

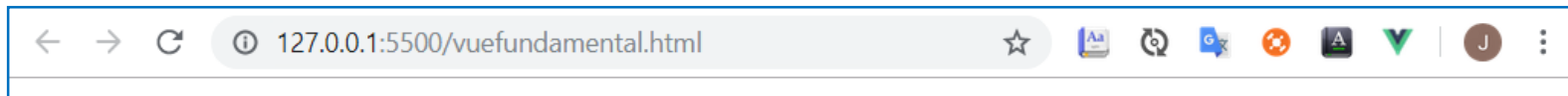
# Vue.js Fundamentals

**Bok, Jong Soon**  
**javaexpert@nate.com**  
**<https://github.com/swacademy/Vue.js>**

# Vue Code Template

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <meta http-equiv="X-UA-Compatible" content="ie=edge">
7      <title>Vue.js Code Template</title>
8      <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9  </head>
10 <body>
11
12 </body>
13 </html>
```

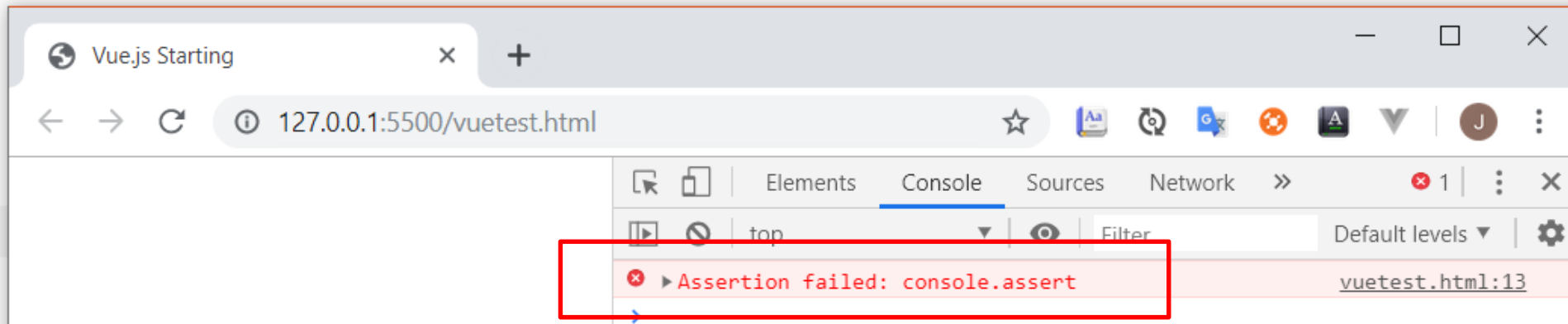
# Vue.js Fundamental



# Vue.js 도입하기

## ■ Vue.js Asserting

```
7   <title>Vue.js Starting</title>
8   <!-- <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script> -->
9 </head>
10 <body>
11   <script>
12     //Loading 및 Vue가 전역 변수로 정의됐는지 확인하기
13     console.assert(typeof Vue !== 'undefined');
14   </script>
15 </body>
16 </html>
```



# Vue.js 도입하기 (Cont.)

```
8     <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9 </head>
10 <body>
11     <div id="app"> {{message}} </div>
12     <script>
13         //Loading 및 Vue가 전역 변수로 정의됐는지 확인하기
14         console.assert(typeof Vue !== 'undefined');
15         let options = {
16             el : '#app',
17             data : {
18                 message : 'Hello, Vue.js!!!'
19             }
20         }
21         new Vue(options);
22     </script>
23 </body>
24 </html>
```

Vue.js Starting

← → ↻ ⓘ 127.0.0.1:5500/vuetest.html

Hello, Vue.js!!!

# Vue Instance

- Vue Application은 Vue 함수를 사용하여 새로운 Vue Instance를 만드는 것부터 시작한다.
- Vue로 화면을 개발하기 위해 필수적으로 생성해야 하는 기본 단위이다.
- Vue 객체 생성은 **new** 연산자를 사용한다.

```
new Vue({  
  ...  
});
```

- 이처럼 생성한 Vue 객체는 Vue.js 프로그램의 최상위 객체로 동작한다.
- 이 객체를 일반적으로 Vue Instance라고 또는 *Root Component*라고 부른다.

# Vue Instance (Cont.)

## ■ Bootstrapping

- Vue.js는 Vue Template과 Instance의 속성을 Binding하고 DOM Event를 Vue Method로 전달하도록 하는 등 Instance를 생성할 때 필요한 작업

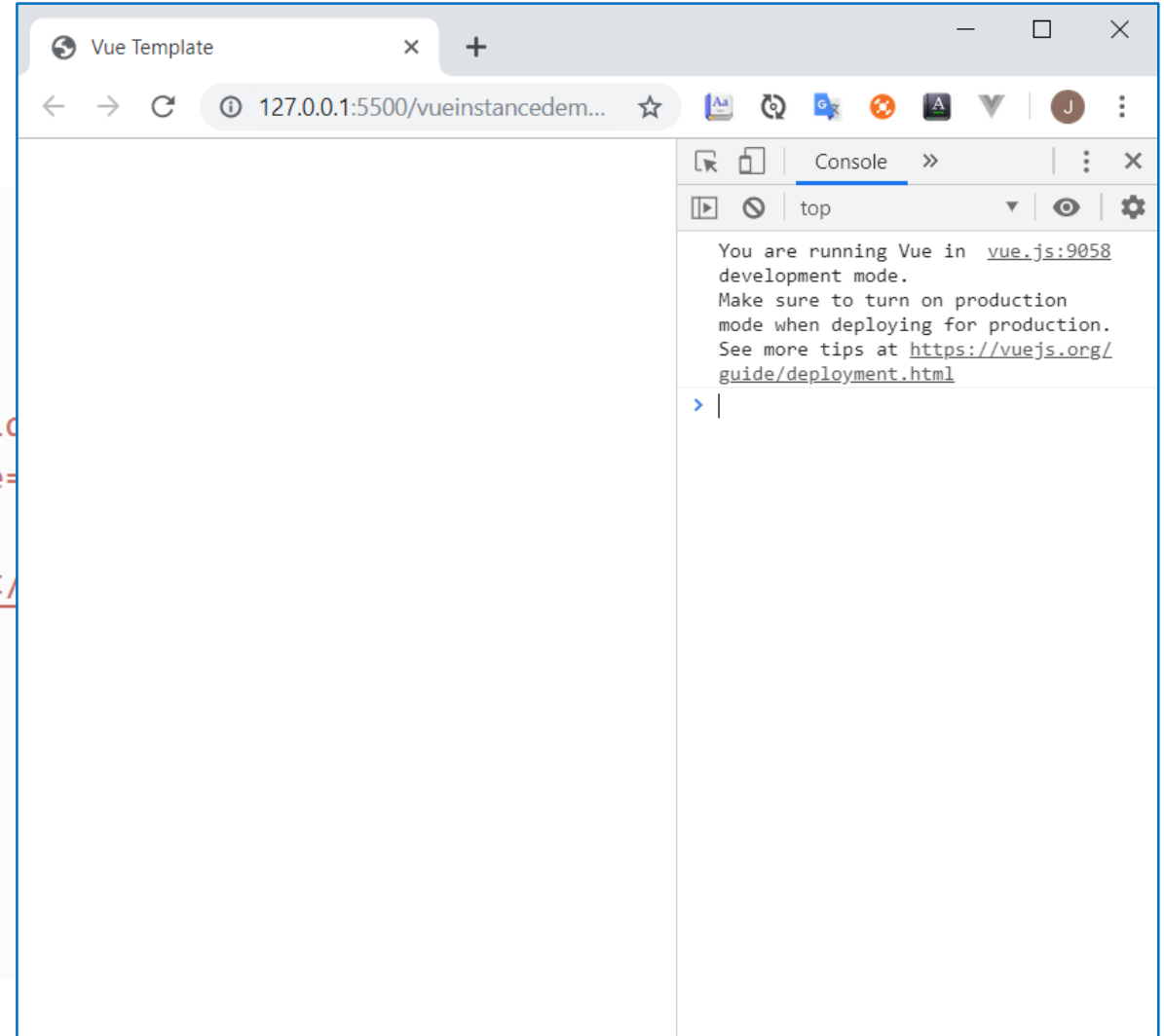
```
var vm = new Vue({  
    //Option 이나 Component 설정 등.  
});
```

- Vue **vm**의 변수 이름은 자유롭게 지정할 수 있지만 관례로 **vm**으로 지정한다.

# Vue Instance (Cont.)

## ■ Vue instance

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-wid
6   <meta http-equiv="X-UA-Compatible" content="ie=
7   <title>Vue.js Code Template</title>
8   <script src="https://unpkg.com/vue@2.6.10/dist/
9 </head>
10 <body>
11   <script>
12     var vm = new Vue();
13   </script>
14 </body>
15 </html>
```

