Vue.js

설치

필수 요소

CDN

- 특별한 설치 없이 html 파일에서 스크립트를 사용
- 프로토 타이핑 또는 학습용 (최신 버전)

```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

- 프로덕션 (특정 버전)

```
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.0"></script>
```

NPM

- Vue를 사용한 애플리케이션을 구축할 때

```
npm install vue
```

Hello

- hello.js, hello.html 파일을 같은 폴더에 생성 후 코딩

hello.js

```
1    let app = new Vue({
2        el:'#title',
3        data: {
4            message:'Hello'
5        }
6    });
```

hello.html

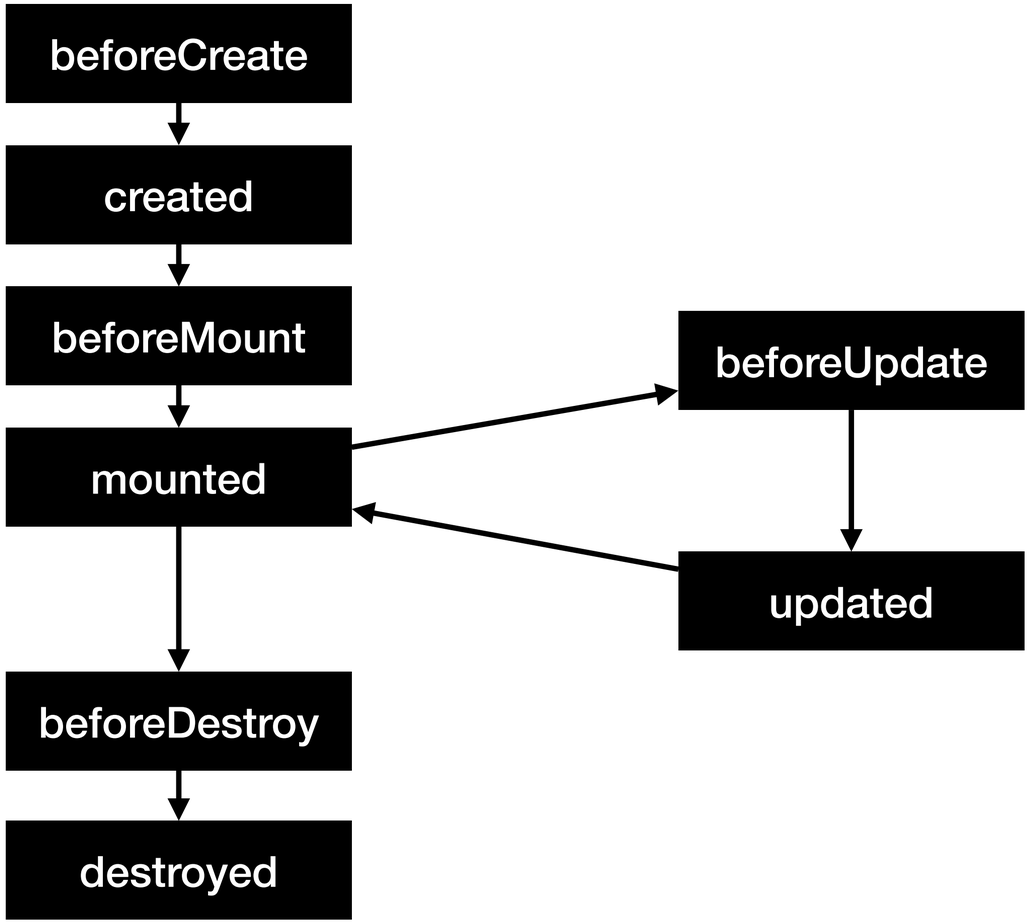
```
1    <!DOCTYPE html>
2    <html>
3        <head>
4            <title>Hello</title>
5        </head>
