

# jQuery

Basic APIs



New jQuery Conference!

# jQuery CONFERENCE 2011

Boston, Massachusetts

October 1<sup>st</sup> & 2<sup>nd</sup> 2011 ♦ The Royal Sonesta Hotel

Dates, Speakers, Sponsors:  
<http://events.jquery.org>

## jQuery is a new kind of JavaScript Library.

jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.

✓ [Lightweight Footprint](#)

✓ [CSS3 Compliant](#)

✓ [Cross-browser](#)

## GRAB THE LATEST VERSION!

CHOOSE YOUR COMPRESSION LEVEL:

- PRODUCTION (31KB, Minified and Gzipped)
- DEVELOPMENT (229KB, Uncompressed Code)

 **Download ( jQuery );**

Current Release: **v1.6.4**

# jQuery

introduction

# Introduction

## JavaScript library

- ◎ 기능과 UI가 많아질 수록 복잡도가 증가한다.
  - 재사용될 수 있는 기능들을 라이브러리화 한다.
- ◎ 자주 사용되는 기능을 모아놓은 자바스크립트 라이브러리들이 많다.
  - Prototype, mooTools, YUI, Dojo, Ext JS, etc.
- ◎ 개발해야 하는 제품의 특성에 따라서 사용한다.
  - 직접 만들 수도 있고 공개 라이브러리를 사용할 수도 있다.
  - 효율성(크기, 기능, 시간 등)을 생각해 본다.

# Introduction

**jQuery** write less, do more

- ◎ 문법이 간결하다.
- ◎ 사용하기 쉽다.
- ◎ 빠르게 배울 수 있다.
- ◎ 다른 라이브러리들과 충돌을 일으키지 않는다.
- ◎ 다양한 플러그인이 존재한다.
- ◎ 브라우저 호환성 문제를 해결해준다.
- ◎ MIT and GPL licenses.

# Introduction

## Installation

- ◎ jquery.com에서 스크립트 파일을 다운로드하여 링크한다.

```
<script src="jquery.js"></script>  
<script>  
// your script goes here.  
</script>
```

- ◎ CDN(Content delivery network)도 사용할 수 있다.
  - Google Ajax API CDN: <http://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js>
  - Microsoft CDN: <http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.6.1.min.js>
  - jQuery CDN: <http://code.jquery.com/jquery-1.6.1.min.js>



# CODA

one-window web development.\*

 **Download**  
click here to get coda

 **Buy Now**  
instant and secure

 **Get Help**  
support and faq's

text editor + file transfer + svn + css + terminal + books + more = *whoah.*  
welcome to coda. grow beautiful code.

 New!


 Sites

 Files

 Editor

 Preview


 CSS

 More

## The story of Coda.

So, we code web sites by hand. And one day, it hit us: **our web workflow was wonky.** We'd have our text editor open, with [Transmit](#) open to save files to the server. We'd be previewing in Safari, adjusting SQL in a Terminal, using a CSS editor and reading references on the web. **"This could be easier,"** we declared. **"And much cooler."**

## What have we added to Coda lately?

 **Plug-ins.** This is huge! Teach Coda new tricks, and extend its functionality, lickety-split. Download plug-ins that others have written, or write your own plug-ins using your favorite scripting language. This is just the beginning — learn more in the [Coda Developer Zone](#). We look forward to seeing what you make!

# jQuery

core



# Core

## jQuery()

- ◎ jQuery( selector, [ context ] )
  - \$()로 축약해서 사용할 수 있다.
  - CSS 선택자를 이용해서 요소를 검색하고 jQuery 객체를 반환한다.
  - Selector Context
    - 검색 범위를 현재 문맥 범위로 좁혀서 사용할 수 있다.

```
$( 'div.foo' ).click(function() {  
    $( 'span', this ).addClass( 'bar' );  
});
```

# Core

## jQuery()

◎ jQuery( element ), jQuery( elementArray )

