트랜지션 효과

좋은 웹 애플리케이션의 요건

- 정확한 기능, 성능, 빠른 로딩 시간 등
- + 부드러운 화면 전환, 애니메이션 처리 등의 기능 제공 여부

트랜지션 효과

- 화면 전환시의 효과
- CSS 스타일을 이용해 트랜지션 기능 지원

Vue.js는...

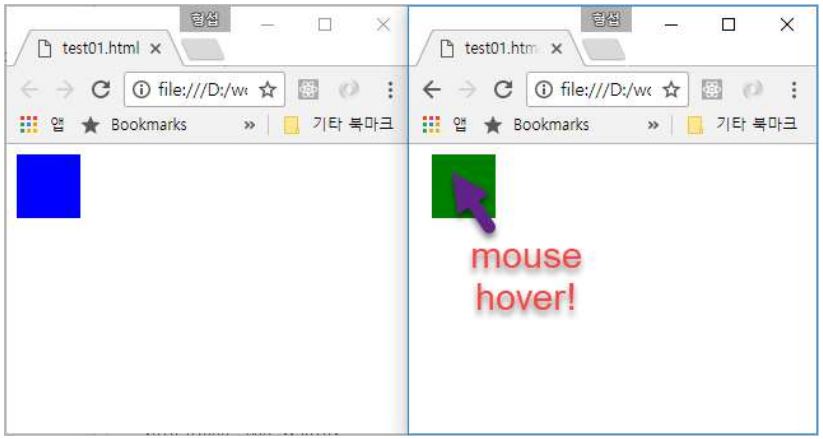
- <transition /> 래퍼 컴포넌트 지원
- 프로그래밍 방식의 동적 트랜지션 효과 지원

CSS 트랜지션 기초

CSS 트랜지션

- CSS 속성 값을 서서히 변경해나가도록 하여 트랜지션을 적용함
- 단순한 HTML + CSS 조합만으로도 가능함
- 예제 13-01 : 13-01.html

```
<style>
.box {
    background-color: blue;
    width:50px; height:50px;
}
.box:hover {
    transition: 0.5s;
    transform: translateX(10px);
    background-color: green;
}
</style>
……
<body>
    <div class="box"></div>
</body>
```

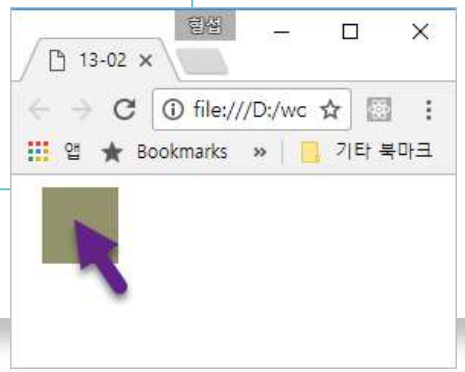


CSS 트랜지션 기초

진행 속도나 단계를 직접 지정하려면…

- @keyframes 속성 사용

```
<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>
……
<body>
    <div class="box"></div>
</body>
```



트랜지션 컴포넌트 기초

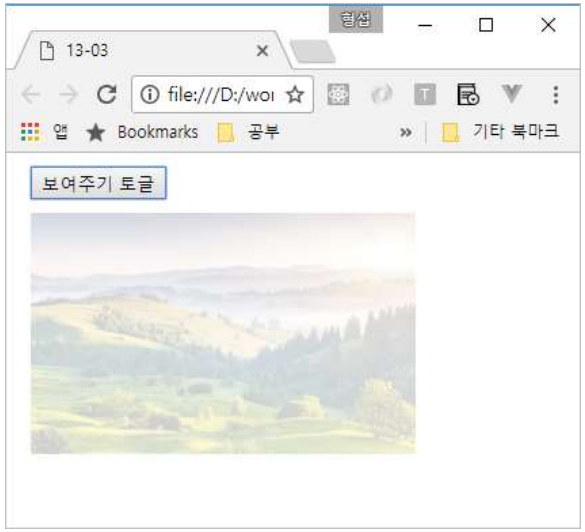
Vue.js에서는...