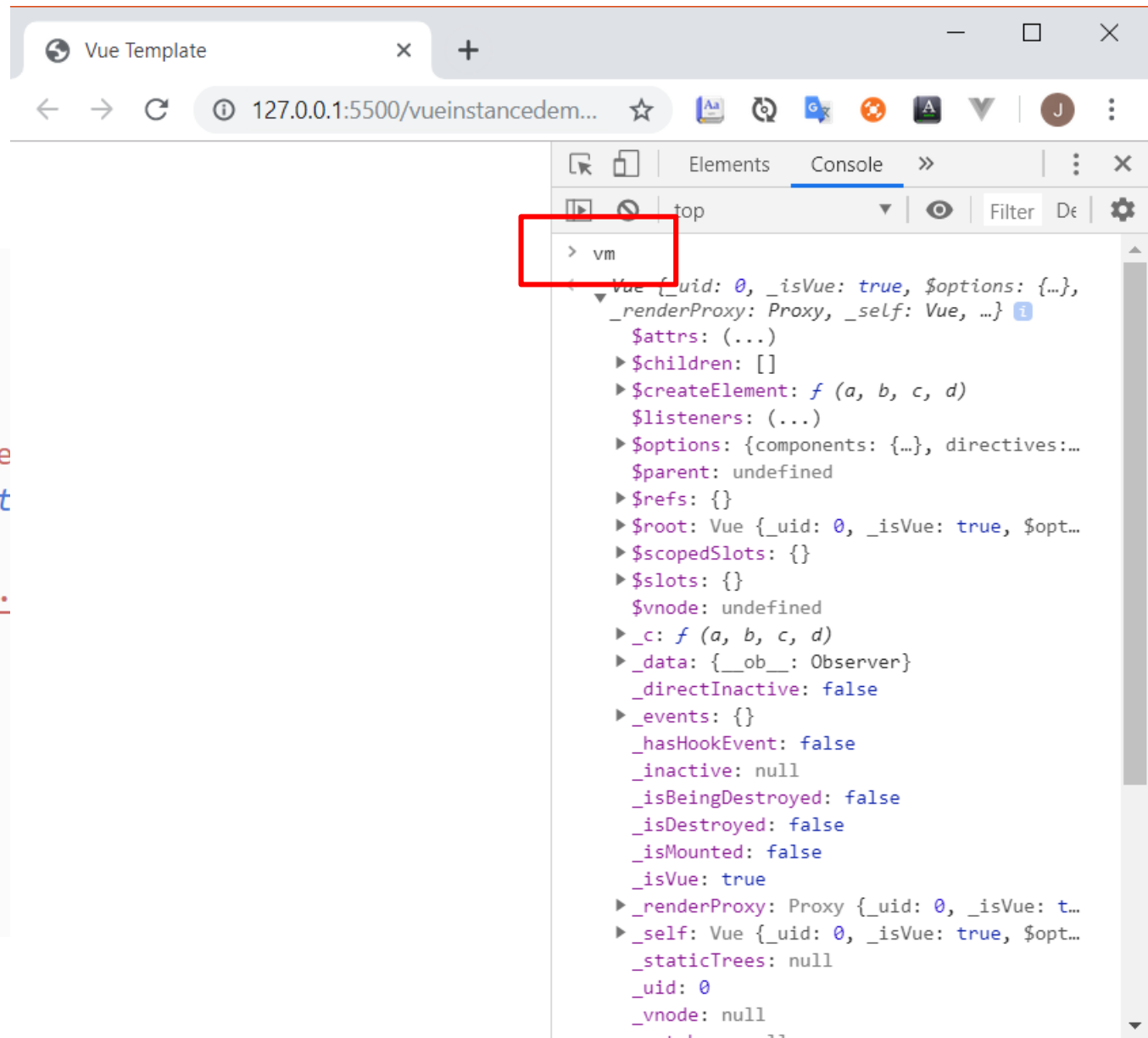




# Vue Instance (Cont.)

## ■ Vue instance

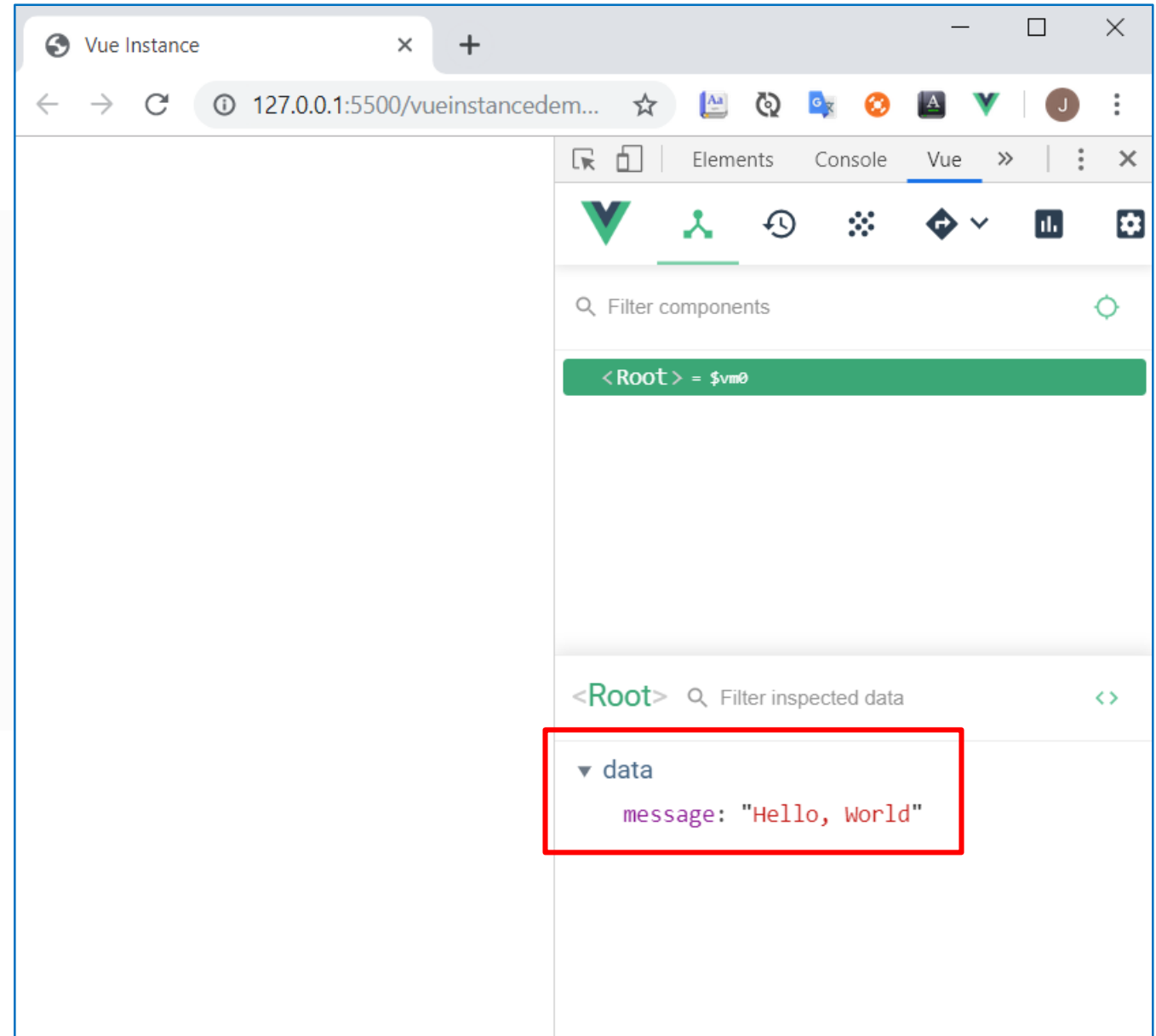
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <title>Vue.js Code Template</title>
8   <script src="https://unpkg.com/vue@2.6.12"></script>
9 </head>
10 <body>
11   <script>
12     var vm = new Vue();
13   </script>
14 </body>
15 </html>
```



# Vue Instance (Cont.)

## ■ Vue instance

```
<div id="app"></div>
<script>
  var vm = new Vue({
    el : '#app',
    data : {
      message : 'Hello, World'
    }
  });
</script>
```



## Vue Instance (Cont.)

- Instance에서 사용할 수 있는 속성과 API
  - **el** : Vue instance가 mount된 요소
  - **template** : 화면에 표시할 요소(HTML, CSS 등)
  - **data** : UI 상태 / 데이터
  - **methods** : 화면의 동작과 event logic을 제어하는 method
  - **created** : Vue의 Lifecycle과 관련된 속성
  - **watch** : data에서 정의한 속성이 변화했을 때 추가 동작을 수행할 수 있게 정의하는 속성
  - **filters** : data를 문자열로 formatting
  - **computed** : data로부터 파생되는 값

```
<script>
  var vm = new Vue({
    el : ,
    template : ,
    data : ,
    methods: ,
    created : ,
    watch : ,
  });
</script>
```

# Vue Component

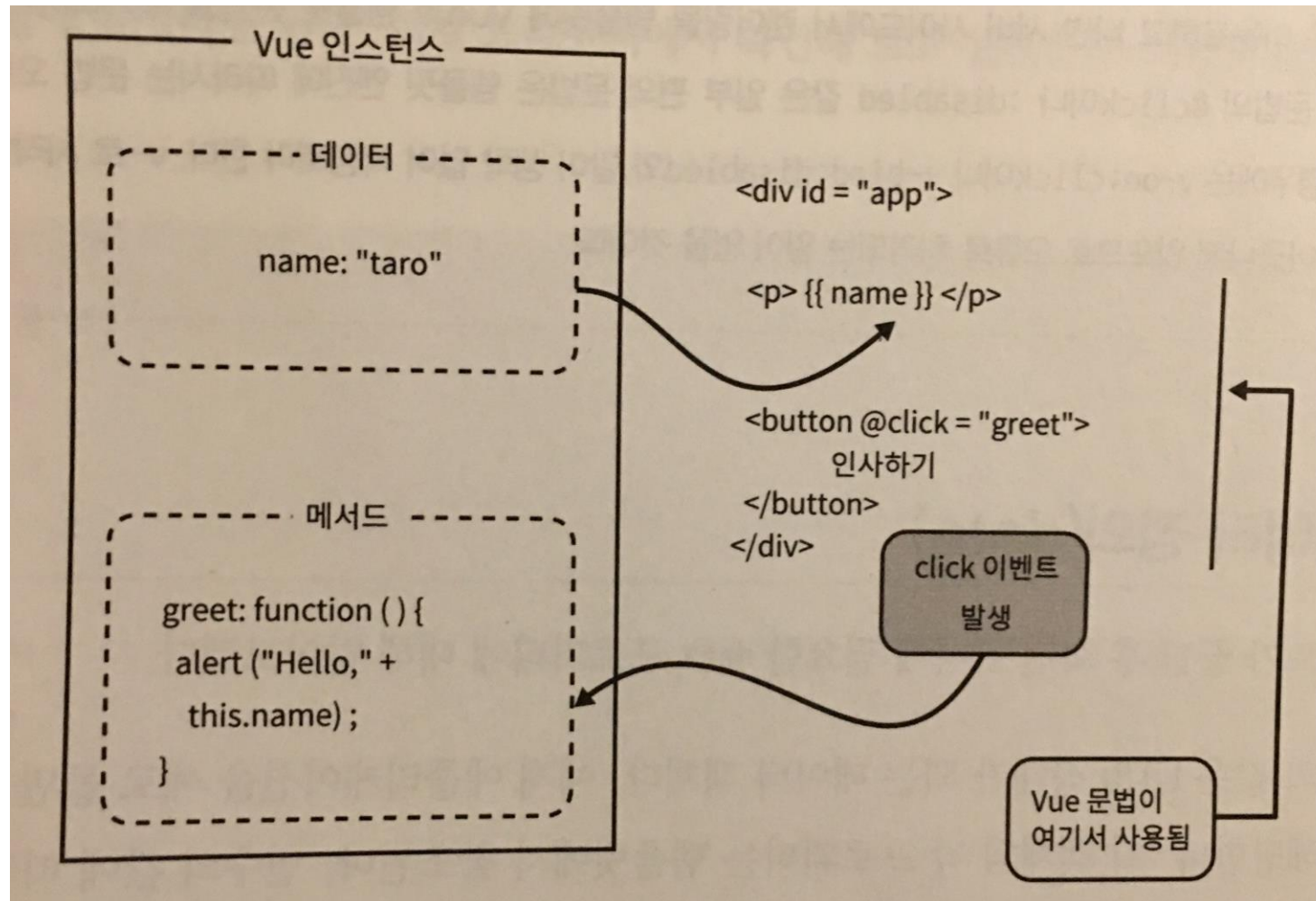
- Programming 에서 Code를 적절한 크기나 역할로 분할하기 위해 함수나 method를 사용하듯이 Vue에서도 Vue instance를 분할할 수 있다.
- 이렇게 Vue instance를 분할한 단위
- **Vue.component**로 전역 component 생성
- Vue instance의 **component** method로 지역 component 생성

# Vue Instance Mount하기

- Vue.js를 사용하려면 생성된 Vue instance를 DOM 요소에 mount 해야 한다.
- Mount란 기본 DOM 요소를 Vue.js가 생성하는 DOM 요소로 치환하는 것
- Mount 하는 방법
  - Vue instance를 생성할 때 options 객체에서 지정하거나
  - Method를 호출해서 나중에 지정 가능.
- Vue instance가 mount되면 mount된 요소 및 그 요소의 자손 node가 치환 된다.
- 따라서, Vue가 영향을 미치는 scope는 해당 요소 안으로 제한된다.

# Vue Instance Mount하기 (Cont.)

## ■ Vue instance mount의 적용 범위



# Vue Instance Mount하기 (Cont.)

## ■ Method를 이용한 mount

- `el` property를 정의하지 않고 `$mount` method 사용
- Vue instance를 생성한 후 언제든지 instance를 mount 가능.

```
var vm = new Vue({  
  ...  
});  
vm.$mount(el);
```

# UI Data 정의하기

- **data** property의 값은 UI의 상태가 되는 data 객체다.
- **data** property의 값은 Vue.js의 reactive system에 포함된다.
  - **data** property의 값이 변경될 때마다 Vue.js가 이를 자동으로 탐지해 표시 내용이 바뀌는 등의 처리 수행.
  - Vue instance를 생성할 때 **data** property를 전달하고 이를 이용해서 template의 내용을 출력하는 것이 Vue.js로 화면의 내용을 출력하는 기본방법.
- **data** property는 객체 혹은 함수를 값으로 가질 수 있다.

