

Vue.js 개요

Vue.js 개요

- **Vue.js**

Vue.js는 SPA 개발을 위한 대표적인 자바스크립트 프레임워크

- **SPA(Single Page Application)**

단일 페이지로 사용 가능한 애플리케이션을 의미하며 페이지 이동 등의 행동이 발생할 경우 웹 페이지 전체가 바뀌는게 아닌, 처음 로딩된 페이지 중에서 변경이 필요한 부분만 바뀜

그로 인해 페이지 전환 속도가 굉장히 빠르고, 이미 로딩이 완료된걸 서버로부터 다시 받아올 필요가 없기 때문에 서버의 자원을 효율적으로 사용할 수 있음

* 웹에서 이동 가능한 모든 페이지에 대한 파일을 받아와서 js에 의해 자체적으로 페이지가 로딩됨

Vue.js 개요

• Vue.js 장점

1. React에 비해 러닝커브가 낮음
2. HTML, CSS, JavaScript를 나눠서 코드를 작성할 수 있음
 - 한국의 개발 문화 특성상 퍼블리셔와의 협업이 좋음
 - HTML, CSS 코드를 그대로 사용 가능하기 때문에 마이그레이션이 쉬움
3. Angular의 장점(데이터바인딩)과 React의 장점(가상 돔)을 모두 가짐
4. Angular, React와 속도 및 성능을 비교했을 때 가장 높은 수치를 보임
 - 가볍고 빠름

• 마이그레이션(Migration)

현재의 시스템에서 다른 시스템으로 이동하는 작업

ex. jQuery를 사용하던 시스템에서 Vuejs로 변경하는 작업 등

Vue.js 개요

DOM 조작 시간 비교

Duration in milliseconds \pm 95% confid

Name Duration for...	vue-v3.2.1	angular-v12.0.1	react-v17.0.1
Implementation notes			
create rows creating 1,000 rows	105.8 \pm 1.8 (1.00)	123.1 \pm 1.3 (1.16)	130.1 \pm 2.7 (1.23)
replace all rows updating all 1,000 rows (5 warmup runs).	102.0 \pm 1.6 (1.00)	114.3 \pm 1.1 (1.12)	114.7 \pm 1.1 (1.12)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16x CPU slowdown.	200.6 \pm 5.0 (1.12)	179.0 \pm 4.6 (1.00)	223.6 \pm 5.0 (1.25)
select row highlighting a selected row. (no warmup runs). 16x CPU slowdown.	29.8 \pm 0.7 (1.00)	74.1 \pm 2.3 (2.49)	109.9 \pm 3.3 (3.68)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	50.7 \pm 0.7 (1.00)	365.4 \pm 2.4 (7.20)	364.9 \pm 2.8 (7.19)
remove row removing one row. (5 warmup runs).	22.3 \pm 0.4 (1.00)	21.1 \pm 0.2 (1.00)	23.6 \pm 0.3 (1.12)
create many rows creating 10,000 rows	970.9 \pm 7.8 (1.00)	1,043.0 \pm 7.4 (1.07)	1,382.9 \pm 23.7 (1.42)
append rows to large table appending 1,000 to a table of 10,000 rows. 2x CPU slowdown.	202.4 \pm 3.7 (1.00)	246.3 \pm 4.3 (1.22)	263.7 \pm 3.5 (1.30)
clear rows clearing a table with 1,000 rows. 8x CPU slowdown.	118.0 \pm 3.6 (1.00)	202.1 \pm 2.3 (1.71)	133.7 \pm 1.6 (1.12)
geometric mean of all factors in the table	1.02	1.55	1.68

시작 시간 비교

Startup metrics (lighthouse with mobi

Name	vue-v3.2.1	angular-v12.0.1	react-v17.0.1
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,104.9 \pm 0.6 (1.00)	2,444.1 \pm 1.7 (1.16)	2,580.3 \pm 2.1 (1.23)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	194.4 \pm 0.0 (1.00)	293.8 \pm 0.0 (1.51)	274.2 \pm 0.0 (1.41)
geometric mean of all factors in the table	1.00	1.32	1.31