- DOM 요소를 직접 사용할 수 있다.
- this를 참조하여 jQuery 객체로 반환하여 사용할 수 있다.

```
$( 'div.foo' ).click( function() {  
    $( this ).slideUp();  
});
```

- AJAX를 통해 받은 XML 문서를 \$에 넘겨서 사용할 수 있다.

```
$.post( 'url.xml', function( data ) {  
    var $child = $( data ).find( 'child' );  
})
```

# Core

## jQuery()

- ◎ jQuery( jQuery object )
  - jQuery 요소를 인수로 받으면 복제된 jQuery 객체를 반환한다.
- ◎ jQuery()
  - 빈 인수를 받으면 .length가 0인 빈 객체를 반환한다.

# Core

## jQuery()

- ◎ jQuery( html, [ ownerDocument ] )
  - HTML 문자열을 인수로 받으면 새로운 요소를 생성한다.
  - 입력된 문자열에 따라 jQuery 내부적으로 createElement나 innerHTML을 사용해서 문서에 적용하게 된다.
  - html, title, head 등의 요소는 브라우저에 따라서 누락될 수 있다.
  - 입력하는 HTML의 문법은 브라우저 호환성을 위해서 정확히 지켜져야 한다.

# Core

## jQuery()

- ◎ jQuery( html, props )
  - 새로운 요소를 생성하고 속성을 정의한다.

```
$( "<div/>", {  
  "class": "test",  
  text: "Click me!",  
  click: function() {  
    $(this).toggleClass("test");  
  }  
}).appendTo("body");
```

# Core

## jQuery()

- ◎ jQuery( callback )

- \$(document).ready()와 같이 DOM이 완료될 때 까지 기다렸다가 콜백 함수를 실행한다.

```
$(function() {  
    // Document is ready  
});
```

# Core

## Chaining

- ◎ 대부분의 jQuery 메서드는 jQuery 객체를 반환한다.
  - 메서드를 연이어서 지정할 수 있다.

```
$('#div.section').hide().addClass('gone');
```

- ◎ 필터가 적용된 경우 .end()로 이전에 참조한 객체로 되돌릴 수 있다.

```
$('#ul.first').find('.foo')  
    .css('background-color', 'red')  
.end().find('.bar')  
    .css('background-color', 'green')  
.end();
```

# jQuery

## selectors



# Selectors

## Basic

- ◎ CSS 셀렉터의 대부분을 지원한다.
  - All Selector ("\*")
  - Class Selector (".class")
  - Element Selector ("element")
  - ID Selector ("#id")
    - id는 한 페이지 안에서 중복되지 않지만 IE는 여러개를 선택하는 버그가 있다.
  - Multiple Selector ("selector1, selector2, selectorN")

# Selectors

## Attribute

- Has Attribute Selector [name]
- Attribute Equals Selector [name="value"]
- Attribute Not Equal Selector [name!="value"]
- Attribute Starts With Selector [name^="value"]
- Attribute Ends With Selector [name\$="value"]

# Selectors

## Attribute

- Multiple Attribute Selector [name="value"][name2="value2"]
- Attribute Contains Selector [name\*="value"]
- Attribute Contains Prefix Selector [name|="value"]
- Attribute Contains Word Selector [name~="value"]

# Selectors

## Basic Filter

- ◎ :first Selector, :last Selector
  - 선택된 요소 중 처음, 마지막 요소를 선택한다.
- ◎ :even Selector, :odd Selector
  - 짝수, 홀수번째 요소를 선택한다.

# Selectors

## Basic Filter

### ◎ :eq() Selector

- n 번째 요소를 선택한다.

```
<table border="1">
  <tr><td>TD #0</td><td>TD #1</td><td>TD #2</td></tr>
  <tr><td>TD #3</td><td>TD #4</td><td>TD #5</td></tr>
  <tr><td>TD #6</td><td>TD #7</td><td>TD #8</td></tr>
</table>
<script>$("td:eq(2)").css("color", "red");</script>
```

