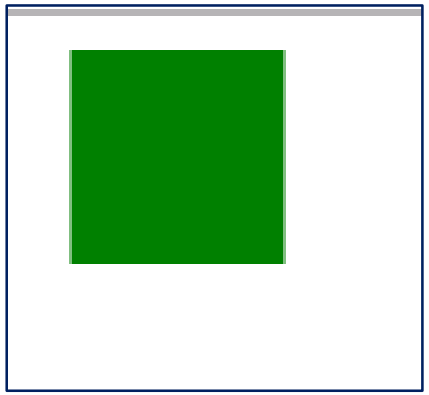
Vue Transition

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

CSS Transition

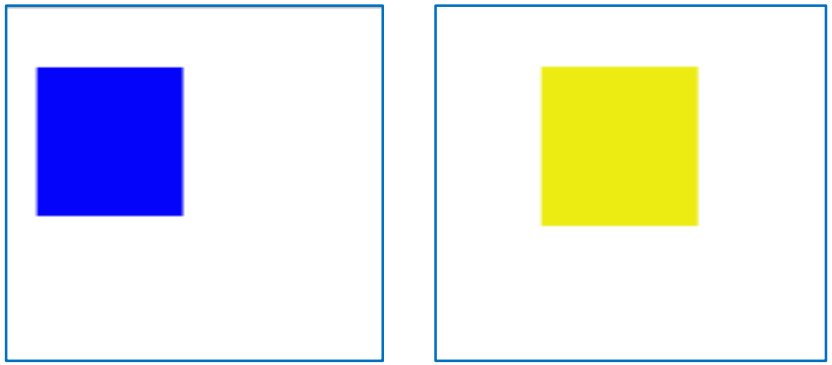


```
    <script src="https://cdn.jsdelivr.net/npm/vue@2.6.10/dist/vue.js"></script>
    <style>
        .box {
            background-color: blue;
            width: 50px;
            height: 50px;
        }

        .box:hover {
            transition: 0.5s;
            transform: translateX(10px);
            background-color: green;
        }
    </style>
</head>
<body>
    <div class="box"></div>
</body>
```

단순히 CSS Transition만 이용하면 두 값 사이를 일정한 속도로 서서히 변화시키면서 Animation 효과를 가져온다.

CSS Transition (Cont.)



진행 속도나 단계를 직접 지정하려면 @keyframes 속성을 사용하면 된다.

```
<style>
    .box {
        background-color: blue;
        width: 50px;
        height: 50px;
        position: absolute;
        top: 20px;
        left: 30px;
    }
    @keyframes shake-box {
        0% {
            transform: translateX(-20px);
            background-color: blue;
        }
        50% {
            transform: translateX(10px);
            background-color: yellow;
        }
        100% {
            transform: translateX(-20px);
            background-color: blue;
        }
    }
    .box:hover {
        animation: shake-box 0.2s infinite;
    }
</style>
```

Transition

- CSS Transition/Animation을 더욱 간단하게 사용할 수 있게 지원해 주는 기능.
- 최근 Web Trend는 UI가 부드럽게 전환되고 Animation 처리 등 기능이 제공되는 여부도 중요해 졌다.
- 화면 전환 시에 부여하는 효과를 **Transition Effect**라고 하며, Web Application에서는 CSS Style을 이용해서 Transition 기능을 지원 가능.
- Vue.js에서는 Element가 DOM에 추가, 제거되거나 변경될 때 Vue.js가 **자동으로** CSS Transition/Animation 관련 Class를 적용하여 Style을 적용해 준다.
- Vue.js의 Transition은 이러한 Design을 위해 제공되는 것
- 굉장히 강력하고 세련된 CSS Transition을 **간단하게** 만들 수 있게 해준다.
- Vue.js는 **Transition Wrapper Component**를 제공하여 간단히 Transition을 적용하는 방법부터 Programming 방식의 동적 상태 전환 기능도 제공.
- Animation 효과는 Transition Wrapper Component로 감싸진 Element, Component, Router View가 삭제되거나 추가되거나 변경될 때 발생.

Transition (Cont.)

```
 9      <style>
10          .box {
11              margin: 10px;
12          }
13
14          .fade-enter-active,
15          .fade-leave-active {
16              transition: opacity .8s
17          }
18
19          .fade-enter,
20          .fade-leave-to {
21              opacity: 0
22          }
23      </style>
```

Vue.js는 CSS Transition class들을 손쉽게 사용할 수 있도록 Transition wrapper Component : <transition></transition> 을 지원

```
25  <body>
26      <div id="app">
27          <div class="box">
28              <button v-on:click="changeVisible">보여주기 토글</button>
29          </div>
30          <div class="box">
31              <transition name="fade">
32                  <img src="images/paris.jpg" v-if="visible" />
33              </transition>
34          </div>
35      </div>
36      <script>
37          var v = new Vue({
38              el: '#app',
39              data: function () {
40                  return {
41                      visible: true
42                  }
43              },
44              methods: {
45                  changeVisible: function () {
46                      this.visible = !this.visible;
47                  }
48              }
49          })
50      </script>
51  </body>
```

Transition (Cont.)

