목록 출력

2020-04-20

이승진

**학습목표**

v-for 지시자

배열의 원소 변경

Vue.set 메소드

$forceUpdate 메소드

**목차**

[1. v-for 지시자(directive) 2](#_Toc38769066)

[1) 태그 반복 생성 2](#_Toc38769067)

[2) display 서식 4](#_Toc38769068)

[3) index 5](#_Toc38769069)

[4) key 6](#_Toc38769070)

[5) 객체 배열 7](#_Toc38769071)

[6) 객체 속성 반복 11](#_Toc38769072)

[7) n 회 반복 15](#_Toc38769073)

[2. 배열의 원소 변경 17](#_Toc38769074)

[1) 변수값 변경 17](#_Toc38769075)

[2) 배열의 원소 변경 19](#_Toc38769076)

[3) this.$forceUpdate() 메소드 21](#_Toc38769077)

[4) Vue.set 메소드 22](#_Toc38769078)

[5) 배열 복제 24](#_Toc38769079)

[6) 기타 배열의 변경 27](#_Toc38769080)

[3. 객체의 속성 변경 28](#_Toc38769081)

[1) 객체의 속성값 변경 28](#_Toc38769082)

[2) 구조분해 할당 (destructing assignment) 30](#_Toc38769083)

[3) 객체 배열 변경 32](#_Toc38769084)

[4) this.$forceUpdate() 35](#_Toc38769085)

[5) 객체 배열 복제 36](#_Toc38769086)

[6) 배열 메소드 39](#_Toc38769087)

[4. 과제 41](#_Toc38769088)

[1) gugu1.html 41](#_Toc38769089)

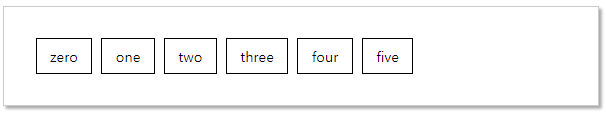
[2) gugu2.html 41](#_Toc38769090)

# v-for 지시자(directive)

## 태그 반복 생성

### for1.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 600px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; }    </style>  </head>  <body>    <div id="app">      <span v-for="s in words">        {{ s }}      </span>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          words: ["zero", "one", "two", "three", "four", "five"]        }      })     </script>  </body>  </html> |



data: {

words: ["zero", "one", "two", "three", "four", "five"]

}

Vue 앱에 words 속성을 만든다.

이 속성의 값은 문자열 배열이다.

<span v-for="s in words">

  {{ s }}

</span>

v-for 지시자는, for 반복문과 비슷하게 작동한다.

words 배열의 원소 각각을 s 지역 변수에 대입하면서 반복한다.

words 배열의 크기 만큼 span 태그들이 반복 생성된다.

{{ s }} 부분에 s 변수의 값이 출력된다.

이 v-for 지시자에 의해서 다음과 같은 태그들이 생성된다.

<span> zero </span>

<span> one </span>

<span> two </span>

<span> three </span>

<span> four </span>

<span> five </span>

## display 서식

span { display: inline-block; margin: 5px; padding: 8px 15px;

  border: 1px solid black; }

display 서식 속성은 태그가 그려지는 형태를 지정한다.

이 속성의 값은 block, inline, inline-block, none 이다.

### display: none;

태그가 웹브라우저 창에 그려지지 않는다.

### display: block;

태그의 width 값이 100% 가 된다.

block 태그들은 margin, padding, width, height 서식이 잘 적용된다.

### display: inline;

태그의 width 값이 그 내용에 딱 맞는 크기가 된다.

padding과 좌우 margin 속성은 잘 적용되지만,

위 아래 margin 속성과 width, height 속성은 상황에 따라 무시될 수 있다.

### inline-block

태그의 width 값이 그 내용에 딱 맞는 크기가 된다.

padding, margin, width, height 속성이 잘 적용된다.

### 디폴트 값

display 속성의 디폴트 값이 block인 태그들은 다음과 같다.

h1, div, p, table, form

display 속성의 디폴트 값이 inline인 태그들은 다음과 같다.

a span input select button

for1.html 예제에서 span 태그의 display 속성 값을 inline-block 으로 설정한 이유는

margin, padding 값이 잘 적용되도록 하기 위해서다.

span { display: inline-block; margin: 5px; padding: 8px 15px;

  border: 1px solid black; }

span 태그의 display 속성 디폴트 값이 inline 이다.

inline 태그에는 margin, padding 값이 잘 적용되지 않는다.

## index

### for2.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 600px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; }    </style>  </head>  <body>    <div id="app">      <span v-for="(s, index) in words">        {{ index }}-{{ s }}      </span>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          words: ["zero", "one", "two", "three", "four", "five"]        }      })     </script>  </body>  </html> |

<span v-for="s in words">

<span v-for="(s, index) in words">

위 두 v-for 지시자가 하는 일은 동일하다. span 태그를 반복해서 생성한다.

words 배열의 원소 각각을 s 지역 변수에 대입하면서 반복한다.

