component', {
   template : '#helloTemplate'
})
</script>
```

DOM 템플릿 구문 작성시 주의사항

- **HTML 요소들은 자식요소로 사용할 수 있는 것이 정해진 경우가 있음**
  - 브라우저가 우선적으로 구문분석을 수행함
  - 예) <select> 태그안에는 <option>이 포함되어야 함.
  - 예제 06-06
    - <select> 태그 내의 <option-component>는 파싱 오류 --> 컴포넌트가 사용되지 못함.
    - is 특성 사용하여 해결!!

```
<div id="app">
  <select>
    <option-component></option-component>
    <option-component></option-component>
  </select>
</div>
```

```
<div id="app">
  <select>
    <option is="option-component"></option>
    <option is="option-component"></option>
  </select>
</div>
```

  - is 특성을 사용하지 않아도 되는 경우(예제 06-08 참조)
    - 단일 파일 컴포넌트로 작성하거나 <script type="text/x-template">으로 작성하는 경우

DOM 템플릿 구문 작성시 주의사항

- **템플릿 안에서 루트 요소는 단하나여야 함.**
  - 템플릿 내부에 여러 요소를 표현해야 한다면 등을 이용해 감싸줌

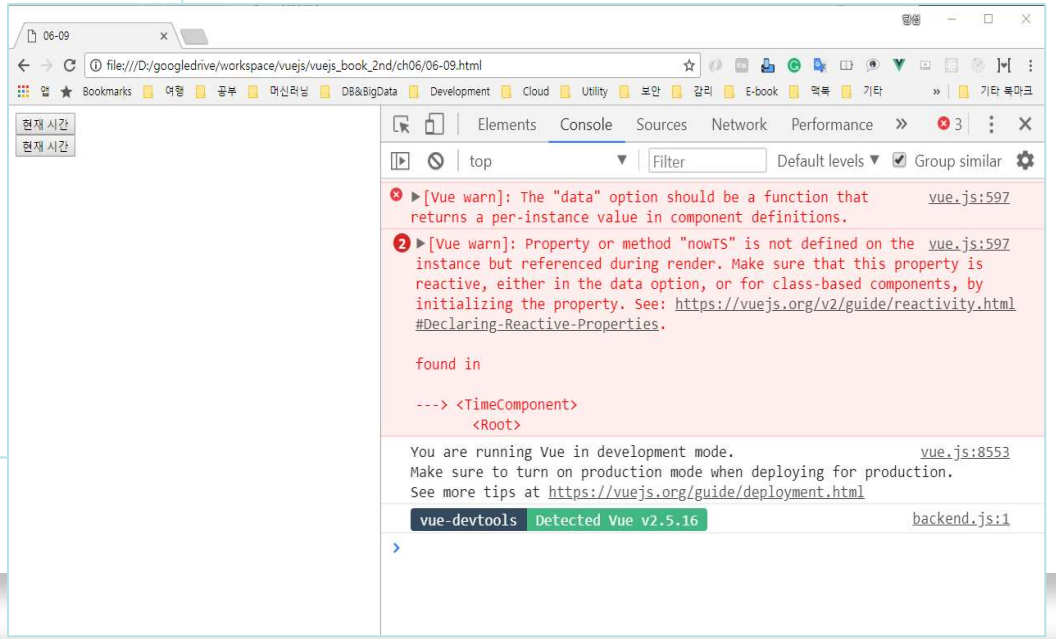
```
<template id="helloTemplate">
    <div>hello</div>
    <div>world</div>
</template>
```

```
<template id="helloTemplate">
  <div>
    <div>hello</div>
    <div>world</div>
  </div>
</template>
```

컴포넌트에서의 data 옵션

- **컴포넌트에서의 data 옵션은 반드시 함수를 통해서 객체를 리턴해주어야 함.**
  - 예제 06-09 : 일단 오류 발생 현상 확인

```
<template id='timeTemplate'>
    <div>
        <span>{{nowTS}}</span>
        <button v-on:click="timeClick">현재 시간</button>
    </div>
</template>
<script type="text/javascript">
  Vue.component('time-component', {
    template : '#timeTemplate',
    data : { nowTS : 0 },
    methods : {
        timeClick : function(e) {
            this.nowTS = (new Date()).getTime();
        }
    }
  })
</script>
```



컴포넌트에서의 data 옵션

왜 그럴까?

- data 옵션에 단순 객체를 지정하면 이 컴포넌트로 여러개의 인스턴스를 만들었을 때 동일한 객체를 참조하게 됨.
- 함수 호출후 리턴된 객체를 사용하면 함수 호출때마다 매번 새로운 객체를 생성하게 되므로 서로 다른 data 옵션을 가지게 됨.
  - 따라서 객체를 리턴하는 함수가 data 옵션에 주어져야 함.

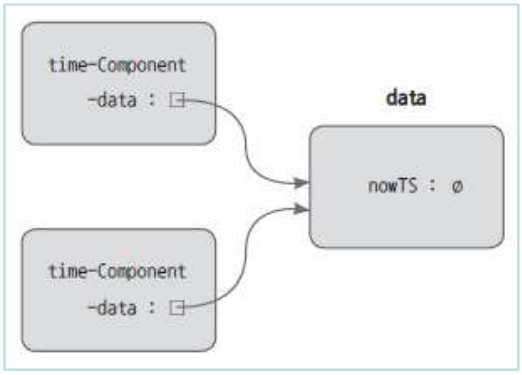
```
<script type="text/javascript">
Vue.component('time-component', {
    template : '#timeTemplate',
    data : function() {
        return { nowTS : 0 };
    },
    methods : {
        timeClick : function(e) {
            this.nowTS = (new Date()).getTime();
        }
    }
})
</script>
```

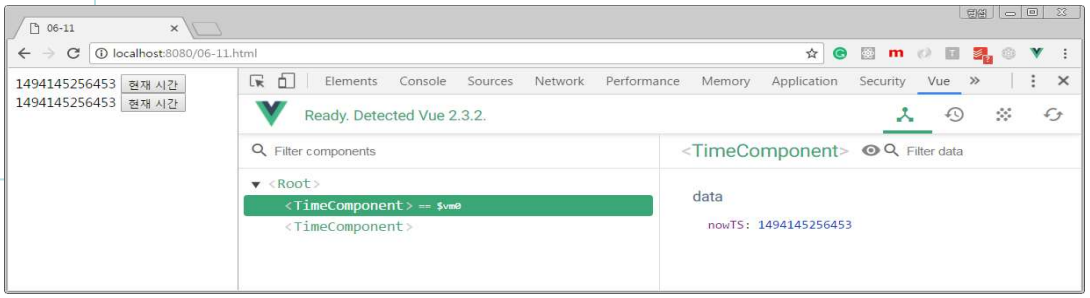
컴포넌트에서의 data 옵션

예제 06-11

- 의도적으로 함수 내부에서 리턴하는 객체를 동일하게 조작해봄
  - 컴포넌트의 버튼을 클릭하면 모든 Timestamp 값이 동일하게 변경됨.

