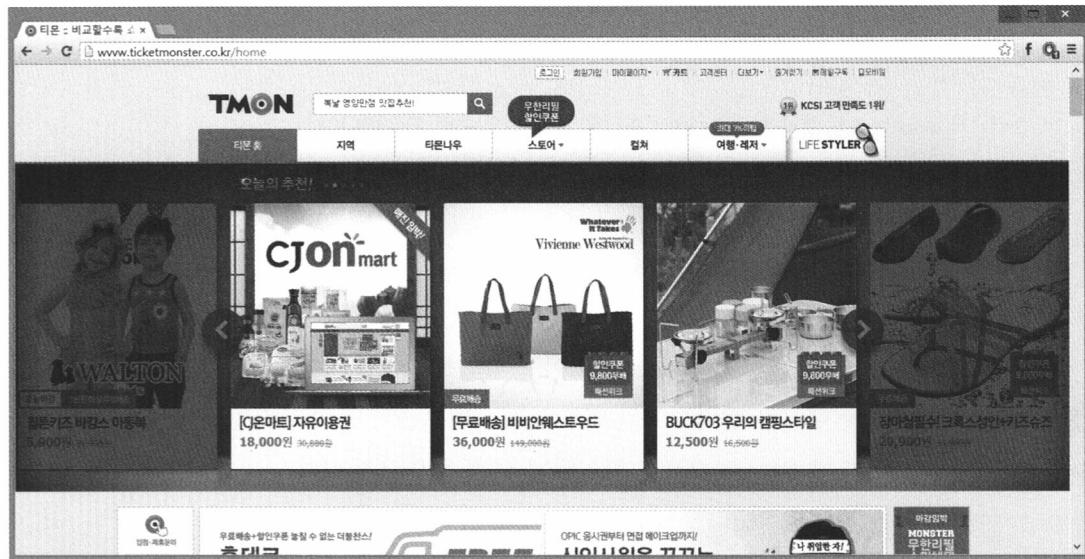


## 이미지 슬라이더

jQuery의 기본적인 내용을 모두 살펴보았습니다. 지금까지는 짧은 예제로 jQuery를 공부했습니다. 18장에서는 jQuery를 활용해 실무에서 사용할 수 있는 간단한 슬라이더를 구현해봅니다. 슬라이더는 그림 18-1처럼 이미지나 글들이 주기적 또는 선택적으로 넘어가는 요소를 의미합니다.

그림 18-1 티켓몬스터의 이미지 슬라이더



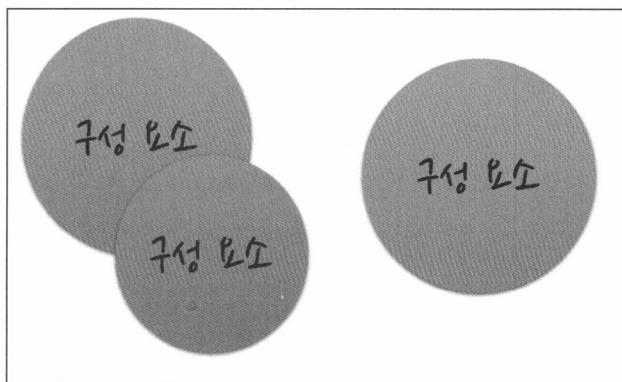
이 장에서는 이 슬라이더를 만들기 위한 기본적인 내용을 배우고, 이를 활용해 슬라이더를 만들어볼 것입니다. 예제가 조금 길어서 따로 장으로 구성했습니다.

## 18.1 jQuery 애니메이션을 위한 준비

불과 5년 전만해도 대부분 플래시로 이미지 슬라이더를 구현했지만, 최근에는 자바스크립트만으로 구현하는 것이 일반적입니다. jQuery를 사용하면 굉장히 쉽게 이러한 애니메이션 구성 요소를 만들 수 있습니다. 이 책에서는 애니메이션 구성 요소를 설명할 때 그림 18-2의 용어를 사용합니다.

그림 18-2 캔버스와 구성 요소

캔버스



자바스크립트로 플래시 또는 실버라이트의 시각적 효과를 구현하려면 다음 세 가지 사항을 CSS로 미리 지정해야 합니다.