메모리 할당 비교

Memory allocation in MBs \pm 95% confid

Please note that currently issue [#916](#) causes wro

Name	vue-v3.2.1	angular-v12.0.1	react-v17.0.1
ready memory Memory usage after page load.	1.3 (1.00)	1.9 (1.55)	1.3 (1.08)
run memory Memory usage after adding 1000 rows.	3.5 (1.00)	4.4 (1.26)	4.7 (1.32)
update each 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	3.9 (1.00)	4.8 (1.24)	5.5 (1.42)
replace 1k rows (5 cycles) Memory usage after clicking create 1000 rows 5 times	4.1 (1.00)	5.4 (1.32)	5.5 (1.33)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	2.7 (1.00)	3.8 (1.42)	3.3 (1.22)
geometric mean of all factors in the table	1.00	1.35	1.27

Vue.js 개요

• Vue.js 단점

1. 중국 개발자(Evan You)에 의해 개발되어 중국의 포지션이 큼
2. 일부 프로젝트에서 재정적 지원 X
3. React에 비해 생태계가 작음
4. 모바일 지원이 부족함

Vue.js 개요

- **Vue.js2 CDN 사용**

HTML 파일의 head 안에 아래 스크립트 작성

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

Vue.js 기초

(선언적 렌더링)

Vue.js 기초

• 선언적 렌더링

템플릿 구문을 사용하여 데이터를 선언적으로 DOM에 렌더링할 수 있음

• 렌더링

서버로부터 HTML 파일을 전달받고 브라우저에 출력되는 과정

• 참고

1. 아래에서 설명되는 것들은 당장 다양한 기능에 대해 설명하지 않고 기초 이 후에 다시 나올 예정
2. 예시들에서는 Vue 인스턴스를 생성할 때 네이밍을 app으로 했지만 VM 인스턴스는 관례적으로 vm(ViewModel)으로 네이밍함

Vue.js 기초

• 참고

1. 아래에서 설명되는 것들은 당장 다양한 기능에 대해 설명하지 않고 기초 이 후에 다시 나올 예정
2. 예시들에서는 Vue 인스턴스를 생성할 때 네이밍을 app으로 했지만 VM 인스턴스는 관례적으로 vm(ViewModel)으로 네이밍함

Vue.js 기초

보간법(Interpolation) 이라고 불림

```
<body>
  <div id="app">
    {{ message }}
  </div>
</body>
</html>
```

아래 data에서 저장된 내용을 출력
태그 안의 내용에 data값을 쓰고 싶을 땐 {{ }} 을 사용함

Vue 인스턴스 생성

```
<script>
  var app = new Vue({
    el: '#app',
    data: {
      message: 'Hello, Vue!'
    }
  })
</script>
```

Vue 인스턴스에 연결할 HTML 요소를 선택

Vue 인스턴스에 데이터를 가지는 블록 (변수)

'Hello, Vue!' 라는 문자열을 message 변수에 저장

Vue.js 기초 (디렉티브)

디렉티브

- **디렉티브(Directive)**

HTML 요소에 추가적인 동작을 적용하기 위한 특별한 속성으로써 Directive는 v- 접두사가 붙어 있음

디렉티브

요소의 title 속성에 message 데이터를 넣음

```
<div id="app-2">
  <!-- v-bind 생략 가능 -->
  <!-- <span v-bind:title="message"> -->
  <span :title="message">
    마우스를 잠시동안 올려보세요.
  </span>
</div>
```

v-bind를 생략하여 사용 가능

```
var app2 = new Vue({
  el: '#app-2',
  data: {
    message: 'v-bind가 동작 했습니다'
  }
})
```

