

CH 10

제이쿼리 응용 다지기

1. 제이쿼리 메소드
2. 제이쿼리 이벤트
3. 제이쿼리 효과
요약

>> 학습목표 <<

- ❖ 스타일 및 DOM 트리 관련 제이쿼리 메소드를 알아본다.
- ❖ 제이쿼리 이벤트 명세 방법을 살펴본다.
- ❖ 다양한 제이쿼리 효과의 종류와 사용 방법을 알아본다.

1.1 제이쿼리 스타일 관련 메소드

● 제이쿼리 CSS3 메소드

- 제이쿼리의 기능 중 하나가 CSS3 스타일 속성을 제어하는 것
- CSS3 관련 메소드를 통해서 HTML5 문서의 스타일을 동적으로 변경 가능

● [표 10-1] CSS3 메소드

형식	기능
<code>\$().css(CSS속성명)</code>	선택된 엘리먼트에 적용된 CSS 스타일 속성값을 반환
<code>\$().css(CSS속성명, CSS속성값)</code> <code>\$().css({CSS속성집합})</code>	선택된 엘리먼트에 CSS 스타일을 적용 속성값을 한꺼번에 설정(맵형식({ '속성명': '속성값', '속성명': '속성값', ... }))
<code>\$().addClass(CSS클래스명)</code>	선택된 엘리먼트에 class 속성값을 설정(CSS 클래스명으로 선언된 스타일을 적용)
<code>\$().removeClass(CSS클래스명)</code>	선택된 엘리먼트의 class 속성값을 제거(적용된 CSS 클래스 스타일을 제거)
<code>\$().toggleClass(CSS클래스명)</code>	선택된 엘리먼트에 class 속성값이 존재하면 제거, 없으면 추가(CSS 클래스 스타일을 적용/해제를 전환)
<code>\$().hasClass(CSS클래스명)</code>	선택된 엘리먼트에 class 속성값 존재 여부를 반환(CSS 클래스 스타일 적용 여부를 반환)
<code>\$().width()</code>	선택된 엘리먼트의 너비 값을 반환
<code>\$().width(너비값)</code>	선택된 엘리먼트의 너비 값을 설정
<code>\$().height()</code>	선택된 엘리먼트의 높이 값을 반환
<code>\$().height(높이값)</code>	선택된 엘리먼트의 높이 값을 설정

DOM CSS 메소드 적용하기

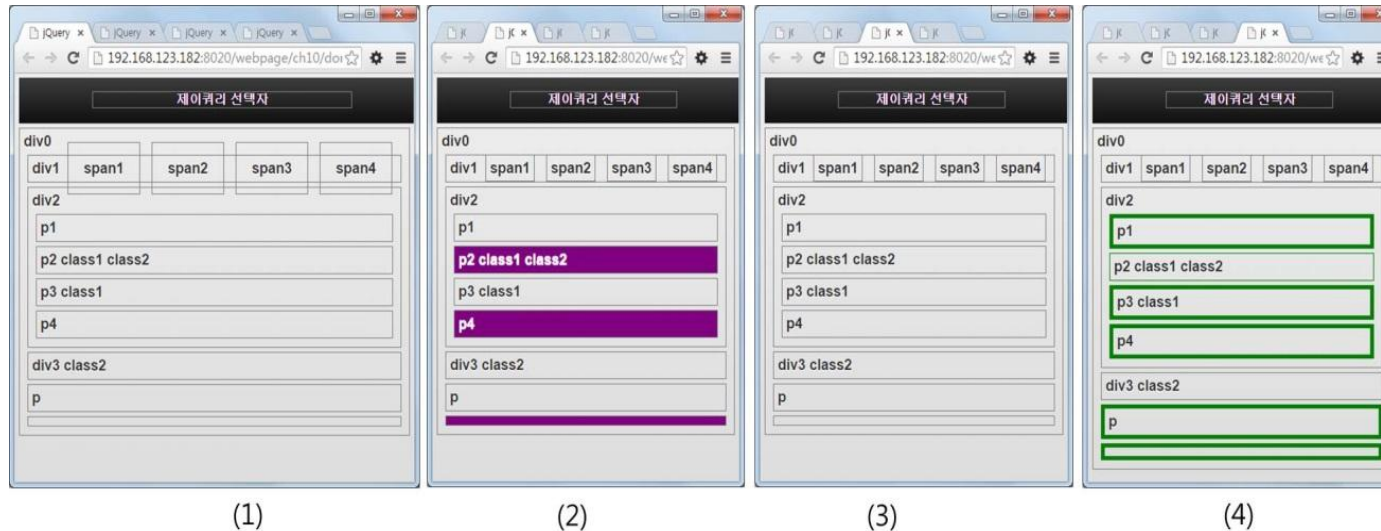
● [그림 10-1] dom-css.html의 실행 결과(예제10-1)

[예제10-1] DOM CSS 메소드 적용하기

/ch10/dom-css.html

```

$('span').css('padding','20px'); // (1)
$('p:odd').css({'background-color':'purple', 'color': 'white'}); // (2)
$('#span2').css('background-color',$('p:eq(1)').css('background-color')); // (3)
$('p').css('border-color','green')
    .not(':eq(1)')
    .css('border-width','thick'); // (4)
  