```
var vm = new Vue({  
  data : {  
    key : value  
  }  
});
```



# UI Data 정의하기 (Cont.)

## ■ product.js

```
1 var items = [  
2   {  
3     name: '연필',  
4     price: 300,  
5     quantity: 0  
6   },  
7   {  
8     name: '공책',  
9     price: 400,  
10    quantity: 0  
11  },  
12  {  
13    name: '지우개',  
14    price: 500,  
15    quantity: 0  
16  }  
17 ]
```

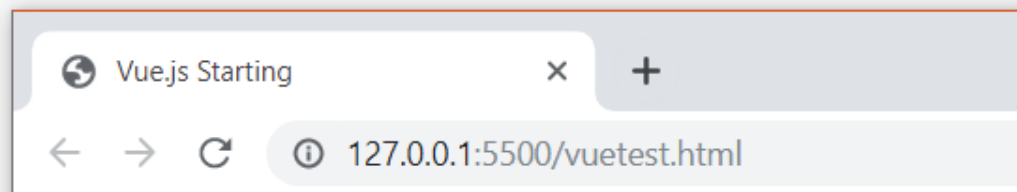
## ■ index.html

```
8   <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>  
9   <script src="product.js"></script>  
10  </head>  
11  <body>
```

```
38  <script>  
39    var vm = new Vue({  
40      el: '#app',  
41      data: {  
42        items: items  
43      },
```

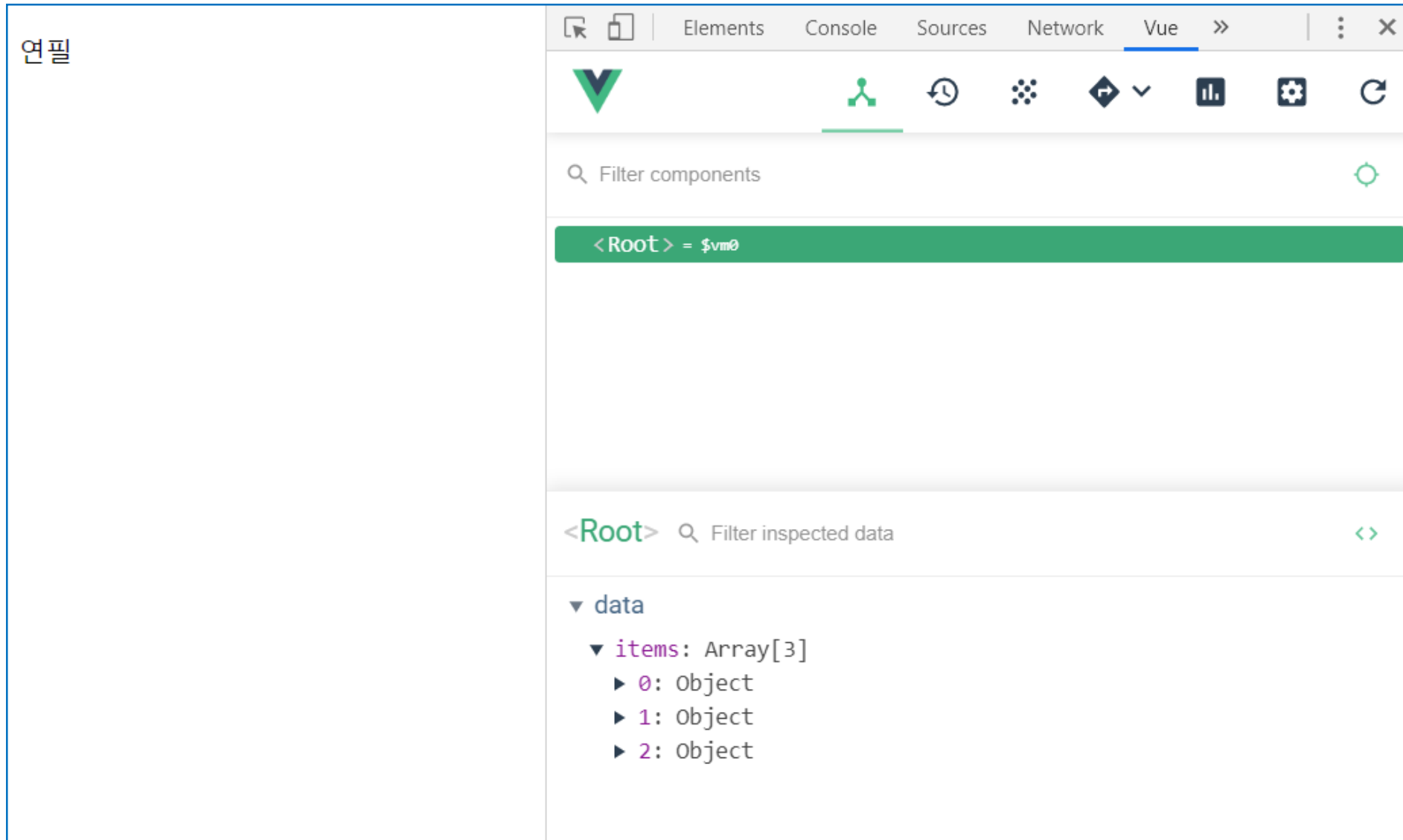
# UI Data 정의하기 (Cont.)

```
8      <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9      <script src="product.js"></script>
10     </head>
11     <body>
12         <div id="app">
13             <p> {{ items[0].name }} </p>
14         </div>
15         <script>
16             let vm = new Vue({
17                 el : '#app',
18                 data : {
19                     items : items
20                 }
21             })
22         </script>
23     </body>
24 </html>
```



연필

# UI Data 정의하기 (Cont.)



# Template 문법

- data 준비 후 data를 적용할 template을 작성해 보자.
- Vue.js는 몇 가지 template 문법 및 기능을 제공한다.
- template은 Vue instance의 data와 View(DOM tree)의 관계를 선언적으로 정의하는 역할을 한다.
  - data가 있다면 View의 내용이 결정된다는 의미.
  - data의 변경에 따라 view를 update하는 기능 → data binding
- Vue.js의 template 문법의 중요한 개념 2가지
  - Mustache 문법을 이용한 data 전개 **{{ }}**
  - Directive를 이용한 HTML 요소 확장

# Template 문법 (Cont.)

## ■ Text로 전개하기

- `{{ }}`
- `{{ }}` 사이에 **data** property에서 정의한 data나 **computed** property, method, **filters**를 참조할 수 있다.

`<p>{{ items[0][.name] }} : {{ items[0].price }} x {{ items[0].quantity }} </p>`

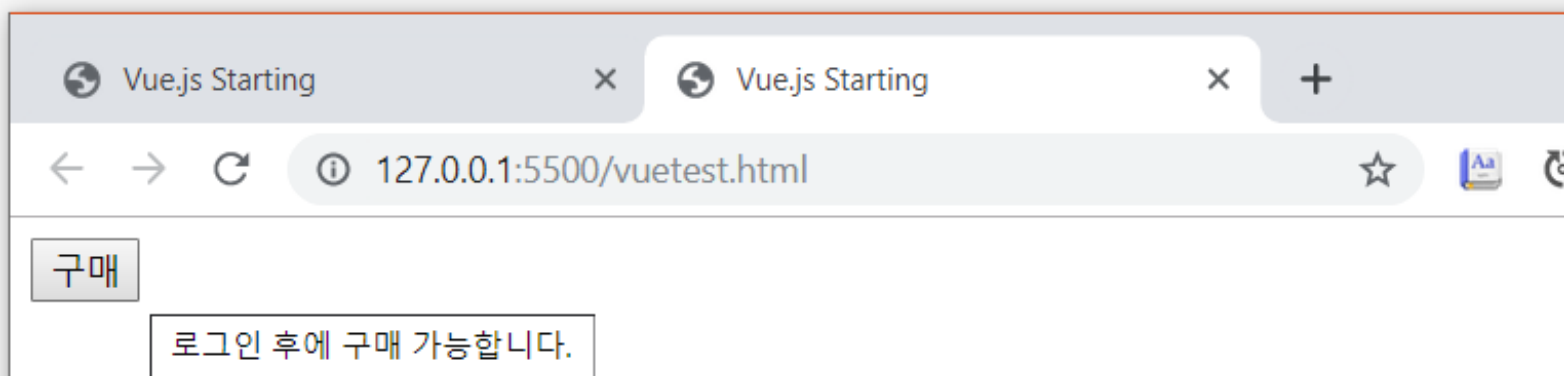
# Template 문법 (Cont.)

- 속성값 전개하기
  - DOM 요소의 속성에도 값을 전개할 수 있다.
  - Vue.js가 제공하는 directive인 **v-bind** 사용

**v-bind : 속성명 = “data를 전개한 속성값 ”**

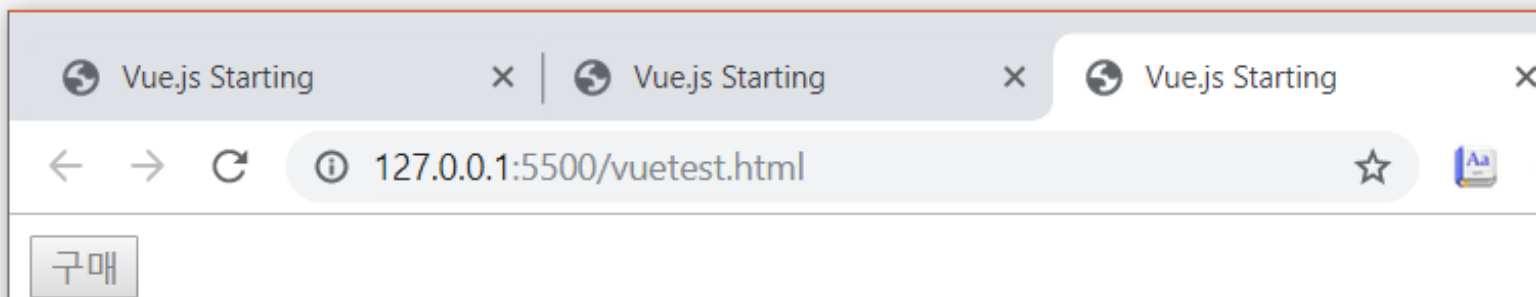
# Template 문법 (Cont.)

```
8     <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9 </head>
10 <body>
11     <button id="b-button" v-bind:title="loggedInButton">구매</button>
12     <script>
13         let vm = new Vue({
14             el : '#b-button',
15             data : {
16                 loggedInButton : '로그인 후에 구매 가능합니다.'
17             }
18         })
19     </script>
20 </body>
21 </html>
22
23
```



# Template 문법 (Cont.)