- 트랜지션 CSS 클래스들을 손쉽게 적용할 수 있도록 <transition /> 래퍼 컴포넌트 지원
  - 모든 요소, 컴포넌트, <router-view />를 감싸주는 것만으로도 효과 적용
- 예제 13-03 : 13-03.html

```
 <style>
   .box { margin:10px; }
   .fade-enter-active, .fade-leave-active {
      transition: opacity .5s }
   .fade-enter, .fade-leave-to { opacity: 0  }
 </style>
……
<body>
 <div id="app">
   <div class="box">
     <button v-on:click="changeVisible">
        보여주기 토글</button>
   </div>
   <div class="box">
     <transition name="fade">
       <img src="hill.jpg" v-if="visible" />
     </transition>
   </div>
</div>
</body>
<script type="text/javascript">
Vue.config.devtools = true;
var v = new Vue({
  el : '#app',
  data : function() {
    return {
      visible:true
    }
  },
  methods : {
    changeVisible : function() {
      this.visible = !this.visible;
    }
  }
})
</script>
```

트랜지션 컴포넌트 기초

예제 13-03 실행 결과



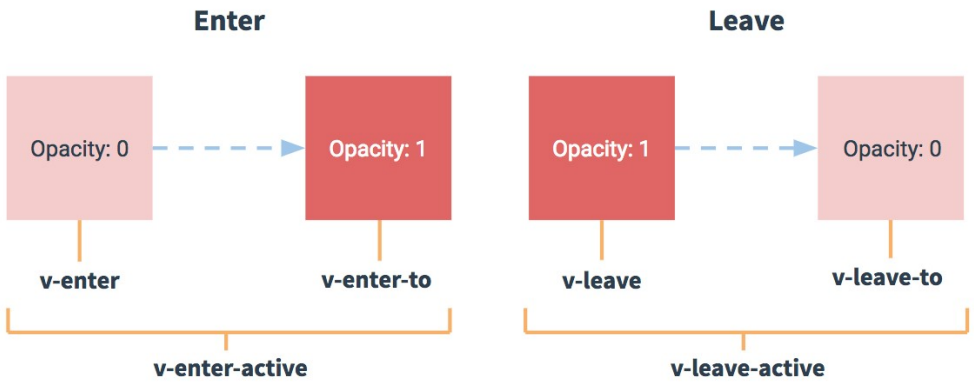
- opacity CSS 속성 : 요소의 불투명도를 지정함(0~1)
  - enter 트랜지션이 시작하면 0에서 1로 0.5초(.5s) 동안 서서히 전환됨
  - leave 트랜지션이 시작하면 1에서 0으로 서서히 전환됨.

트랜지션 컴포넌트 기초

트랜지션 CSS 클래스 개념

표 13-01 트랜지션 CSS 클래스

트랜지션 CSS 클래스	설명
v-enter	요소가 나타나기 시작할 때 적용할 클래스
v-enter-active	요소가 나타나는 트랜지션이 진행되는 동안 적용할 클래스
v-enter-to	요소가 나타나는 트랜지션이 완료될 때 적용할 클래스
v-leave	요소가 사라지기 시작할 때 적용할 클래스
v-leave-active	요소가 사라지는 트랜지션이 진행되는 동안 적용할 클래스
v-leave-to	요소가 사라지는 트랜지션이 완료될 때 적용할 클래스