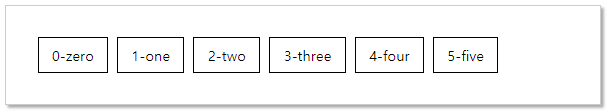
반복의 순서에 해당하는 인덱스 값이 index 지역 변수에 대입된다.

{{ index }}

index 변수 값이 출력된다.

{{ s }}

s 변수 값이 출력된다.



## key

### for3.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 600px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; }    </style>  </head>  <body>    <div id="app">      <span v-for="(s, index) in words" v-bind:key="index">        {{ index }}-{{ s }}      </span>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          words: ["zero", "one", "two", "three", "four", "five"]        }      })     </script>  </body>  </html> |

Vue 앱의 속성 값이 변경되면, 그 값이 출력되는 태그 부분도 즉시 변경된다.

words 배열의 크기가 늘어나거나 줄어들면,

v-for 지시자에 의해 생성되는 span 태그들의 수도 즉시 늘어나거나 줄어든다.

예를 들어, words 배열의 어떤 원소가 제거되면,

그 원소에 해당하는 span 태그도 즉시 제거된다.

Vue 엔진이 그 원소에 해당하는 span 태그를 찾아 제거해야 한다.

이렇게 Vue 엔진이 태그를 찾는 작업을 도와주기 위해,

v-for 지시자에 의해 반복 생성되는 태그에 v-bind:key 속성을 부여하자.

이 속성의 값은 유일해야 한다.

전체 페이지에서 유일할 필요까지는 없고, 현재 v-for 반복에서만 유일하면 된다.

반복 순서에 해당하는 인덱스 값도, 현재 반복에서 유일하므로, v-bind:key 속성값이 될 수 있다.

만약 배열의 원소가 중복되지 않는다면, index 보다는 배열의 원소를

v-bind:key 속성값으로 사용하는 것이 좋다.

## 객체 배열

### for4.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 500px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      table { border-collapse: collapse; width: 400px; }      td { padding: 5px; border: 1px solid #aaa;  text-align: center; }      tr:nth-child(1) { background-color: #ddd; }    </style>  </head>  <body>    <div id="app">      <table>        <tr>          <td>ID</td><td>이름</td><td>나이</td>        </tr>        <tr v-for="person in persons" v-bind:key="person.id">          <td>{{ person.id }}</td>          <td>{{ person.name }}</td>          <td>{{ person.age }}</td>        </tr>      </table>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          persons: [ { id: 101, name: "홍길동", age: 16 },                     { id: 102, name: "임꺽정", age: 18 },                     { id: 103, name: "장길산", age: 19 },                     { id: 104, name: "이몽룡", age: 17 } ]        }      })     </script>  </body>  </html> |

data: {

  persons: [ { id: 101, name: "홍길동", age: 16 },

     { id: 102, name: "임꺽정", age: 18 },

             { id: 103, name: "장길산", age: 19 },

             { id: 104, name: "이몽룡", age: 17 } ]

}

앱의 persons 속성 값은 객체 배열이다.

<tr v-for="person in persons" v-bind:key="person.id">

persons 배열의 원소 각각을 person 지역 변수에 대입하면서, tr 태그를 반복 생성한다.

이렇게 반복 생성되는 태그의 키(v-bind:key) 값으로 person.id 값을 지정한다.

<td>{{ person.id }}</td>

<td>{{ person.name }}</td>

<td>{{ person.age }}</td>

person 객체의 id, name, age 속성값을 <td> 태그에 출력한다.



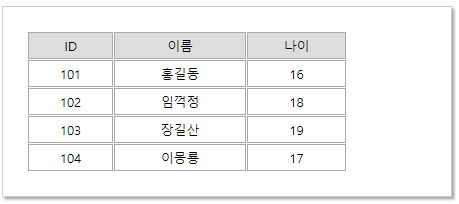
tr:nth-child(1) { background-color: #ddd; }

부모 태그(table)로부터 첫번째 자식인 tr 태그에만 이 서식을 적용한다.

즉 첫번째 행(tr)의 배경색이 회색이 된다.

table { border-collapse: collapse; width: 400px; }

border-collapse: collapse 서식을 제거하고 실행하면 아래와 같이 그려진다.