디렉티브

• 디렉티브의 종류

종류	설명
v-if	조건이 참(true)일 경우에만 렌더링을 진행
v-show	조건이 참(true)일 경우에만 렌더링을 하되, display: none 처리 진행
v-for	반복문을 실행하여 렌더링을 진행
v-bind 또는 :	요소의 속성(title, src, class)을 지정하고 Vue 인스턴스의 데이터를 사용
v-on 또는 @	이벤트 리스너를 호출하여 사용
v-model	사용자의 입력을 받는 요소의 값을 뷰 인스턴스의 데이터와 연결됨(양방향 바인딩)
v-text	innerText 속성에 연결되며 문자열을 HTML 인코딩하여 문자열만 출력
v-html	v-text와 비슷하지만 HTML 인코딩을 하지 않아 HTML 태그가 적용된 내용이 출력 * XSS 공격 등 보안에 취약하여 사용을 권장하지 않음
v-cloak	vue 인스턴스가 완전히 로드되기 이전에 감춰주는 역할을 하며 일반적으로 CSS와 사용 [v-clock] { display: none; }
v-pre	해당 태그와 자식 태그들은 vue가 무시하고 건너 뛴

디렉티브

• 원시 HTML (v-html)

v-html 디렉티브를 사용하여 실제 HTML 자체를 출력할 수 있음
→ HTML 인코딩을 하지 않기 때문
→ XSS 공격에 취약하므로 되도록 사용하지 않는것을 권장
(기초적인 보안 설정은 되어있음)

```
let vm = new Vue({  
  el : "#divTag",  
  data : {  
    message : '<p>메시지!!!</p>'  
  }  
})
```

→

```
<div id="divTag">  
  {{ message }}  
  <p v-html="message"></p>  
</div>
```

→ <p>메시지!!!</p>
→ 메시지!!!