CSS 애니메이션 처리

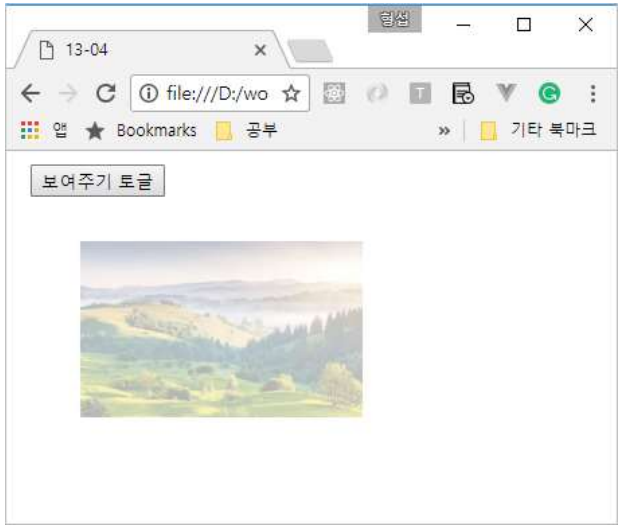
<transition /> 래퍼 컴포넌트가 @keyframes 지원

- 예제 13-04 : 13-04.html

```
<style>
.box { margin:10px; }
.elastic-enter-active { animation: elastic-in 0.5s; }
.elastic-leave-active {
   animation: elastic-in 0.5s reverse;  }
@keyframes elastic-in {
  0% { transform: scale(0); opacity:0;}
  70% { transform: scale(1.2); opacity:0.5; }
  100% { transform: scale(1); opacity:1; }
}
</style>
……
<body>
  <div id="app">
    <div class="box">
      <button v-on:click="changeVisible">
          보여주기 토글</button>
    </div>
    <div class="box">
      <transition name="elastic">
        <img src="hill.jpg" v-if="visible" />
      </transition>
    </div>
  </div>
</body>
<script type="text/javascript">
Vue.config.devtools = true;
var v = new Vue({
  el : '#app',
  data : function() {
    return {
      visible:true
    }
  },
  methods : {
    changeVisible : function() {
      this.visible = !this.visible;
    }
  }
})
</script>
……
```