### for5.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 500px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      table { border-collapse: collapse; width: 400px; }      td { padding: 5px; border: 1px solid #aaa;  text-align: center; }      tr:nth-child(1) { background-color: #ddd; }    </style>  </head>  <body>    <div id="app">      <table>        <tr>          <td>index</td><td>ID</td><td>이름</td><td>나이</td>        </tr>        <tr v-for="(person, index) in persons" v-bind:key="person.id">          <td>{{ index }}</td>          <td>{{ person.id }}</td>          <td>{{ person.name }}</td>          <td>{{ person.age }}</td>        </tr>      </table>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          persons: [ { id: 101, name: "홍길동", age: 16 },                     { id: 102, name: "임꺽정", age: 18 },                     { id: 103, name: "장길산", age: 19 },                     { id: 104, name: "이몽룡", age: 17 } ]        }      })     </script>  </body>  </html> |

<tr v-for="person in persons" v-bind:key="person.id">

<tr v-for="(person, index) in persons" v-bind:key="person.id">

위 두 v-for 지시자가 하는 일은 동일하다. tr 태그를 반복해서 생성한다.

persons 배열의 원소 각각을 person 지역 변수에 대입하면서 반복한다.

반복의 순서에 해당하는 인덱스 값이 index 지역 변수에 대입된다.

이렇게 반복 생성되는 태그의 키(v-bind:key) 값으로 person.id 값을 지정한다.



## 객체 속성 반복

### for6.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 500px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;  border: 1px solid black; }    </style>  </head>  <body>    <div id="app">      <span v-for="value in person">        {{ value }}      </span>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          person: { id: 101, name: "홍길동", age: 16, "3": "three", "2": "two", 1: "one" }        }      })     </script>  </body>  </html> |

data: {

   person: { id: 101, name: "홍길동", age: 16, "3": "three", "2": "two", 1: "one" }

}

Vue 앱의 person 속성값은 객체이다.

이 객체의 속성은 다음과 같다.

|  |  |
| --- | --- |
| 속성명 | 값 |
| id | 101 |
| name | "홍길동" |
| age | 16 |
| 3 | "three" |
| 2 | "two" |
| 1 | "one" |

<span v-for="value in person">

앱의 person 속성의 값은 객체이다.

이 객체의 속성값 각각을 value 지역 변수에 대입하며, span 태그를 반복 생성한다.

<span v-for="value in person">

{{ value }}

</span>

v-for 지시자에 의해서 반복 생성되는 span 태그는 다음과 같다.

<span> one </span>

<span> two </span>

<span> three </span>

<span> 101 </span>

<span> 홍길동 </span>

<span> 16 </span>



javascript 객체의 속성이 탐색되는 순서는 다음과 같다.

- 속성명이 숫자인 속성이 먼저 탐색된다.

- 속성명이 숫자인 속성의 탐색 순서는, 그 속성명 오름차순이다.

- 그 다음 속성명이 문자인 속성이 탐색된다.

- 속성명이 문자인 숙성의 탐색 순서는, 속성이 생성된 순서이다.

### for7.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 500px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;  border: 1px solid black; }    </style>  </head>  <body>    <div id="app">      <span v-for="(value, key) in person" v-bind:key="key">        {{ key }}, {{ value }}      </span>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          person: { id: 101, name: "홍길동", age: 16, "3": "three", "2": "two", 1: "one" }        }      })     </script>  </body>  </html> |

<span v-for="(value, key) in person" v-bind:key="key">

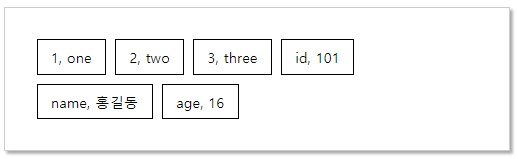
앱의 person 속성의 값은 객체이다.

이 객체의 속성값, 속성명 각각을 value, key 지역 변수에 대입하며, span 태그를 반복 생성한다.

이렇게 반복 생성되는 태그의 키(v-bind:key) 값으로, key 변수값(person 객체의 속성명)을 지정한다.

value, key 변수의 이름은 중요하지 않다, 순서가 중요하다.

(속성값, 속성명) 순서이다.



### for8.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 500px;       border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;  border: 1px solid black; }    </style>  </head>  <body>    <div id="app">      <span v-for="(value, key, index) in person" v-bind:key="key">        {{ index }}, {{ key }}, {{ value }}      </span>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          person: { id: 101, name: "홍길동", age: 16, "3": "three", "2": "two", 1: "one" }        }      })     </script>  </body>  </html> |

<span v-for="(value, key, index) in person" v-bind:key="key">

앱의 person 속성의 값은 객체이다.

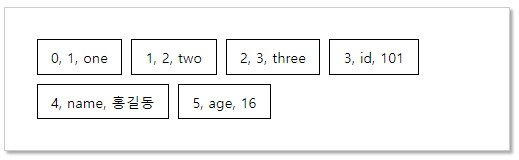
이 객체의 속성값, 속성명 각각을 value, key 지역 변수에 대입하며, span 태그를 반복 생성한다.

반복의 순서에 해당하는 인덱스 값이 index 지역 변수에 대입된다.