# Selectors

## Basic Filter

### ◎ :lt() Selector, :gt() Selector

- n 보다 작은, n 보다 큰 요소를 선택한다.

```
<table border="1">
  <tr><td>TD #0</td><td>TD #1</td><td>TD #2</td></tr>
  <tr><td>TD #3</td><td>TD #4</td><td>TD #5</td></tr>
  <tr><td>TD #6</td><td>TD #7</td><td>TD #8</td></tr>
</table>
<script>$("td:lt(4)").css("color", "red");</script>
```

# Selectors

## Basic Filter

- ◎ :header Selector
  - 제목 요소를 선택한다.
- ◎ :not() Selector
  - 선택자에 해당되지 않는 요소를 선택한다.
- ◎ :animated Selector
  - 현재 움직이고 있는 요소 선택한다.

# Selectors

## Child Filter

- ◎ :first-child Selector, :last-child Selector
  - 부모 요소의 처음, 마지막 요소를 선택한다.
- ◎ :only-child Selector
  - 형제가 없는 요소를 선택한다.



# Selectors

## Child Filter

### ◎ :nth-child() Selector

- 부모 요소의 n 번째 자식 요소를 선택한다.

```
<ul>
  <li>John</li>
  <li>Karl</li>
  <li>Brandon</li>
</ul>
<ul>
  <li>Sam</li>
</ul>
<ul>
  <li>Glen</li>
  <li>Tane</li>
  <li>Ralph</li>
  <li>David</li>
</ul>
<script>$("ul li:nth-child(2)").append("<span> - 2nd!</span>");</script>
```

# Selectors

## Content Filter

### ◎ :contains() Selector

- 특정 텍스트를 포함하는 요소를 선택한다.

```
<div>John Resig</div>  
<div>George Martin</div>  
<div>Malcom John Sinclair</div>  
<div>J. Ohn</div>  
<script>  
$("div:contains('John')").css("text-decoration", "underline");  
</script>
```

# Selectors

## Content Filter

- ◎ :empty Selector

- 빈요소를 선택한다.

- ◎ :parent Selector

- 자식이 있는 부모 요소를 선택한다.

- ◎ :has() Selector

- 셀렉터에 해당하는 요소를 포함하고 있는 요소를 선택한다.

# Selectors

## Form

- ◎ :input Selector, :checkbox Selector, :radio Selector
- ◎ :text Selector, :password Selector, :file Selector
- ◎ :button Selector, :submit Selector, :image Selector, :reset Selector
- ◎ :focus selector
- ◎ :checked Selector
- ◎ :selected Selector
- ◎ :enabled Selector, :disabled Selector

# Selectors

## Hierarchy

- ◎ Child Selector ("parent > child")
  - 부모의 자식 요소를 선택한다.
- ◎ Descendant Selector ("ancestor descendant")
  - 하위 요소를 모두 선택한다.
- ◎ Next Adjacent Selector ("prev + next")
  - prev 다음에 인접한 next 요소를 선택한다.

# Selectors

## Hierarchy

- ◎ Next Siblings Selector ("prev ~ siblings")
  - prev 이후에 나오는 형제 요소들을 선택한다.

```
<div>div (doesn't match since before #prev)</div>  
<span id="prev">span#prev</span>  
<div>div sibling</div>  
<div>div sibling <div id="small">div niece</div></div>  
<span>span sibling (not div)</span>  
<div>div sibling</div>  
<script>$("#prev ~ div").css("border", "3px groove blue");</script>
```

# Selectors

## Visibility Filter

- ◎ :visible Selector
  - 보이는 요소를 선택한다.
- ◎ :hidden Selector
  - 보이지 않는 요소를 선택한다.