- 캔버스의 width 스타일 속성과 height 스타일 속성은 필수로 지정합니다.
- 캔버스의 position 스타일 속성은 relative로 지정합니다.
- 캔버스의 overflow 스타일 속성은 hidden으로 지정합니다.
- 구성 요소의 position 스타일 속성은 absolute로 지정합니다.

이 절에서는 왜 이러한 CSS 속성을 지정해야 하는지 살펴봅시다. body 태그를 다음과 같이 지정합니다. 외부의 div 태그가 캔버스이고 내부의 div 태그들이 구성 요소입니다. h1 태그는 애니메이션 구성 요소가 제 위치를 찾는지 확인하기 위해 놓은 것입니다.

코드 18-1 body 태그 구성

```
<body>
  <h1>Test Header</h1>
  <div id="canvas">
```

```
<div class="inner_box"></div>
<div class="inner-box"></div>
<div class="inner-box"></div>
<div class="inner-box"></div>
<div class="inner-box"></div>
</div>
<h1>Test Header</h1>
</body>
```

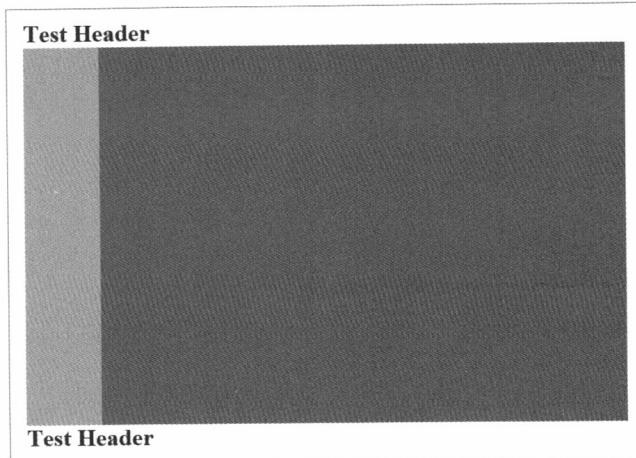
style 태그는 코드 18-2처럼 지정해 각 구성 요소를 구분할 수 있게 합시다.

#### 코드 18-2 style 태그 구성

```
<style>
  * { margin: 0px; padding: 0px; }
  #canvas {
    background: gray;
  }
  .inner-box {
    background: orange;
    width: 100px;
    height: 100px;
  }
</style>
```

현재 코드를 실행하면 그림 18-3처럼 구성됩니다. 각 구성 요소가 순서대로 배치됩니다. 이때 캔버스가 내부에 있는 구성 요소에 의해 자동으로 크기가 지정됩니다.

그림 18-3 실행 결과



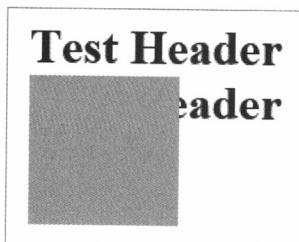
`animate()` 메서드로 `left` 속성 또는 `top` 속성을 지정하려면 `position` 스타일 속성이 `absolute`나 `relative`여야 하겠죠? 절대 좌표로 애니메이션을 구현하는 것이 일반적이므로 구성 요소는 대부분 `absolute` 속성을 사용합니다.

#### 코드 18-3 position 속성 적용

```
.inner-box {  
    background: orange;  
  
    width: 100px;  
    height: 100px;  
  
    position: absolute;  
}
```

하지만 코드를 실행하면 문제를 쉽게 찾을 수 있습니다. 구성 요소의 `position` 속성이 `absolute`일 경우에는 캔버스의 `height` 속성이 사라져 그림 18-4처럼 캔버스 뒤에 있는 `h1` 태그가 위로 올라와 있는 것을 볼 수 있습니다.

그림 18-4 캔버스 공간이 사라진 상태



캔버스의 `width` 스타일 속성과 `height` 스타일 속성은 반드시 지정해야 합니다. 따라서 코드 18-4처럼 캔버스의 `width` 속성과 `height` 속성을 지정하는 것이 포인트입니다.

#### 코드 18-4 캔버스 크기 강제 지정

```
#canvas {  
    background: gray;  
    width: 600px;  
    height: 400px;  
}
```

플래시나 실버라이트의 구성 요소를 보면 HTML 태그 내에 삽입할 때 width 속성과 height 속성을 반드시 지정하므로 별로 특이한 내용은 아닙니다.