```
<script type="text/javascript">
var data = { nowTS : 0 };
Vue.component('time-component', {
    template : '#timeTemplate',
    data : function() {
        return data;
    },
    methods : {
        timeClick : function(e) {
            this.nowTS = (new Date()).getTime();
        }
    }
})
</script>
```



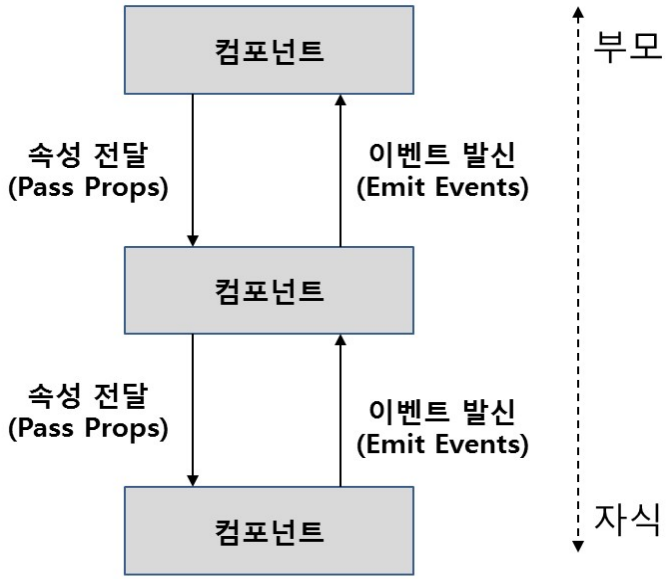


- data 옵션의 함수안에서 정의된 객체를 리턴해야 함!!!

props와 event

부모 - 자식 컴포넌트 간 통신

- Vue 컴포넌트들이 부모-자식 관계로 형성되었을 때 각 컴포넌트의 내부 데이터는 캡슐화되기 때문에 접근이 불가능함.
- 부모 -> 자식 : 속성(props)으로 전달(pass)
- 자식 -> 부모 : 이벤트(event) 발신(emit), v-on 디렉티브 사용



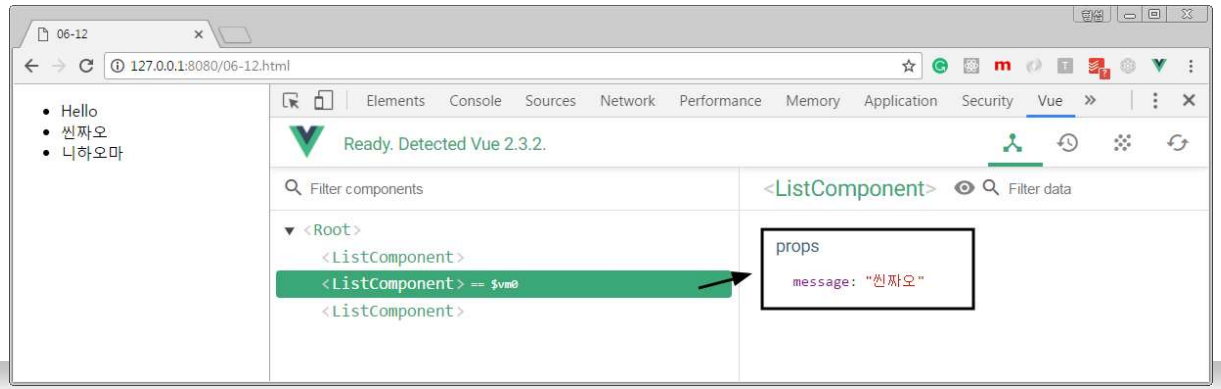
props와 event

props를 이용한 정보 전달

- Vue 컴포넌트를 정의할 때 props 옵션에 props 명을 배열로 나열함
- 예제 06-12

```
<template id="listTemplate">
    <li>{{message}}</li>
</template>
<script type="text/javascript">
Vue.component('list-component', {
    template : '#listTemplate',
    props : [ 'message' ]
})
</script>
```

```
<div id="app">
    <ul>
        <list-component message="Hello"></list-component>
        <list-component message="씬짜오"></list-component>
        <list-component message="니하오마"></list-component>
    </ul>
</div>
```



props와 event

props를 이용한 정보 전달(이어서)

- props를 표기할 때 주의 사항
  - 속성명이 camel casing을 사용했다면 속성명을 전달할 태그의 특성명은 kebob casing을 사용해야 함.
  - 만일 특성명에서도 camelCasing을 사용한다면 속성 값이 전달되지 않음.
- 예제 06-13

```
<template id="listTemplate">
    <li>{{message}}</li>
</template>
<script type="text/javascript">
Vue.component('list-component', {
    template : '#listTemplate',
    props : [ 'myMessage' ]
})
</script>
```