이렇게 반복 생성되는 태그의 키(v-bind:key) 값으로, key 변수값(person 객체의 속성명)을 지정한다.

value, key, index 변수의 이름은 중요하지 않다, 순서가 중요하다.

(속성값, 속성명, 인덱스) 순서이다.



## n 회 반복

### for9.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 500px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;  border: 1px solid black; }    </style>  </head>  <body>    <div id="app">      <span v-for="i in size">        {{ i }}      </span>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          size: 14        }      })     </script>  </body>  </html> |

<span v-for="i in size">

지역 변수 i 의 값이 1 부터 size 이 될 때까지 1씩 증가하며

span 태그를 반복 생성한다.

즉 span 태그가 14 개 생성된다.

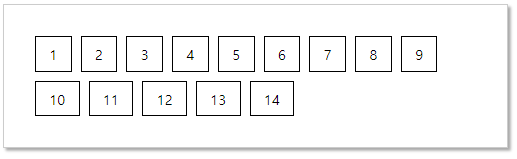
<span> 1 </span>

<span> 2 </span>

<span> 3 </span>

... 생략 ...

<span> 14 </span>



### for10.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      table { border-collapse: collapse; margin: auto; }      td { padding: 15px; border: 1px solid black;  text-align: center; }    </style>  </head>  <body>    <div id="app">      <table>        <tr v-for="r in row\_size">          <td v-for="c in col\_size">            {{ (r - 1) \* row\_size + c }}          </td>        </tr>      </table>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          row\_size: 4,          col\_size: 4        }      })     </script>  </body>  </html> |

중첩된 v-for 구현

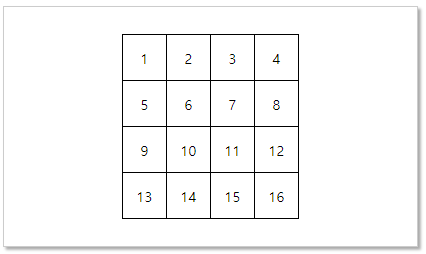
<tr v-for="r in row\_size">

tr 태그가 row\_size 개 생성된다.

<td v-for="c in col\_size">

tr 태그 내부에 td 태그가 col\_size 개 생성된다.

tr 태그가 row\_size개 이므로, td 태그는 총 row\_size \* col\_size 개 생성된다.



# 배열의 원소 변경

## 변수값 변경

### update1.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 300px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; cursor: pointer; }      div#app > div { margin-top: 20px; }      button { padding: 0.5em 2em; }    </style>  </head>  <body>    <div id="app">      <span>        {{ i }}      </span>      <div>        <button type="button" v-on:click="increase">++</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          i : 0        },        methods: {          increase() {            this.i++;          }        }      })     </script>  </body>  </html> |

data: {

i : 0

}

앱의 i 속성값은 0 이다.

{{ i }}

앱의 i 속성의 값이 출력된다. i 속성값이 변경되면 즉시 다시 출력된다.



++ 버튼을 누르면 increase 메소드가 호출된다.

<button type="button" v-on:click="increase">++</button>

methods: {

  increase() {

    this.i++;

  }

}

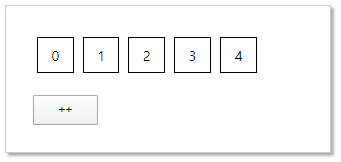
increase 메소드에서 i 속성값을 ++ 한다.

i 속성값이 변경되면, {{ i }} 부분에 즉시 다시 출력된다.

## 배열의 원소 변경

### update2.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 300px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; cursor: pointer; }      div#app > div { margin-top: 20px; }      button { padding: 0.5em 2em; }    </style>  </head>  <body>    <div id="app">      <span v-for="i in a">        {{ i }}      </span>      <div>        <button type="button" v-on:click="increase">++</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          a: [ 0, 1, 2, 3, 4]        },        methods: {          increase() {            for (let i=0; i < this.a.length; ++i)              this.a[i] += 1;            console.log(this.a);          }        }      })     </script>  </body>  </html> |



앱의 a 속성값은 int 배열이다.

++ 버튼을 누르면 increase 메소드가 호출된다.

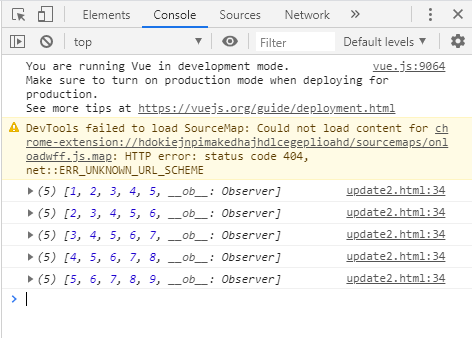
increase 메소드에서 a 배열의 각 원소의 값을 1 증가시킨다.

그런데 화면에 변화가 없다.

웹브라우저에서 F12 단축키를 눌러서 개발자 창을 열고, Console 탭을 보자.

console.log(this.a);

이 문장의 출력을 Console 탭에서 확인하자.