# jQuery

traversing



# Traversing

## Tree Traversal

### ◎ .children()

- 자식요소를 취한다. 셀렉터를 추가로 명시할 수 있다.

```
<p>Hello (this is a paragraph)</p>
<div><span>Hello Again (this span is a child of the a div)</span></div>
<p>And <span>Again</span> (in another paragraph)</p>
<div>And One Last <span>Time</span> (most text directly in a div)</div>
<script>
$("div").children().css("border-bottom", "3px double red");
</script>
```

# Traversing

## Tree Traversal

- ◎ `.siblings()`

- 형제 요소를 취한다.

```
<div><span>Hello</span></div>  
<p class="selected">Hello Again</p>  
<p>And Again</p>  
<script>  
$("p").siblings(".selected").css("background", "yellow");  
</script>
```

# Traversing

## Tree Traversal

### ◎ .closest()

- 현재 요소를 포함해서 셀렉터에 해당하는 가까운 조상 요소를 취한다.

### ◎ .find()

- 자손 요소 중에 셀렉터, jQuery 객체, DOM 요소에 해당하는 요소를 취한다.

```
<p><span>Hello</span>, how are you?</p>
<p>Me? I'm <span>good</span>.</p>
<div>Did you <span>eat</span> yet?</div>
<script>
var $spans = $('span');
$("p").find( $spans ).css('color','red');
</script>
```

# Traversing

## Tree Traversal

- ◎ `.next()`
  - 다음에 오는 요소를 취한다.
- ◎ `.nextAll()`
  - 다음에 오는 모든 요소를 취한다.
- ◎ `.nextUntil()`
  - 선택터에 해당하는 요소 전까지의 다음 요소를 취한다.

# Traversing

## Tree Traversal

- ◎ .prev()
  - 이전에 오는 요소를 취한다.
- ◎ .prevAll()
  - 이전에 오는 모든 요소를 취한다.
- ◎ .prevUntil()
  - 셀렉터에 해당하는 요소 전까지의 이전 요소를 취한다.

# Traversing

## Tree Traversal

- ◎ .parent()
  - 부모 요소를 취한다.
- ◎ .parents()
  - 루트 요소까지의 모든 부모 요소를 취한다.
- ◎ .parentsUntil()
  - 선택터와 매칭되는 요소 전까지의 조상 요소를 취한다.
- ◎ .offsetParent()
  - 위치가 지정된 가장 가까운 조상 요소를 취한다.

# Traversing

## Filtering

- ◎ `.eq()`
  - `n` 번째 요소를 취한다.
- ◎ `.first()`
  - 첫번째 요소를 취한다.
- ◎ `.last()`
  - 마지막 요소를 취한다.
- ◎ `.slice()`
  - 시작과 끝 사이에 해당하는 요소를 취한다.

# Traversing

## Filtering

- ◎ .has()

- 셀렉터에 해당되는 요소나 해당 DOM 요소를 가지고 있는 요소를 취한다.

- ◎ .not()

- 해당되는 요소를 제외한다.

- ◎ .is()

- 현재 요소들이 셀렉터나 요소, jQuery 객체에 해당하는지를 확인한다.



# Traversing

## Filtering

- ◎ .map()

- 현재 선택된 요소들을 콜백함수에 보내서 새로운 jQuery 객체를 반환한다.

- ◎ .filter()

- 셀렉터나 함수, jQuery 객체, 요소에 해당하는 요소를 취한다.

# Traversing

## Miscellaneous Traversing

- ◎ .add()

- 인수의 요소가 추가된 jQuery 객체를 반환한다.

- ◎ .end()

- 최근 필터링 결과를 끝내고 체인을 이전 상태로 되돌린다.

```
<p><span>Hello</span>, how are you?</p>  
<script>  
$("p").find("span").end().css("border", "2px red solid");  
</script>
```

# Traversing

## Miscellaneous Traversing

### ◎ .contents()

- 안에 포함하고 있는 자식들을 가져온다. .children()과 유사하지만 .contents()는 텍스트 노드도 같이 포함한다.

```
<div class="container">
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
  sed do eiusmod tempor incididunt ut labore et dolore magna
  aliqua.</p>
  <br /><br />
  <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco
  laboris nisi ut aliquip ex ea commodo consequat.</p>
  <br /><br />
  <p>Duis aute irure dolor in reprehenderit in voluptate velit
  esse cillum dolore eu fugiat nulla pariatur.</p>
</div>
<script>
$( '.container' ).contents().filter(function() {
  return this.nodeType == 3;
}).wrap('<p></p>').end().filter('br').remove();
</script>
```