6        <body>
7            <h1 id="title">{{ message }}</h1>
8
9            <script src="https://cdn.jsdelivr.net/npm/vue"></script>
10           <script src="./hello.js"></script>
11       </body>
12   </html>
```

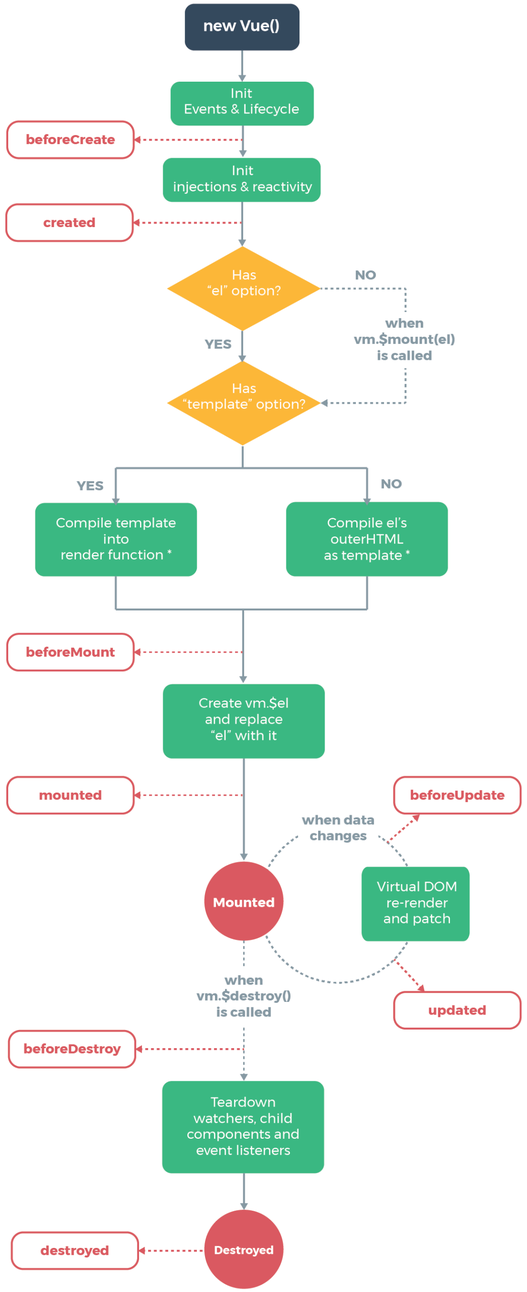
Vue instance

- Root Vue instance
  - Vue 애플리케이션의 기본 요소.
  - `new Vue({ /* options */ });`를 통해 생성
  - 옵션: 사용할 element, data, method 등을 선언

Vue instance

- Lifecycle





https://vuejs.org/v2/guide/instance.html#Instance-Lifecycle-Hooks

* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

Hello

- hello.js, hello.html 수정 -> data의 범위 확인

hello.js