```



맵 방식과 메소드 체인 방식

- CSS3 메소드를 통해 CSS3 스타일을 지정할 때 2가지 방식이 가능하다.
- 맵(map) 방식
 - { }안에 ':'으로 구분한 속성과 값의 쌍을 ','로 구분하여 여러 개 나열하는 방법

```
$( 'p:odd' ).css( { 'background-color': 'purple', 'color': 'white' } );
```

- 메소드 체인(method chain) 방식
 - 메소드 호출 뒤에 마침표(.)을 찍고 또 다른 메소드를 호출하도록 함으로써 한 문장 안에서 체인처럼 연속적으로 메소드를 호출하는 방법
 - 앞의 메소드가 반환하는 제이쿼리 객체에 뒤에 연결된 메소드가 추가로 적용되는 방식
 - 전체 코드 길이가 줄어드는 이점

```
$( 'p:odd' ).css( 'background-color', 'purple' ).css( 'color', 'white' );
```

맵 방식과 메소드 체인 방식 예제

● 다음 예를 각각 실행할 경우의 결과

- (1) `$('#div:eq(2)').css('border-width', 'thick');`
- (2) `$('#div:eq(2)').find('.class2').css('border-width', 'thick');`
- (3) `$('#div:eq(2)').find('.class2').text('노드 변경').css('border-width', 'thick');`
- (4) `$('#div:eq(2)').find('.class2').text('노드 변경').append('<h5>노드 추가</h5>').css('border-width', 'thick');`



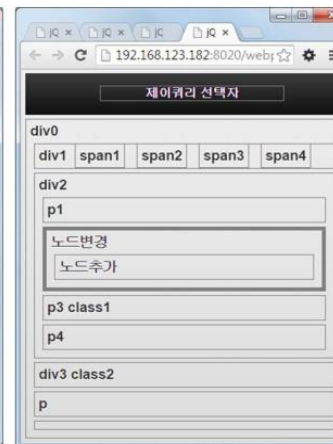
(1)



(2)



(3)



(4)

스타일 클래스 메소드 적용하기

• [그림 10-2] style-class.html의 실행 결과(예제10-2)

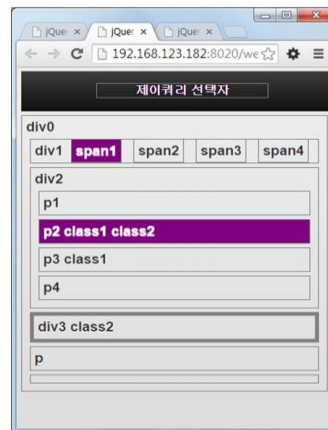
[예제10-2] 스타일 클래스 메소드 적용하기

/ch10/style-class.html

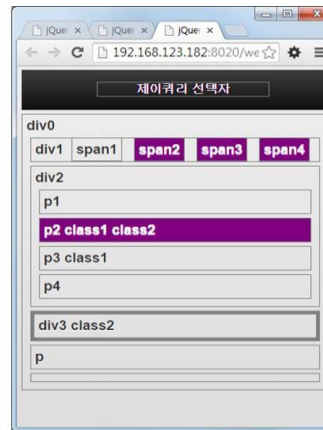
```
<style type="text/css">
    .class2 { border-width: thick; }
    .u_bgpurple { background-color: purple; color: white; }
    .u_dotted { border-style : dotted; }
</style>
... 생략 ...
$('span:first').addClass('u_bgpurple');           // (1)
$('p:eq(1)').removeClass('class2').addClass('u_bgpurple'); // (2)
$('span').toggleClass('u_bgpurple');              // (3)
```



(1)



(1)(2)



(1)(2)(3)

1.2 DOM 트리 관련 메소드(1)

● DOM 탐색 메소드

- 선택자 이외에도 여러 DOM 탐색 메소드를 통해 엘리먼트에 접근 가능
- DOM 트리 관련 탐색 메소드와 필터링 메소드는 선택자의 기능을 확장 및 보완
 - 탐색(traversing) 메소드 : DOM 트리의 선택된 위치를 기준으로 원하는 노드(주로 엘리먼트)를 찾음
 - 선택자 대신 현재의 참조 엘리먼트를 기준으로 탐색 메소드를 사용하는 것이 효과적인 경우에 사용(예: 이전 또는 다음 엘리먼트를 접근하는 경우)
 - `$()` 함수 선택자를 이용 특정 노드를 찾은 후, 그 위치를 기준으로 다른 노드를 탐색