# Traversing

## Miscellaneous Traversing

- ◎ .andSelf()
  - 이전 체이닝 스택과 자기 자신을 합친다.

```
<div>
  <p>First Paragraph</p>
  <p>Second Paragraph</p>
</div>
<script>
$("div").find("p").andSelf().addClass("border");
</script>
```

# Traversing

## Collection Manipulation

### ◎ .each()

- \$객체를 순서대로 탐색하면서 각각의 객체에 함수를 실행한다.

```
<ul>
  <li>foo</li>
  <li>bar</li>
</ul>
<script>
$('li').each(function(index) {
  alert(index + ': ' + $(this).text());
});
</script>
```

# jQuery

manipulation

# Manipulation

## General Attributes

### ◎ .attr()

- HTML 속성 값을 취하거나 추가, 변경한다.

```
$('#greatphoto').attr('alt', 'Beijing Brush Seller');
```

- JSON 객체를 사용해서 여러 속성을 동시에 적용할 수 있다.

```
$('#greatphoto').attr({  
  alt: 'Beijing Brush Seller',  
  title: 'photo by Kelly Clark'  
});
```

### ◎ .removeAttr()

- HTML 속성을 지운다.

# Manipulation

## General Attributes

### ◎ .prop()

- 자바스크립트 속성 값을 취하거나 추가, 변경한다.
- `<input type="checkbox" checked="checked" />` 의 경우 (jQuery 1.6 이상)
  - `$('#input').attr('checked') == 'checked'` (string type)
  - `$('#input').prop('checked') == true` (boolean type)

### ◎ .removeProp()

- 자바스크립트 속성을 지운다.



# Manipulation

## General Attributes

- ◎ `.val()`
  - 값(value)를 취한다.

```
$('input:text.items').val(function(index, value) {  
    return value + ' ' + this.className;  
});
```

# Manipulation

## Class Attribute

- ◎ `.addClass()`

- 클래스(class)를 추가한다.

- ◎ `.removeClass()`

- 한개 또는 여러개의 클래스를 지운다.

```
$("#p").removeClass("myClass noClass").addClass("yourClass");
```

- ◎ `.hasClass()`

- 클래스가 지정되어 있는지를 확인한다.

```
var hasFoo = $('p').hasClass('foo');
```

# Manipulation

## Class Attribute

### ◎ .toggleClass()

- 현재의 클래스 값에 따라서 클래스를 추가하거나 지운다.

```
<p class="blue">Click to toggle</p>
<p class="blue highlight">highlight</p>
<p class="blue">on these</p>
<p class="blue">paragraphs</p>
<script>
$("p").click(function () {
    $(this).toggleClass("highlight");
});
</script>
```

# Manipulation

## DOM Insertion, Inside

### ◎ .text()

- 텍스트 콘텐츠를 취하거나 추가, 변경한다.

```
$('div.demo-container').text('<p>This is a test.</p>');
```

### ◎ .html()

- HTML 콘텐츠를 취하거나 추가, 변경한다.

```
$('div.demo-container').html('<p>All new content. <em>You bet!</em></p>');
```

# Manipulation

## DOM Insertion, Inside

- ◎ .prepend()
  - 인수를 jQuery 객체의 시작 지점에 자식 노드로 추가한다.
- ◎ .prependTo()
  - jQuery 객체를 인수의 시작 지점에 자식 노드로 추가한다.

# Manipulation

## DOM Insertion, Inside

- ◎ `.append()`
  - 인수를 jQuery 객체의 끝나는 지점에 자식 노드로 추가한다.
- ◎ `.appendTo()`
  - jQuery 객체를 인수의 끝나는 지점에 자식 노드로 추가한다.

# Manipulation

## DOM Insertion, Outside

- ◎ `.before()`
  - 인수를 jQuery 객체의 이전에 형제 노드로 추가한다.
- ◎ `.insertBefore()`
  - jQuery 객체를 인수의 이전에 형제 노드로 추가한다.

