2주차

템플릿 문법

보간법

#문자열

데이터 바인딩의 기본 형태는 Mustache 구문 ({{}}, 이중 중괄호) 를 사용하는 텍스트 형태

```
<span>메세지 : {{msg}}</span>
<span v-once>{{msg}}</span>
```

mustache 구문 내 msg는 vue 데이터 객체의 msg 속성 값으로 대체되고 msg 속성 값이 변경될때 마다 갱신됩니다.

v-once 를 사용하여 데이터 변경 시 업데이트 되지 않게 만들 수 있습니다.

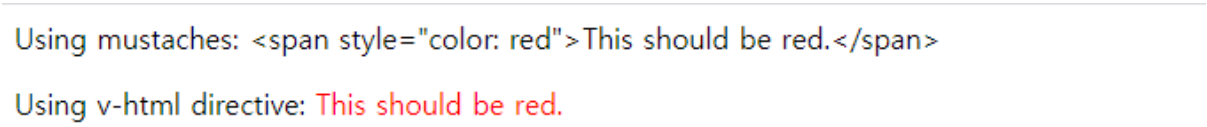
#원시 HTML

이중 중괄호는 HTML이 아닌 일반 텍스트로 데이터를 해석합니다. 실제 HTML 을 출력하려면

v-html 를 사용해야합니다.

```
<div id="app">
  <p>Using mustaches: {{ rawHtml }}</p>
  <p>Using v-html directive: <span v-html="rawHtml"></span></p>
</div>
```

```
<script>
var app = new Vue({
  el: '#app',
data: { rawHtml : '<span style="color: red">This should be red.</span>' } }) </script>
```



mustaches 구문은 일반 텍스트로 노출합니다. 위의 소스를 보면 p태그 안 바닥 페이지에 rawHtml 라는 데이터를 노출시켰다. rawHtml 데이터가 HTML 구문의 데이터라고 할지라도 일반 텍스트로 노출된다는 의미입니다. html 영역을 핸들링 하는 v-html 은 신뢰할수 있는 콘텐츠에서만 사용해야한다고 경고하고 있습니다.

#속성

mustaches는 html 속성에서 사용할 수 없습니다 대신 v-bind 를 사용해야합니다.

```
<div v-bind:id="dynamicId"></div>

<div id="app">
  <p>Using mustaches: {{ rawHtml }}</p>
  <p>Using v-html directive: <span v-html="rawHtml"></span></p>
  <div v-bind:id="dynamicId">dynamicId test</div>
</div>
<script>
```

```
  var app = new Vue({
   el: '#app',
  data: { rawHtml : '<span style="color: red">This should be red.</span>',
  dynamicId: 'divDynamicId' }
})
</script>
```

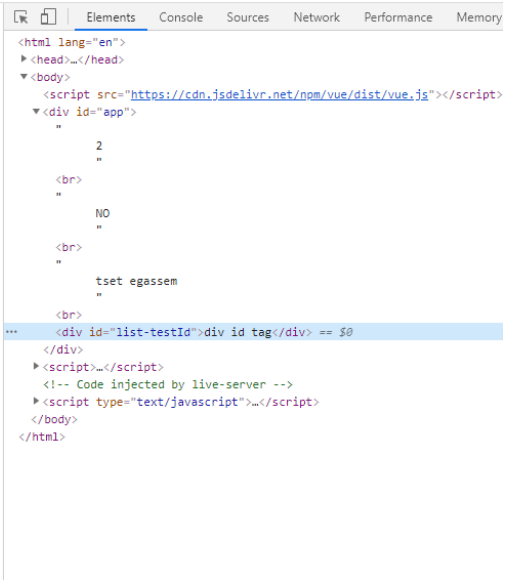
mustaches 는 html 태그 영역이 아닌 텍스트 영역에서 사용하는 방식으로 html 속성에서는 사용할수 없습니다. 그러나 때로 html 속성으로 화면을 제어할 필요가 있는데 있때 'v-' 접두사가 있는 특수 속성을 활용하면 됩니다.

ex) dynamicId 라는 객체에 divDynamicId 속성을 넣었고 v-bind:id로 id를 선언하면 div의 id는 divDynamicId로 선언이 됩니다.