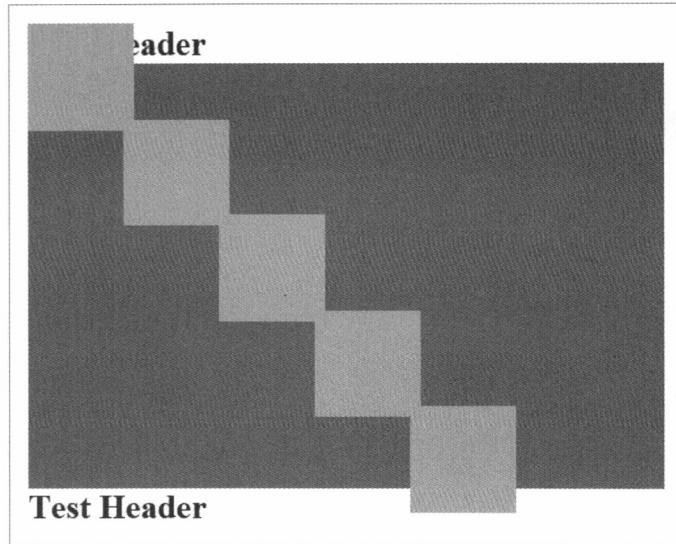
jQuery 구성 요소를 만들려면 한 가지 문제를 더 해결해야 하는데요. 잠시 코드 18-5처럼 각각의 구성 요소에 left 스타일 속성과 top 스타일 속성을 지정해봅시다.

#### 코드 18-5 절대 좌표 지정

```
<script>
    $(document).ready(function () {
        $('.inner-box').each(function (index) {
            $(this).css({
                left: index * 90,
                top: index * 90
            });
        });
    });
</script>
```

코드를 실행하면 그림 18-5처럼 각각의 구성 요소가 위치를 잡습니다. 부모의 위치를 무시하고 각각의 구성 요소가 페이지의 절대 좌표를 기준으로 위치했습니다.

그림 18-5 절대 좌표의 문제점



부모의 position 스타일 속성이 absolute 또는 relative일 때만 자식 구성 요소가 부모의 위치를 기준으로 위치를 잡습니다. 따라서 구성 요소뿐만 아니라 캔버스에도 position 스타일 속성을 지정해야 합니다. 일반적으로 캔버스의 position 스타일 속성에는 relative를 부여합니다.

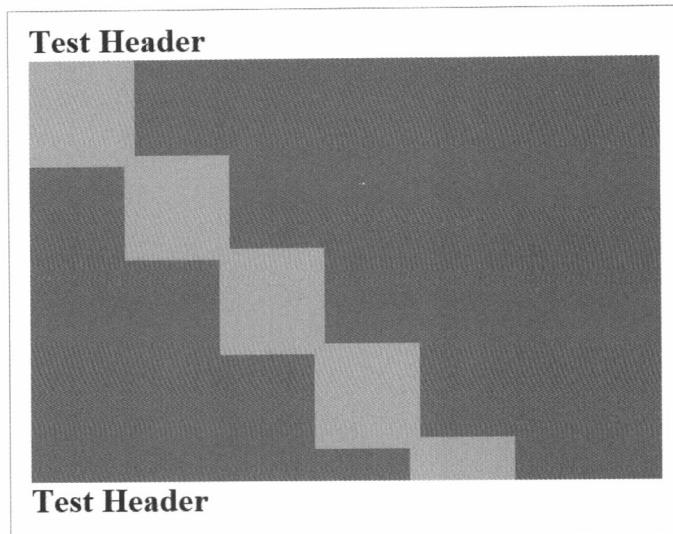
이어서 캔버스에는 overflow 스타일 속성도 입력합니다. overflow 스타일 속성은 내용물이 사용할 수 있는 범위를 초과하면 어떻게 처리하는지를 나타냅니다. overflow 스타일 속성에 hidden 값을 입력하면 캔버스의 공간을 넘는 내용물을 보이지 않게 합니다.

#### 코드 18-6 캔버스의 position 속성과 overflow 속성 지정

```
#canvas {  
    background: gray;  
    width: 600px;  
    height: 400px;  
    position: relative;  
    overflow: hidden;  
}
```

코드를 실행하면 그림 18-6처럼 출력됩니다. 캔버스 내부의 구성 요소에 애니메이션을 적용하기에 적당한 상태가 됐습니다.