CSS 애니메이션 처리

예제 13-04 실행 결과

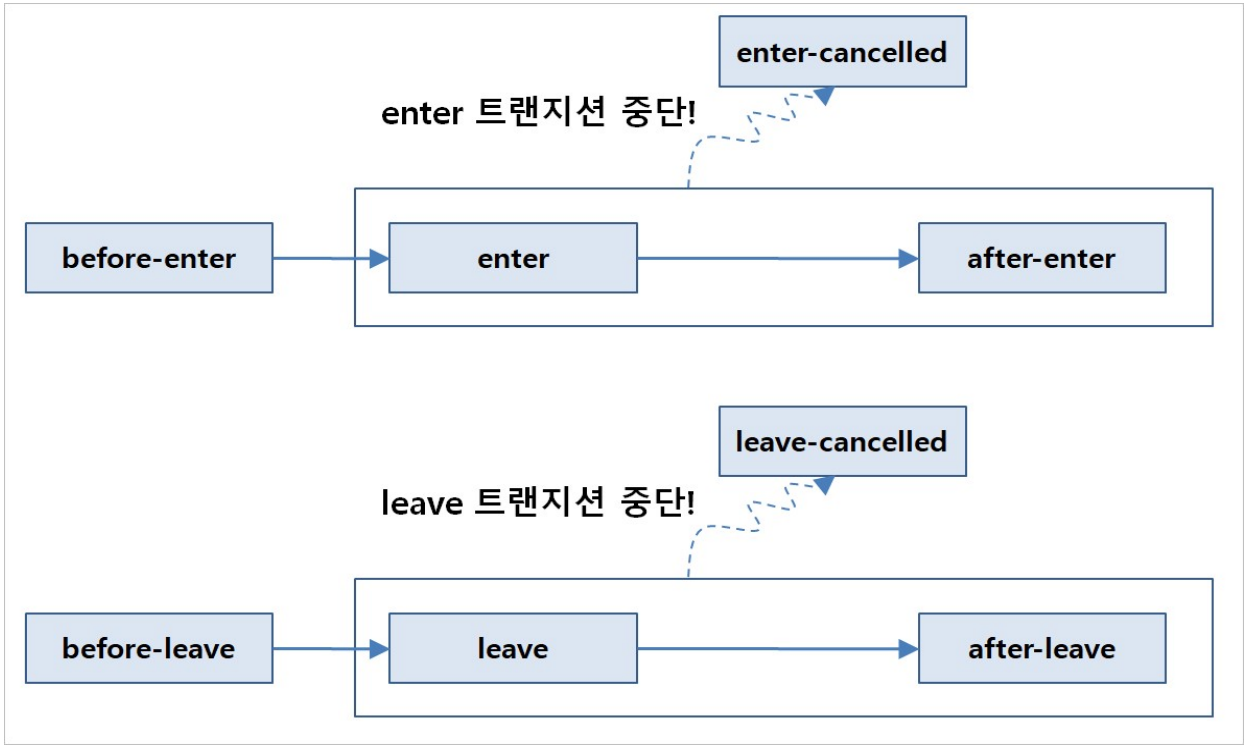


- elastic 효과

트랜지션 이벤트 훅

이벤트 훅

- 트랜지션이 시작되거나 종료될 때의 이벤트 훅을 제공함.



트랜지션 이벤트 훅

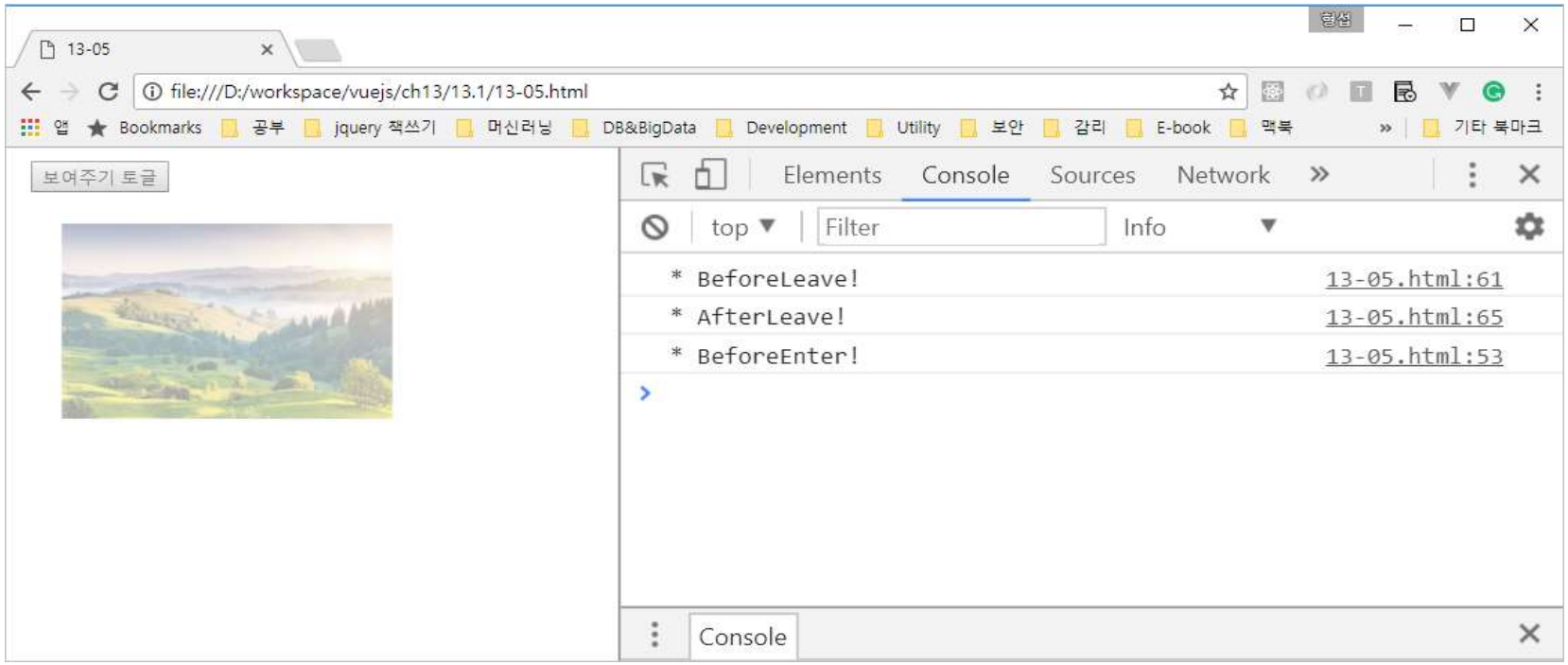
- 예제 13-05 : 13-05.html

```
<div class="box">
  <transition name="elastic"
    @before-enter="elasticBeforeEnter"
    @after-enter="elasticAfterEnter"
    @before-leave="elasticBeforeLeave"
    @after-leave="elasticAfterLeave">
    <img src="hill.jpg" v-if="visible" />
  </transition>
</div>

……
<script type="text/javascript">
Vue.config.devtools = true;
var v = new Vue({
  el : '#app',
  data : function() {
    return {
      visible:true,
      buttonEnabled:true
    }
  },
  methods : {
    changeVisible : function() {
      this.visible = !this.visible;
    },
    elasticBeforeEnter : function() {
        console.log("* BeforeEnter!");
        this.buttonEnabled = false;
    },
    elasticAfterEnter : function() {
        console.log("* AfterEnter!");
        this.buttonEnabled = true;
    },
    elasticBeforeLeave : function() {
        console.log("* BeforeLeave!");
        this.buttonEnabled = false;
    },
    elasticAfterLeave : function() {
        console.log("* AfterLeave!");
        this.buttonEnabled = true;
    }
  }
})
</script>
```