```
let app = new Vue({
    el:'#title',
    data: {
        message:'Hello',
        name:'everyone'
    }
});
```

hello.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Hello</title>
    </head>
    <body>
        <div id="title">
            <h1>{{message}}, {{name}}</h1>
        </div>
        <h1>{{message}}</h1>

        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="./hello.js"></script>
    </body>
</html>
```

Hello

- hello.js, hello.html 수정 -> 객체 타입 데이터

hello.js

```
let app = new Vue({
    el:'#title',
    data: {
        message:{
            greetings:'Hello',
            name:'user!'
        }
    }
});
```

hello.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Hello</title>
    </head>
    <body>
        <div id="title">
            <h1>{{message.greetings}}, {{message.name}}</h1>
        </div>

        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="./hello.js"></script>
    </body>
</html>
```

Template syntax
문자열

- 문자열 표기

```
<span>메시지: {{ msg }}</span>
```

- 데이터 객체의 해당 속성이 변경될 때 마다 갱신

https://kr.vuejs.org/v2/guide/syntax.html

Template syntax

Attributes

- html의 attribute에는 {{ }} 를 사용할 수 없으며 v-bind를 사용. href, src, alt 등

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        address:'https://www.naver.com'
    }
});
```

hello.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Vue</title>
    </head>
    <body>
        <div id="app">
            <a v-bind:href='address'>Naver</a>
        </div>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="./hello.js"></script>
    </body>
</html>
```

v-bind:href 는 :href 와 동일하게 동작함.
v-bind: 로 시작하는 속성은
:로 줄여쓸 수있음.

https://kr.vuejs.org/v2/guide/syntax.html

Template syntax
Expressions

- Javascript의 표현식 사용

```HTML
{{ number + 1 }}

{{ ok ? 'YES' : 'NO' }}

{{ message.split('').reverse().join('') }}

<div v-bind:id="'list-' + id"></div>
```

```HTML
<!-- 아래는 구문입니다, 표현식이 아닙니다. -->
{{ var a = 1 }}

<!-- 조건문은 작동하지 않습니다. 삼항 연산자를 사용해야 합니다. -->
{{ if (ok) { return message } }}
```

X

https://kr.vuejs.org/v2/guide/syntax.html

조건문

- v-if, v-else-if, v-else: DOM 자체가 수정됨

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        age:20
    }
});
```

hello.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Vue</title>
    </head>
    <body>
        <div id="app">
            <p v-if="age>=20">Adult</p>
            <p v-else-if="age<20 && age > 12">Teen ager</p>
            <p v-else>Kid</p>
        </div>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="./hello.js"></script>
    </body>
</html>
```

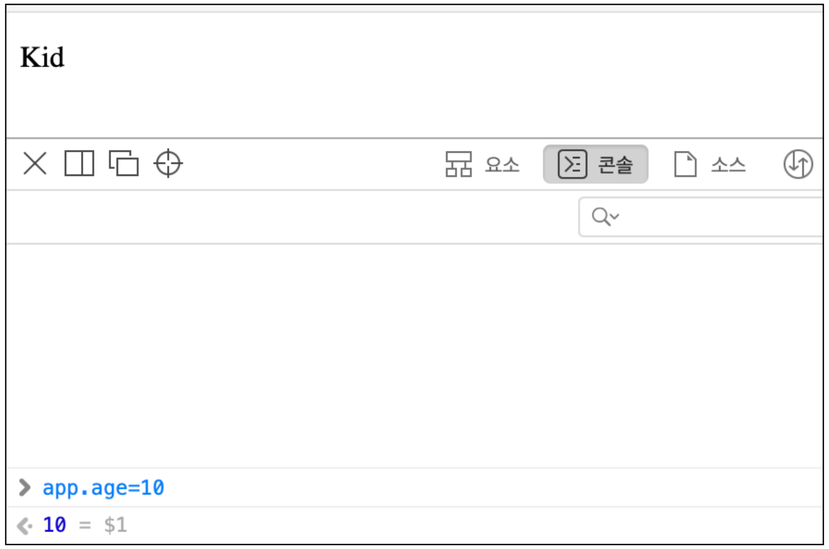
https://kr.vuejs.org/v2/guide/index.html#조건문과-반복문

조건문

- 브라우저의 콘솔 창에서 다음과 같이 값을 변화시켜가면서 확인할 수 있다.

Chrome (F12)

Safari (option + command+c)



반복문

- for-each의 구조: li에 사용

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        todos:[
            {text:'JavaScript 배우기'},
            {text:'Vue 배우기'},
            {text:'멋진 것 만들기'}
        ]
    }
});
```

hello.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Vue</title>
    </head>
    <body>
        <div id="app">
            <ul>
                <li v-for="todo in todos">{{todo.text}}</li>
            </ul>
        </div>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="./hello.js"></script>
    </body>
</html>
```

https://kr.vuejs.org/v2/guide/index.html#조건문과-반복문

반복문

- for-each의 구조: div에 사용

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        todos:[
            {text:'JavaScript 배우기'},
            {text:'Vue 배우기'},
            {text:'멋진 것 만들기'}
        ]
    }
});
```