버튼을 누를 때 마다 a 배열의 원소의 값이 1 씩 증가하고 있다.

그런데 화면이 다시 그려지지 않고 있다.

Vue 앱의 속성의 값이 배열인 경우에,

그 배열의 원소의 값이 변경되는 것을 Vue 엔진이 감지하지 못한다.

그래서, 배열의 원소의 값이 변경되더라도, 화면이 다시 그려지지 않는다.

## this.$forceUpdate() 메소드

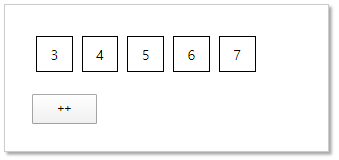
### update3.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 300px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; cursor: pointer; }      div#app > div { margin-top: 20px; }      button { padding: 0.5em 2em; }    </style>  </head>  <body>    <div id="app">      <span v-for="i in a">        {{ i }}      </span>      <div>        <button type="button" v-on:click="increase">++</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          a: [ 0, 1, 2, 3, 4]        },        methods: {          increase() {            for (let i=0; i < this.a.length; ++i)              this.a[i] += 1;            this.$forceUpdate();          }        }      })     </script>  </body>  </html> |

노란색으로 칠한 부분을 제외하면, 앞의 예제와 동일하다.

this.$forceUpdate(); 메소드는

화면이 다시 그려지도록 강제한다.



++ 버튼을 클릭할 때 마다 화면이 다시 그려지기 때문에, 숫자들이 1 씩 증가하는 것을 볼 수 있다.

## Vue.set 메소드

### update4.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 300px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; cursor: pointer; }      div#app > div { margin-top: 20px; }      button { padding: 0.5em 2em; }    </style>  </head>  <body>    <div id="app">      <span v-for="i in a">        {{ i }}      </span>      <div>        <button type="button" v-on:click="increase">++</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          a: [ 0, 1, 2, 3, 4]        },        methods: {          increase() {            for (let i=0; i < this.a.length; ++i)              Vue.set(this.a, i, this.a[i] + 1);          }        }      })     </script>  </body>  </html> |

노란색으로 칠한 부분을 제외하면, 앞의 예제와 동일하다.

### Vue.set(배열, 인덱스, 값)

배열의 인덱스 위치에 값을 대입한다.

Vue.set(this.a, i, this.a[i] + 1);

this.a 배열의 i 인덱스 위치에 (this.a[i] + 1) 값을 대입한다.

따라서 위 코드는 아래 코드와 동일하다.

this.a[i] = (this.a[i] + 1);

위 코드는 아래 코드와 동일하다.

this.a[i] += 1;

그런데 아래의 코드로 배열의 원소의 값을 수정되는 것은 Vue 엔진이 감지하지 못한다.

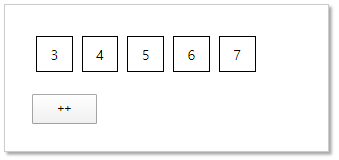
this.a[i] = (this.a[i] + 1);

this.a[i] += 1;

그래서 화면이 다시 그려지지 않는다.

Vue.set 메소드로 배열의 원소의 값이 수정되는 것은 Vue 엔진이 감지한다.

그래서 화면이 즉시 다시 그려진다.



++ 버튼을 클릭할 때 마다 화면이 다시 그려지기 때문에, 숫자들이 1 씩 증가하는 것을 볼 수 있다.

## 배열 복제

### update5.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 300px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; cursor: pointer; }      div#app > div { margin-top: 20px; }      button { padding: 0.5em 2em; }    </style>  </head>  <body>    <div id="app">      <span v-for="i in a">        {{ i }}      </span>      <div>        <button type="button" v-on:click="increase">++</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          a: [ 0, 1, 2, 3, 4]        },        methods: {          increase() {            let b = this.a.slice(0);            for (let i=0; i < b.length; ++i)              b[i] += 1;            this.a = b;          }        }      })     </script>  </body>  </html> |

노란색으로 칠한 부분을 제외하면, 앞의 예제와 동일하다.

#### 배열.slice(0)

배열의 slice 메소드를 이용하여 배열을 복제할 수 있다.

배열을 복제한 새 배열을 생성하여 리턴한다.

slice 메소드의 사용법을 복습하시오.



let b = this.a.slice(0);

this.a 배열을 복제하여 새 배열을 생성하고, 그 새 배열에 대한 참조를 지역변수 b에 대입한다.

메모리 구조가 위 그림에서 아래 그림으로 변화한다.



for (let i=0; i < b.length; ++i)

  b[i] += 1;

b 배열의 모든 원소를 1씩 증가시킨다



this.a = b;

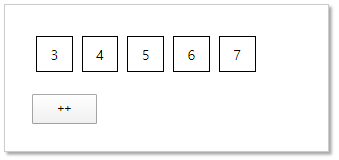
b 배열에 대한 참조를 this.a 속성에 대입한다.