그림 18-6 실행 결과



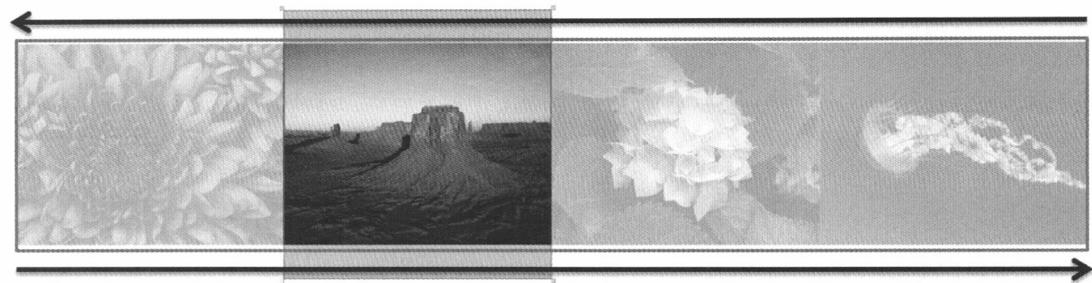
처음 언급한 4가지 사항을 꼭 기억하고, 다음 절로 넘어갑시다.

## 18.2 HTML 구성

이 절을 읽기 전에 18장의 그림만 미리 살펴보세요. 내용이 길어서 어떤 결과가 나오는지 바로 확인할 수 없으므로 미리 살펴보는 것이 좋습니다. 이 절에서 만들 애니메이션 요소는 이미지 슬라이더입니다.

이미지 슬라이더는 소극장과 같습니다. 극장 안에는 모든 소품과 연기자들이 준비하고 있지만, 관객의 눈에 보이는 것은 일부분이지요. 이미지 슬라이더도 사용자의 눈에 보이는 것은 일부분입니다. 보이지 않는 곳에서 모든 이미지가 미리 준비하고 있어야 합니다. 그럼 18-7은 이미지 슬라이더의 전형적인 형태입니다.

그림 18-7 이미지 슬라이더의 기본 구성



캔버스 내에서 전체 이미지를 담은 상자가 좌우로 움직이는 효과를 냅니다. 이때 사용자에게 보여지는 화면을 한정해 이미지가 바뀌는 효과를 주는 것입니다. 18장에서는 이미지만 웠다갔다하는 것이 아니라 텍스트도 함께 움직이는 이미지 슬라이더를 만듭니다.

코드 18-7처럼 전체적인 HTML 페이지를 구성합니다.

코드 18-7 HTML 페이지 구성

```
<!DOCTYPE html>
<html>
<head>
<style>

</style>
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script>
```

```
</script>
</head>
<body>

</body>
</html>
```

캔버스를 구성할 div 태그를 코드 18-8처럼 만듭니다. 이전 결과 마찬가지로 애니메이션 요소가 위치를 제대로 차지하는지 확인하려고 h1 태그를 입력했습니다.

#### 코드 18-8 캔버스 생성

```
<body>
  <h1>Test Header</h1>
  <div class="animation_canvas">

    </div>
    <h1>Test Header</h1>
  </body>
```

본격적으로 이미지 슬라이더를 구성해보겠습니다. 지금부터 캔버스 내부에 그림 18-8처럼 세 가지 id를 갖는 공간을 만듭니다.