hello.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Vue</title>
    </head>
    <body>
        <div id="app">
            <div v-for="todo in todos">
                <p>{{todo.text}}</p>
            </div>
        </div>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="./hello.js"></script>
    </body>
</html>
```

https://kr.vuejs.org/v2/guide/index.html#조건문과-반복문

반복문

- for-each의 구조: index 받기

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        todos:[
            {text:'JavaScript 배우기'},
            {text:'Vue 배우기'},
            {text:'멋진 것 만들기'}
        ]
    }
});
```

hello.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Vue</title>
    </head>
    <body>
        <div id="app">
            <div v-for="(todo, index) in todos">
                <p>{{index}}::{{todo.text}}</p>
            </div>
        </div>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="./hello.js"></script>
    </body>
</html>
```

https://kr.vuejs.org/v2/guide/index.html#조건문과-반복문

반복문

- for-each의 구조: object의 값들 출력하기

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        user:{
            userid:'user123',
            name:'Jane Vue',
            email:'user123@email.com'
        }
    }
});
```

hello.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Vue</title>
    </head>
    <body>
        <div id="app">
            <div v-for="(value, name) in user">
                <p>{{name}} - {{value}}</p>
            </div>
        </div>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="./hello.js"></script>
    </body>
</html>
```

https://kr.vuejs.org/v2/guide/index.html#조건문과-반복문

반복문

- for-each의 구조: range for

hello.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Vue</title>
    </head>
    <body>
        <div id="app">
            <div v-for="n in 10">
                <p>{{n}}</p>
            </div>
        </div>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="./hello.js"></script>
    </body>
</html>
```

https://kr.vuejs.org/v2/guide/index.html#조건문과-반복문

Event Handling

- Click, mouseover, submit, keyup.x 등의 이벤트를 감지하고 이를 처리함
- 간단한 코드는 다음과 같이 바로 작성

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        counter:0
    }
});
```

hello.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Vue</title>
    </head>
    <body>
        <div id="app">
            <button v-on:click="counter+=1">+</button>
            <p>{{counter}}</p>
        </div>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="./hello.js"></script>
    </body>
</html>
```

https://kr.vuejs.org/v2/guide/events.html

Event Handling

- method를 이용한 처리

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        counter:0
    },
    methods:{
        addCounter:function(){
            this.counter += 1
        }
    }
});
```

또는

```
    methods:{
        addCounter(){
            this.counter += 1
        }
    }
```

hello.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Vue</title>
    </head>
    <body>
        <div id="app">
            <button v-on:click="addCounter">+</button>
            <p>{{counter}}</p>
        </div>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="./hello.js"></script>
    </body>
</html>
```

Event Handling

- method parameter

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        message:'Select a user',
        users:[ {name:'User'}, {name:'Admin'} ]
    },
    methods:{
        sayHi(name){
            this.message = 'Hi, ' + name
        }
    }
});
```

hello.html

```
<body>
    <div id="app">
        <p>{{message}}</p>
        <ul>
            <li v-for="user in users">
                <p v-on:click="sayHi(user.name)">{{user.name}}</p>
            </li>
        </ul>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue"></script>
    <script src="./hello.js"></script>
</body>
```

v-on:click 는 @click 와 동일하게 동작함.
v-on: 로 시작하는 속성은
@로 줄여쓸 수있음.