# Manipulation

## DOM Insertion, Outside

- ◎ .after()
  - 인수를 jQuery 객체의 이후에 형제 노드로 추가한다.
- ◎ .insertAfter()
  - jQuery 객체를 인수의 이후에 형제 노드로 추가한다.



# Manipulation

## DOM Removal

- ◎ `.remove()`
  - DOM에서 요소를 삭제한다.
- ◎ `.detach()`
  - DOM에서 요소를 삭제한다. 데이터가 유지되기 때문에 저장했다가 다른 곳에 사용할 수 있다.
- ◎ `.empty()`
  - DOM에서 자식 요소를 모두 삭제한다.

# Manipulation

## DOM Replacement

### ◎ .replaceWith()

- 인수의 요소로 jQuery 객체를 대체한다.

```
<buttendiv>First</buttendiv>
<buttendiv>Second</buttendiv>
<buttendiv>Third</buttendiv>

<script>
$("button").click(function () {
    $(this).replaceWith( "<div>" + $(this).text() + "</div>" );
});
</script>
```

### ◎ .replaceAll()

- jQuery 객체로 인수의 요소를 대체한다.

# Manipulation

## DOM Insertion, Around

### ◎ .wrap()

- 감싸는 요소를 추가한다.

```
<div><p>Hello</p></div>  
<div><p>cruel</p></div>  
<div><p>World</p></div>  
<script>$ ("p") .wrap ("<div></div>");</script>
```

### ◎ .wrapAll()

- 전체를 감싸는 요소를 추가한다.

```
<div><p>Hello</p>  
<p>cruel</p>  
<p>World</p></div>  
<script>$ ("p") .wrapAll ("<div></div>");</script>
```

# Manipulation

## DOM Insertion, Around

- ◎ `.unwrap()`
  - 감싸고 있는 부모 요소를 제거한다.
- ◎ `.wrapInner()`
  - 내부의 콘텐츠를 감싸는 요소를 추가한다.

```
<p><b>Hello</b></p>  
<p><b>cruel</b></p>  
<p><b>World</b></p>  
<script>$ ("p") .wrapInner ("</b>");</script>
```

# Manipulation

## Copying

- ◎ .clone()

- 요소를 복제한다.

```
<b>Hello</b><p><b>Hello</b>, how are you?</p>
```

```
<script>  
    $("b").clone().prependTo("p");  
</script>
```

# Manipulation

## Style Properties

### ◎ .css()

- 스타일 속성에 따른 값을 취하거나 추가, 변경한다.
- 높이 등을 취할 때에는 단위도 같이 반환한다.

```
$('#mydiv').css('color', 'green');
```

- index를 취해 함수를 사용할 수 있다.

```
$('#div.example').css('width', function(index) {  
    return index * 50;  
});
```

# Manipulation

## Style Properties

### ◎ .height()

- 요소의 높이를 취하거나 설정한다.

```
var height = $('div#intro').height();
```

### ◎ .innerHeight()

- 패딩 영역을 포함한 요소의 내부 높이를 취한다.

### ◎ .outerHeight()

- 패딩, 보더 영역을 포함한 요소의 외부 높이를 취한다.
- 마진 영역 포함을 선택할 수 있다.

# Manipulation

## Style Properties

- ◎ `.width()`

- 요소의 너비를 취하거나 설정한다.

- ◎ `.innerWidth()`

- 패딩 영역을 포함한 요소의 내부 너비를 취한다.

- ◎ `.outerWidth()`

- 패딩, 보더 영역을 포함한 요소의 외부 너비를 취한다.



# Manipulation

## Style Properties

- ◎ `.position()`
  - 부모 요소로 부터의 위치 값을 취한다.
- ◎ `.offset()`
  - 문서로 부터의 위치 값을 취한다.

# Manipulation

## Style Properties

- ◎ `.scrollLeft()`

- 좌우 스크롤 바의 위치를 취한다.