형식	기능
<code>\$().find()</code>	선택된 엘리먼트 중에서 조건을 충족하는 모든 자손 엘리먼트를 반환
<code>\$().children()</code>	선택된 엘리먼트 중에서 [조건을 충족하는] 모든 자식 엘리먼트들을 반환
<code>\$().parent()</code>	선택된 엘리먼트 중에서 부모 엘리먼트를 반환
<code>\$().parents()</code>	선택된 엘리먼트 중에서 [조건을 충족하는] 모든 조상 엘리먼트들을 반환
<code>\$().siblings()</code>	선택된 엘리먼트 중에서 자신을 제외한 [조건을 충족하는] 모든 형제 엘리먼트들을 반환
<code>\$().prev()</code>	선택된 엘리먼트 중에서 바로 앞에 위치한 [조건을 충족하는] 형제 엘리먼트를 반환
<code>\$().prevAll()</code>	선택된 엘리먼트 중에서 앞에 위치한 모든 [조건을 충족하는] 모든 형제 엘리먼트들을 반환
<code>\$().next()</code>	선택된 엘리먼트 중에서 바로 다음에 위치한 [조건을 충족하는] 형제 엘리먼트를 반환
<code>\$().nextAll()</code>	선택된 엘리먼트 중에서 다음에 위치한 [조건을 충족하는] 모든 형제 엘리먼트들을 반환

DOM 탐색 메소드 적용하기

• [그림 10-3] dom-traversal.html의 실행 결과(예제10-3)

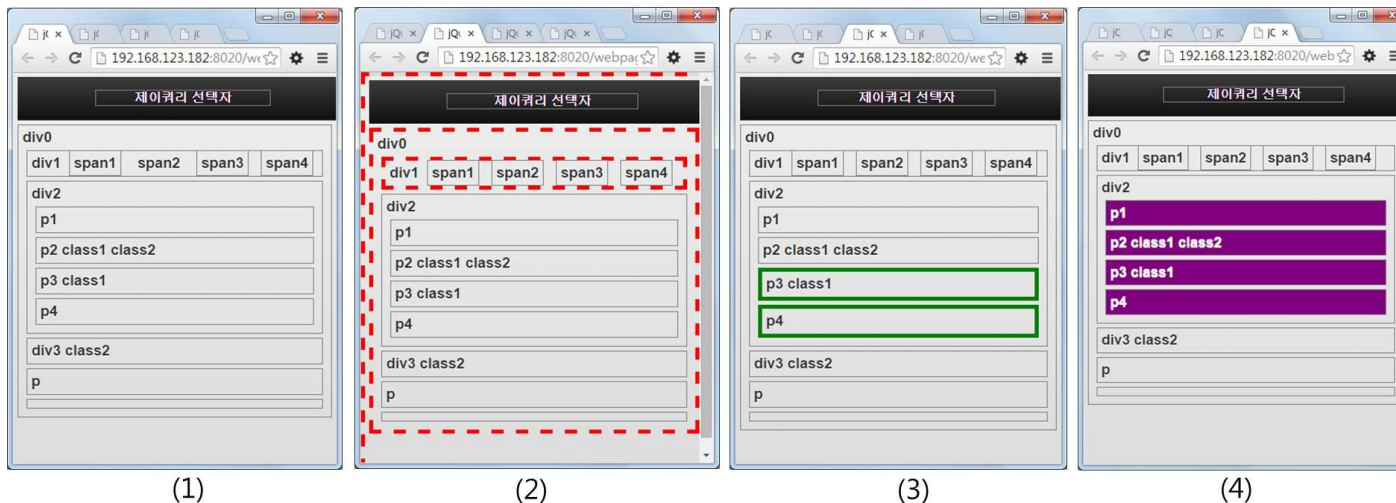
[예제10-3] DOM 탐색 메소드 적용하기

/ch10/dom-traversal.html

```

$('span:eq(2)').prev().css('border-style', 'hidden');           // (1)
$('span').parents().css('border', 'dashed thick red');         // (2)
$('#p2').nextAll().css('border', 'solid thick green');         // (3)
$('#div2').find('p').css({'background-color':'purple', 'color': 'white'}); // (4)

```



DOM 트리 관련 메소드(2)

● DOM 필터링 메소드

- DOM 트리의 선택된 노드 집합(제이쿼리 객체 집합)에서 다시 특정 조건을 만족하는 노드만을 탐색하는 필터링 메소드

형식	기능(첨자가 0부터 시작됨)
<code>\$().filter()</code>	선택된 엘리먼트들 중에서 필터링 조건을 충족하는 엘리먼트를 반환
<code>\$().slice(시작첨자[, 종료첨자])</code>	선택된 엘리먼트들 중에서 시작첨자부터 종료첨자 이전까지의 엘리먼트를 반환
<code>\$().first()</code>	선택된 엘리먼트들 중에서 첫 번째 엘리먼트를 반환
<code>\$().last()</code>	선택된 엘리먼트들 중에서 마지막 엘리먼트를 반환
<code>\$().eq(n)</code>	선택된 엘리먼트들 중에서 첨자 n인 엘리먼트를 반환
<code>\$().has()</code>	선택된 엘리먼트들 중에서 특정 자손 엘리먼트를 갖는 엘리먼트를 반환
<code>\$().is()</code>	선택된 엘리먼트들 중에서 특정 조건을 만족하는 엘리먼트가 있으면 'true' 반환
<code>\$().not()</code>	선택된 엘리먼트들 중에서 특정 조건을 만족하지 않는 엘리먼트를 반환