Vue 앱 객체의 a 속성의 값이 변경되었다. (a 멤버 변수의 값이 변경되었다)

Vue 엔진이 이 변경을 감지한다.

그래서 화면이 즉시 다시 그려진다.



++ 버튼을 클릭할 때 마다 화면이 다시 그려지기 때문에, 숫자들이 1 씩 증가하는 것을 볼 수 있다.

## 기타 배열의 변경

### update6.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 600px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; cursor: pointer; }      div#app > div { margin-top: 20px; }      button { padding: 0.5em 2em; }    </style>  </head>  <body>    <div id="app">      <span v-for="i in a">        {{ i }}      </span>      <div>        <button type="button" v-on:click="increase">++</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          a: [ 0, 1, 2, 3, 4]        },        methods: {          increase() {            let value = this.a.length;            this.a.push(value);          }        }      })     </script>  </body>  </html> |

노란색으로 칠한 부분을 제외하면, 앞의 예제와 동일하다.

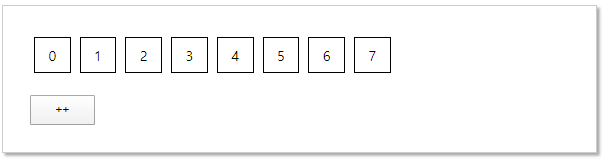
Vue 엔진이 감지하지 못하는 것은, 배열의 원소의 값이 변경되는 것 뿐이다.

그 외 다른 변경은 모두 감지한다.

배열의 메소드 호출에 의한 변경도 모두 감지한다.

++ 버튼을 클릭하면, 배열의 크기가 증가하는 것이 실시간으로 그려진다.

배열의 push 메소드는, 배열의 끝에 새 값을 추가한다. 배열의 크기가 1 증기한다.



# 객체의 속성 변경

## 객체의 속성값 변경

### update7.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 300px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; cursor: pointer; }      div#app > div { margin-top: 20px; }      button { padding: 0.5em 2em; }    </style>  </head>  <body>    <div id="app">      <span v-for="value in person">        {{ value }}      </span>      <div>        <button type="button" v-on:click="increase">++</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          person: { id: 101, name: "홍길동", age: 16 }        },        methods: {          increase() {            this.person.id += 1;            this.person.age += 1;          }        }      })     </script>  </body>  </html> |

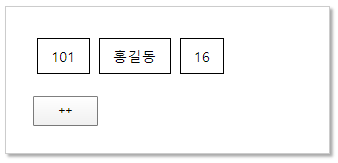
Vue 앱의 person 속성의 값은 객체이다.



Vue 엔진이 감지하지 못하는 것은, 배열의 원소의 값이 변경되는 것 뿐이다.

그 외 다른 변경은 모두 감지한다.

객체의 속성 변경도 감지되고, 즉시 화면이 다시 그려진다.



++ 버튼을 클릭하면 id 속성값고 age 속성값이 증가하고 화면이 다시 그려진다.

## 구조분해 할당 (destructing assignment)

### destructing3.js

|  |
| --- |
| let a, b;  a = 5;  b = 6;  console.log(a, b);  let temp = a;  a = b;  b = temp;  console.log(a, b); |

위 코드를 웹브라우저에서 실행할 필요 없다. (웹브라우저에서 실행 가능하다.)

서버의 node.js에서 실행하자.

단축키 Ctrl+F5를 누르면 서버의 콘솔창에서 실행된다.

출력

|  |
| --- |
| 5 6  6 5 |

let temp = a;

a = b;

b = temp;

a 변수의 값과 b 변수의 값을 서로 교환(swap)하는 코드이다.

### destructing4.js

|  |
| --- |
| let a, b;  [a, b] = [5, 6];  console.log(a, b);  [a, b] = [b, a];  console.log(a, b); |

[a, b] = [5, 6];

a 변수에 5 값이 대입되고,

b 변수에 6 값이 대입된다.

위 코드에 사용된 문법이 구조분해 할당(destructing assignment) 이다.

[a, b] = [b, a];

a 변수에 b 값이 대입되고,

b 변수에 a 값이 대입된다.

a 변수의 값과 b 변수의 값을 서로 교환(swap)하는 코드이다.

출력

|  |
| --- |
| 5 6  6 5 |

### destructing5.js

|  |
| --- |
| let a = [];  a[0] = 3;  a[1] = 4;  console.log(a);  let temp = a[0];  a[0] = a[1];  a[1] = temp;  console.log(a); |

a[0] = 3;

a[1] = 4;

a[0]에 3을 대입하고, a[1]에 4를 대입한다.

let temp = a[0];

a[0] = a[1];

a[1] = temp;

a[0] 값과 a[1] 값을 서로 교환(swap)하는 코드이다.