- ◎ `.scrollTop()`

- 상하 스크롤 바의 위치를 취한다.

# jQuery

event

# Event

## Event handling

- ◎ 이벤트가 발생할 때 실행될 함수를 지정한다.

```
$('#a:first').click(function(ev) {  
    $(this).css({backgroundColor: 'orange'});  
    return false; // Or ev.preventDefault();  
});  
$('#a:first').click();
```

# Event

## Keyboard Events

### ◎ .focusin()

- 포커스를 받을 때 발생하는 focusin 이벤트를 붙인다.

### ◎ .focusout()

- 포커스를 잃을 때 발생하는 focusout 이벤트를 붙인다.
- blur는 버블링이 발생하지 않지만 focusout은 발생한다.

# Event

## Keyboard Events

- .keydown()

- 키가 내려갈 때 발생하는 keydown 이벤트를 붙인다.

- .keyup()

- 키가 올라올 때 발생하는 keyup 이벤트를 붙인다.

- .keypress()

- 키가 눌릴 때 발생하는 keypress 이벤트를 붙인다.

# Event

## Mouse Events

- `.click()`

- 클릭할 때 발생하는 click 이벤트를 붙인다.

- `.dblclick()`

- 더블 클릭할 때 발생하는 click 이벤트를 붙인다.

- `.mouseup()`

- 마우스 버튼을 올릴 때 발생하는 mouseup 이벤트를 붙인다.

- `.mousedown()`

- 마우스 버튼을 내릴 때 발생하는 mousedown 이벤트를 붙인다.

# Event

## Mouse Events

### ◎ .toggle()

- 두개 이상의 핸들러를 순차적으로 실행되도록 한다.
- 링크나 버튼에 선언될 경우 .preventDefault()를 수행하기 때문에 원래의 동작은 하지 않는다.

```
$( "td" ).toggle (
    function () {
        $(this).addClass ("selected");
    },
    function () {
        $(this).removeClass ("selected");
    }
);
```



# Event

## Mouse Events

- ◎ .mouseover()

- 마우스 포인터가 올라갈 때 발생하는 mouseover 이벤트를 붙인다.

- ◎ .mouseout()

- 마우스 포인터가 내려갈 때 발생하는 mouseout 이벤트를 붙인다.

- ◎ .mousemove()

- 마우스 포인터가 움직일 때 발생하는 mousemove 이벤트를 붙인다.

# Event

## Mouse Events

- ◎ .mouseenter()

- 마우스 포인터가 들어올 때 발생하는 이벤트를 붙인다.

- ◎ .mouseleave()

- 마우스 포인터가 나갈 때 발생하는 이벤트를 붙인다.

# Event

## Mouse Events

### ◎ .hover()

- 마우스 포인터가 들어오고 나갈 때 발생하는 이벤트에 동작을 설정한다.

```
$( "td" ).hover (
    function () {
        $(this).addClass ("hover");
    },
    function () {
        $(this).removeClass ("hover");
    }
);
```

# Event

## Form Events

- ◎ .focus()

- 포커스를 받을 때 발생하는 focus 이벤트를 붙인다.

- ◎ .blur()

- 포커스를 잃을 때 발생하는 blur 이벤트를 붙인다.

# Event

## Form Events

### ◎ .change()

- 값이 변경될 때 발생하는 change 이벤트를 붙인다.

### ◎ .select()

- 선택될 때 발생하는 select 이벤트를 붙인다.

### ◎ .submit()

- 서식이 전송될 때 발생하는 submit 이벤트를 붙인다.
- return false로 서식 전송을 막을 수 있다.

```
$('#form').submit(function () {  
    return false;  
});
```

# Event

## Document Loading

- ◎ .load()

- 페이지의 로딩이 완료되었을 때 발생하는 load 이벤트를 붙인다.

- ◎ .ready()

- DOM의 로딩이 완료되었을 때 작동될 함수를 지정한다.

- ◎ .unload()

- 페이지를 벗어날 때 발생하는 unload 이벤트를 붙인다.