```
<div id="app">
    <ul>
        <list-component my-message="Hello"></list-component>
        <list-component my-message ="씬짜오"></list-component>
        <list-component my-message ="니하오마"></list-component>
    </ul>
</div>
```

props와 event

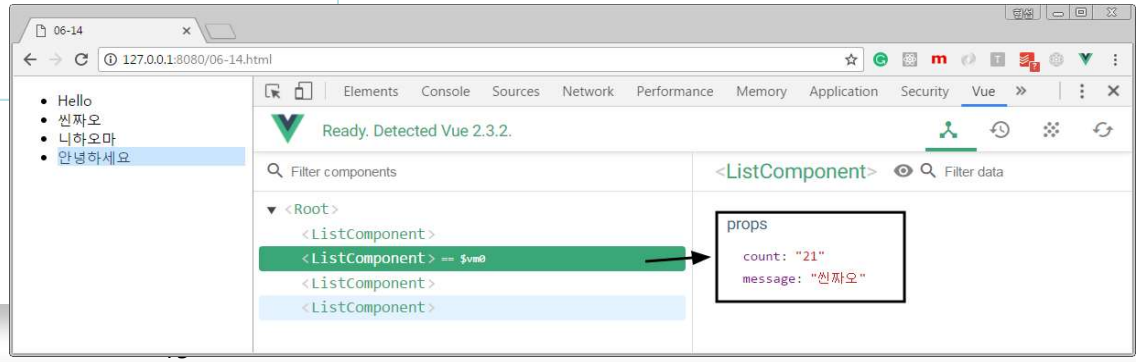
props의 엄격한 유효성 검증이 필요하다면?

- 속성명의 배열이 아닌 객체 형태를 사용할 수 있음.
  - 하지만 값을 전달할 때는 주의가 필요함.
- 예제 06-14~15

```
<template id="listTemplate">
    <li>{{message}}</li>
</template>
<script type="text/javascript">
Vue.component('list-component', {
    template : '#listTemplate',
    props : {
        message : { type:String, default:'안녕하세요' },
        count : { type:Number, required:true }
    }
})
</script>
```

```
<div id="app">
    <ul>
        <list-component message="Hello"  count="100"></list-component>
        <list-component message="씬짜오"  count="21"></list-component>
        <list-component message="니하오마"></list-component>
        <list-component count="1000"></list-component>
    </ul>
</div>
```

타입이상!! count가 문자열!!

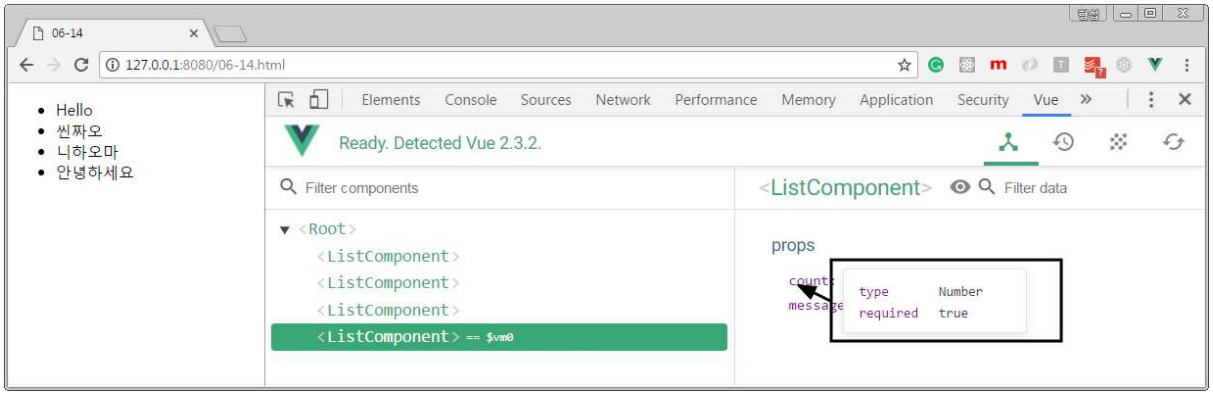


props와 event

props의 엄격한 유효성 검증이 필요하다면?(이어서)

- 정확한 타입으로 전달하려면 v-bind 디렉티브를 사용함

```
<div id="app">
    <ul>
        <list-component message="Hello" v-bind:count="100"></list-component>
        <list-component message="씬짜오" :count="21"></list-component>
        <list-component message="니하오마"></list-component>
        <list-component v-bind:count="1000"></list-component>
    </ul>
</div>
```



props와 event

props의 엄격한 유효성 검증이 필요하다면?(이어서)

- 속성으로 전달할 값이 배열이나 객체이고 기본값(default value)을 지정하고 싶다면?
  - 반드시 함수를 사용해야 함. 예제 06-16

```
<template id="listTemplate">
    <li>{{message}}</li>
</template>
<script type="text/javascript">
Vue.component('list-component', {
    template : '#listTemplate',
    props : {
        message : { type:String, default:'안녕하세요' },
        count : { type:Number, required:true },
        countries : {
            type:Array,
            default: function() {
                return ['대한민국'];
            }
        }
    }
})
</script>
```

```
<div id="app">
    <ul>
        <list-component message="Hello" v-bind:count="100"
            v-bind:countries="['미국', '영국', '호주']"></list-component>
        <list-component message="씬짜오" :count="21"
            :countries="['베트남']"></list-component>
        <list-component message="니하오마"
            :countries="['중국', '타이완']"></list-component>
        <list-component v-bind:count="1000"></list-component>
    </ul>
</div>
```

props와 event

props 예제 : 예제 06-17