DOM 필터링 메소드 적용하기

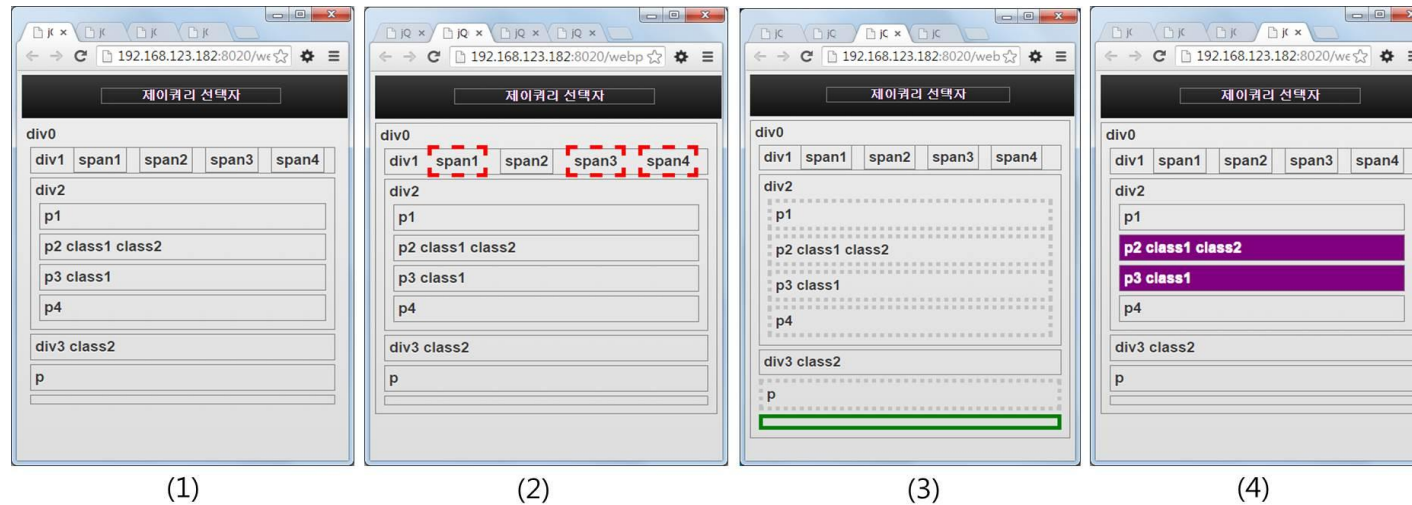
● [그림 10-4] dom-filtering.html의 실행 결과(예제 10-4)

[예제10-4] DOM 필터링 메소드 적용하기

/ch10/dom-filtering.html

```

$('div').first().css('border-style', 'hidden'); // (1)
$('span').not('#span2').css('border', 'dashed thick red'); // (2)
$('p').css('border', 'dotted thick silver').filter('#div0 p:last').css('border', 'solid thick green'); // (3)
$('p').slice(1,3).css({'background-color':'purple', 'color': 'white'}); // (4)
  
```



DOM 트리 관련 메소드(3)

● DOM 트리 엘리먼트 조작 메소드

- 정적인 HTML5 문서에 동적인 특성을 제공하는 제이쿼리의 대표적 기능
- DOM 트리에 새로운 노드를 추가하거나 변경 또는 제거를 자유롭게 수행

형식	기능
<code>\$().append()</code>	선택된 엘리먼트의 마지막 자식 엘리먼트로 추가
<code>\$().prepend()</code>	선택된 엘리먼트의 첫 번째 자식 엘리먼트로 추가
<code>\$().after()</code>	선택된 엘리먼트의 뒤에 형제 엘리먼트로 추가
<code>\$().before()</code>	선택된 엘리먼트의 앞에 형제 엘리먼트로 추가
<code>\$().empty()</code>	선택된 엘리먼트의 내용을 비움(자식 엘리먼트만 제거)
<code>\$().remove()</code>	선택된 엘리먼트를 제거(자식 엘리먼트도 포함하여 제거)
<code>\$().replaceWith()</code>	선택된 엘리먼트를 특정 엘리먼트로 바꿈
<code>\$().clone()</code>	선택된 엘리먼트를 복사
<code>\$().wrap()</code>	선택된 엘리먼트를 특정 엘리먼트로 둘러쌈(부모 엘리먼트로 삽입)
<code>\$().unwrap()</code>	선택된 엘리먼트의 부모 엘리먼트를 제거

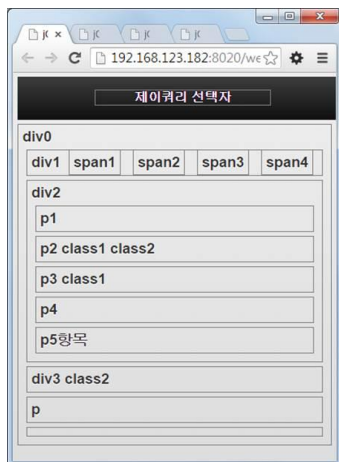
DOM 엘리먼트 메소드 적용하기

● [그림 10-5] dom-element.html의 실행 결과(예제 10-5)