출력

|  |
| --- |
| 3 4  4 3 |

### destructing6.js

|  |
| --- |
| let a = [];  [a[0], a[1]] = [3, 4];  console.log(a);  [a[0], a[1]] = [a[1], a[0]];  console.log(a); |

[a[0], a[1]] = [3, 4];

a[0]에 3을 대입하고, a[1]에 4를 대입한다.

[a[0], a[1]] = [a[1], a[0]];

a[0] 값과 a[1] 값을 서로 교환(swap)하는 코드이다.

출력

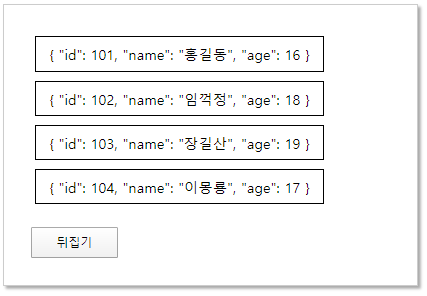
|  |
| --- |
| 3 4  4 3 |

## 객체 배열 변경

### update8.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; cursor: pointer; }      div#app > div { margin-top: 20px; }      button { padding: 0.5em 2em; }    </style>  </head>  <body>    <div id="app">      <span v-for="person in persons">        {{ person }}      </span>      <div>        <button type="button" v-on:click="reverse">뒤집기</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          persons: [ { id: 101, name: "홍길동", age: 16 },                     { id: 102, name: "임꺽정", age: 18 },                     { id: 103, name: "장길산", age: 19 },                     { id: 104, name: "이몽룡", age: 17 } ]        },        methods: {          reverse() {            let a = this.persons;            [a[0], a[3]] = [a[3], a[0]];            [a[1], a[2]] = [a[2], a[1]];          }        }      })     </script>  </body>  </html> |

앱의 persons 속성의 값은 객체 배열이다.



뒤집기 버튼을 클릭하면, perersons 배열의 원소 순서가 뒤집어진다.

즉 persons 배열의 원소의 값들이 바뀐다.

배열의 원소의 값이 바뀌는 것을 Vue 엔진이 감지하지 못하기 때문에

화면이 다시 그려지지 않는다.



let a = this.persons;

실행 결과



[a[0], a[3]] = [a[3], a[0]];

[a[1], a[2]] = [a[2], a[1]];

실행 결과



## this.$forceUpdate()

### update9.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; cursor: pointer; }      div#app > div { margin-top: 20px; }      button { padding: 0.5em 2em; }    </style>  </head>  <body>    <div id="app">      <span v-for="person in persons">        {{ person }}      </span>      <div>        <button type="button" v-on:click="reverse">뒤집기</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          persons: [ { id: 101, name: "홍길동", age: 16 },                     { id: 102, name: "임꺽정", age: 18 },                     { id: 103, name: "장길산", age: 19 },                     { id: 104, name: "이몽룡", age: 17 } ]        },        methods: {          reverse() {            let a = this.persons;            [a[0], a[3]] = [a[3], a[0]];            [a[1], a[2]] = [a[2], a[1]];            this.$forceUpdate();          }        }      })     </script>  </body>  </html> |

노란색으로 칠한 부분을 제외하면, 앞의 예제와 동일하다.

this.$forceUpdate(); 메소드는 화면이 다시 그려지도록 강제하기 때문에

뒤집기 버튼을 클릭하면 배열의 원소의 순서가 뒤집어지고 화면이 다시 그려진다.

## 객체 배열 복제

### update10.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; cursor: pointer; }      div#app > div { margin-top: 20px; }      button { padding: 0.5em 2em; }    </style>  </head>  <body>    <div id="app">      <span v-for="person in persons">        {{ person }}      </span>      <div>        <button type="button" v-on:click="reverse">뒤집기</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          persons: [ { id: 101, name: "홍길동", age: 16 },                     { id: 102, name: "임꺽정", age: 18 },                     { id: 103, name: "장길산", age: 19 },                     { id: 104, name: "이몽룡", age: 17 } ]        },        methods: {          reverse() {            let a = this.persons.slice(0);            [a[0], a[3]] = [a[3], a[0]];            [a[1], a[2]] = [a[2], a[1]];            this.persons = a;          }        }      })     </script>  </body>  </html> |

노란색으로 칠한 부분을 제외하면, 앞의 예제와 동일하다.



let a = this.persons.slice(0);

배열이 복제된다.

실행 결과



slice 메소드로 배열을 복제하는 것은, 얕은 복사(shallow copy)이기 때문에,

배열만 복제되고, 배열의 원소가 참조하는 대상 객체는 복제되지 않는다.

깊은 복사(deep copy)를 하면

배열도 복제되고, 배열의 원소가 참조하는 대상 객체도 복제된다.