```
<template id="listTemplate">
   <div>
   <table id="list">
      <thead>
         <tr>
            <th>번호</th><th>이름</th><th>전화번호</th><th>주소</th>
         </tr>
      </thead>
      <tbody id="contacts" >
         <tr v-for="contact in contacts">
            <td>{{contact.no}}</td>
            <td>{{contact.name}}</td>
            <td>{{contact.tel}}</td>
            <td>{{contact.address}}</td>
         </tr>
      </tbody>
   </table>
   </div>
</template>
<script type="text/javascript">
Vue.component('contactlist-component', {
   template : '#listTemplate',
   props : [ 'contacts' ]
})
</script>
```

props와 event

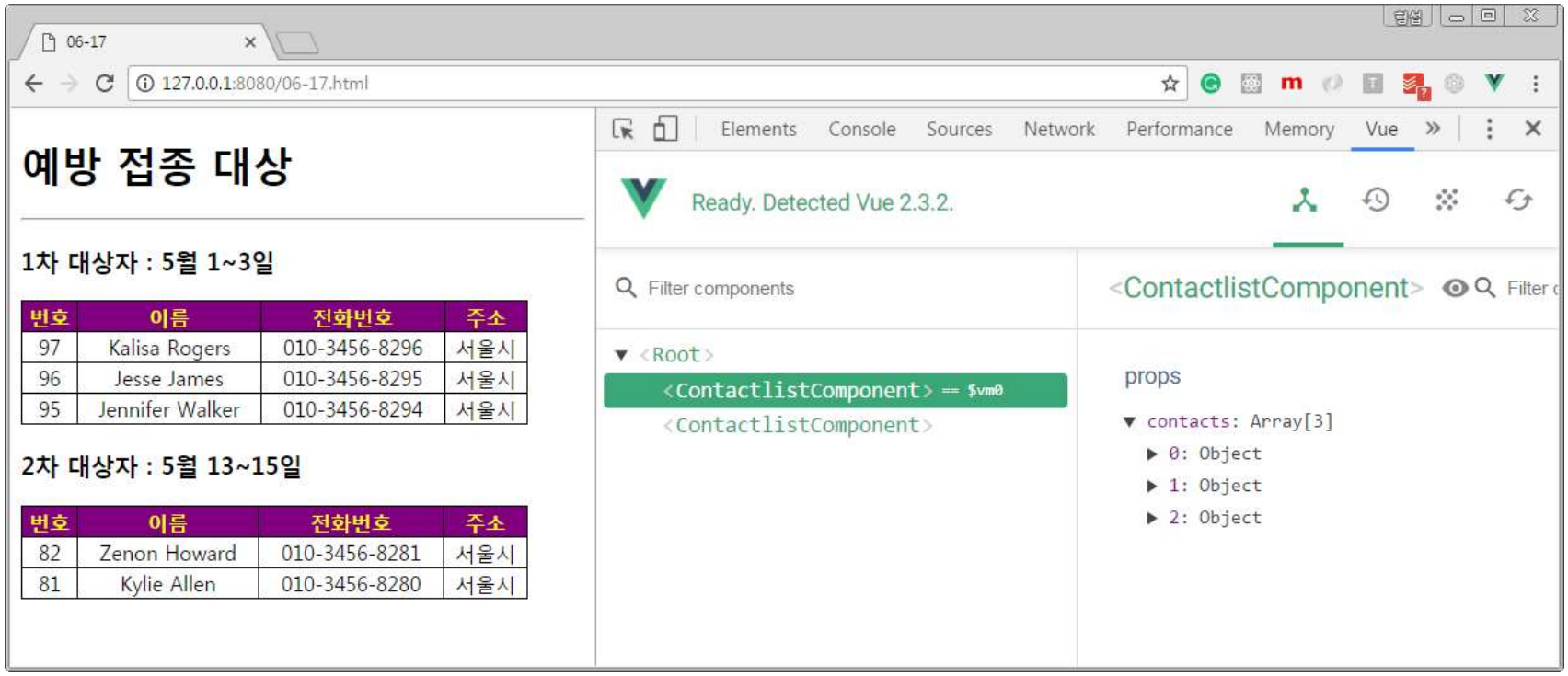
props 예제 : 예제 06-17 (이어서)

```
<div id="app">
    <h1>예방 접종</h1>
    <hr />
    <h3>1차 대상자 : 5월 1~3일</h3>
    <contactlist-component :contacts="list1"></contactlist-component>
    <h3>2차 대상자 : 5월 13~15일</h3>
    <contactlist-component :contacts="list2"></contactlist-component>
</div>
```

```
<script type="text/javascript">
Vue.config.devtools = true
var vm = new Vue({
    el : "#app",
    data : {
        list1 : [
            {"no":97,"name":"Kalisa Rogers","tel":"010-3456-8296","address":"서울시" },
            {"no":96,"name":"Jesse James","tel":"010-3456-8295","address":"서울시" },
            {"no":95,"name":"Jennifer Walker","tel":"010-3456-8294","address":"서울시" }
        ],
        list2 : [
            {"no":82,"name":"Zenon Howard","tel":"010-3456-8281","address":"서울시"},
            {"no":81,"name":"Kylie Allen","tel":"010-3456-8280","address":"서울시"}
        ]
    }
})
</script>
```

props와 event

props 예제 : 예제 06-17 (이어서)



props와 event

event를 이용한 정보 전달

- 자식컴포넌트에서 이벤트를 발신(emit)하고
- 부모컴포넌트에서 v-on 디렉티브를 이용해 수신한다.

예제 06-18

- 자식 컴포넌트

```
<!-- child Component 시작 -->
<style>
    .buttonstyle { width:120px; height:30px;
          text-align: center; }
</style>
<template id="childTemplate">
    <div>
        <button class="buttonstyle"
            v-on:click="clickEvent"
            v-bind:data-lang="buttonInfo.value">
            {{ buttonInfo.text }}
        </button>
    </div>
</template>
```

```
<script type="text/javascript">
Vue.component('child-component', {
    template : '#childTemplate',
    props : [ 'buttonInfo' ],
    methods : {
        clickEvent : function(e) {
            this.$emit('timeclick', e.target.innerText,
                e.target.dataset.lang);
        }
    }
})
</script>
<!-- child Component 끝 -->
```

props와 event

예제 06-18 (이어서)

- 부모 컴포넌트