```
8      <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9  </head>
10 <body>
11     <button id="b-button" v-bind:disabled="!canBuy">구매</button>
12     <script>
13         let vm = new Vue({
14             el : '#b-button',
15             data : {
16                 canBuy : false
17             }
18         })
19     </script>
20 </body>
21 </html>
22
```





# Filters

- 일반적인 text formatting 기능 제공
- Vue 생성자 options 중 하나
- Mustache 문법을 이용한 text 전개와 v-bind directive의 속성값 표현식에  
만 사용 가능.
- 사용 예
  - Data 객체를 YYYY/mm/dd와 같은 format으로 변환
  - 숫자 0.5를 percent 단위('50%')로 변환
- Vue 생성자에서 options 중 filters에 인자 하나를 받는 함수 형태로 정의
- 이 인자는 나중에 filter가 받게 될 값이 된다.

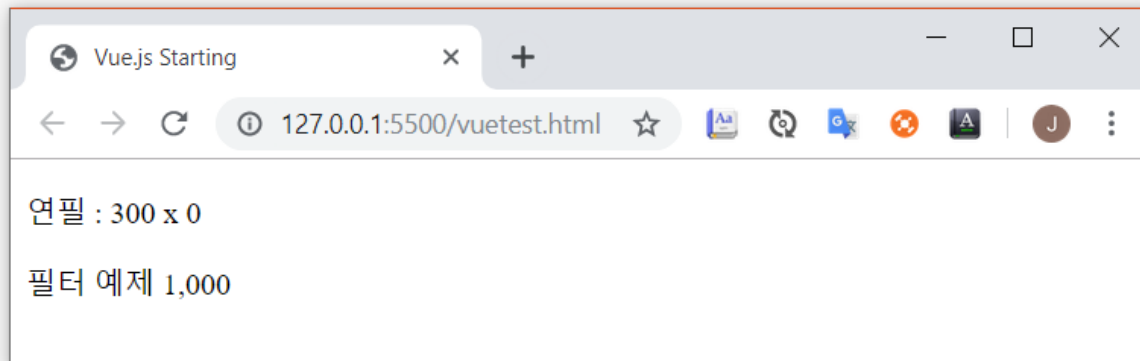
## Filters (Cont.)

- 정의해 둔 filter는 template에서 **{{ }}** 문법과 **|** (pipe) 연산자를 조합해서 사용한다.
- pipe 연산자(**|**) 왼쪽에 오는 값이 filter 의 인자가 된다.

```
filters : {  
    필터명 : function(value) {  
        return ...  
    }  
}  
  
{{ 값 | 필터명 }}
```

# Filters (Cont.)

```
8     <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9     <script src="product.js"></script>
10 </head>
11 <body>
12   <div id="app">
13     <p> {{ items[0].name }} : {{ items[0].price }} x {{ items[0].quantity }}</p>
14     <p> 필터 예제 {{ 1000 | numberWithDelimiter }} </p>
15   </div>
16   <script>
17     let vm = new Vue({
18       el : '#app',
19       data : {
20         items : items
21       },
22       filters : {
23         numberWithDelimiter : function(value){
24           if(!value) return '0';
25           return value.toString().replace(/(\d)(?=(\d{3})+$/g, '$1,');
26         }
27       }
28     })
29   </script>
```



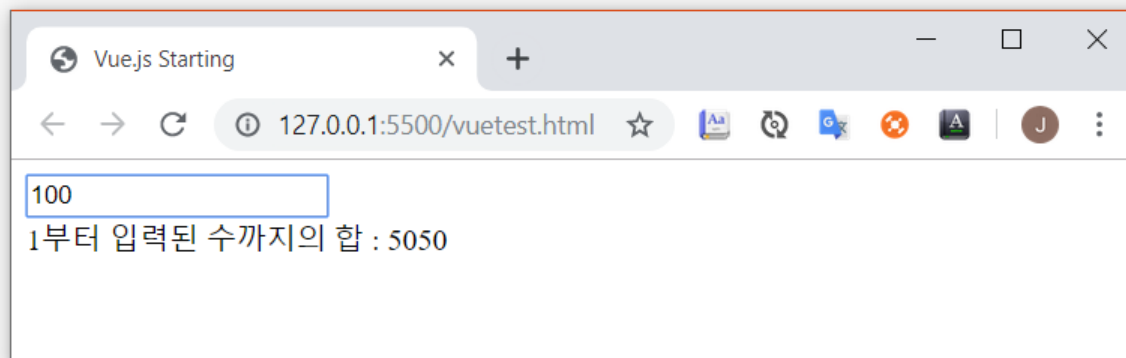
# computed

- 어떤 data에서 파생된 data를 property로 공개하는 기능
- Vue 생성자 options 중 하나
- 대체로 복잡한 식을 template에 나타내는 용도로 사용

```
new Vue({  
  ...  
  computed : {  
    //함수 형태로 구현함, 참조할 때는 property처럼 동작.  
    computed property 이름 : function(){  
      return ...  
    }  
  }  
});
```

# computed (Cont.)

```
8   <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9   </head>
10  <body>
11    <div id="app">
12      <input type="text" v-model="num" /><br />
13      1부터 입력된 수까지의 합 : <span> {{ sum }} </span>
14    </div>
15    <script>
16      let vm = new Vue({
17        el : '#app',
18        data : {
19          num : 0
20        },
21        computed : {
22          sum : function(){
23            let n = Number(this.num);
24            if (Number.isNaN(n) || n < 1) return 0;
25            return ((1 + n) * n) / 2;
26          }
27        }
28      })
29    </script>
```



## computed (Cont.)

### ■ **this** 참조하기

- **computed** property 참조할 때
- Method를 이용해서 data 객체와 **computed** property 참조할 때

■ 이 때 **this**가 가리키는 대상은 Vue instance 자신이다.

```
computed : {  
    someFunc : function(){  
        return this.item * 3;  
    }  
}
```

# Directive

- Vue.js는 표준 HTML에 독자적으로 정의한 속성을 추가해 이 속성값 표현식의 변화에 따라 DOM을 조작한다.
- 이러한 특별한 속성을 말한다.
- Directive로 사용되는 속성은 이름이 **v-**로 시작한다.
- Vue instance가 mount된 요소와 그 요소의 자손 요소에서만 사용 가능.
- Directive 속성은 JavaScript 표현식을 값으로 갖는다.

# Directive (Cont.)

## ■ 기본 directive

### ● v-text

#### ■ {{ }}

■ **innerText** 속성에 연결됨.

■ Tag 문자열을 HTML encoding하여 나타내기 때문에 화면에는 Tag 문자열이 그대로 나타남.

### ● v-html

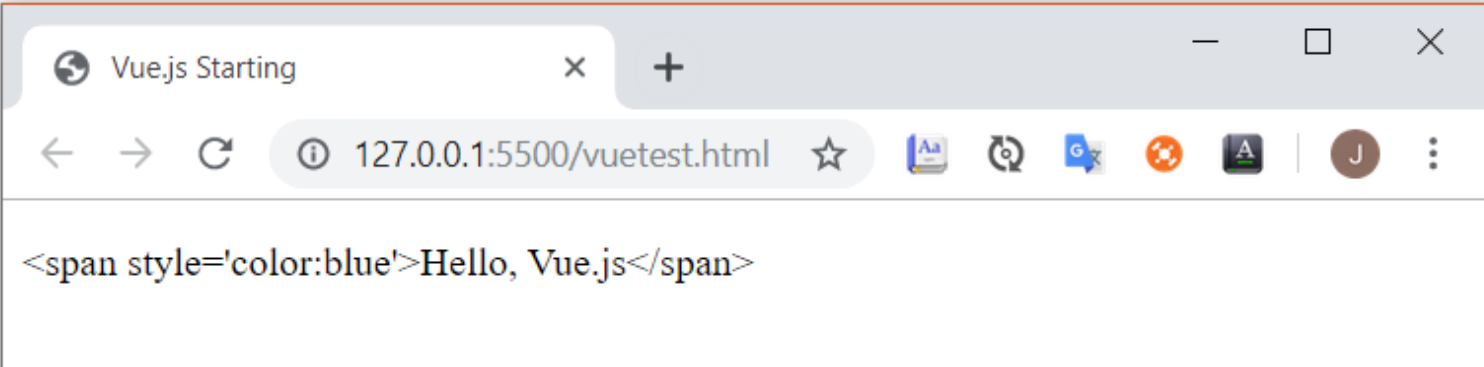
■ **innerHTML** 속성에 연결됨.

■ Tag 문자열을 parsing 하여 화면에 나타냄.



## Directive (Cont.)

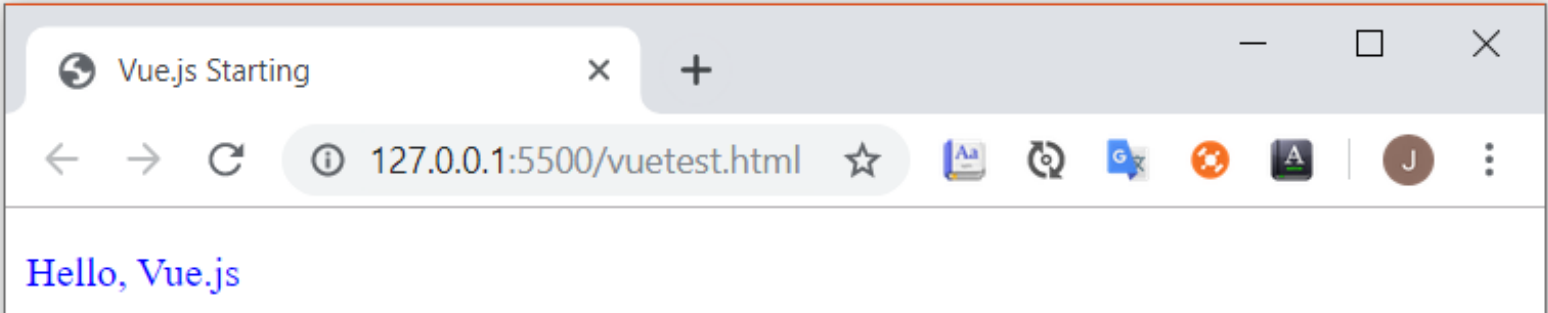
```
8     <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9 </head>
10 <body>
11     <div id="app">
12         <p v-text="message"></p>
13     </div>
14 <script>
15     let vm = new Vue({
16         el : '#app',
17         data : {
18             message : "<span style='color:blue'>Hello, Vue.js</span>"
19         },
20     })
21 </script>
22 </body>
23 </html>
24
```



The screenshot shows a web browser window titled "Vue.js Starting". The address bar displays the URL "127.0.0.1:5500/vuetest.html". The browser's developer tools or console area shows the rendered HTML output: `<span style='color:blue'>Hello, Vue.js</span>`. This output corresponds to the `message` property defined in the Vue instance's `data` object in the code above.

## Directive (Cont.)