트랜지션 이벤트 훅

예제 13-05 실행 결과



트랜지션 이벤트 훅

트랜지션 이벤트 훅을 사용하면...

- velocity.js와 같은 JS 기반의 고성능 애니메이션 라이브러리를 사용할 수 있음
- 패키지 다운로드
  - npm install --save velocity-animate
- 단일 파일 컴포넌트에서 사용
  - import Velocity from 'velocity-animate'

Velocity.js란?

Velocity.js는 웹사이트의 애니메이션을 지원하는 브라우저에서 실행되는 무료로 사용할 수있는 오픈소스 자바스크립트 라이브러리입니다. Velocity.js의 문법은 CSS로 작성하던 애니메이션을 쉽게 작성할 수 있도록 디자인되었으며 CSS 기반 애니메이션 효과와 비교해 거의 동일한 수준의 성능을 제공합니다. 더 자세한 내용은 http://velocityjs.org/ 를 참조하세요.

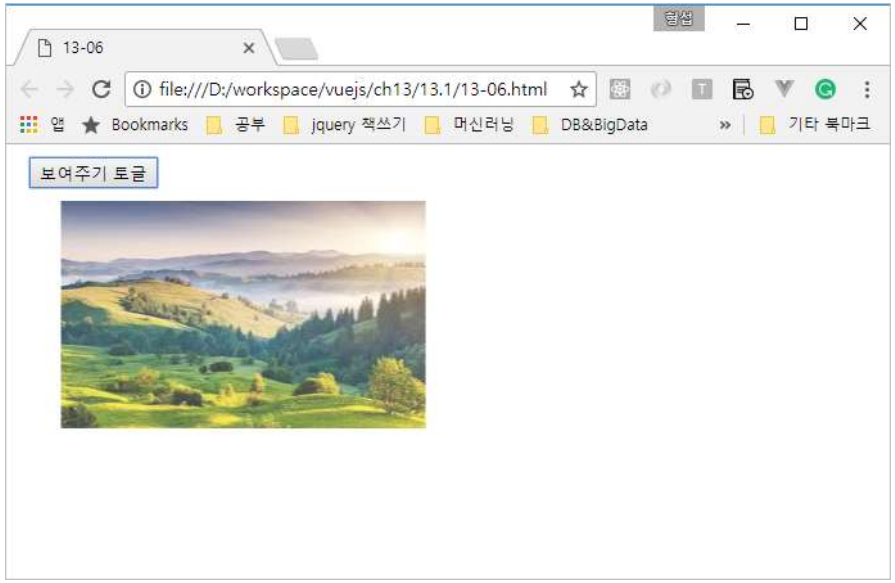
트랜지션 이벤트 훅

- 예제 13-06 : 13-06.html

```
……
  methods : {
    changeVisible : function() {
      this.visible = !this.visible;
    },
    beforeEnter: function (el) {
      el.style.opacity = 0
    },
    enter: function (el, done) {
      Velocity(el, { opacity: 0, scale:0.2 }, { duration: 200 })
      Velocity(el, { opacity: 0.7, scale:1.2 }, { duration: 200 })
      Velocity(el, { opacity: 1, scale:1 }, { complete: done })
    },
    leave: function (el, done) {
      Velocity(el, { translateX: '0px', opacity: 1 }, { duration: 100 })