```
<!-- parent Component 시작 -->
<template id="parent-template">
    <div>
        <child-component v-for="s in buttons" v-bind:button-info="s" v-on:timeclick="timeclickEvent">
        </child-component>
        <hr />
        <div>{{ msg }}</div>
    </div>
</template>
<script type="text/javascript">
Vue.component('parent-component', {
    template : '#parent-template',
    props : [ 'buttons' ],
    data : function() {
        return { msg:"" };
    },
    methods : {
        timeclickEvent : function(k, v) {
            this.msg = k + ", " +v;
        }
    }
})
</script>
<!-- parent Component 끝 -->
```

props와 event

예제 06-18 (이어서)

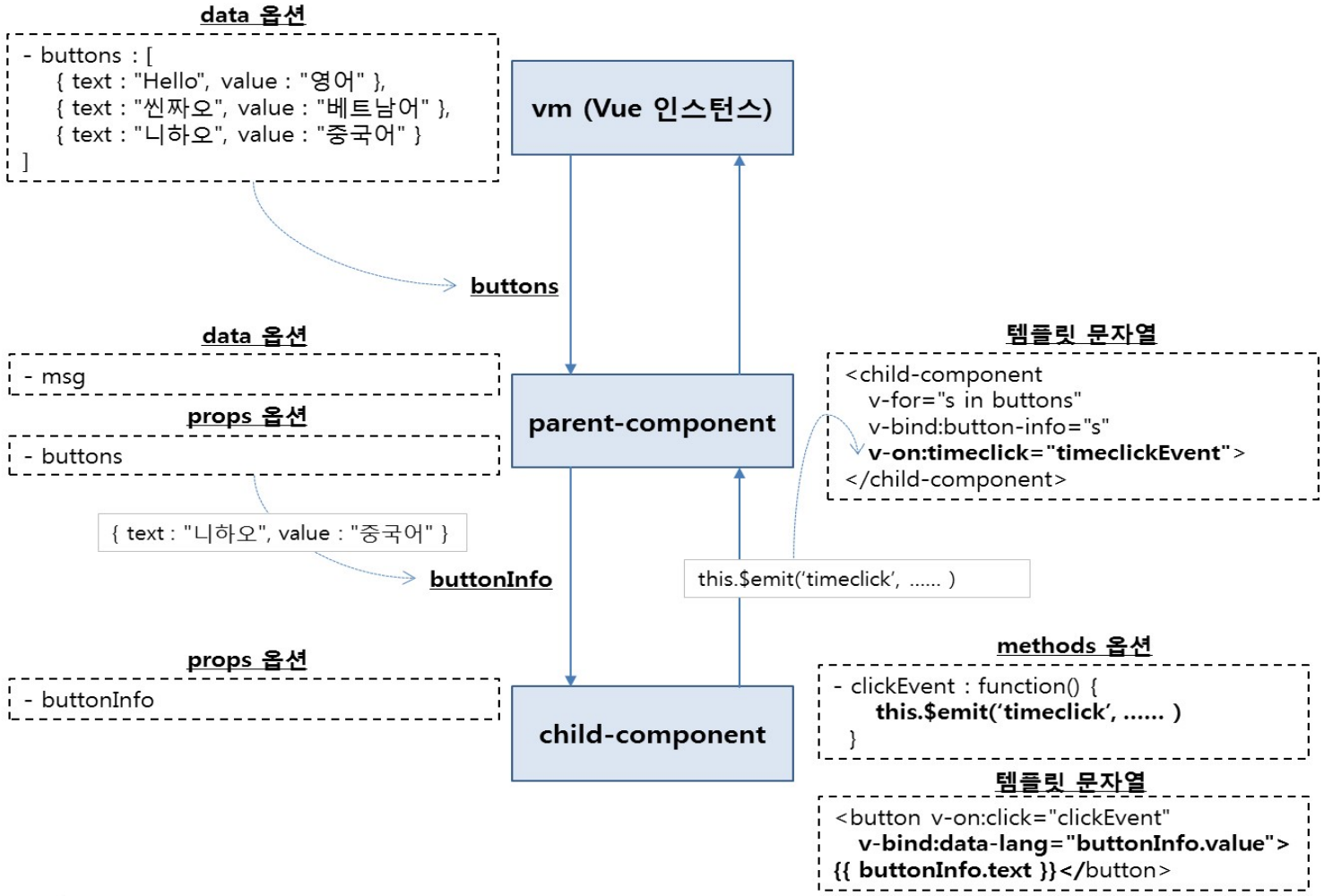
- 컨테이너

```
……
<div id="app">
  <parent-component :buttons="buttons"></parent-component>
</div>
……
<script type="text/javascript">
Vue.config.devtools = true;
var vm = new Vue({
    el : "#app",
    data : {
        buttons : [
            { text : "Hello", value : "영어" },
            { text : "씬짜오", value : "베트남어" },
            { text : "니하오", value : "중국어" }
        ]
    }
})
</script>
```

props와 event

예제 06-18 (이어서)

- 전체 구조



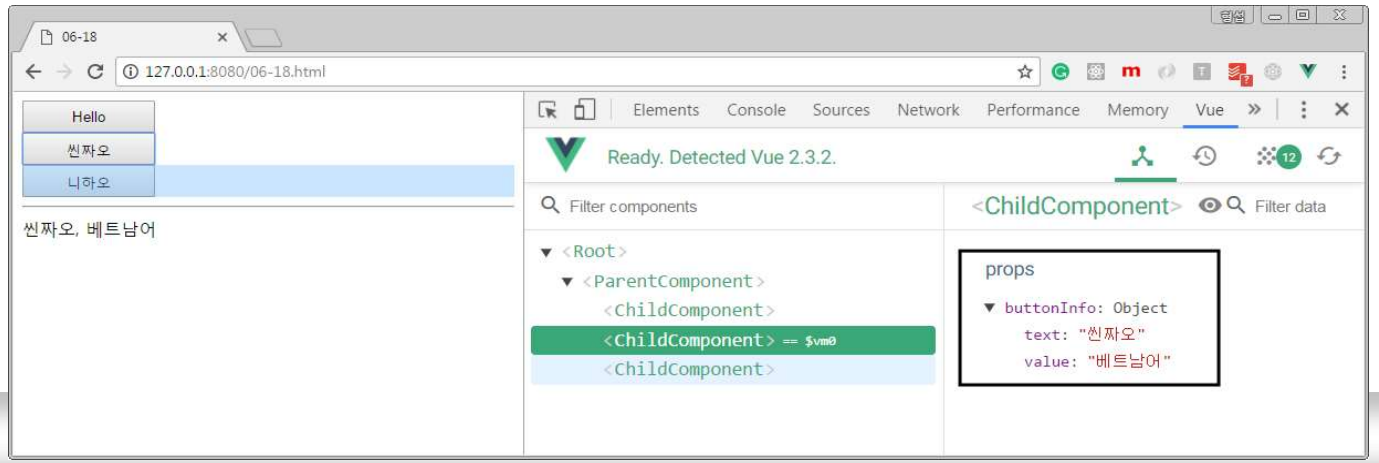
props와 event

예제 06-18(이어서)

- 핵심 코드

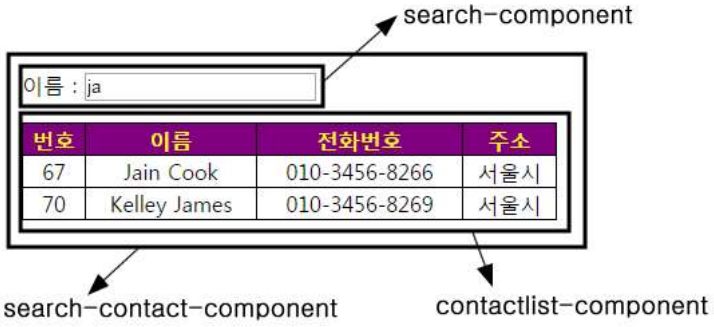
```
[ 자식 컴포넌트에서... ]
this.$emit('timeclick', e.target.innerText, e.target.dataset.lang);