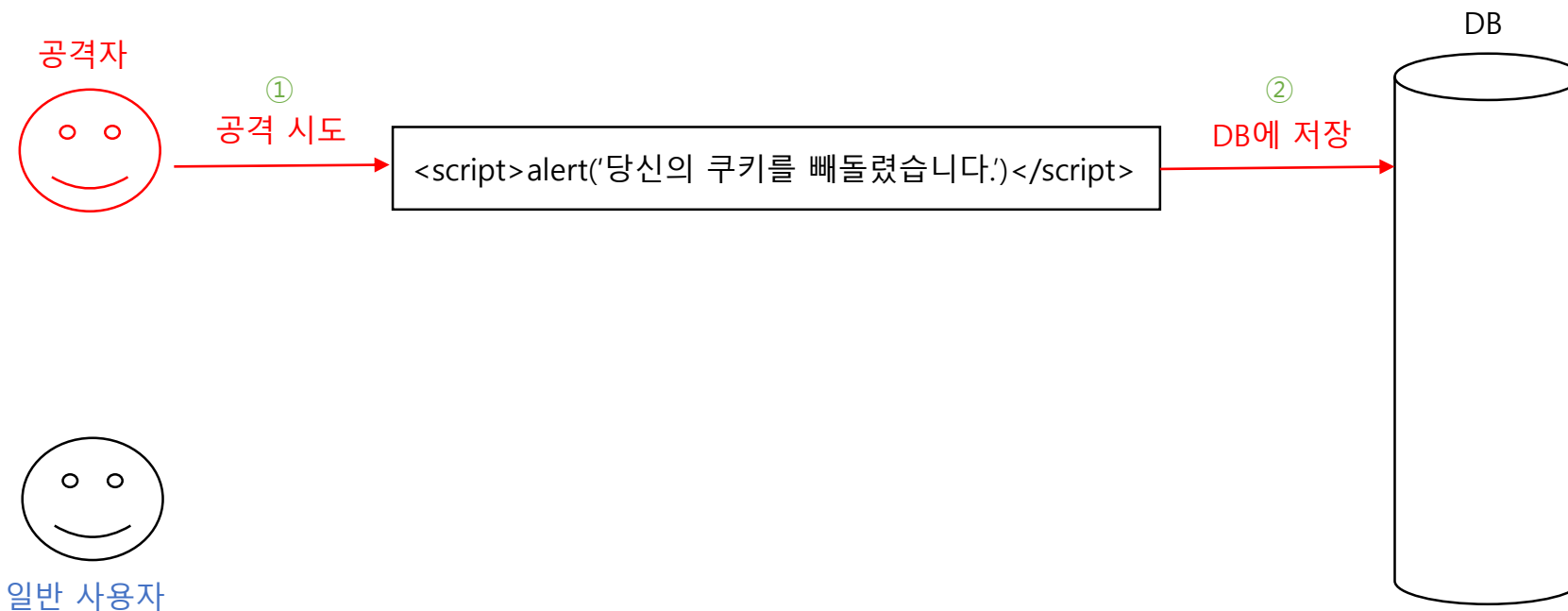


웹사이트에서 임의의 HTML을 동적으로 렌더링하려면 XSS 취약점으로 쉽게 이어질 수 있으므로 매우 위험할 가능성이 있습니다. 신뢰할 수 있는 콘텐츠에서만 HTML 보간을 사용하고 사용자가 제공한 콘텐츠에서는 절대 사용하면 안됩니다.

디렉티브

• 참고 - XSS

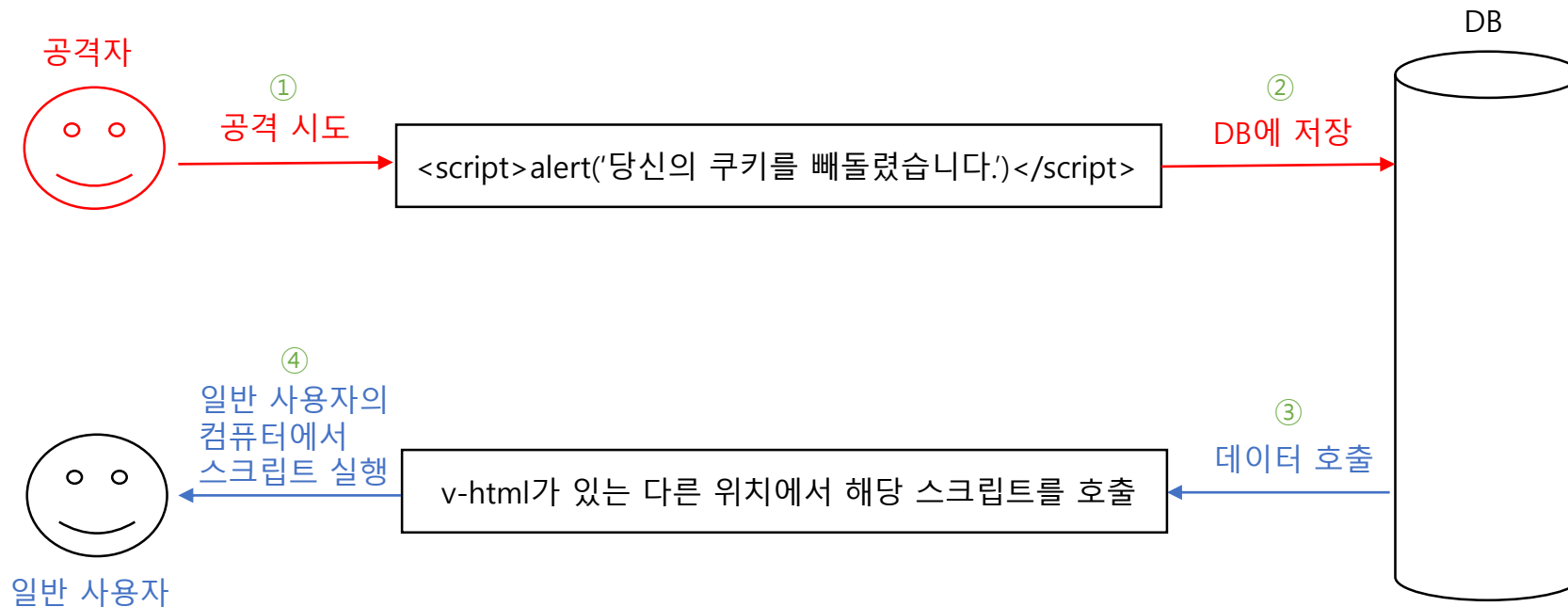
XSS(Cross-Site Scripting)은 SQL injection과 함께 가장 기초적인 웹 해킹 방법 중 하나이며, 권한이 없는 사용자가 웹 사이트에 스크립트를 삽입하는 공격 기법