그림 18-8 이미지 슬라이더의 HTML 태그 구성

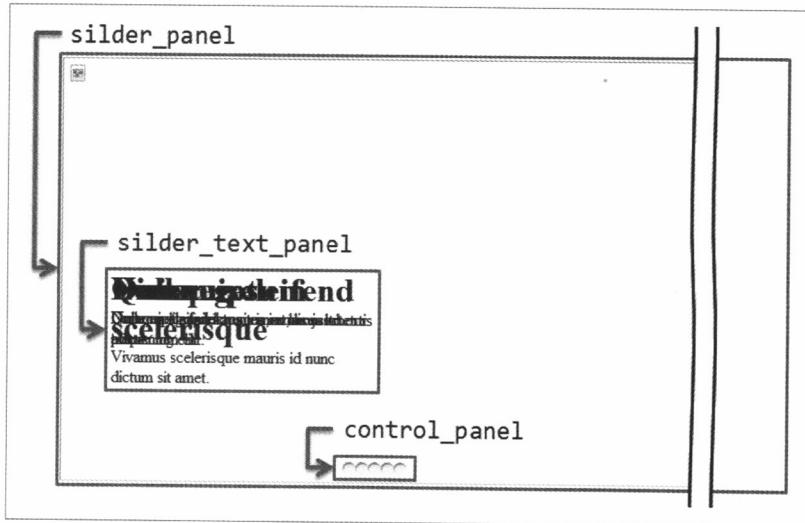


그림 18-8의 각 공간에는 다음 특성이 있습니다.

- **slider\_panel** 이미지를 담는 패널로, 움직이며 이미지가 전환되는 효과를 만듭니다.
- **slider\_text\_panel** 글자를 담는 패널입니다.
- **control\_panel** 컨트롤 버튼을 담는 패널입니다.

이를 토대로 코드 18-9처럼 body 태그를 구성합니다.

---

#### 코드 18-9 body 태그 공간 구성

---

```
<body>
    <h1>Test Header</h1>
    <div class="animation_canvas">
        <div class="slider_panel">

        </div>
        <div class="slider_text_panel">

        </div>
        <div class="control_panel">

        </div>
    </div>
    <h1>Test Header</h1>
</body>
```

---

각각의 공간에 코드 18-10처럼 내용을 넣습니다.

---

#### 코드 18-10 body 태그 구성 완료

---

```
<body>
    <h1>Test Header</h1>
    <div class="animation_canvas">
        <div class="slider_panel">
            <img class="slider_image" />
            <img class="slider_image" />
            <img class="slider_image" />
            <img class="slider_image" />
            <img class="slider_image" />
        </div>
        <div class="slider_text_panel">
            <div class="slider_text">
                <h1>Lorem ipsum</h1>
            </div>
        </div>
    </div>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
</div>
<div class="slider_text">
    <h1>Nulla eget</h1>
    <p>Nulla eget sapien mauris, ornare elementum elit.</p>
</div>
<div class="slider_text">
    <h1>Quisque eleifend</h1>
    <p>Quisque eleifend augue nec lacus lobortis porta.</p>
</div>
<div class="slider_text">
    <h1>Donec</h1>
    <p>Donec a ligula lectus, eu iaculis justo.</p>
</div>
<div class="slider_text">
    <h1>Vivamus scelerisque</h1>
    <p>Vivamus scelerisque mauris id nunc dictum sit amet.</p>
</div>
</div>
<div class="control_panel">
    <div class="control_button"></div>
    <div class="control_button"></div>
    <div class="control_button"></div>
    <div class="control_button"></div>
    <div class="control_button"></div>
</div>
<h1>Test Header</h1>
</body>
```

내용을 왜 이렇게 넣었는지는 결과를 참고하세요.

“그림 18-9를 보면 control\_button은 이미지인데, 왜 img 태그를 구성하지 않는 것인가요?”

일반적으로 control\_button과 같은 요소는 div 태그로 만들고 스타일시트에서 background 속성으로 이미지를 설정합니다. 이렇게 하면 스타일시트의 hover 필터와 active 필터를 사용할 수 있거든요. 이와 관련된 부분은 다음 절에 살펴봅니다.

## 18.3 스타일시트 구성

스타일시트와 관련되어 어려운 부분은 이전에 설명했으므로 이해되지 않는 부분은 없을 겁니다. 이 책의 주제가 CSS가 아니므로 간단하게 넘어갑니다.

코드 18-11 style 태그 구성

```
<style>
  * { margin: 0px; padding: 0px; }

  /* Animation Canvas */
  .animation_canvas {
    /* overflow: hidden; */
    position: relative;
    width: 600px; height: 400px;
  }

  /* Slider Panel */
  .slider_panel { width: 3000px; position: relative; }
  .slider_image { float: left; width: 600px; height: 400px; }

  /* Slider Text Panel */
  .slider_text_panel { position: absolute; top: 200px; left: 50px; }
  .slider_text { position: absolute; width: 250px; height: 150px; }

  /* Control Panel */
  .control_panel {
    position: absolute; top: 380px;
    left: 270px; height: 13px;
    overflow: hidden;
  }
  .control_button {
    width: 12px; height: 46px;
    position: relative;
    float: left; cursor: pointer;
    background: url('point_button.png');
  }
  .control_button:hover { top: -16px; }
  .control_button.active { top: -31px; }

</style>
```

코드를 보고 조금 이상한 것을 느끼는 독자가 있을 것입니다.