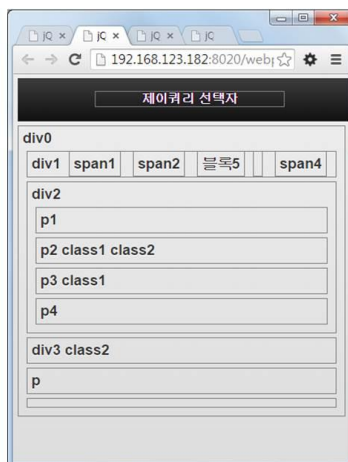
[예제 10-5] DOM 엘리먼트 메소드 적용하기

/ch10/dom-element.html

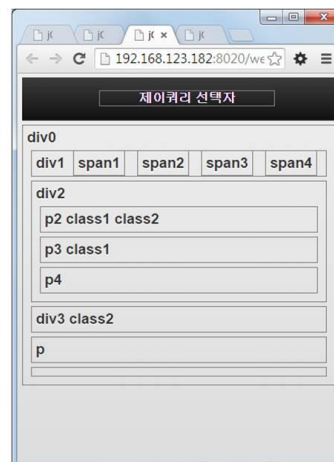
```
$('#div2').append($('
```



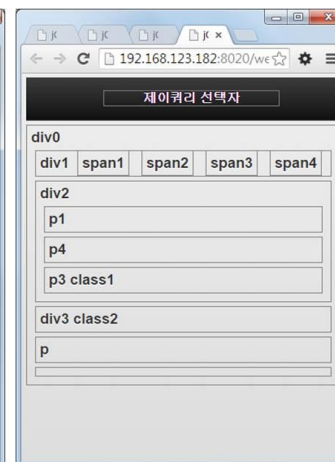
(1)



(2)



(3)



(4)

1.3 기타 메소드

● 프로그래밍 관련 메소드

- 사용자와의 상호 작용을 위한 DOM 트리와의 정보 교환에 관련된 제이쿼리 메소드

형식	기능
<code>\$().html()</code>	선택된 엘리먼트의 내용을 HTML5 형식의 문자열로 반환(마크업 포함)
<code>\$().html(HTML5문자열)</code>	선택된 엘리먼트 밑에 HTML5 문자열을 엘리먼트로 변환하여 추가
<code>\$().text()</code>	선택된 엘리먼트의 텍스트 내용을 텍스트 형식의 문자열로 반환
<code>\$().text(문자열)</code>	선택된 엘리먼트 밑에 텍스트 내용으로 문자열을 추가
<code>\$().size()</code>	선택된 엘리먼트의 개수를 반환 <code>\$().length</code> 와 기능 동일
<code>\$().get(첨자)</code>	선택된 엘리먼트 중에서 첨자(0부터 시작)에 해당하는 엘리먼트 객체를 반환
<code>\$().index()</code>	선택된 (첫)엘리먼트의 형제 엘리먼트와의 상대적인 첨자를 반환
<code>\$().each(콜백함수)</code>	선택된 엘리먼트들을 차례로 순환하면서 콜백 함수를 반복 호출

프로그래밍 메소드 적용하기

- [그림 10-6] dom-programming.html의 실행 결과(예제10-6)

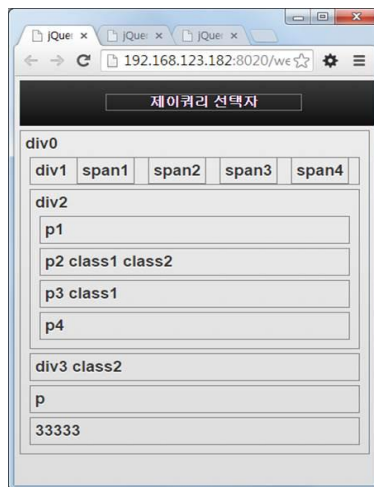
[예제10-6] 프로그래밍 메소드 적용하기

/ch10/dom-programming.html

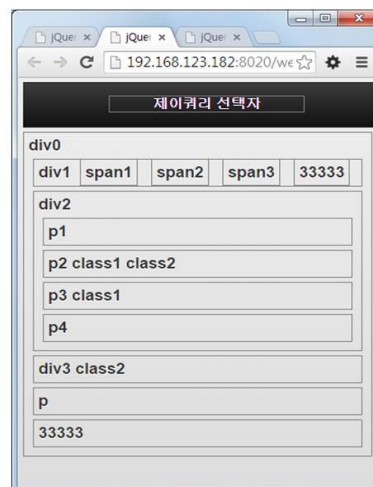
```

$('p:eq(5)').text('33333');           // (1)
$('span:eq(3)').text($('p:last').text()); // (2)
$('p:first').html($('div:eq(1)').html()); // (3)

```



(1)



(1)(2)



(1)(2)(3)

DOM 트리 관련 메소드(4)