보여주기 토글

Transition (Cont.)

1. Transition 효과를 적용하고 싶은 요소를 아래와 같이 **<transition>** tag로 감싸기만 하면 Transition 전용 Class를 사용할 수 있다.

```
<transition>
    <div v-show='show'>Element</div>
</transition>
```

2. Transition Class를 Style로 정의한다.
3. Vue.js가 ** ** tag의 추가/소멸 시점에 Class를 동적으로 조작해서 Transition을 적용시킨다.
4. 실제로 **<transition>**으로 감싼 Element가 DOM에 추가될 때는 **enter**, DOM에서 제거될 때에는 **leave**라는 문자를 포함한 Class가 Transition Class로 적용된다.

Transition (Cont.)

- **Transition CSS Class**

Transition CSS Class	Description
v-enter	Element가 나타나기 시작할 때 적용할 Class
v-enter-active	Element가 나타나는 Transition이 진행되는 동안 적용할 Class
v-enter-to	Element가 나타나는 Transition이 완료될 때 적용할 Class
v-leave	Element가 사라지기 시작할 때 적용할 Class
v-leave-active	Element가 사라지는 Transition이 진행되는 동안 적용할 Class
v-leave-to	Element가 사라지는 Transition이 완료될 때 적용할 Class

Transition CSS Class 개념

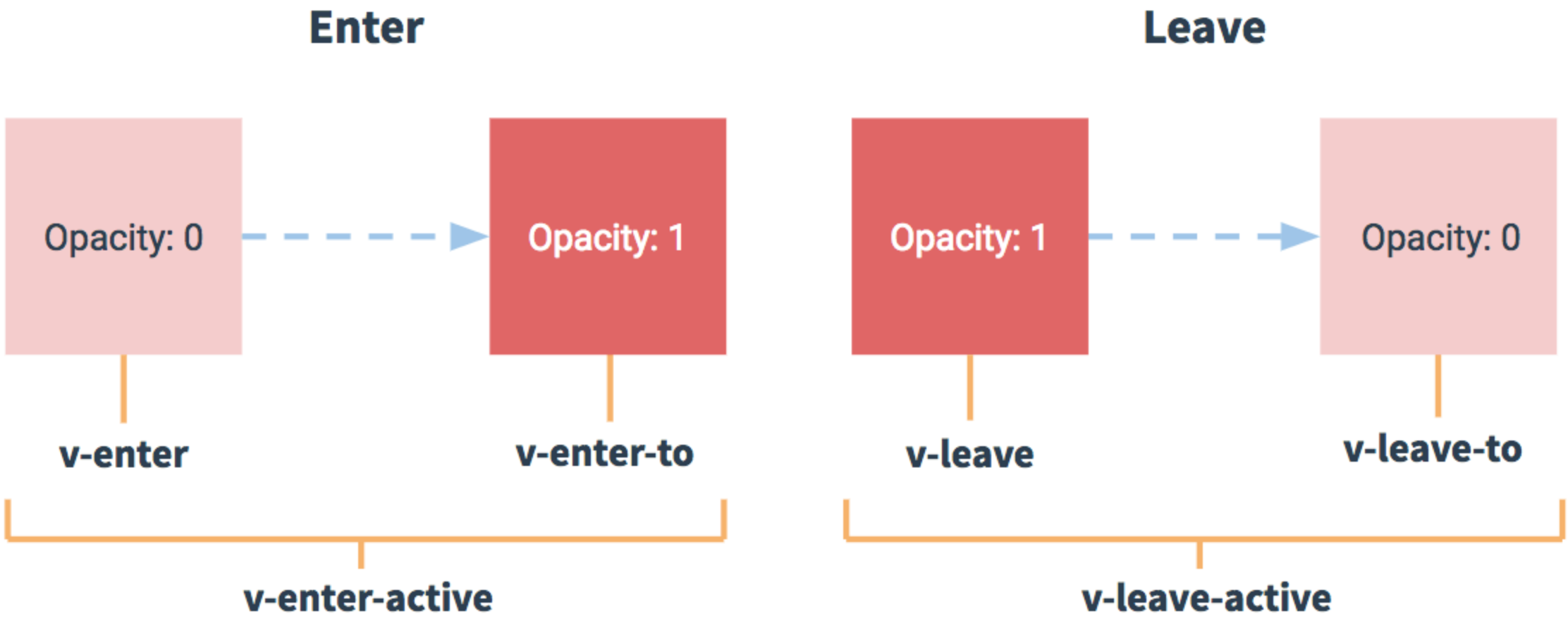