```
<li v-for="user in users">
    <p @click="sayHi(user.name)">{{user.name}}</p>
</li>
```

Form input binding

- 데이터와 view의 양방향 모델링

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        comment:''
    }
});
```

hello.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Vue</title>
    </head>
    <body>
        <div id="app">
            <input v-model="comment">
            <p>{{comment}}</p>
        </div>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="./hello.js"></script>
    </body>
</html>
```

Form input binding

- textarea

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        comment:''
    }
});
```

hello.html

```
<body>
    <div id="app">
        <span>여러 줄을 가지는 메시지:</span>
        <p style="white-space: pre-line">{{ comment }}</p>
        <br>
        <textarea v-model="comment" placeholder="여러줄을 입력해보세요"></textarea>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue"></script>
    <script src="./hello.js"></script>
</body>
```

Form input binding

- checkbox: 단일 값은 boolean으로, 여러 개의 값은 배열로 처리

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        value:false,
        values:[]
    }
});
```

hello.html

```
<body>
    <div id="app">
        <input type="checkbox" id="checkbox_single" v-model="value">
        <label for="checkbox_single">{{ value }}</label>

        <div>
            <input type="checkbox" id="checkbox_one" value="one" v-model="values">
            <label for="checkbox_one">one</label>
            <input type="checkbox" id="checkbox_two" value="two" v-model="values">
            <label for="checkbox_two">two</label>
            <input type="checkbox" id="checkbox_three" value="three" v-model="values">
            <label for="checkbox_two">three</label>
            <p>Checked: {{ values }}</p>
        </div>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue"></script>
    <script src="./hello.js"></script>
</body>
```

Form input binding

- v-model.lazy: 각 입력에 동기화 하는 것이 아니라 값이 수정 된 다음 동기화

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        value:'hello',
        value2:'hello'
    }
});
```

hello.html

```
<body>
    <div id="app">
        <input v-model.lazy='value'>
        <p>{{value}}</p>
        <input v-model='value2'>
        <p>{{value2}}</p>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue"></script>
    <script src="./hello.js"></script>
</body>
```

두 input에 글자를 입력해보고 엔터키를 입력해본다.

Form input binding

- v-model.number: 사용자 입력이 숫자로 형변환 되도록 설정

hello.js

```
1   let app = new Vue({
2       el:'#app',
3       data: {
4           age:20
5       }
6   });
```

hello.html

```
7   <body>
8       <div id="app">
9           <input v-model.number='age'>
10          <p>{{age}}</p>
11      </div>
12      <script src="https://cdn.jsdelivr.net/npm/vue"></script>
13      <script src="./hello.js"></script>
14  </body>
```

input에 숫자가 아닌 글자를 입력해본다.

Form input binding

- v-model.trim: 입력을 trim (문자열 양 끝의 공백을 제거)

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        message:''
    }
});
```

hello.html

```
<body>
    <div id="app">
        <input v-model.trim='message'>
        <p>{{message}}</p>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue"></script>
    <script src="./hello.js"></script>
</body>
```

문자열의 시작과 끝에 공백을 입력해본다.

Computed properties

- 원본 데이터의 값을 바로 사용하지 않고 처리한 뒤 사용해야 하는 경우 이를 computed property로 만들어 간단하게 사용할 수 있다.
- 예) 두 변수의 값을 합성한 문자열 사용: 이름(아이디)
- 예) 원본 데이터의 값을 바로 사용하지 않는 경우: true/false 대신 재고있음/매진

- 함수 호출과 다른 점: 참조 하고 있는 데이터가 업데이트 되지 않으면 저장된(캐싱된) 값을 바로 반환함

Computed properties

- computed property

hello.js

```
let app = new Vue({
    el:'#app',
    data: {
        user_id:'user123',
        user_grade:'Silver'
    },
    computed:{
        user_info:function() {
            return this.user_id+'(' + this.user_grade + ')'
        }
    }
});
```

hello.html

```
<body>
    <div id="app">
        <p>{{user_info}}</p>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue"></script>
    <script src="./hello.js"></script>
</body>
```

Component

- 재사용 가능한 Element를 등록하고 이를 코드에 사용함
- 화면을 구성할 template, data, html attribute 값을 정의하는 props 등을 옵션에 추가
- 만드는 법

```
Vue.component(‘tag_name’, {
    // 옵션
});
```

Component

- 전역 등록
- new Vue() 앞에 다음과 같이 정의한다.