```
8     <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9 </head>
10 <body>
11     <div id="app">
12         <p v-html="message"></p>
13     </div>
14     <script>
15         let vm = new Vue({
16             el : '#app',
17             data : {
18                 message : "<span style='color:blue'>Hello, Vue.js</span>"
19             },
20         })
21     </script>
22 </body>
23 </html>
```



# Directive (Cont.)

## ■ class와 style 연결하기

- **v-bind**
- class directive → **v-bind:class**
- style directive → **v-bind:style**
- class는 각 값을 공백으로 구분("classA classB")
- style은 세미콜론(;)으로 구분("color : red ; background : yellow")

## ■ v-bind 생략 표기법

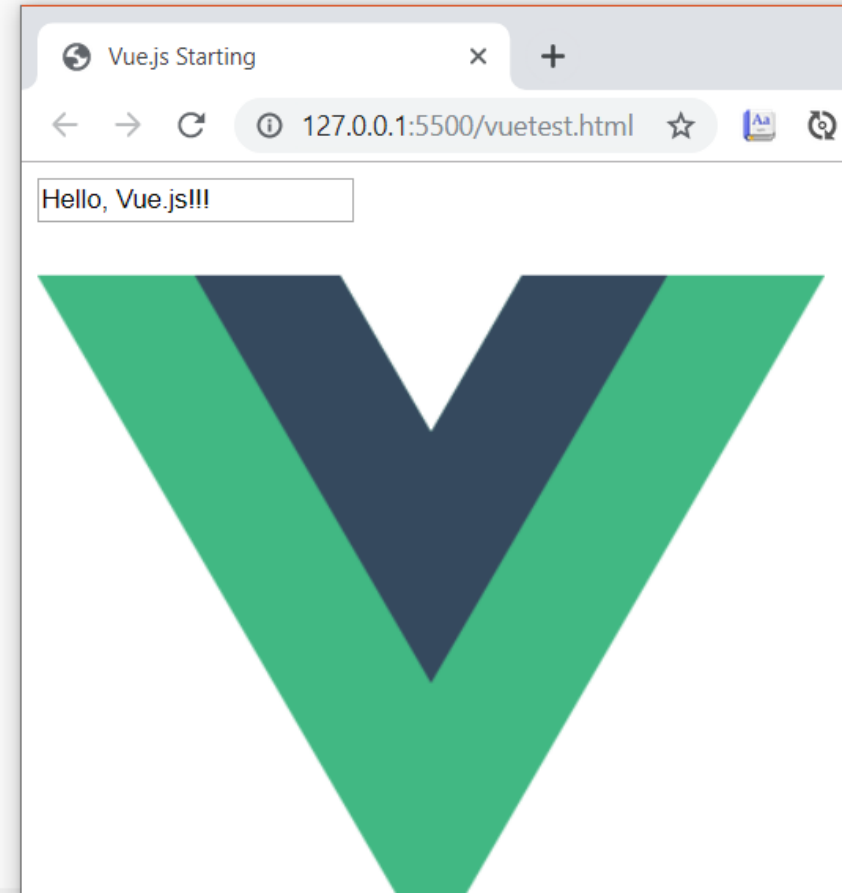
- v-bind는 가장 많이 사용하는 directive 중 하나.
- 따라서 간결한 표기를 위한 생략 표기법 지원

**v-bind:disabled → :disabled**

**v-bind:src → :src**

# Directive (Cont.)

```
8   <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9 </head>
10 <body>
11   <div id="app">
12     <input type="text" id="a" v-bind:value="message">
13     <br />
14     
15   </div>
16   <script>
17     let model = {
18       message : 'Hello, Vue.js!!!',
19       imagePath : 'https://vuejs.org/images/logo.png'
20     };
21     let vm = new Vue({
22       el : '#app',
23       data : model
24     })
25   </script>
26 </body>
```



# Directive (Cont.)

## ■ v-model

- 양방향 Data binding

```
<div id="app">
  <h1> {{ message }} </h1>
  <input v-model="message">
</div>

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

Greeting You!

Greeting You!

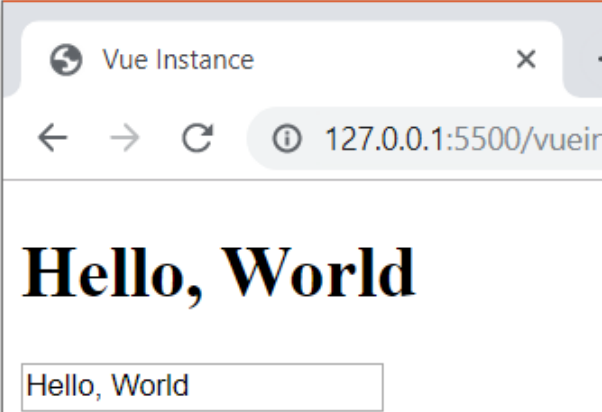
Vue.js가 좋네요.

Vue.js가 좋네요.

# Directive (Cont.)

## ■ jQuery와 비교

```
<script src="https://code.jquery.com/jquery-3.4.1.js"
  integrity="sha256-WpOohJOqMqqyKL9FccASB900KwACQJpFTUUBLTYOVvVU="
  crossorigin="anonymous"></script>
</head>
<body>
  <div id="app">
    <h1>Greeting You! </h1>
    <input id="message">
  </div>
  <script>
    $('#message').on('keyup', function(){
      var message = $('#message').val();
      $('#h1').text(message);
    });
  </script>
</body>
```

A screenshot of a web browser window. The title bar says "Vue Instance". The address bar shows "127.0.0.1:5500/vueir". The main content area displays "Hello, World" in a large, bold, black serif font. Below the text is a text input field containing the text "Hello, World".

Vue Instance

127.0.0.1:5500/vueir

# Hello, World

Hello, World

# Directive

## ■ v-model

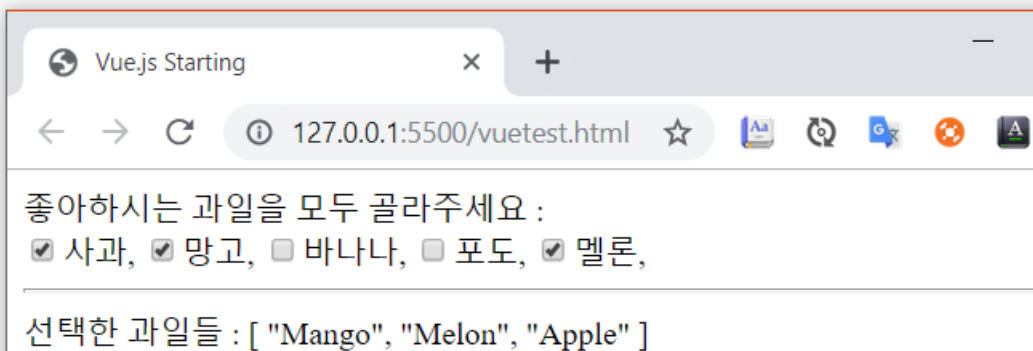
```
<div id="app">
  <div>좋아하시는 과일을 모두 골라주세요 : </div>
  <input type="checkbox" value="Apple" v-model="fruits">사과,
  <input type="checkbox" value="Mango" v-model="fruits">망고,
  <input type="checkbox" value="Banana" v-model="fruits">바나나,
  <input type="checkbox" value="Grape" v-model="fruits">포도,
  <input type="checkbox" value="Melon" v-model="fruits">멜론,
</div>
```

```
<hr />
```

```
<div id="app2">
  선택한 과일들 : <span v-html="fruits"></span>
</div>
```

```
<script>
```

```
  let model = {
    fruits : []
  };
  let vm = new Vue({
    el : '#app',
    data : model
  });
  let vm2 = new Vue({
    el : '#app2',
    data : model
  });
</script>
```



# Lab

- Hello, {{ name }}
- input tag 추가하고 이를 name과 Binding 하기
- User가 이름을 입력하거나 이름을 변경할 때마다 즉시 변경되어야 한다.

Hello Mr. Anderson

Enter your name:

```
{  
  "name": "Mr. Anderson"  
}
```



# Directive (Cont.)

## ■ 조건에 따른 rendering

- **v-show**, **v-if**, **v-else**, **v-else-if**
- template 에서 어떤 요소를 조건에 따라 표시하거나 표시하지 않도록
- 속성값의 표현식을 평가해서 값이 참이 됐을 때만 해당 요소 표시하고 거짓이면 표시하지 않는다.
- 유효성 검사 오류 메시지 표시, Login을 한 사용자에게만 보여줄 contents 등.
- **v-show**와 **v-if**의 차이점
  - DOM 요소 유지 여부
  - **v-else** 동반 여부
  - 평가 값이 빈번하게 바뀌는 경우 → **v-show**
  - 자주 바뀌지 않는 경우 → **v-if**

```
<div>  
  Hello <span v-if="show">Vue.js</span>  
</div>  
new Vue({  
  data : {  
    show : false  
  }  
})
```

# Lab.

- User는 이름과 함께 성별을 입력한다.
- User가 남성(male)일 경우 Hello Mister {{ name }}
- User가 여성(female)일 경우 Hello Miss {{ name }}
- 성별이 남성이나 여성이 아닌 경우 경고 메시지를 'So you can't decide. Fine!'이라고 출력한다.

Hello, Mister Anderson.

Enter your gender:

male

Enter your name:

Anderson

```
{  
  "gender": "male",  
  "name": "Anderson"  
}
```

# Lab.

- 개인 정보를 저장해서 출력해보자.
- 개인정보에는 이름(name), 몸무게(weight), 키(height), 좋아하는 음식(favoriteFood)가 있다.
- v-for를 이용해서 index : key = value 형식으로 출력해보자.

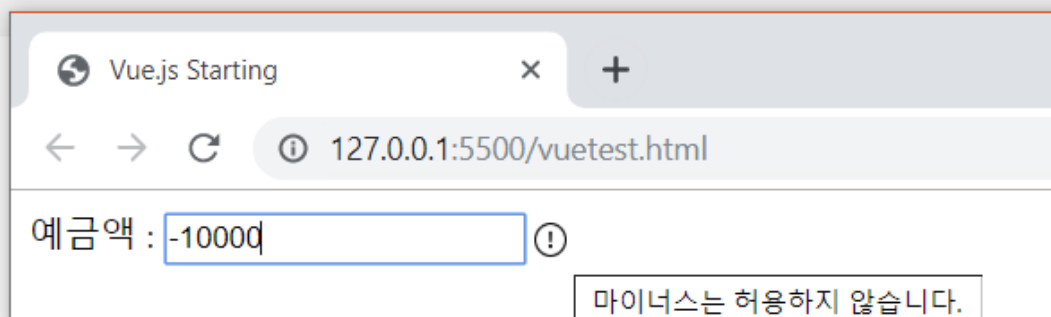
This is me!

0: name = Kostas
1: height = 1.73m
2: weigh = 65kg
3: eyeColor = brown
4: favoriteFood = Ntolmas

```
{  
  "quotes": {  
    "name": "Kostas",  
    "height": "1.73m",  
    "weigh": "65kg",  
    "eyeColor": "brown",  
    "favoriteFood": "Ntolmas"  
  }  
}
```

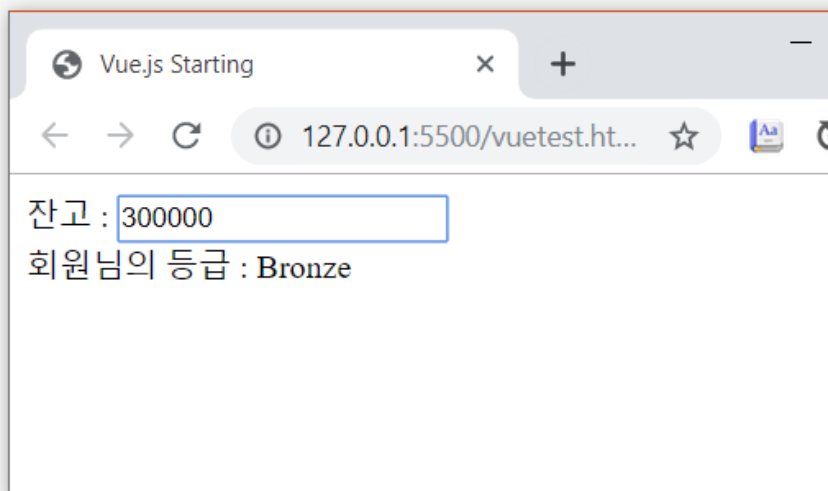
## Directive (Cont.)

```
8     <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9 </head>
10 <body>
11     <div id="app">
12         예금액 : <input type="text" v-model="amount" />
13         
16     </div>
17     <script>
18         let vm = new Vue({
19             el : '#app',
20             data : {
21                 amount : 0,
22                 imagePath : 'http://sample.bmaster.kro.kr/img/error.png'
23             }
24         });
25
26     </script>
```