디렉티브

• 참고 - XSS

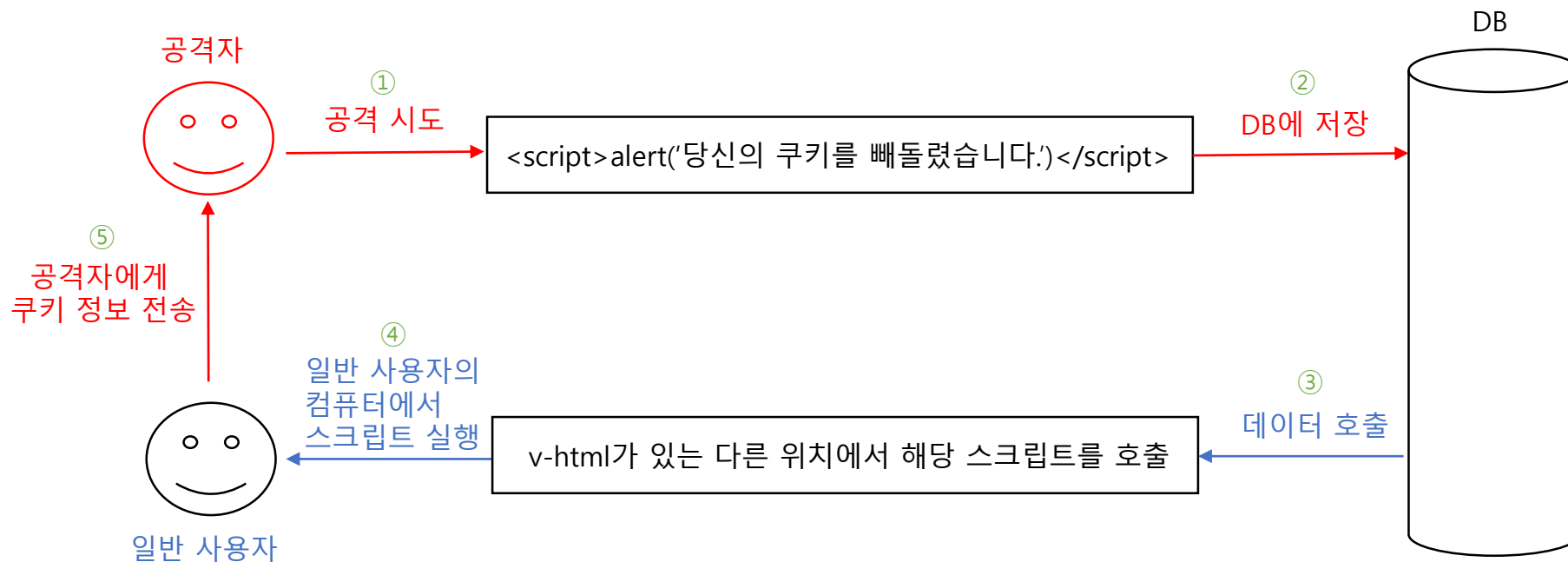
XSS(Cross-Site Scripting)은 SQL injection과 함께 가장 기초적인 웹 해킹 방법 중 하나이며, 권한이 없는 사용자가 웹 사이트에 스크립트를 삽입하는 공격 기법



디렉티브

• 참고 - XSS

XSS(Cross-Site Scripting)은 SQL injection과 함께 가장 기초적인 웹 해킹 방법 중 하나이며, 권한이 없는 사용자가 웹 사이트에 스크립트를 삽입하는 공격 기법