      Velocity(el, { translateX: '20px', opacity: 1 }, { duration: 100, loop:2 })
      Velocity(el, { translateX: '0px', opacity: 1 }, { duration: 200 })
      Velocity(el, { translateX: '100px', opacity: 0 }, { complete:done })
    }
  }
……
```

트랜지션 이벤트 훅

예제 13-06 실행 결과



- 나타날 때는 Elastic 효과
- 사라질때는 까딱까딱~ 휘익!

리스트에 대한 트랜지션

- **<transition /> 래퍼 컴포넌트**
  - 단일 요소, 단일 컴포넌트에 대한 트랜지션 효과 적용
- **<transition-group /> 래퍼 컴포넌트**
  - v-for 디렉티브를 이용해 반복 렌더링 하는 요소들에 대해 트랜지션 적용
  - v-for로 반복되는 요소에 반드시 key 특성을 부여해야 함
    - 각 반복 요소에 대하 고유값이 주어지도록...
  - <transition-group>의 tag 특성에 반드시 이전 감싸는 요소를 명시

리스트에 대한 트랜지션

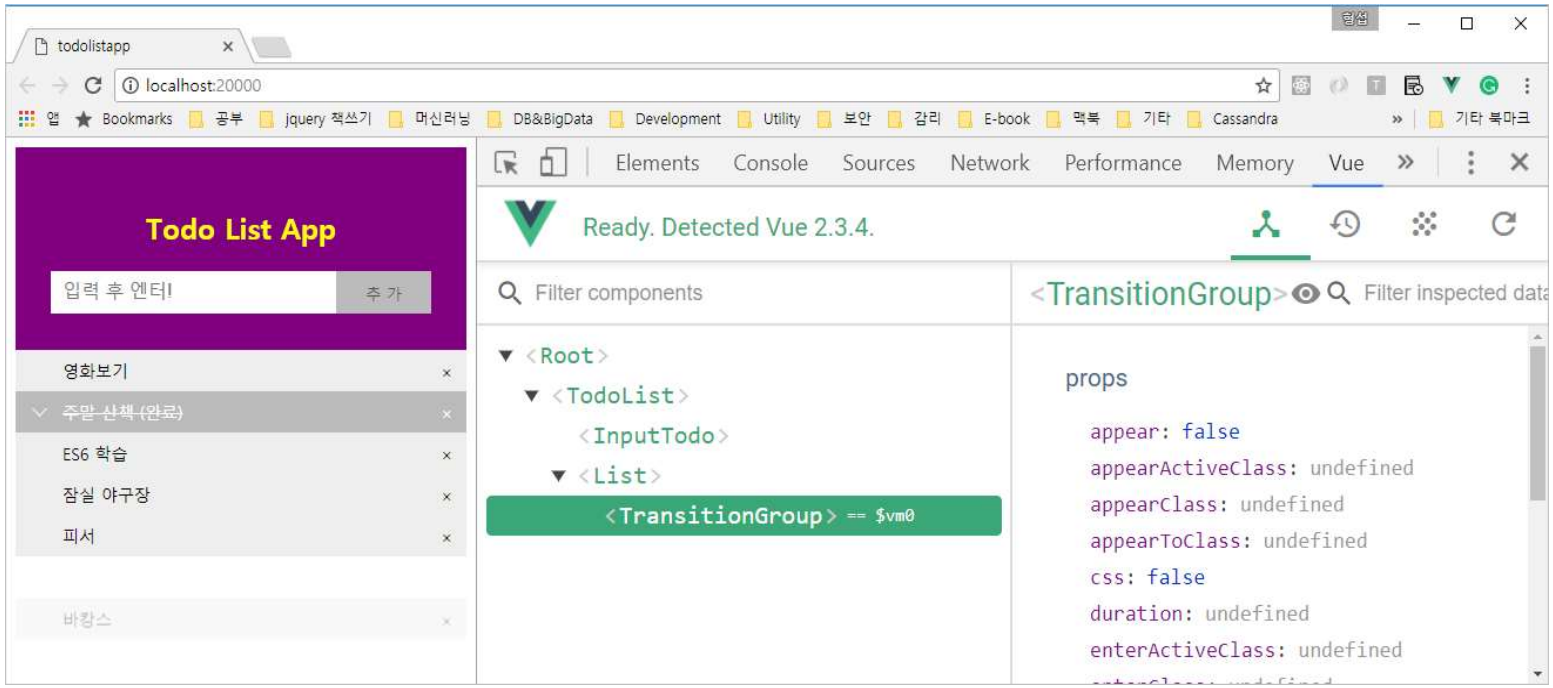
예제 13-07

```
<ul id="todolist">
  <li v-for="(a, index) in todolist" v-bind:class="checked(a.done)"
      v-on:click="doneToggle({index:index})">
      ......
  </li>
</ul>
```

```
<transition-group name="list" tag="ul">
  <li v-for="(a, index) in todolist" v-bind:class="checked(a.done)"
     v-on:click="doneToggle({index:index})" :key="index">
     ......
  </li>
</transition-group>
```