# Directive (Cont.)

```
8   <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9 </head>
10 <body>
11   <div id="app">
12     잔고 : <input type="text" v-model="balance" />
13     <br />
14     회원님의 등급 :
15     <span v-if="balance >= 1000000">Gold</span>
16     <span v-else-if="balance >= 500000">Silver</span>
17     <span v-else-if="balance >= 200000">Bronze</span>
18     <span v-else>Basic</span>
19   </div>
20   <script>
21     let vm = new Vue({
22       el : '#app',
23       data : {
24         balance : 0
25       }
26     });
27
28   </script>
```



# Directive (Cont.)

## ■ 반복 rendering

### ● v-for

```
<ul>
```

```
  <li v-for="item in items"> {{ item }} </li>
```

```
</ul>
```

```
new Vue({
```

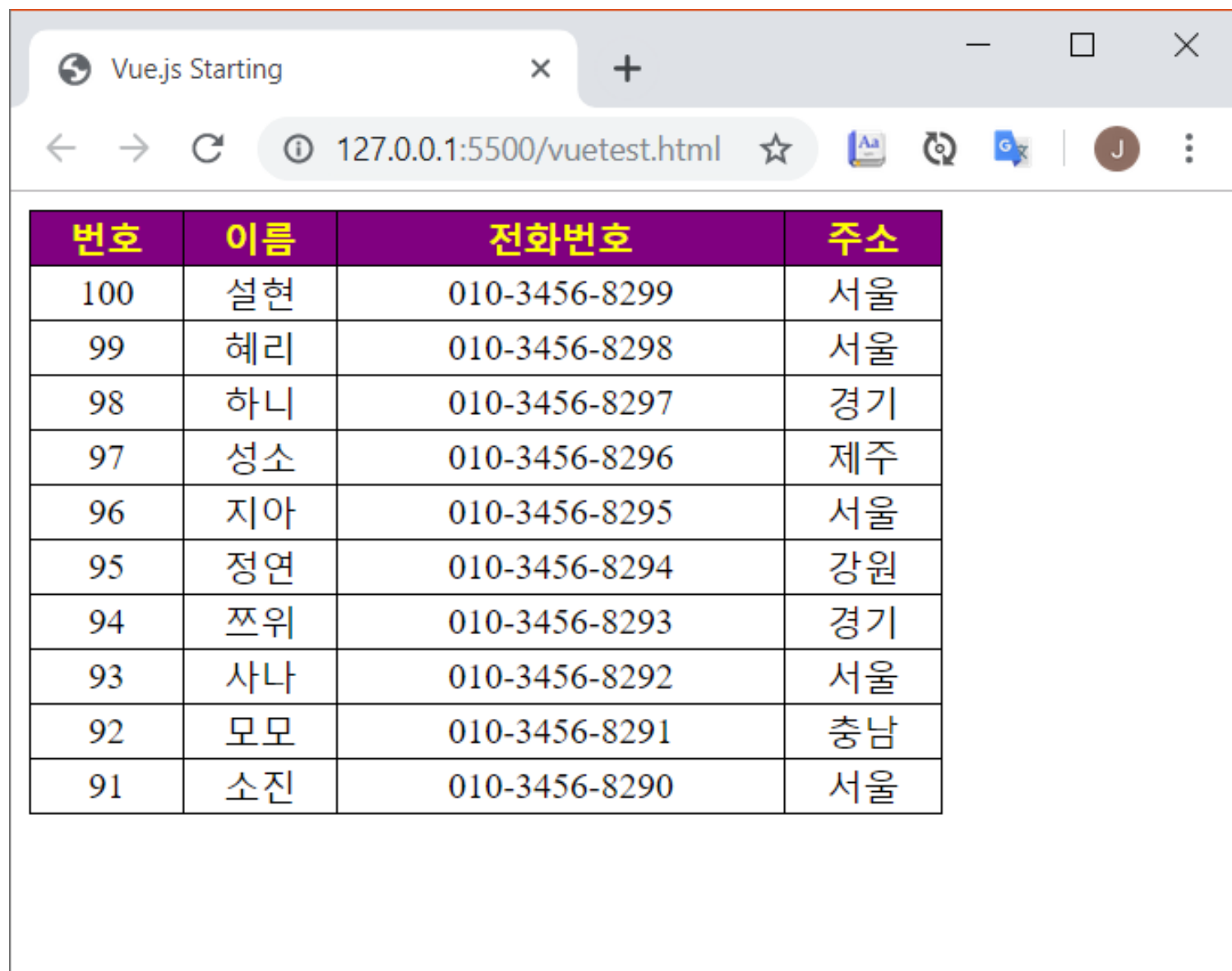
```
  data : {
```

```
    items : ['shirts', 'jeans', 'hats']
```

```
  }
```

```
})
```

## Directive (Cont.)



A screenshot of a web browser window. The tab is titled 'Vue.js Starting'. The address bar shows the URL '127.0.0.1:5500/vuetest.html'. The page content is a table with four columns: '번호' (Number), '이름' (Name), '전화번호' (Phone Number), and '주소' (Address). The table contains 10 rows of data, with the first row having a purple header. The browser interface includes navigation buttons (back, forward, refresh), a search bar, and various extension icons.

번호	이름	전화번호	주소
100	설현	010-3456-8299	서울
99	헤리	010-3456-8298	서울
98	하니	010-3456-8297	경기
97	성소	010-3456-8296	제주
96	지아	010-3456-8295	서울
95	정연	010-3456-8294	강원
94	쯔위	010-3456-8293	경기
93	사나	010-3456-8292	서울
92	모모	010-3456-8291	충남
91	소진	010-3456-8290	서울

# Directive (Cont.)

```
8      <style>
9          #list { width:400px; border:1px solid black; border-collapse: collapse;}
10         #list td, #list th {border:1px solid black; text-align: center;}
11         #list > thead > tr { color:yellow; background-color:purple}
12     </style>
13     <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
14 </head>
15 <body>
16     <div id="example">
17         <table id="list">
18             <thead>
19                 <tr>
20                     <th>번호</th><th>이름</th><th>전화번호</th><th>주소</th>
21                 </tr>
22             </thead>
23             <tbody id="contacts">
24                 <tr v-for="contact in contacts">
25                     <td>{{ contact.no }} </td>
26                     <td>{{ contact.name }}</td>
27                     <td>{{ contact.tel }}</td>
28                     <td>{{ contact.address }}</td>
29                 </tr>
30             </tbody>
31         </table>
32     </div>
```

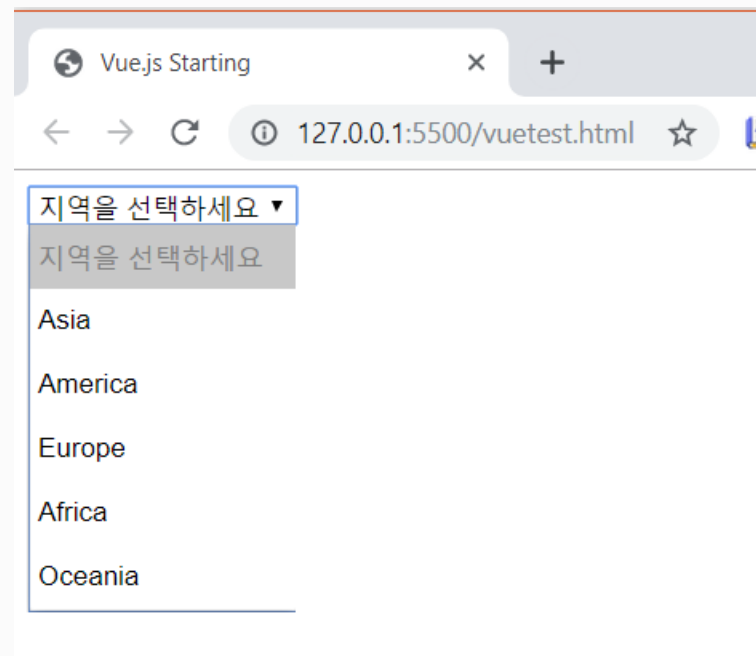


# Directive (Cont.)

```
33     <script>
34         let model = {
35             "pageno" : 1,
36             "pagesize" : 10,
37             "totalcount" : 100,
38             "contacts" :[
39                 { "no" : 100, "name" : "설현", "tel" : "010-3456-8299", "address" : "서울"},
40                 { "no": 99, "name": "혜리", "tel": "010-3456-8298", "address": "서울" },
41                 { "no": 98, "name": "하니", "tel": "010-3456-8297", "address": "경기" },
42                 { "no": 97, "name": "성소", "tel": "010-3456-8296", "address": "제주" },
43                 { "no": 96, "name": "지아", "tel": "010-3456-8295", "address": "서울" },
44                 { "no": 95, "name": "정연", "tel": "010-3456-8294", "address": "강원" },
45                 { "no": 94, "name": "쯔위", "tel": "010-3456-8293", "address": "경기" },
46                 { "no": 93, "name": "사나", "tel": "010-3456-8292", "address": "서울" },
47                 { "no": 92, "name": "모모", "tel": "010-3456-8291", "address": "충남" },
48                 { "no": 91, "name": "소진", "tel": "010-3456-8290", "address": "서울" }
49             ]
50         }
51         let list = new Vue({
52             el : '#example',
53             data : model
54         });
55     </script>
```

# Directive (Cont.)

```
8   <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9 </head>
10 <body>
11   <div id="example">
12     <select id="regions">
13       <option disabled="disabled" selected>지역을 선택하세요</option>
14       <option v-for="(val, key) in regions" v-bind:value="key">{{ val }}</option>
15     </select>
16   </div>
17   <script>
18     let regions = {
19       "A" : "Asia",
20       "B" : "America",
21       "C" : "Europe",
22       "D" : "Africa",
23       "E" : "Oceania"
24     };
25
26     let list = new Vue({
27       el : '#example',
28       data : {
29         regions:regions
30       }
31     });
32   </script>
```



## Directive (Cont.)

← → ↻ ⓘ 127.0.0.1:5500/vuetest.html ☆ Aa 🔍

번호	이름	전화번호	주소
1	설현	010-3456-8299	서울
2	혜리	010-3456-8298	서울
5	지아	010-3456-8295	서울
8	사나	010-3456-8292	서울
10	소진	010-3456-8290	서울

# Directive (Cont.)

```
8      <style>
9          #list { width:400px; border:1px solid black; border-collapse: collapse;}
10         #list td, #list th {border:1px solid black; text-align: center;}
11         #list > thead > tr { color:yellow; background-color:purple}
12     </style>
13     <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
14 </head>
15 <body>
16     <div id="example">
17         <table id="list">
18             <thead>
19                 <tr>
20                     <th>번호</th><th>이름</th><th>전화번호</th><th>주소</th>
21                 </tr>
22             </thead>
23             <tbody id="contacts">
24                 <tr v-for="(contact, index) in contacts"
25                     v-if="contact.address.indexOf('서울') > -1">
26                     <td>{{ index + 1 }} </td>
27                     <td>{{ contact.name }}</td>
28                     <td>{{ contact.tel }}</td>
29                     <td>{{ contact.address }}</td>
30                 </tr>
31             </tbody>
32         </table>
33     </div>
```

# Directive (Cont.)

```
33     <script>
34         let model = {
35             "pageno" : 1,
36             "pagesize" : 10,
37             "totalcount" : 100,
38             "contacts" :[
39                 { "no" : 100, "name" : "설현", "tel" : "010-3456-8299", "address" : "서울"},
40                 { "no": 99, "name": "혜리", "tel": "010-3456-8298", "address": "서울" },
41                 { "no": 98, "name": "하니", "tel": "010-3456-8297", "address": "경기" },
42                 { "no": 97, "name": "성소", "tel": "010-3456-8296", "address": "제주" },
43                 { "no": 96, "name": "지아", "tel": "010-3456-8295", "address": "서울" },
44                 { "no": 95, "name": "정연", "tel": "010-3456-8294", "address": "강원" },
45                 { "no": 94, "name": "쯔위", "tel": "010-3456-8293", "address": "경기" },
46                 { "no": 93, "name": "사나", "tel": "010-3456-8292", "address": "서울" },
47                 { "no": 92, "name": "모모", "tel": "010-3456-8291", "address": "충남" },
48                 { "no": 91, "name": "소진", "tel": "010-3456-8290", "address": "서울" }
49             ]
50         }
51         let list = new Vue({
52             el : '#example',
53             data : model
54         });
55     </script>
```