Vue.js 기초

(조건문, 반복문, 사용자 입력 핸들링)

Vue.js 기초

- 조건문(v-if)

엘리먼트가 표시 될지에 대한 여부를 제어할 수 있는 기능으로써 v-if가 해당함

```
<div id="app">
  <p v-if="isTrue">트루 메시지는 보입니다!</p>
</div>
```

```
<!-- 조건문 -->
<script>
  var app = new Vue({
    el: '#app',
    data: {
      isTrue: true
    }
  })
</script>
```

Vue.js 기초

• 반복문(v-for)

배열의 데이터를 바인딩하여 값을 출력함

```
<div id="app">  messageList의 데이터가 한번씩  
  <ul>          message 변수에 들어가며 반복  
    <li v-for="message in messageList">  
      {{ message.text }}  
    </li>  
  </ul>  
</div>
```

```
var app = new Vue({  
  el: '#app',  
  data: {  
    messageList: [  
      {text: 'message1'},  
      {text: 'message2'},  
      {text: 'message3'}  
    ]  
  }  
})
```

배열명

배열 키

배열 값

Vue.js 기초

• 사용자 입력 핸들링(v-on)

이벤트 리스너를 호출하여 사용 가능

```
<div id="app">
  <p>{{ message }}</p>
  <button v-on:click="reverseMessage">
    버튼을 눌러주세요
  </button>
</div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    message: '클릭 되었어요!'
  },
  methods: {
    reverseMessage: function () {
      this.message = this.message.split('').reverse().join('')
    }
  }
})
```

Vue.js 기초

• 설명

1. methods : vue 인스턴스 안에 함수를 동작시킬 때 필요함
2. reverseMessage라는 변수에 익명 함수를 사용
3. methods 안에서 data에 있는 변수를 찾아갈 때는 this를 사용해야 함
4. this.message.split('').reverse().join('')
 - this.message 나의 data에 있는 message 변수
 - split('') : 각각의 문자가 배열의 요소로 분리
 - reverse() : 배열의 순서를 역으로 뒤집음
 - join('') : 배열의 요소를 하나의 문자열로 결합

```
<div id="app">
  <p>{{ message }}</p>
  <button v-on:click="reverseMessage">
    버튼을 눌러주세요
  </button>
</div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    message: '클릭 되었어요!'
  },
  methods: {
    reverseMessage: function () {
      this.message = this.message.split('').reverse().join('')
    }
  }
})
```

Vue.js 기초

- 사용자 입력 핸들링(v-model)

사용자의 입력과 앱의 상태를 양방향으로 바인딩

```
<div id="app">
  <p>{{ message }}</p>
  <input v-model="message">
</div>
```

```
<!-- 이벤트 입력 핸들링 -->
<script>
  var app = new Vue({
    el: '#app',
    data: {
      message: '입력해보세요!'
    }
  })
</script>
```


Vue.js 기초 (Component)

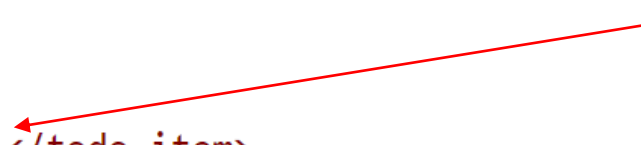
Component

• 컴포넌트(Component)

Vue.js의 핵심 기능 중 하나로써 재사용 가능한 조각으로 나누어 관리하여 재사용성이 증가하고, 각자 나누어 관리하기 때문에 코드가 몰리는 현상이 사라져 가독성과 유지보수성이 좋아짐

```
<div id="app">  
  <todo-item></todo-item>  
</div>
```

```
Vue.component('todo-item', {  
  template: '<li>할일 항목 하나입니다.</li>'  
});  
  
var app = new Vue({  
  el: '#app'  
})
```