Image from : https://vuejs.org/v2/guide/transitions.html

Transition Wrapper Component

- Vue.js는 Element를 노출시키는 등의 Transition Animation을 구현하기 위해 사용하는 Component.
- 자신이 감싸고 있는 Component 혹은 Element의 노출 상태에 따라 Animation 처리를 맡는 Component.
- Animation의 시작부터 끝에 이르기까지 각 과정에서 v-enter, v-enter-active등 Vue.js에 미리 정의된 Class를 적용 및 해제하는 역할 수행.
- 해당 Class가 적용되는 Element(Directive)와 조건

```
<transition>
    <!– transition component,  여기에 속하는 element를 대상으로 Animation을 만듦 -->
</transition>
```

Transition Wrapper Component (Cont.)

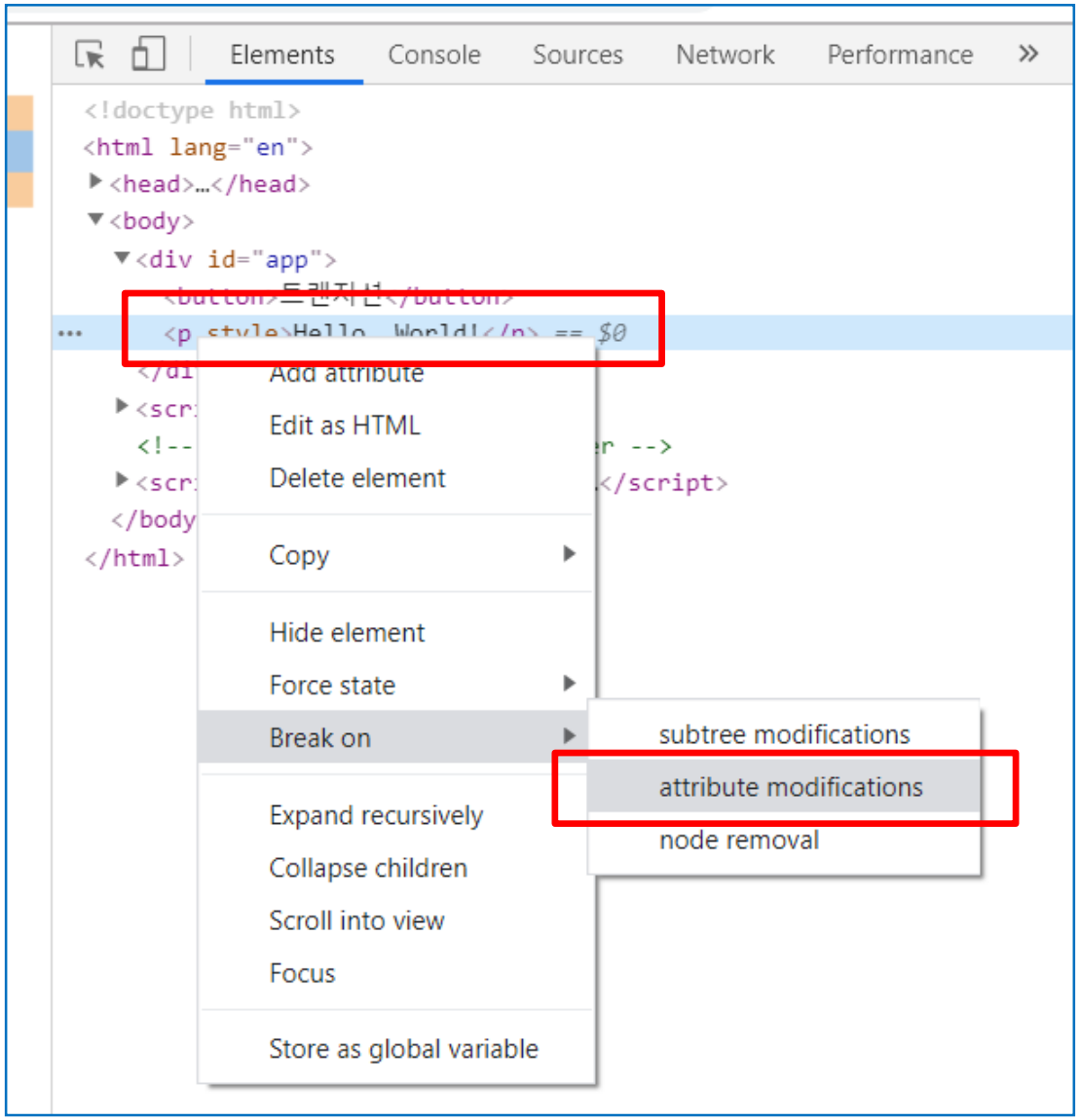
- Vue.js는Transition Component는 자신이 감싸고 있는 Component 혹은 Element가 노출되거나 사라질 때(enter / leave) Transition을 추가한다.
- 노출 혹은 사라짐의 조건
  - `v-if` 조건의 평가 값이 변한 경우
  - `v-show` 조건의 평가 값이 변한 경우
  - 동적 Component의 `is` 속성값이 변한 경우
- 기본으로 제공되는 Class명은 접두사 `v-`가 붙지만, `name` 속성에서 Class명을 변경가능하다.
- `<transition name='fade'>`라고 지정하면, `v-enter`가 아니라 `fade-enter`를 사용하게 된다.