“animation\_canvas 클래스의 overflow 속성이 주석 처리 됐네요?”

예제를 만드는 동안은 주석을 유지해주세요. 그 편이 예제를 만드는데 훨씬 이해하기 쉬울 것입니다. 예제를 모두 완성한 후에 주석을 제거해주세요.

“control\_button 클래스의 hover와 active에 top 속성을 지정했네요?”

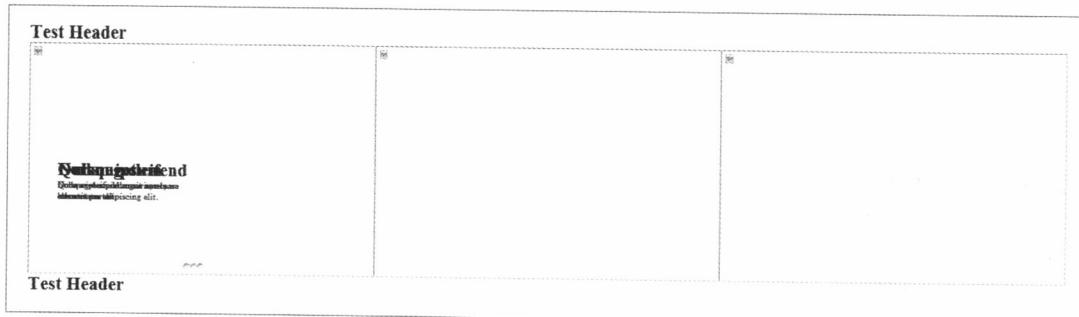
point\_button.png 파일은 다음과 같이 세로로 길게 늘어진 모양의 그림입니다. 이 그림에 표시 부분을 제한하고 위 아래로 옮김으로써 이미지가 변하는 것과 비슷한 효과를 나타냅니다. 아주 많이 사용하는 기술입니다.

그림 18-9 point\_button.png 파일



최종적으로 완성한 후에 control\_button을 살펴보면 쉽게 이해할 수 있습니다. 현재까지 만든 분량은 그림 18-10과 같습니다.

그림 18-10 현재 결과



## 18.4 자바스크립트 구성

이제 script 태그 부분을 입력해보겠습니다. 다음 순서로 진행합니다.

- 초기 텍스트 위치를 지정합니다.
- 이벤트를 연결합니다.
- 각각의 텍스트와 컨트롤 버튼의 위치를 알 수 있게 data-index를 할당합니다.
- 이동하는 함수를 만듭니다.
- 초기 슬라이더의 위치를 설정합니다.

slider\_text의 위치를 설정합니다. 사실 이 부분은 스타일시트에서 처리해도 되는 부분입니다.

### 코드 18-12 초기 텍스트 위치 지정

```
<script>
$(document).ready(function () {
    // 초기 텍스트 위치를 지정합니다.
    $('.slider_text').css('left', -300);
});
</script>
```

각 요소에 each() 메서드로 data-index를 부여합니다.

“왜 data-index 속성을 사용하나요?”

그냥 필자가 부여한 이름입니다. 원하는 이름을 사용해도 됩니다. 참고로 사용자가 추가로 속성을 입력할 때는 data-로 시작하는 것이 좋습니다. 웹 표준에 있는 권고 사안이기도 합니다.

### 코드 18-13 data-index 속성 부여

```
<script>
$(document).ready(function () {
    // 초기 텍스트 위치 지정 및 data-index 할당
    $('.slider_text').css('left', -300).each(function (index) {
        $(this).attr('data-index', index);
    });

    // 컨트롤 버튼의 클릭 리스너 지정 및 data-index 할당
    $('.control_button').each(function (index) {
```

```

        $(this).attr('data-index', index);
    });
});
</script>

```

일단 지금까지의 결과를 실행하고 요소 검사로 살펴보면 그림 18-11처럼 data-index 속성이 부여된 것을 알 수 있습니다.