[a[0], a[3]] = [a[3], a[0]];

[a[1], a[2]] = [a[2], a[1]];

지역 변수 a가 참조하는 배열의 순서를 변경한다.

실행결과



this.persons = a;

실행결과



Vue 앱 객체의 persons 속성의 값이 변경되었다. (persons 멤버 변수 값이 변경되었다)

Vue 엔진이 이 변경을 감지한다.

그래서 화면이 즉시 다시 그려진다.

## 배열 메소드

### update11.html

|  |
| --- |
| <html lang="ko">  <head>    <meta charset="utf-8" />    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>    <style>      div#app { padding: 30px; margin: 30px auto; max-width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa;  }      span { display: inline-block; margin: 5px; padding: 8px 15px;        border: 1px solid black; cursor: pointer; }      div#app > div { margin-top: 20px; }      button { padding: 0.5em 2em; }    </style>  </head>  <body>    <div id="app">      <span v-for="person in persons">        {{ person }}      </span>      <div>        <button type="button" v-on:click="reverse">뒤집기</button>      </div>    </div>    <script type="text/javascript">      var app = new Vue({        el: '#app',        data: {          persons: [ { id: 101, name: "홍길동", age: 16 },                     { id: 102, name: "임꺽정", age: 18 },                     { id: 103, name: "장길산", age: 19 },                     { id: 104, name: "이몽룡", age: 17 } ]        },        methods: {          reverse() {            this.persons.reverse();          }        }      })     </script>  </body>  </html> |

노란색으로 칠한 부분을 제외하면, 앞의 예제와 동일하다.

배열의 reverse 메소드는, 배열의 원소의 순서를 뒤집는다.



this.persons.reverse();

실행결과



Vue 엔진이 감지하지 못하는 것은, 배열의 원소의 값이 변경되는 것 뿐이다.

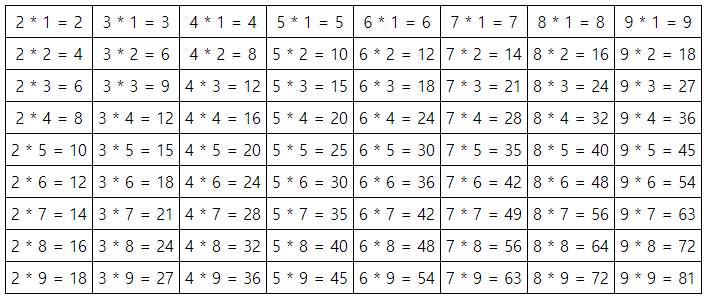
그 외 다른 변경은 모두 감지한다.

배열의 메소드 호출에 의한 변경도 모두 감지한다.

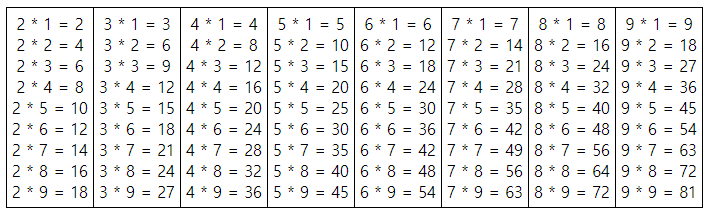
그래서 뒤집기 버튼을 클릭하면, 배열의 순서가 뒤집어지고, 즉시 화면이 다시 그려진다.

# 과제

## gugu1.html



## gugu2.html



위와 같이 출력되도록 구현하시오.

### 주의사항

구구단을 미리 다 입력해서 구현하면 당연히 안되고,

중첩된 v-for 를 이용해서 계산해서 출력하는 방식으로 구현해야 됨.

경계선도 위와 같이 그려져야 함.

어떤 태그를 사용해서 구현해도 됨.

### 잔소리

저 학년 학생들은, 이제 막 시작하는 단계라서 이 과제 구현이 쉽지 않을 텐데,

(아직도 기억할지 모르겠지만) 어렸을 때 처음 구구단 외우고 나눗셈 배울 때도 힘들었다.

하루에 몇 시간씩 4~5년 꾸준히 훈련해야 한 사람의 개발자가 될 수 있으니,

벌써 지치거나 좌절하지 말고, 매일 몇 시간씩 훈련하자.

전문 직종 중에서 몇 년 훈련이 필요 없는 분야가 있을 것 같은가?

처음부터 쉽게 잘하는 친구들과 자신을 비교하며 우울한 학생이 있을 텐데,

그 학생들은 처음부터 (프로그래밍이 재미있어서) 혼자 이것 저것 구현해 보는 훈련(놀이)를 했기 때문이지,

하나를 배우면 열을 깨우치는 천재라서가 아니다.

성공회대 학생 중에 그런 천재는 없어요.

과제와 시험공부만 해서, 영어 번역가나 통역가가 결코 될 수 없듯이

과제와 시험공부만 해서, 개발자가 결코 될 수 없어요.