● DOM 트리 속성 조작 메소드

- 엘리먼트의 시작 태그 안에 포함된 속성 이름과 속성값은 DOM 트리에서 엘리먼트 노드의 하위 노드
- 속성에 대해서도 메소드를 통해 DOM 트리에 추가하거나 제거 가능

형식	기능
<code>\$().attr(속성명)</code>	선택된 엘리먼트들 중 첫 번째 엘리먼트의 특정 속성값을 반환
<code>\$().attr(속성명,속성값)</code> <code>\$().attr({속성집합})</code> <code>\$().attr(속성명,함수())</code>	선택된 모든 엘리먼트에 속성값을 설정 여러 속성값을 한꺼번에 설정(맵형식({속성명:속성값, 속성명:속성값, ... })) 함수 반환 값을 속성값으로 설정
<code>\$().removeAttr(속성명)</code>	선택된 모든 엘리먼트의 특정 속성을 제거
<code>\$().val()</code>	선택된 첫 번째 엘리먼트(폼 관련)의 value 속성값을 반환
<code>\$().val(속성값)</code>	선택된 모든 엘리먼트(폼 관련)의 value 속성값을 설정

DOM 속성 메소드 적용하기

● [그림 10-7] dom-attribute.html의 실행 결과(예제10-7)

[예제10-7] DOM 속성 메소드 적용하기

/ch10/dom-attribute.html

```

alert('학번 disabled 속성값 : ' + $('#s_id').attr('disabled'));           // (1)
$('#s_name').attr('disabled', false);                                   // (2)
$('input:radio').attr('checked', false);                               // (4)
$('input[name="s_tel"]').removeAttr('placeholder');                  // (3)
var age = $('input[type="number"]').val();
age = eval(age) + 15;
$('input[type="number"]').val(age);                                     // (5)

```

[예제8-11] native-form.html 실행 화면

(1) ~ (5) 속성 메소드 적용 후 실행 화면

each() 메소드와 this

● each() 메소드

- 보통 이름없는 콜백 함수(반복 함수)를 입력 인자로 사용
 - 콜백 함수는 \$()가 반환하는 제이쿼리 객체 집합의 개수만큼 반복해서 호출됨
 - 콜백 함수 안에서 반복 호출할 때마다 각 제이쿼리 객체(엘리먼트)들을 \$(this)로 접근
 - C언어에서 for 반복문 안의 'i' 반복 제어 변수와 같은 역할

● 예)

- 비순서 리스트

```
<ul>  
<li>봄</li> <li>여름</li> <li>가을</li> <li>겨울</li>  
</ul>
```

- 다음 제이쿼리 메소드를 실행하면 4개 항목의 계절 이름을 순차적으로 경고 창에 반복해서 표시

```
$('.li').each(function(){  
    alert($(this).text());  
});
```

2.1 제이쿼리 이벤트

- 제이쿼리 이벤트(event)

- 웹 페이지와 사용자 사이의 동적인 상호작용을 지원
- 사용자의 동작과 관련하여 이벤트 메소드를 사용하면 사용자의 동작에 동적으로 반응하는 웹 페이지를 만들 수 있음
- 이벤트 핸들러에 해당하는 제이쿼리 함수를 호출하는 기능

- 이벤트 핸들러(event handler)

- 리스너(listener)
- 기다리던 이벤트가 발생하면 이를 감지해서 적절한 처리를 하도록 하는 함수
- 주로 이벤트와 제이쿼리 메소드를 연결
- 자바스크립트보다 훨씬 간단하게 이벤트 처리가 가능
- 예) `$(document).ready()`

2.2 이벤트 핸들러 연결 및 해제

● 이벤트 메소드 직접 연결

- 표준 이벤트 유형을 단축형 메소드로 직접 사용
- 제이쿼리 선택자에 의해 지정된 객체에 이벤트 유형으로 명시한 사건이 발생하면 함수가 호출되어 실행
- 함수 안에 명시된 코드는 이벤트에 의해 함수가 호출되어야만 실행

```
$( '선택자' ).이벤트유형( 함수명 );
```

● bind() 메소드 간접 연결

- bind() 메소드
 - 이벤트 유형별로 이벤트 핸들러 함수를 연결하는 역할
 - 하나 또는 여러 개의 이벤트가 발생할 때 함수가 호출되어 실행되도록 함

이벤트 통해 메소드 호출

● 이벤트를 통해 함수를 호출하는 예

- (1) 이벤트 메소드를 사용하여 이벤트와 이벤트 핸들러 함수를 직접 연결
- (2) bind() 메소드를 사용하여 이벤트 유형과 이벤트 핸들러 함수를 간접 연결
- (3) 함수 이름 없이 한 번에 이벤트와 함수를 연결 가능

```
function fn_hi() {
    alert('hi');
}
... 생략 ...
$('div').click(fn_hi);           // (1)
$('div').bind('click', fn_hi);  // (2)
$('div').click(function() { alert('hi'); }); // (3)
```