Component

• 컴포넌트 예시2

```
<div id="app">
  <ol>
    <todo-item v-for="item in todoList"
      :todo="item"
      :key="item.idx">
    </todo-item>
  </ol>
</div>
```

컴포넌트 안에있는
todoList 가져옴

컴포넌트명

```
Vue.component('todo-item', {
  props: ['todo'], 부모 컴포넌트 → 자식 컴포넌트
  template: '<li>{{ todo.text }}</li>'
})
let app = new Vue({
  el: '#app',
  data: {
    todoList: [
      { idx: 0, text: '밥먹기' },
      { idx: 1, text: '잠자기' },
      { idx: 2, text: '쉬기' }
    ]
  }
})
```

Vue.js 기초 (Computed)

Computed

- **Computed**

Vue 컴포넌트 내에서 데이터를 가공하거나 연산한 결과를 반환하며
특정 계산이 필요한 로직의 경우 Computed를 사용

Computed

- Computed를 사용하지 않을 경우

아래와 같이 사용할 경우 해당 코드가 많아질수록 유지 보수에 어려움이 생김
→ 수정 사항이 생길 시 하나 하나 다 찾아서 수정해야하기 때문

```
<div id="original">  
  message.split('').reverse().join('')  
</div>
```

Computed

• Computed를 사용할 경우

Computed를 만들어주고 필요한 곳에서 꺼내씀으로써 가독성과 유지보수성 향상
→ 자바의 getter와 비슷한 역할을 수행함
→ 그 외에도 반복되는 HTML 코드를 줄여서 가독성과 유지보수성이 향상됨

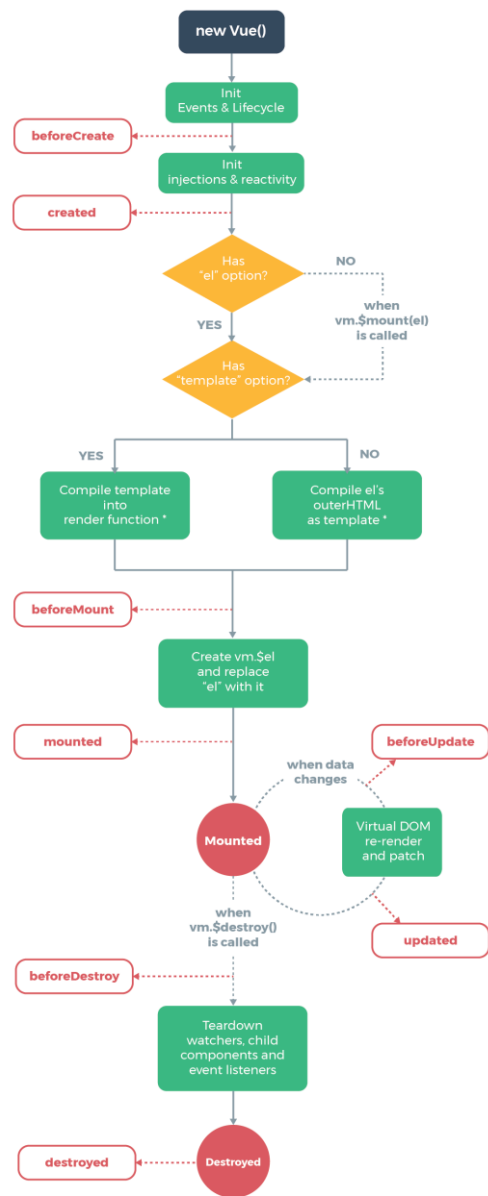
```
<div id="example">
  <p>원본 메시지: "{{ message }}"</p>
  <p>역순 메시지: "{{ reversedMessage }}"</p>
</div>
```

```
var vm = new Vue({
  el: '#example',
  data: {
    message: '안녕하세요'
  },
  computed: {
    // 계산된 getter
    reversedMessage: function () {
      return this.message.split('').reverse().join('')
    }
  }
})
```

Vue.js 기초

(라이프사이클)

라이프사이클



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