## Directive (Cont.)

← → ↻ ⓘ 127.0.0.1:5500/vuetest.html ☆ 📄 🔍

번호	이름	전화번호	주소
1	설현	010-3456-8299	서울
2	혜리	010-3456-8298	서울
3	하니	010-3456-8297	경기
4	성소	010-3456-8296	제주
5	지아	010-3456-8295	서울
6	정연	010-3456-8294	강원
7	쯔위	010-3456-8293	경기
8	사나	010-3456-8292	서울
9	모모	010-3456-8291	충남
10	소진	010-3456-8290	서울

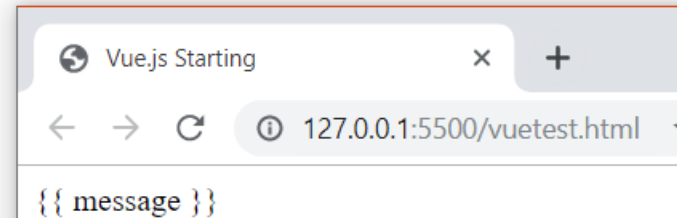


# Directive (Cont.)

## ■ 기타 directive

- **v-pre**
- HTML 요소에 대한 컴파일을 수행하지 않는다.

```
8   <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9 </head>
10 <body>
11   <div id="example">
12     <span v-pre> {{ message }} </span>
13   </div>
14   <script>
15     let vm = new Vue({
16       el : '#example',
17       data : {
18         message : 'Hello, Vue.js!!!'
19       }
20     });
21   </script>
22 </body>
23 </html>
```



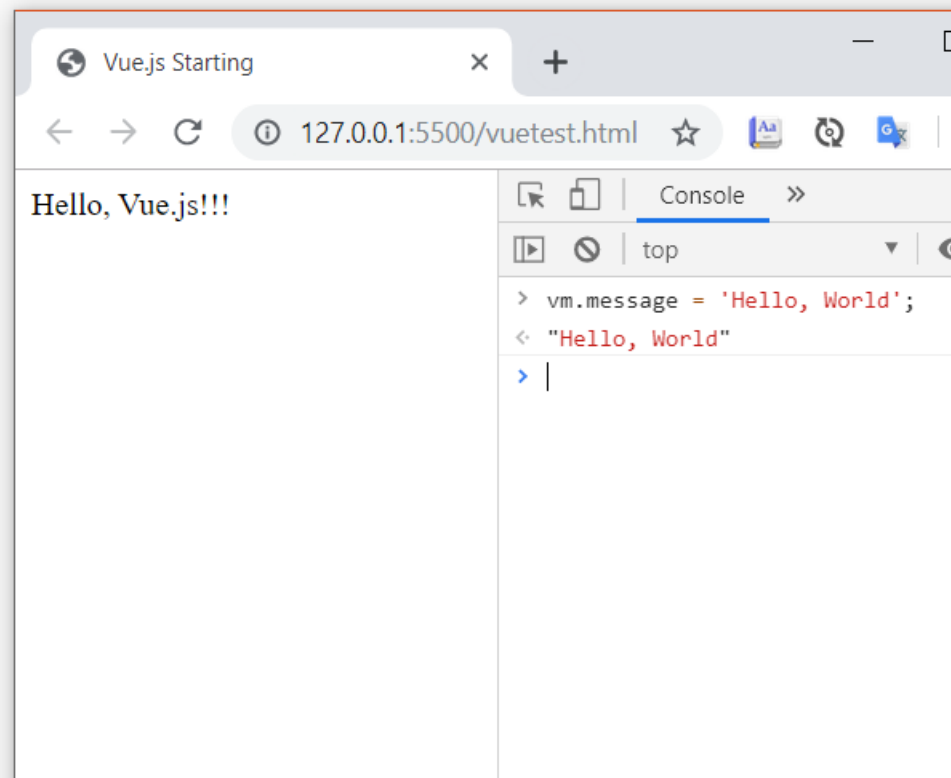


# Directive (Cont.)

## ■ 기타 directive

- **v-once**
- HTML 요소를 단 한번만 rendering 한다.

```
8   <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9 </head>
10 <body>
11   <div id="example">
12     <span v-once> {{ message }} </span>
13   </div>
14   <script>
15     let vm = new Vue({
16       el : '#example',
17       data : {
18         message : 'Hello, Vue.js!!!'
19       }
20     });
21   </script>
22 </body>
23 </html>
```



## Directive (Cont.)

국가명 :

번호	국가명	수도	지역
1	미국	워싱턴DC	america
12	나미비아	빈트후크	africa

## Directive (Cont.)

```
8      <style>
9          #List {
10              width: 400px;
11              border: 1px solid ■black;
12              border-collapse: collapse;
13          }
14
15          #List td, #List th {
16              border: 1px solid ■black;
17              text-align: center;
18          }
19
20          #List > thead > tr {
21              color: ■yellow;
22              background-color: ■purple;
23          }
24      </style>
25      <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
26  </head>
```

# Directive (Cont.)

```
27 <body>
28   <div id="exmaple">
29     <p>
30       국가명 : <input type="text" v-model="countryname" placeholder="국가명" />
31     </p>
32     <table id="list">
33       <thead>
34         <tr>
35           <th>번호</th>
36           <th>국가명</th>
37           <th>수도</th>
38           <th>지역</th>
39         </tr>
40       </thead>
41       <tbody id="contacts">
42         <tr v-for="c in filtered">
43           <td>{{c.no}}</td>
44           <td>{{c.name}}</td>
45           <td>{{c.capital}}</td>
46           <td>{{c.region}}</td>
47         </tr>
48       </tbody>
49     </table>
50   </div>
```

## Directive (Cont.)

```
50     </div>
51     <script type="text/javascript">
52         var model = {
53             countryname: "",
54             countries: [
55                 { no: 1, name: "미국", capital: "워싱턴DC", region: "america" },
56                 { no: 2, name: "프랑스", capital: "파리", region: "europe" },
57                 { no: 3, name: "영국", capital: "런던", region: "europe" },
58                 { no: 4, name: "중국", capital: "베이징", region: "asia" },
59                 { no: 5, name: "태국", capital: "방콕", region: "asia" },
60                 { no: 6, name: "모로코", capital: "라바트", region: "africa" },
61                 { no: 7, name: "라오스", capital: "비엔티안", region: "asia" },
62                 { no: 8, name: "베트남", capital: "하노이", region: "asia" },
63                 { no: 9, name: "피지", capital: "수바", region: "oceania" },
64                 { no: 10, name: "솔로몬 제도", capital: "호니아라", region: "oceania" },
65                 { no: 11, name: "자메이카", capital: "킹스턴", region: "america" },
66                 { no: 12, name: "나미비아", capital: "빈트후크", region: "africa" },
67                 { no: 13, name: "동티모르", capital: "딜리", region: "asia" },
68                 { no: 14, name: "멕시코", capital: "멕시코시티", region: "america" },
69                 { no: 15, name: "베네수엘라", capital: "카라카스", region: "america" },
70                 { no: 16, name: "서사모아", capital: "아피아", region: "oceania" }
71             ]
72         }
```

## Directive (Cont.)

```
74     var clist = new Vue({
75         el: "#exmaple",
76         data: model,
77         computed: {
78             filtered: function () {
79                 var cname = this.countryname.trim();
80                 return this.countries.filter(function (item, index) {
81                     if (item.name.indexOf(cname) > -1) {
82                         return true;
83                     }
84                 });
85             }
86         }
87     });
88 </script>
89 </body>
90 </html>
```

# Directive (Cont.)

## ■ Event handling

- **v-on**
- Event가 일어난 시점에 속성값으로 지정된 표현식 실행
- DOM API의 **addEventListener**과 같다.

**v-on:이벤트 이름="값을 평가할 표현식 "**

- v-on 생략 표기법  
**v-on : click → @click**

# Directive (Cont.)

```
8   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
9   <style>
10    .layout1 {
11      margin: 30px 30px 30px 30px;
12    }
13  </style>
14  <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
15 </head>
16 <body>
17   <div id="example" class="container layout1">
18     <p><input type="text" v-model="amount" class="form-control" /></p>
19     <p>
20       <button id="deposit" @click="balance += parseInt(amount)" class="btn btn-primary">예금</button>
21       <button id="withdraw" @click="balance -= parseInt(amount)" class="btn btn-primary">인출</button>
22     </p>
23     <h3>계좌 잔고 : {{balance}}</h3>
24   </div>
25   <script type="text/javascript">
26     var vm = new Vue({
27       el: "#example",
28       data: {
29         amount: 0,
30         balance: 0,
31       }
32     })
33   </script>
```

예금 인출

계좌 잔고 : 50



# Method

- Vue instance의 method
- Vue instance의 생성자 options 객체에서 **methods** property로 정의

```
methods : {  
  메소드명 : function(){  
    //원하는 처리  
  }  
}
```

# Lab

## ■ product.js

```
1  var items = [  
2      {  
3          name: '연필',  
4          price: 300,  
5          quantity: 0  
6      },  
7      {  
8          name: '공책',  
9          price: 400,  
10         quantity: 0  
11     },  
12     {  
13         name: '지우개',  
14         price: 500,  
15         quantity: 0  
16     }  
17 ]
```

## Lab (Cont.)

```
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <meta http-equiv="X-UA-Compatible" content="ie=edge">
7    <title>Vue.js Code Template</title>
8    <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9    <script src="product.js"></script>
10 </head>
```

# Lab (Cont.)

```
11 <body>
12   <h1>Shopping Mall</h1>
13   <div id="app">
14     <ul>
15       <li v-for="item in items" v-bind:key="item.name">
16         {{ item.name }}의 개수: <input type="number" v-model="item.quantity" min="0">
17       </li>
18     </ul>
19     <hr>
20     <div v-bind:style="errorMessageStyle">
21       <ul>
22         <li v-for="item in items" v-bind:key="item.name">
23           {{ item.name }}: {{ item.price }} x {{ item.quantity }} =
24           {{ item.price * item.quantity | numberWithDelimiter }} 원
25         </li>
26       </ul>
27       <p>{{ items[0].name }}: {{ items[0].price }} x {{ items[0].quantity }}</p>
28       <p>소계: {{ totalPrice | numberWithDelimiter }} 원</p>
29       <p>합계(세포함): {{ totalPriceWithTax | numberWithDelimiter }} 원</p>
30       <p v-show="!canBuy">
31         {{ 1000 | numberWithDelimiter }}원 이상부터 구매 가능
32       </p>
33       <!-- 버튼을 클릭하면 메시드가 호출됨 -->
34       <button v-bind:disabled="!canBuy" v-on:click="doBuy">구매</button>
35     </div>
36   </div>
```

# Lab (Cont.)

```
38 <script>
39   var vm = new Vue({
40     el: '#app',
41     data: {
42       items: items
43     },
44     filters: {
45       numberWithDelimiter: function (value) {
46         if (!value) {
47           return '0'
48         }
49         return value.toString().replace(/(\d)(?=(\d{3})+$/g, '$1,')
50       }
51     },
52     methods: {
53       doBuy: function () {
54         // 실제라면 서버와 통신하는 시점
55         alert(this.totalPriceWithTax + '원 판매됨!')
56         this.items.forEach(function (item) {
57           item.quantity = 0
58         })
59       }
60     },
```