[ 부모 컴포넌트에서 ]
<child-component v-for="s in buttons" v-bind:button-info="s" v-on:timeclick="timeclickEvent">
</child-component>
.....
methods : {
    timeclickEvent : function(k, v) {
        this.msg = k + ", " +v;
    }
}
```

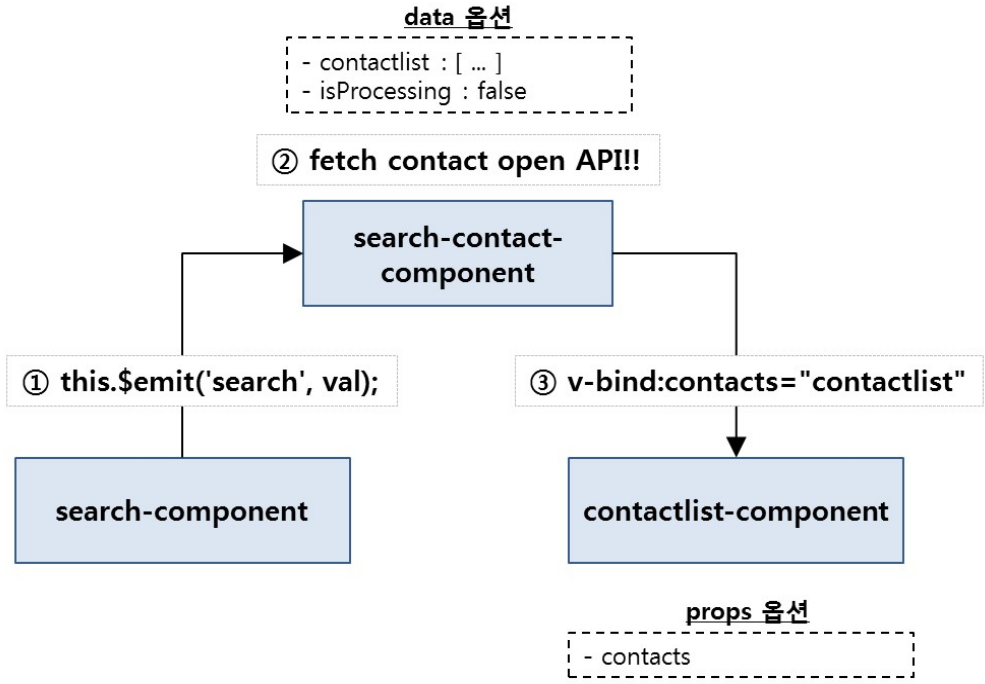


props와 event

props와 event 예제

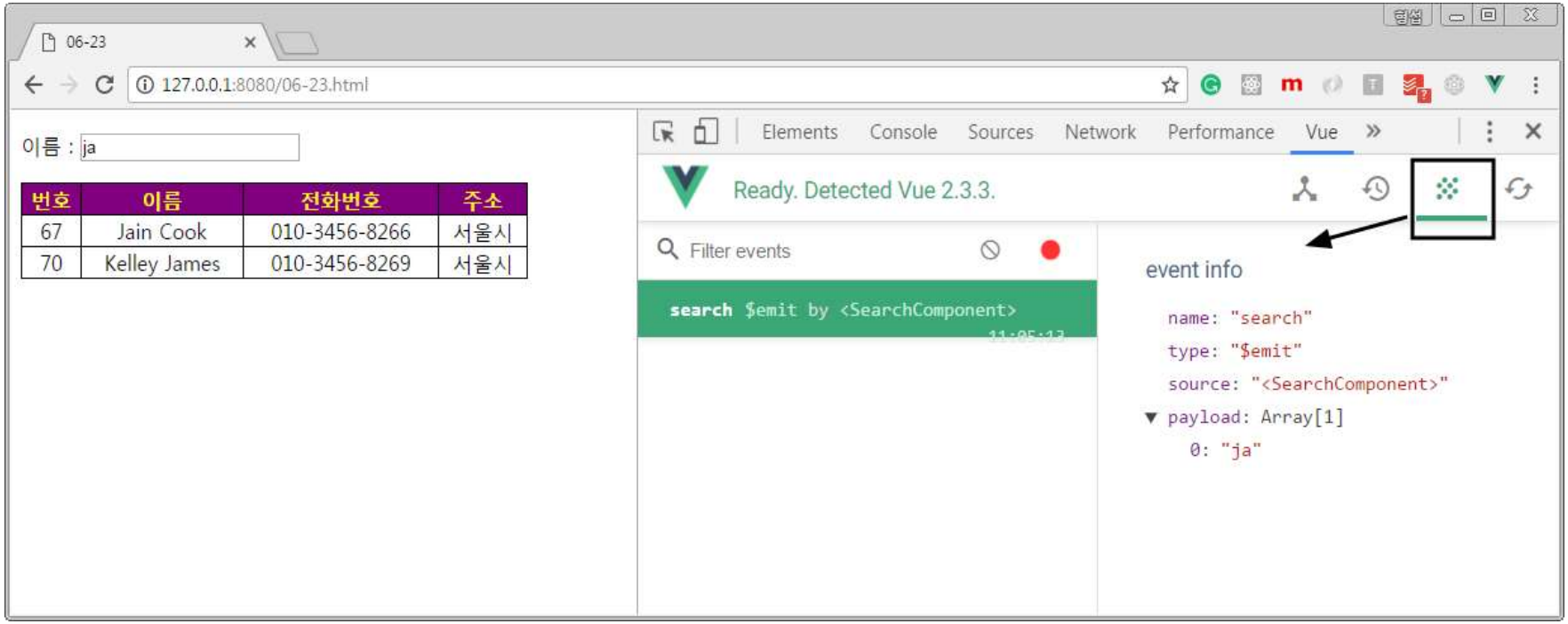


- 예제 06-19~23



props와 event

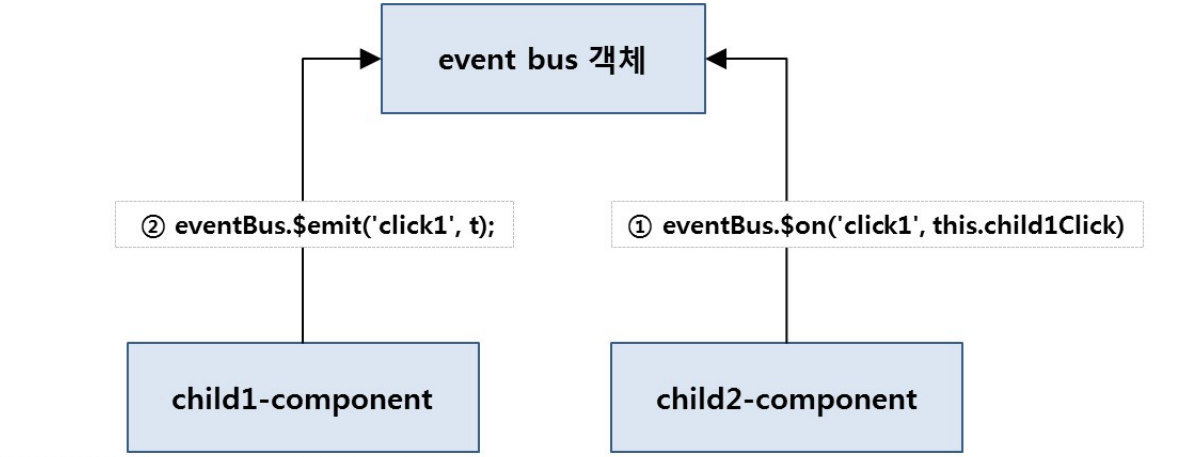
props와 event 예제(이어서)



이벤트 버스 객체를 이용한 통신

이벤트 버스 객체의 사용 케이스

- 부모 - 자식 관계가 아닌 컴포넌트
  - 형제 관계의 컴포넌트끼리 통신해야 하는 경우