#javascript 표현

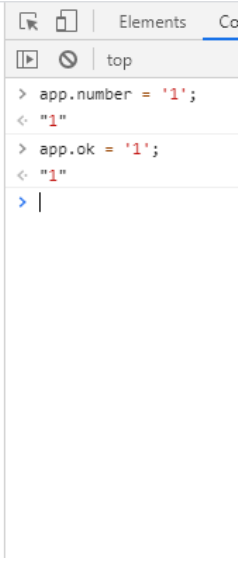
```
<div id="app">
  {{ number + 1 }}
  <br>
  {{ ok ? 'YES' : 'NO' }}
  <br>
  {{ message.split('').reverse().join('') }}
  <br>
  <div v-bind:id="'list-' + id">div id tag</div>
</div>

<script>
var app = new Vue({
  el: '#app',
  data: {
    number : 1,
    ok : null,
    message : 'message test',
    id : 'testId'
    }
  })
</script>
```

2
NO
tset egassem
div id tag



11
YES
tset egassem
div id tag

11
YES
tset egassem
div id tag



단 , 한가지 제한사항으로는 각 바인딩에 하나의 단일 표현식만 포함 될 수 있으므로 아채럼 작성하면 X

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

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

mustaches 구문안에서 js 표현식을 써도 됩니다. 하지만 위 코드처럼 number의 값을 문자열로 넣으면 숫자로 자동 변환이 되지 않으므로 11로 표현 될 수 있습니다. (Number("string") 사용하면 +2)

디렉티브

디렉티브는 v- 접두사가 있는 속성입니다. 디렉티브 속성값은 단일 js 표현식이 됩니다. (v-for는 예외)

`<p v-if="seen">이제 나를 볼 수 있어요</p>`

여기서, `v-if` 디렉티브는 `seen` 표현의 진실성에 기반하여 `<p>` 엘리먼트를 제거 또는 삽입합니다.

v-if 와 v-show의 차이

→ v-if를 사용할 때 초기 랜더링에서 조건이 거짓이면 아무것도 하지 않음. v-show는 display : none 강태 → v-if는 토글비요잉 높지만 v-show는 초기 랜더링 비용이 높지 않음, 매우 자주 전환 할 경우 v-sho , 자주 변경할 필요없을 땐 v-if

이름	설명

v-text	요소의 textcontext 변경
v-html	innerHTML 변경 (xss 공격에 취약)
v-once	엘리먼트나 컴포넌트를 한번만 랜더링 처리, 상위 컴포넌트에 선언되면 하위에도 적용
v-pre	해당 엘리먼트와 모든 자식 엘리먼트 컴파일 건너뜀
v-cloak	인스턴스가 준비될때까지 컴파일 되지 않는 mustache 바인딩을 숨기는데 사용 가능
v-model	양방향 데이터 바인딩
v-bind	단방향 데이터 바인딩
v-on	이벤트 바인딩 사용

#전달인자

일부 디렉티브는 콜론으로 표시되는 "전달인자"를 사용 할 수 있습니다. 예를 들어 v-bind 디렉티브는 반응적으로 HTML 속성을 갱신하는데 사용됩니다.

```
</head>
<body>
    <div id ="app">
        <a v-bind:href="url">페이지 이동</a>
        <br/>
        <a v-on:click="doSomething">url 정보 daum으로 변경</a>
        <pre>
            {{$data}}
        </pre>
    </div>
</body>
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

<script>
    new Vue({
        el: "#app",
        data:{
            url:'https://naver.com'
        },
        methods:{
            doSomething(){
                this.url = "https://daum.net"
            }
        }
    })
</script>
</html>
```

url의 값을 v-bind 디렉티브에 href로 전달하여 a태그 이벤트 실행 시 이동합니다.

doSomething 이라는 메소드를 v-on 디렉티브에 click시 실행합니다.

v-bind:href는 네이버 이동 선언 , v-on:click 은 doSomething 메소드를 이용하여 url를 다음으로 변경처리

#동적 전달인자

js 표현식을 대괄호로 묶어 디렉티브 전달인자로도 사용 가능합니다

```
<template>
  <h1
    :[attr]="msg"    // [attr] 동적 전달인자
    @[event]="add">  // [event] 동적 전달인자
```

```
    {{ msg }}
  </h1></template>
```

```
<script>
export default {
  data() {
    return {
      msg: 'active',
      attr: 'class',
      event: 'click'
    }
  },
  methods: {
    add() {
      this.msg += '!'
    }
  }
}
</script>
```

#수식어

수식어는 . 으로 표시 되는 특수 접미사로, 디렉티브를 특별한 방법으로 바인딩 해야하는 점을 알립니다. 예로 `.prevent` 수식어는 트리거된 이벤트에서 `event.preventDefault()`를 호출하도록 `v-on` 디렉티브에게 알려줍니다.