**그림 18-11** data-index 속성 확인

```

▼<div class="slider_text_panel">
▶ <div class="slider_text" style="left: -300px; " data-index="0">...</div>
▶ <div class="slider_text" style="left: -300px; " data-index="1">...</div>
▶ <div class="slider_text" style="left: -300px; " data-index="2">...</div>
▶ <div class="slider_text" style="left: -300px; " data-index="3">...</div>
▶ <div class="slider_text" style="left: -300px; " data-index="4">...</div>
</div>
▼<div class="control_panel">
  <div class="control_button" data-index="0"></div>
  <div class="control_button" data-index="1"></div>
  <div class="control_button" data-index="2"></div>
  <div class="control_button" data-index="3"></div>
  <div class="control_button" data-index="4"></div>
</div>

```

기본적인 설정은 완료됐으니 이미지를 움직이는 이벤트를 작성하겠습니다. moveSlider() 함수를 만듭니다. control\_button을 클릭했을 때 moveSlider() 함수를 호출하고 처음 시작할 때 moveSlider() 함수를 호출해 랜덤한 위치로 이동합니다.

**코드 18-14** 이벤트 연결

```

<script>
$(document).ready(function () {
    // 슬라이더를 움직여주는 함수
    function moveSlider(index) {

    }

    // 초기 텍스트 위치 지정 및 data-index 할당
    $('.slider_text').css('left', -300).each(function (index) {
        $(this).attr('data-index', index);
    });

    // 컨트롤 버튼의 클릭 리스너 지정 및 data-index 할당
    $('.control_button').each(function (index) {

```

```
        $(this).attr('data-index', index);
    }).click(function () {
        var index = $(this).attr('data-index');
        moveSlider(index);
    });

    // 초기 슬라이더 위치 지정
    var randomNumber = Math.round(Math.random() * 4);
    moveSlider(randomNumber);
});
</script>
```

---

moveSlider( ) 함수는 다음 방식으로 구성하는데요. 특별히 다루지 않은 부분이 없으므로 쉽게 이해할 수 있을 것입니다.

#### 코드 18-15 moveSlider( ) 함수

---

```
// 슬라이더를 움직여주는 함수
function moveSlider(index) {
    // 슬라이더를 이동합니다.
    var willMoveLeft = -(index * 600);
    $('.slider_panel').animate({ left: willMoveLeft }, 'slow');

    // control_button에 active 클래스를 부여/제거합니다.
    $('.control_button[data-index=' + index + ']').addClass('active');
    $('.control_button[data-index!=' + index + ']').removeClass('active');

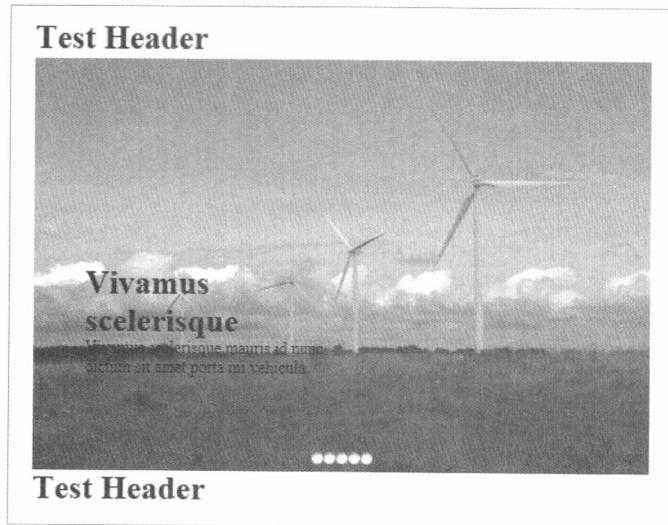
    // 글자를 이동합니다.
    $('.slider_text[data-index=' + index + ']').show().animate({
        left: 0
    }, 'slow');

    $('.slider_text[data-index!=' + index + ']').hide('slow', function () {
        $(this).css('left', -300);
    });
}
```

---

코드를 실행하면 그림 18-12와 같은 완성된 결과를 볼 수 있습니다.

그림 18-12 최종 완성 결과



이 장에서는 분량 관계상 생략했지만 17장에서 배운 easing 플러그인과 함께 사용하면 조금 더 멋진 효과를 연출할 수 있습니다.