```
Vue.component('child1-component',  {
    methods : {
        clickEvent : function() {
            ......
            eventBus.$emit('click1', t);
        }
    }
});
```

```
Vue.component('child2-component',  {
    created : function() {
        eventBus.$on('click1', this.child1Click)
    },
    methods : {
        child1Click : function(time) {
            this.timelist.push(time);
        }
    }
});
```

이벤트 버스 객체를 이용한 통신

예제 06-24

- 이벤트 버스 객체 및 첫번째 자식 컴포넌트

```
<!-- 이벤트 버스 객체 -->
<script type="text/javascript">
    var eventBus = new Vue();
</script>
<!-- 첫번째 자식 컴포넌트 시작-->
<template id="chidl1Template">
    <div>
        <button v-on:click="clickEvent">child1 button!!</button>
        <div>{{currentTime}}</div>
    </div>
</template>
<script type="text/javascript">
Vue.component('child1-component', {
    template : '#chidl1Template',
    data : function() {
        return { currentTime : '' };
    },
    methods : {
        clickEvent : function() {
            var d = new Date();
            var t = d.toLocaleTimeString() + " " + d.getMilliseconds() + "ms";
            eventBus.$emit('click1', t);
            this.currentTime = t;
        }
    }
});
</script>
<!-- 첫번째 자식 컴포넌트 끝-->
```

이벤트 버스 객체를 이용한 통신

예제 06-24(이어서)

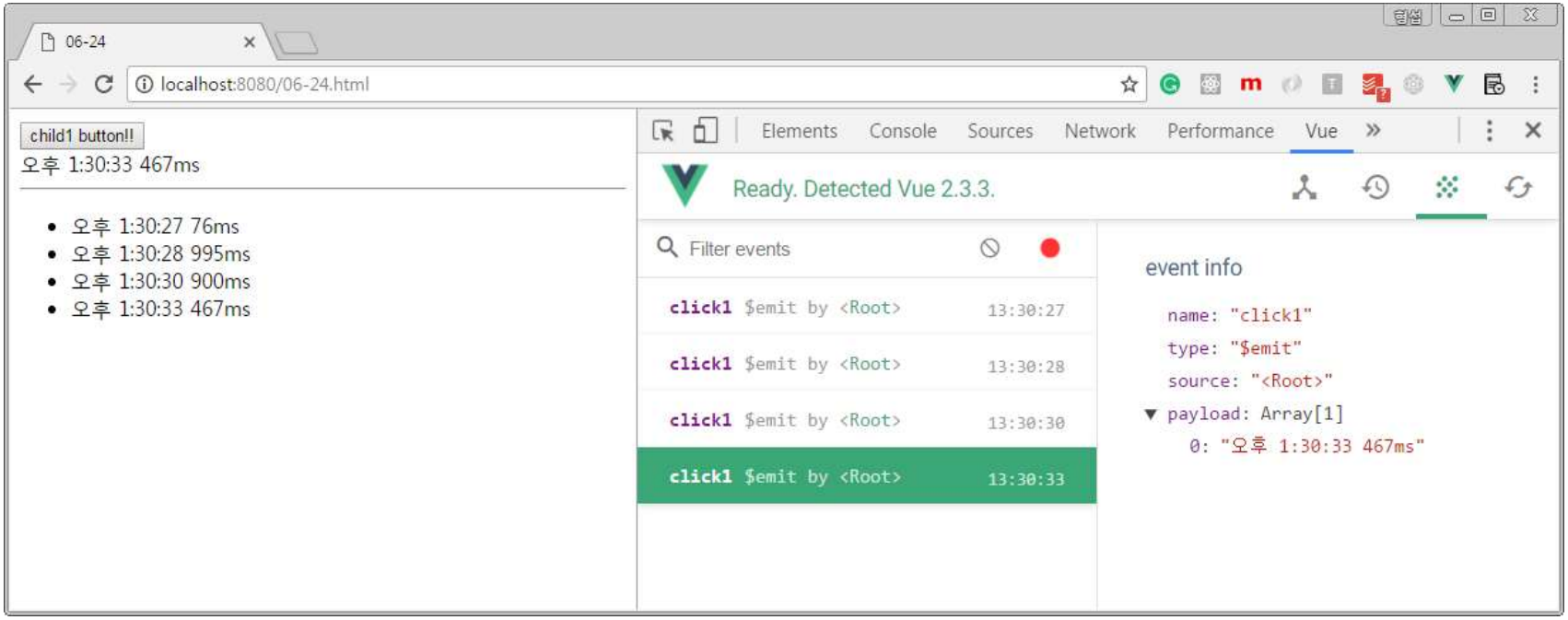
- 두번째 자식 컴포넌트