● 이벤트 연결 해제

- unbind() 메소드를 이용

```
$('#선택자').unbind('이벤트유형'); // 선택자 객체에 연결된 이벤트를 해제
$('#선택자').unbind();             // 선택자 객체에 연결된 모든 이벤트를 해제
```

2.3 이벤트 메소드(1)

● 이벤트 연결/해제 메소드

- 이벤트 발생시 실행할 함수를 이벤트와 연결하거나 연결을 해제하는 제이쿼리 메소드

형식	기능
<code>\$().bind()</code>	특정 엘리먼트(제이쿼리 객체)에 이벤트 핸들러를 연결
<code>\$().unbind()</code>	bind()로 연결된 이벤트 핸들러를 해제
<code>\$().one()</code>	특정 엘리먼트(제이쿼리 객체)에 이벤트 핸들러를 단 한번 연결 후 해제(일회용)
<code>\$().trigger()</code>	특정 엘리먼트(제이쿼리 객체)에 직접 이벤트를 발생시켜 이벤트 핸들러 함수를 실행
<code>\$().live()</code>	동적으로 미래에 생성될 특정 엘리먼트(제이쿼리 객체)에 이벤트 핸들러를 연결
<code>\$().die()</code>	live()로 연결된 이벤트 핸들러를 해제

이벤트 메소드[2]

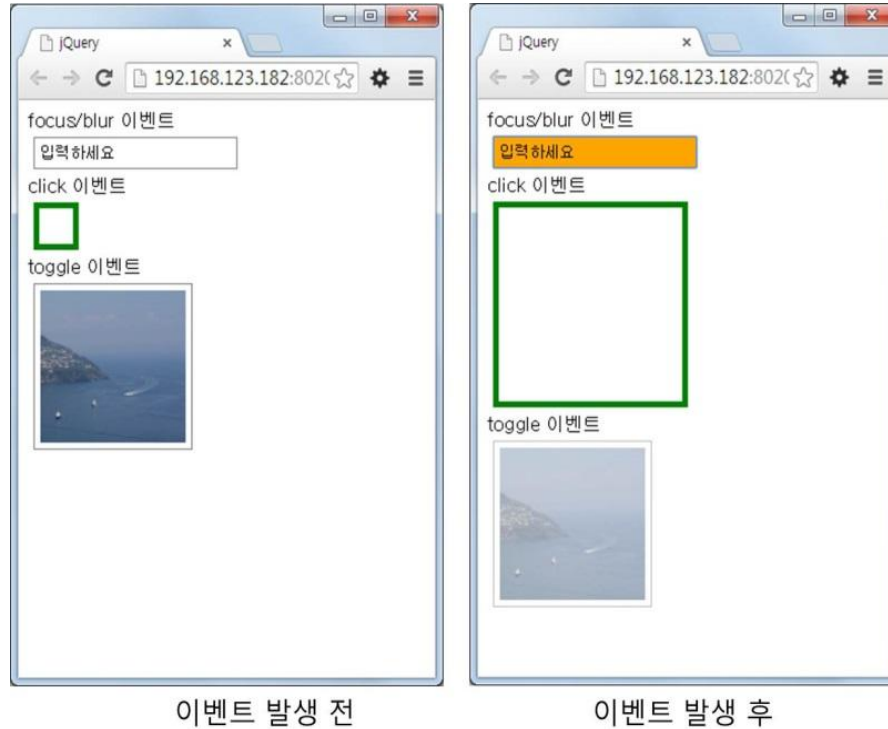
● 이벤트 단축형 메소드

- 표준 이벤트 유형은 bind() 메소드 없이도 단축형 메소드로 사용 가능
- 이벤트 유형 이름을 메소드 이름으로 사용 가능
- 직접 이벤트 핸들러를 설정하는 단축형 이벤트 메소드

분류	이벤트 메소드	기능
마우스	\$().click()	엘리먼트 표시 영역을 마우스로 클릭할 때 동작
	\$().hover()	엘리먼트 표시 영역 안으로 마우스 포인터가 들어올 때(또는 나갈 때) 동작
	\$().toggle()	마우스를 클릭할 때마다 두 함수를 번갈아 동작
폼	\$().focus()	폼 엘리먼트 표시 영역이 포커스를 얻을 때 동작
	\$().blur()	폼 엘리먼트 표시 영역이 포커스를 잃을 때 동작
	\$().change()	폼 엘리먼트 표시 영역의 값이 변경될 때 동작
	\$().select()	폼 엘리먼트 표시 영역의 텍스트 일부를 선택할 때 동작
키보드	\$().keydown()	키보드를 눌렀을 때 동작
문서	\$().ready()	브라우저에 문서가 읽혀질 때 동작 DOM을 완전히 로드 했을 때 실행할 함수를 지정
	\$().load()	브라우저에 문서 관련 모든 자원이 읽혀질 때 동작 엘리먼트의 모든 하위 엘리먼트를 로드 했을 때 실행할 함수를 지정
	\$().unload()	브라우저에서 문서가 사라질 때 동작 현재 페이지를 떠나거나 이동할 경우 실행할 함수를 지정