리스트에 대한 트랜지션

예제 13-07 실행 결과



연락처 애플리케이션에 트랜지션 적용하기

지금까지 학습한 내용을 바탕으로 연락처 앱에 적용

- 예제 13-09 : src/App.vue 변경
  - <router-view>전체에 <transition /> 적용
  - 화면 전환시 flip 효과!!

```
<template>
  <div id="container">
      ……(생략)
      <transition name="flip" mode="out-in">
        <router-view></router-view>
      </transition>
      <loading v-show="isloading"></loading>
  </div>
</template>
<script>
……(생략)
</script>
<style scoped>
……(생략)
.flip-enter-active {   transition: all .4s ease-in-out; }
.flip-leave-active {   transition: all .4s ease-in-out;  }
.flip-enter, .flip-leave-to {   transform: scaleY(0) translateZ(0);   opacity: 0;  }
</style>
```

연락처 애플리케이션에 트랜지션 적용하기

velocity를 이용한 예제

- 연락처 추가,수정, 사진 변경 화면에 트랜지션 효과
  - 나타날 때 elastic 효과
  - 사라질 때 까딱까딱~ 휙! 효과
- 예제 13-10

```
<template>
    <div>
    ……(생략)
    <transition
        v-on:before-enter="beforeEnter"
        v-on:enter="enter"
        v-on:leave="leave">
        <router-view></router-view>
    </transition>
    </div>
</template>
```

```
methods : {
  ……(생략)
  beforeEnter: function (el) {
    el.style.opacity = 0
  },
  enter: function (el, done) {
    Velocity(el, { opacity: 0, scale:0.2 }, { duration: 200 })
    Velocity(el, { opacity: 0.7, scale:1.2 }, { duration: 200 })
    Velocity(el, { opacity: 1, scale:1 }, { complete: done })
  },
  leave: function (el, done) {
    Velocity(el, { translateX: '0px', opacity: 1 }, { duration: 100 })
    Velocity(el, { translateX: '20px', opacity: 1 }, { duration: 100, loop:2 })
    Velocity(el, { translateX: '0px', opacity: 1 }, { duration: 200 })
    Velocity(el, { translateX: '100px', opacity: 0 }, { complete:done })
  }
}
```