# Lab (Cont.)

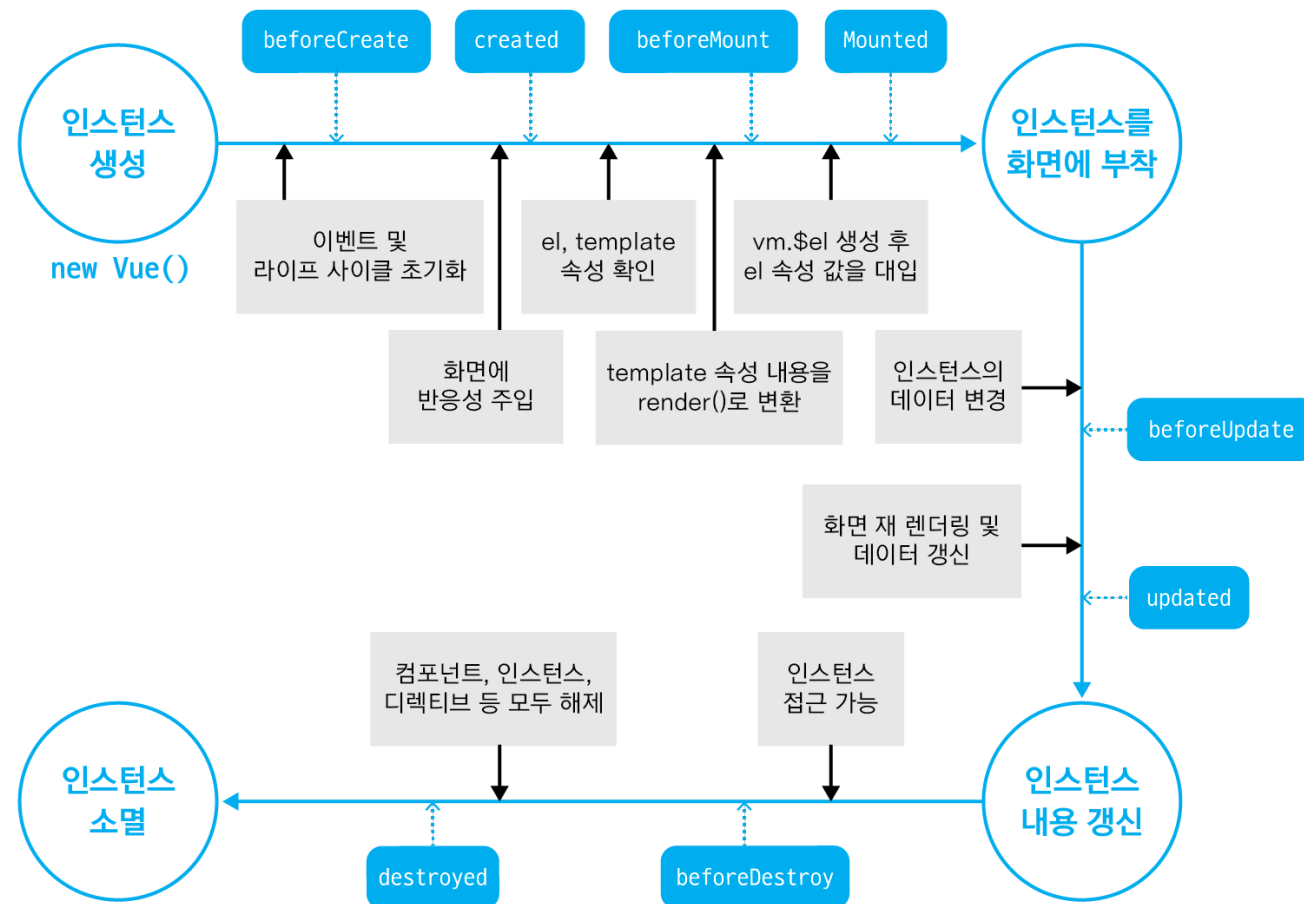
```
61         computed: {
62             totalPrice: function () {
63                 return this.items.reduce(function (sum, item) {
64                     return sum + (item.price * item.quantity)
65                 }, 0)
66             },
67             totalPriceWithTax: function () {
68                 return Math.floor(this.totalPrice * 1.10)
69             },
70             canBuy: function () {
71                 return this.totalPrice >= 1000
72             },
73             errorMessageStyle: function () {
74                 // canBuy가 거짓이면 붉게 표시
75                 return {
76                     border: this.canBuy ? '' : '1px solid red',
77                     color: this.canBuy ? '' : 'red'
78                 }
79             }
80         }
81     });
82 </script>
83 </body>
```

# Instance Lifecycle

- Vue의 Instance가 생성되어 소멸되기까지 거치는 과정
- Instance가 생성되고 나면 Library 내부적으로 다음과 같은 과정이 진행된다.
  - **data** 속성의 초기화 및 관찰(Reactivity 주입)
  - Vue Template Code Compile(Virtual DOM → DOM 변환)
  - Instance를 DOM에 부착

# Instance Lifecycle (Cont.)

## ■ Instance의 Lifecycle Diagram



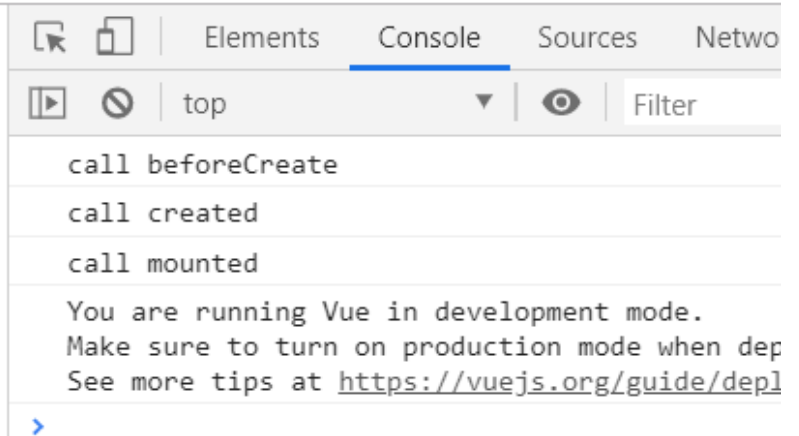


# Instance Lifecycle (Cont.)

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

<script>
  var vm = new Vue({
    el : '#app',
    data : {
      message : 'Hello, Vue.js!!!'
    },
    beforeCreate : function() {
      console.log('call beforeCreate');
    },
    created : function(){
      console.log('call created');
    },
    mounted : function(){
      console.log('call mounted');
    },
    updated : function(){
      console.log('call updated');
    }
  });
</script>
```

Hello, Vue.js!!!



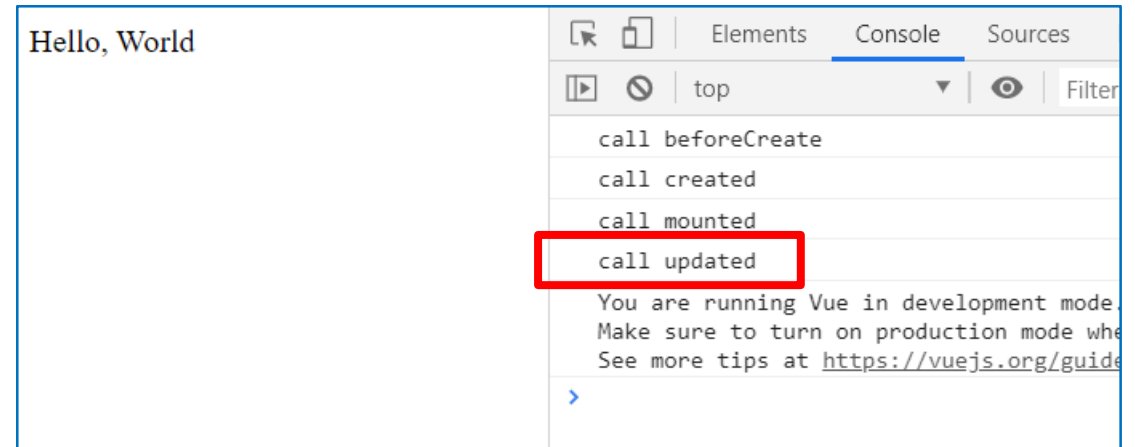
The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays the following logs:

- call beforeCreate
- call created
- call mounted

Below the logs, a message states: "You are running Vue in development mode. Make sure to turn on production mode when deploying to production. See more tips at <https://vuejs.org/guide/deployment>".

# Instance Lifecycle (Cont.)

```
var vm = new Vue({
  el : '#app',
  data : {
    message : 'Hello, Vue.js!!!'
  },
  beforeCreate : function() {
    console.log('call beforeCreate');
  },
  created : function(){
    console.log('call created');
  },
  mounted : function(){
    console.log('call mounted');
    this.message = 'Hello, World';
  },
  updated : function(){
    console.log('call updated');
  }
});
```



message값 변경

# Instance Lifecycle (Cont.)

## ■ Lifecycle Hook

- Vue의 Lifecycle을 이해해야 하는 이유는 바로 Lifecycle Hook 때문이다.
- Lifecycle Hook으로 Instance의 특정 시점에 원하는 Logic을 구현할 수 있다.
- 예를 들어, Component가 생성되고 바로 Data를 Server에서 받아오고 싶으면 **created**나 **beforeMount** Lifecycle Hook을 사용할 수 있다.
- 옆 Code는 Instance가 생성되고 바로 Axios로 HTTP GET Request를 보내서 Data를 받아오는 Code.

```
new Vue({  
  methods: {  
    fetchData() {  
      axios.get(url);  
    }  
  },  
  created: function() {  
    this.fetchData();  
  }  
})
```

# Instance Lifecycle (Cont.)

## ■ 자주 사용되는 Lifecycle Hook List

### ● created

#### ■ beforeCreate 다음 단계

- data 속성과 methods 속성이 정의되었기 때문에 **this.data** 또는 **this.fetchData()**와 같은 Logic들을 이용하여 **data** 속성과 **methods** 속성에 정의된 값에 접근하여 Logic을 실행할 수 있다.
- 하지만, 아직 Instance가 화면 요소에 부착되기 전이기 때문에 **template** 속성에 정의된 dom 요소에 접근할 수 없다.

### ● beforeMount

- **created** 단계 이후 **template** 속성에 지정한 Markup 속성을 **render()** 함수로 변환한 후 **el** 속성에 지정한 화면 요소(DOM)에 Instance를 부착하기 전에 호출되는 단계.
- **render()**가 호출되기 직전의 Logic 추가할 때 좋다.

# Instance Lifecycle (Cont.)

## ■ 자주 사용되는 Lifecycle Hook List

### ● Mounted

- **el** 속성에 지정한 화면 요소에 Instance가 부착된 후 호출되는 단계.
- **template** 속성에 정의한 화면 요소(DOM)에 접근할 수 있어서 화면 요소를 제어하는 Logic을 수행하기 좋은 단계.
- 하지만, DOM에 Instance가 부착되고 바로 호출되기 때문에 하위 Component나 외부 Library에 의해 추가된 화면 요소들이 최종 HTML Code로 변환되는 시점과 다를 수 있다.

### ● Destroyed

- Vue Instance가 소멸되고 난 후 호출되는 단계.
- Vue Instance에 정의한 모든 속성이 제거되고 하위에 선언한 Instance들 또한 모두 소멸한다.