Transition Wrapper Component (C

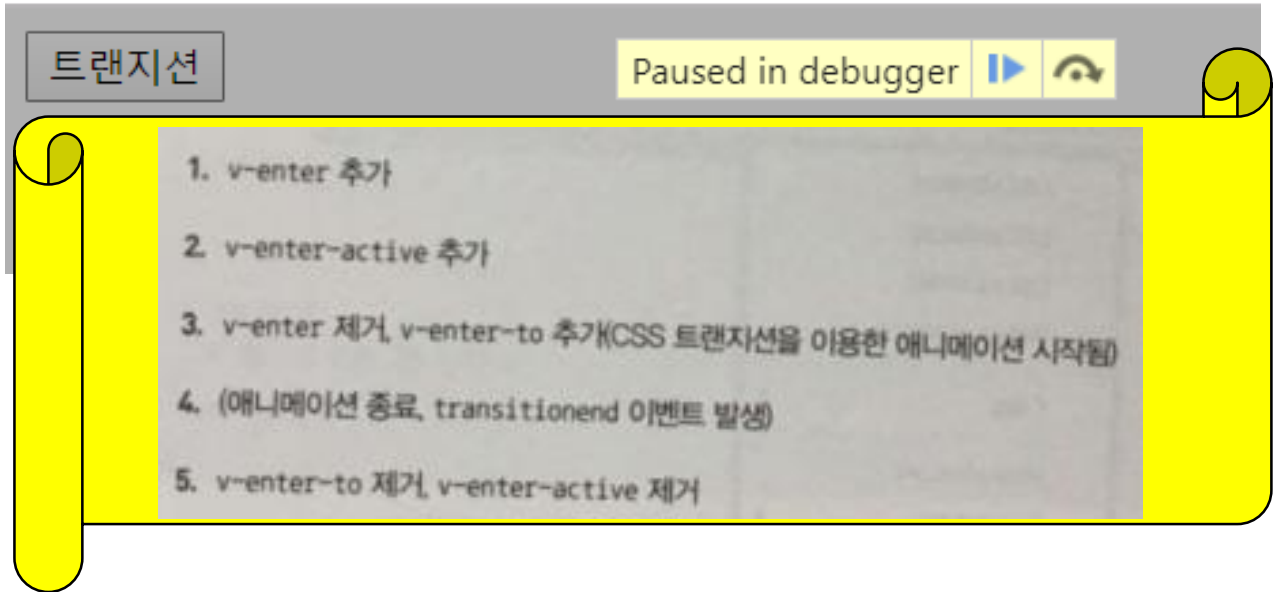
```
8     <script src="https://cdn.jsdelivr.net/npm/vue@2.6.10/dist/vue.js"
9     <link rel="stylesheet" href="css/style.css">
10  </head>
11  <body>
12      <div id="app">
13          <button @click="isShown = !isShown">트랜지션</button>
14          <transition>
15              <p v-show="isShown">Hello, World!</p>
16          </transition>
17      </div>
18      <script>
19          var v = new Vue({
20              el: '#app',
21              data: function () {
22                  return {
23                      isShown : false
24                  }
25              }
26          })
27      </script>
28  </body>
```


```
1   .v-enter-active,
2   .v-leave-active {
3       /* Animation 지속시간, Easing 등을 설정 */
4       transition:opacity 500ms ease-out;
5   }
6
7   /* Fade-in */
8   .v-enter {
9       /* Fade-in 처음 상태 */
10      opacity: 0;
11  }
12
13  .v-enter-to {
14      /* Fade-in 끝 상태 */
15      opacity: 1;
16  }
17
18  /* Fade-out */
19  .v-leave {
20      /* Fade-out 처음 상태 */
21      opacity: 1;
22  }
23
24  .v-leave-to {
25          /* Fade-out 끝 상태 */
26      opacity: 0;
27  }
```