```
Vue.component('tag_name', {
    // 옵션
});
```

Component

- 기본 사용법

hello.js

```
1   Vue.component('hello', {
2       props:['name'],
3       template:'<p>{{message}}</p>',
4       data: function(){
5           return { message: 'hello!, ' + this.name }
6       }
7   });
8
9   let app = new Vue({
10      el:'#app'
11  });
```

data 를 함수로 구현해야 하는 점을 주의한다.
template는 하나의 root element를 가져야 한다.
복잡한 구조라면 div 아래 자식을 두는 등의
방법을 사용한다.

hello.html

```
7       <body>
8           <div id="app">
9               <hello name="user"></hello>
10              <hello name="students"></hello>
11          </div>
12          <script src="https://cdn.jsdelivr.net/npm/vue"></script>
13          <script src="./hello.js"></script>
14      </body>
```

생성된 Component는 반복적으로 사용 가능

Component

- 지역 등록
- 하나의 Vue instance 에서만 사용할 수 있는 컴포넌트

new Vue() 함수의 option 부분에

components: { ‘tag_name’, {

    // 옵션

}} 로 선언 가능.

Component

hello.js

```
 9  let app = new Vue({
10      el:'#app',
11      components:{
12          'local-hello':{
13              template:'<p>This is local component'
14          }
15      }
16  });
```

hello.html

```
 8      <div id="app">
 9          <hello name="user"></hello>
10          <hello name="students"></hello>
11          <local-hello></local-hello>
12      </div>
```

Event

- Vue에서 부모 컴포넌트는 자식 컴포넌트에 데이터를 전달할 수 있음
- 자식 컴포넌트가 부모 컴포넌트의 데이터를 변경할 수는 **없음**.
- 자식 컴포넌트가 이벤트를 발생시키고, 이를 이용해 부모의 데이터를 수정해야 함.

Event

- 다음과 같이 작성하고 테스트

hello.js

```
1   Vue.component('add',{
2       template:'<button @click="add">{{count}}</button>',
3       data() {
4           return {count:0}
5       },
6       methods:{
7           add(){
8               this.count += 1
9           }
10      }
11  })
12  let app = new Vue({
13      el:'#app',
14      data:{
15          totalCount:0
16      }
17  });
```

hello.html

```
7       <body>
8           <div id="app">
9               <h3>{{totalCount}}</h3>
10              <add></add><br>
11              <add></add><br>
12              <add></add><br>
13          </div>
14          <script src="https://cdn.jsdelivr.net/npm/vue"></script>
15          <script src="./hello.js"></script>
16      </body>
```

Event

hello.js

```
1   Vue.component('add',{
2       template:'<button @click="add">{{count}}</button>',
3       data() {
4           return {count:0}
5       },
6       methods:{
7           add(){
8               this.count += 1;
9               this.$emit('event-add', 1);
10              }
11      }
12  });
13
14  let app = new Vue({
15      el:'#app',
16      data:{
17          totalCount:0
18      },
19      methods:{
20          updateTotal(count){
21              this.totalCount += count
22          }
23      }
24  });
```

hello.html

```
7       <body>
8           <div id="app">
9               <h3>{{totalCount}}</h3>
10              <add @event-add="updateTotal"></add><br>
11              <add @event-add="updateTotal"></add><br>
12              <add @event-add="updateTotal"></add><br>
13          </div>
14          <script src="https://cdn.jsdelivr.net/npm/vue"></script>
15          <script src="./hello.js"></script>
16      </body>
```

Template

- Component에서 사용하는 template을 스트링으로 정의할 때의 문제점
- Component의 크기가 커지고 복잡해 질 경우
  - 코드가 복잡해진다.
  - 유지보수가 어렵다
  - 줄 바꿈 등의 처리가 어렵다.
- 별도의 파일을 작성하여 모듈화 할 수 있으나 이 경우 complie이 필요하며
- 별도의 프로젝트로 생성해 작성 가능하다.