<form v-on:submit.prevent="onSubmit"> ... </form>

v-on

- 약어: `@`
- 예상됨: `Function | Inline Statement | Object`
- 전달인자: `event`
- 수식어:
  - `.stop` - `event.stopPropagation()` 을 호출합니다.
  - `.prevent` - `event.preventDefault()` 을 호출합니다.
  - `.capture` - 캡처 모드에서 이벤트 리스너를 추가합니다.
  - `.self` - 이벤트가 이 엘리먼트에서 전달된 경우에만 처리 됩니다
  - `.{keyCode | keyAlias}` - 특정 키에 대해서만 처리 됩니다.
  - `.native` - 컴포넌트의 루트 엘리먼트에서 네이티브 이벤트를 수신합니다.
  - `.once` - 단 한번만 처리됩니다.
  - `.left` - (2.2.0) 왼쪽 버튼 마우스 이벤트 트리거 처리기.
  - `.right` - (2.2.0) 오른쪽 버튼 마우스 이벤트 트리거 처리기.
  - `.middle` - (2.2.0) 가운데 버튼 마우스 이벤트 트리거 처리기.
  - `.passive` - (2.3.0+) DOM 이벤트를 `{ passive: true }` 와 연결합니다.

```
<!-- 메소드 핸들러 -->
<button v-on:click="doThis"></button>

<!-- dynamic event (2.6.0+) -->
<button v-on:[event]="doThis"></button>

<!-- 인라인 구문 -->
<button v-on:click="doThat('hello', $event)"></button>

<!-- 약어 -->
<button @click="doThis"></button>

<!-- shorthand dynamic event (2.6.0+) -->
<button @[event]="doThis"></button>

<!-- 전파 금지 -->
<button @click.stop="doThis"></button>

<!-- 기본 동작 방지 -->
<button @click.prevent="doThis"></button>

<!-- 표현식이 없는 기본 동작 방지 -->
```

```
<form @submit.prevent></form>

<!-- 수식어 체이닝 -->
<button @click.stop.prevent="doThis"></button>

<!-- 키 별칭을 이용한 키 입력 수식어 -->
<input @keyup.enter="onEnter">

<!-- 키 코드를 이용한 키 입력 수식어 -->
<input @keyup.13="onEnter">

<!-- the click event will be triggered at most once -->
<button v-on:click.once="doThis"></button>

<!-- 객체 구문 (2.4.0+) -->
<button v-on="{ mousedown: doThis, mouseup: doThat }"></button>
```

약어

v- 접두사는 템플릿의 Vue 특정 속성을 식별하기 위한 시각적인 신호 역할을 합니다. 이 기능은 Vue.js를 사용하여 기존의 마크업에 동적인 동작을 적용할 때 유용하지만 일부 자주 사용되는 디렉티브에 대해 너무 장황하다고 느껴질 수 있습니다. 동시에 Vue.js가 모든 템플릿을 관리하는 SPA를 만들 때 v- 접두어의 필요성이 떨어집니다. 따라서 가장 자주 사용되는 두개의 디렉티브인 v-bind와 v-on에 대해 특별한 약어를 제공합니다.

이들은 일반적인 HTML과 조금 다르게 보일 수 있습니다. 하지만 :와 @는 속성 이름에 유효한 문자이며 Vue.js를 지원하는 모든 브라우저는 올바르게 구문 분석을 할 수 있습니다. 또한 최종 렌더링 된 마크업에는 나타나지 않습니다. 약어는 완전히 선택사항이지만 나중에 익숙해지면 편할 것 입니다.

출처:

#v-bind 약어

```
<!-- 전체 문법 -->
<a v-bind:href="url"> ... </a>

<!-- 약어 -->
<a :href="url"> ... </a>

<!-- shorthand with dynamic argument (2.6.0+) -->
<a :[key]="url"> ... </a>
```

#v-on 약어

```
<!-- 전체 문법 -->
<a v-on:click="doSomething"> ... </a>

<!-- 약어 -->
<a @click="doSomething"> ... </a>

<!-- shorthand with dynamic argument (2.6.0+) -->
<a @[event]="doSomething"> ... </a>
```