Transition Wrapper Component (Cont.)




```
7866     */
7867   function addClass (el, cls) {  el = p {__vOriginalDisplay: ""
7868     /* istanbul ignore if */
7869     if (!cls || !(cls = cls.trim())) {  cls = "v-enter"
7870       return
7871     }
7872
7873     /* istanbul ignore else */
7874     if (el.classList) {  el = p {__vOriginalDisplay: "", __vue_
7875       if (cls.indexOf(' ') > -1) {  cls = "v-enter"
7876         cls.split(whitespaceRE).forEach(function (c) { return e
7877       } else {
7878         el.classList.add(cls);
7879       }
7880     } else {
```



Lab. Transition Wrapper Component

```
    <script src="https://cdn.jsdelivr.net/npm/vue@2.6.10/dist/vue.js"></script>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div id="app">
        <button @click="count++">변경하기</button>
        <transition>
            <div v-bind:key="count">{{ count }}</d
        </transition>
    </div>
    <script>
        var v = new Vue({
            el: '#app',
            data: {
                count : 0
            }
        })
    </script>
```

```
.v-enter-active, .v-leave-active {
    transition:opacity 1s;
}

.v-leave-active {
    position: absolute;
}

.v-enter, .v-leave-to {
    opacity: 0;
}
```

변경하기

6

List Transition

- 여러 개의 Element를 Group化해서 추가, 삭제, 이동 Animation을 부여할 수 있다.
- **<transition-group>** tag 사용
- **tag** 속성으로 tag 이름 사용
- 일반적으로 **v-for**를 적용할 요소에 Transition을 적용하는 경우에 사용.

```
<transition-group name='list' tag='ul'>
    <li v-for='item in list' v-bind:key='item.id'></li>
</transition-group>
```

- **tag** 속성을 생략한 경우에는 **span** 요소로 wrapping 됨.
- 주의할 점은, Component가 아닐 경우에도 반드시 **key**를 설정해야 한다.
- 이유는, Transition을 사용하면 내부적으로 순서가 바뀌기 때문에 반드시 **key**를 지정해서 Element들을 구분할 수 있게 해야 한다.

List Transition (Cont.)

- List의 요소를 추가 또는 제거하거나 조건으로 Rendering 상태를 변경할 때, 해당 Element에 enter와 leave 계열의 Transition Class가 적용됨.
- 순서가 바뀌었을 때도 Transition이 이루어진다.
- Element가 추가 또는 삭제 될 때 정렬 순서 변경으로 요소가 움직일 때, 해당 Element에 **.v-move**라는 Transition Class가 적용된다.

```
/* 1초 동안 요소를 움직이기  */
.v-move {
     transition : transform 1s;
}
```

Lab. List Transition

```
8     <script src="https://cdn.jsdelivr.net/npm/vue@2.6.10/dist/vue.js"></script>
9     <script src="https://cdn.jsdelivr.net/npm/lodash@4.17.15/lodash.min.js"></script>
10    <link rel="stylesheet" href="css/style.css">
11  </head>
13    <div id="app">
14        <button @click="order=!order">변경하기</button>
15        <!-- transition-group tag로 지정한 속성은 wrap element에 추가하기-->
16        <transition-group tag='ul' class='list'>
17            <li v-for='item in sortedList' v-bind:key='item.id'>
18                {{ item.name }} {{ item.price }} 원
19            </li>
20        </transition-group>
21    </div>
22    <script>
23        var v = new Vue({
24            el: '#app',
25            data: {
26                order : false,
27                list : [
28                    { id:1, name : '사과', price : 100},
29                    { id:2, name : '바나나', price : 200},
30                    { id:3, name : '딸기', price : 300}
31                ]
32            },
33            computed : {
34                // order 값에 따라 list의 순서를 반전하는 산출 속성
35                sortedList : function(){
36                    return _.orderBy(this.list, 'price', this.order ? 'desc' : 'asc')
37                }
38            }
39        })
40    </script>
```

```
1   /* 1초 동안 요소를 움직이기 */
2   .v-move {
3       transition : transform 1s;
4   }
5
```

변경하기

- 딸기 300 원
- 바나나 200 원
- 사과 100 원