2.4 이벤트 활용 예

- [그림 10-8] jq-event.html의 실행 결과(예제 10-8)



3.1 제이쿼리 효과

● 효과 유형 메소드

- 특정 영역을 서서히 사라졌다가 다시 나타나게 하는 등의 간단한 애니메이션 효과를 메소드로 제공한다. (다양한 효과를 일정 시간 동안 계속 수행)

● 제이쿼리에서 제공하는 기본 효과 유형 메소드

효과 유형	기능
<code>show([ms],[function()])</code>	선택된 엘리먼트를 화면에서 보이게 함
<code>hide([ms],[function()])</code>	선택된 엘리먼트를 화면에서 사라지게 함
<code>toggle([ms],[function()])</code>	선택된 엘리먼트가 화면에 보였다가 사라지는 상태를 반복함 <code>show()</code> 와 <code>hide()</code> 를 번갈아 수행함
<code>slideUp([ms],[function()])</code>	선택된 엘리먼트의 높이를 점차 위로 감소시켜 화면에서 사라지게 함 위로 접는 효과
<code>slideDown([ms],[function()])</code>	선택된 엘리먼트의 높이를 점차 아래로 증가시켜 화면에서 보이게 함 아래로 펼치는 효과
<code>slideToggle([ms],[function()])</code>	선택된 엘리먼트의 높이를 변경하여 화면에서 사라지거나 보이게 함
<code>fadeIn([ms],[function()])</code>	선택된 엘리먼트의 불투명도를 점차 높여서 보이게 함
<code>fadeOut([ms],[function()])</code>	선택된 엘리먼트의 불투명도를 점차 낮춰서 사라지게 함
<code>fadeToggle([ms],[function()])</code>	선택된 엘리먼트의 불투명도를 변경하여 사라지거나 보이게 함

● 효과 유형 메소드의 3가지 입력 인자 형식

```

show( ) ; // (1) 빈 입력인자
show(600) ; 또는 show('slow') ; // (2) 수치, 문자열 입력인자
show(200, function( ) { ... } ) ; // (3) 콜백함수 입력인자

```

3.2 사용자 정의 효과 생성하기

● animate() 메소드

- 기본적인 효과 이외에 맞춤형 효과를 사용자가 직접 정의하여 사용
- 수치값을 사용하는 CSS3 스타일 속성값은 모두 사용 가능

효과 유형	기능
animate([properties][,ms][,function()])	선택된 엘리먼트에 대해 직접 설정한 효과를 통해 맞춤형 애니메이션 효과를 적용함

- 첫 번째 입력 인자인 특성(properties) 객체
 - 다양한 움직임 효과를 줄 수 있는 CSS3 속성과 속성값을 지정
 - CSS3 속성 중 크기나 길이, 비율 등 숫자를 사용하는 속성들을 맵 방식으로 명세
 - 예) 지속시간 : 1/1000초 단위의 숫자나 slow, normal, fast 문자열 중 하나

animate(특성객체, 지속시간, 콜백함수)

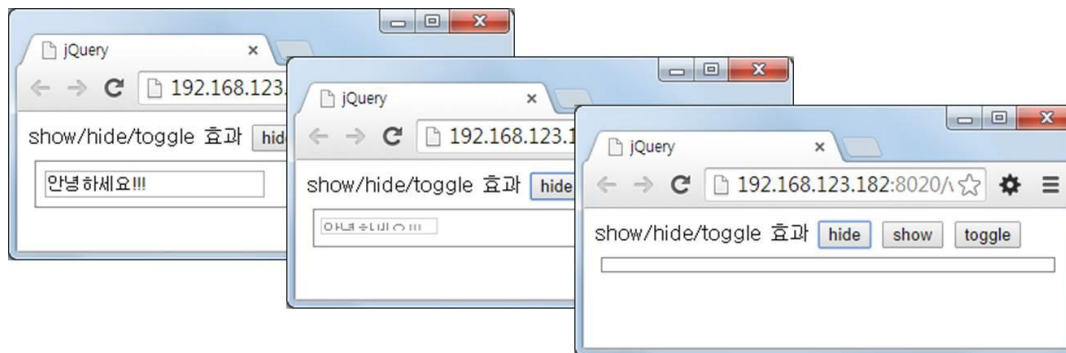
■ 사용자 정의 효과의 예

```

$('p').animate({font-size: "3em"}, 2000);      // 문단 글자 크기를 3배로 확대하는 효과
$('div').animate({height: "25%"}, "slow");     // div 영역의 높이를 1/4로 축소하는 효과
$('div:has(img)').animate({width: "0px"}, 1000); // 이미지가 포함된 div 영역을 왼쪽으로 접는 효과
  
```

3.3 제이쿼리 효과 활용

- 효과 활용 예 : `show()`, `hide()`, `toggle()`
 - [그림 10-9] `jq-effect1.html`의 실행 결과(예제10-9)



- 효과 활용 예 : `fadeOut()`, `fadeIn()`, `fadeToggle()`
 - [그림 10-10] `jq-effect2.html`의 실행 결과(예제10-10)