```
<!-- 두번째 자식 컴포넌트 시작-->
<template id="chidl2Template">
    <ul>
        <li v-for="t in timelist">{{t}}</li>
    </ul>
</template>
<script type="text/javascript">
Vue.component('child2-component', {
    template : '#chidl2Template',
    data : function() {
        return {    timelist : [ ]  };
    },
    created : function() {
        eventBus.$on('click1', this.child1Click);
    },
    methods : {
        child1Click : function(time) {
            this.timelist.push(time);
        }
    }
});
</script>
<!-- 두번째 자식 컴포넌트 끝-->
```

이벤트 버스 객체를 이용한 통신

예제 06-24(이어서)

- 실행 결과



이벤트 버스 객체를 이용한 통신

이벤트 버스 객체를 이용한 TodoListApp 예제

이벤트 버스 객체를 이용한 통신

이벤트 버스 객체를 이용한 TodoListApp 예제 (이어서)

- 기본 틀(예제 06-25)의 일부 : 이벤트 버스 객체와 Vue 인스턴스

```
<!--이벤트 버스 객체 시작-->
<script type="text/javascript">
    var eventBus = new Vue();
</script>
<!--이벤트 버스 객체 끝-->

<body>
    <div id="todolistapp">
        <div id="header" class="header">
            <h2>Todo List App</h2>

        </div>

    </div>
</body>
<script type="text/javascript">
Vue.config.devtools = true;
var vm = new Vue({
    el : "#todolistapp"
})
</script>
```

이벤트 버스 객체를 이용한 통신

이벤트 버스 객체를 이용한 TodoListApp 예제 (이어서)

- list-component : 예제 06-26

```
<template id="list-template">
    <ul id="todolist">
        <li v-for="(a, index) in todolist"
            v-bind:class="checked(a.done)"
            v-on:click="doneToggle(index)">
            <span>{{ a.todo }}</span>
            <span v-if="a.done"> (완료)</span>
            <span class="close"
                v-on:click.stop="deleteTodo(index)">
                &#x00D7;</span>
        </li>
    </ul>
</template>
<script type="text/javascript">
Vue.component('list-component', {
    template : '#list-template',
    created : function() {
        eventBus.$on('add-todo', this.addTodo);
    },
    data : function() {
        return {
            todolist : [
                { todo : "영화보기", done:false },
                { todo : "주말 산책", done:true },
                { todo : "ES6 학습", done:false },
                { todo : "잠실 야구장", done:false },
            ]
        }
    },
    methods : {
        checked : function(done) {
            if(done) return { checked:true };
            else return { checked:false };
        },
        addTodo : function(todo) {
            if (todo !== "") {
                this.todolist.push({ todo : todo, done:false });
            }
        },
        doneToggle : function(index) {
            this.todolist[index].done = !this.todolist[index].done;
        },
        deleteTodo : function(index) {
            this.todolist.splice(index,1);
        }
    }
})
</script>
```

이벤트 버스 객체를 이용한 통신

이벤트 버스 객체를 이용한 TodoListApp 예제 (이어서)

- input-component : 예제 06-27

```
<!-- input-component 시작-->
<style>
.input {
    border: none; width: 75%; height:35px; padding: 10px;
    float: left; font-size: 16px;
}
.addbutton {
    padding: 10px; width: 25%; height:35px;
    background: #d9d9d9;
    color: #555; float: left; text-align: center;
    font-size: 13px; cursor: pointer; transition: 0.3s;
}
.addbutton:hover { background-color: #bbb; }
</style>
<template id="input-template">
    <div>
        <input class="input" type="text" id="task"
          v-model.trim="todo" placeholder="입력 후 엔터!"
          v-on:keyup.enter="addTodo">
        <span class="addbutton" v-on:click="addTodo">
          추 가</span>
    </div>
</template>
```

```
<script type="text/javascript">
Vue.component('input-component', {
    template : '#input-template',
    data : function() {
        return { todo : "" }
    },
    methods : {
        addTodo : function() {
            eventBus.$emit('add-todo', this.todo);
            this.todo = "";
        }
    }
})
</script>
<!-- input-component 끝-->
```

이벤트 버스 객체를 이용한 통신

이벤트 버스 객체를 이용한 TodoListApp 예제 (이어서)

- 예제 06-28

```
<body>
    <div id="todolistapp">
        <div id="header" class="header">
            <h2>Todo List App</h2>
            <input-component></input-component>
        </div>
        <list-component></list-component>
    </div>
</body>
```