# Event

## Browser Events

- ◎ .error()

- 오류가 발생할 때 발생하는 error 이벤트를 붙인다.

- ◎ .resize()

- 화면 크기가 변경될 때 발생하는 resize 이벤트를 붙인다.

- ◎ .scroll()

- 스크롤이 움직일 때 발생하는 scroll 이벤트를 붙인다.

# Event

## Event Handler Attachment

- ◎ .bind()
  - 요소에 이벤트 핸들러를 지정한다.
- ◎ .unbind()
  - 지정된 이벤트 핸들러를 삭제한다.
- ◎ .one()
  - .bind()와 동일하지만 이벤트가 한번 실행된 후 .unbind()된다.



# Event

## Event Handler Attachment

### ◎ .live()

- 앞으로 생성될 요소에도 이벤트 핸들러를 지정한다.
- 이벤트 델리게이션(event delegation)을 사용해서 새로 생성되는 요소에도 이벤트가 적용된다는 점에서 .bind()와 다르다.

### ◎ .die()

- .live()로 생성된 이벤트 핸들러를 삭제한다.

# Event

## Event Handler Attachment

### ◎ .delegate()

- 특정 요소에 이벤트 핸들러를 지정한다.
- .live()와 동일하지만 이벤트를 루트 요소가 아니라 특정 DOM 요소에 지정할 수 있다.

### ◎ .undelegate()

- .delegate()로 생성된 이벤트 핸들러를 삭제한다.

# Event

## Event Handler Attachment

### ◎ .trigger()

- 요소에 지정된 이벤트 핸들러와 동작을 실행시킨다.

```
$('#foo').bind('click', function() {  
    alert($(this).text());  
});  
$('#foo').trigger('click');
```

### ◎ .triggerHandler()

- .trigger()와 동일하지만 지정된 핸들러 동작만 수행한다.

# jQuery

effect

# Effect

## Basics

### ◎ .show()

- 선택된 요소를 보여준다.

### ◎ .hide()

- 선택된 요소를 감춘다.

```
$( '.target' ).hide( 'slow' );
```

### ◎ .toggle()

- 선택된 요소를 상태에 따라서 감추거나 보여준다.

# Effect

## Fading

- ◎ `.fadeIn()`

- 선택된 요소를 투명도를 조절하여 서서히 보여준다.

- ◎ `.fadeOut()`

- 선택된 요소를 투명도를 조절하여 서서히 감춘다.

```
$( '.target' ).fadeOut(2000);
```

# Effect

## Fading

- ◎ .fadeTo()

- 선택된 요소를 지정된 투명도로 서서히 조절한다.

- ◎ .fadeToggle()

- 선택된 요소를 투명도를 조절하여 감추거나 보여준다.

# Effect

## Sliding

### ◎ .slideUp()

- 선택된 요소의 높이를 줄여서 감춘다.

```
$('.target').slideUp('fast');
```

### ◎ .slideDown()

- 선택된 요소의 높이를 원상태로 복원한다.

### ◎ .slideToggle()

- 선택된 요소를 슬라이드 효과로 감추거나 보여준다.



# Effect

## Custom

### ◎ .animate()

- CSS 속성(숫자)으로 애니메이션을 만든다.
- width, height, left, scrollTop, scrollLeft 등의 사용이 가능하다.
- shorthand는 지원되지 않는다.

```
$('#clickme').click(function() {  
    $('#book').animate({  
        opacity: 0.25,  
        left: '+=50',  
        height: 'toggle'  
    }, 5000, function() {  
        // Animation complete.  
    });  
});
```

# Effect

## Custom

### ◎ .stop()

- 현재 보여지고 있는 애니메이션을 멈춘다.

### ◎ .delay()

- 실행을 주어진 시간만큼 연기한다.

```
<p><button>Run</button></p>
<div class="first"></div>
<div class="second"></div>

<script>
  $("button").click(function() {
    $("div.first").slideUp(300).delay(800).fadeIn(400);
    $("div.second").slideUp(300).fadeIn(400);
  });
</script>
```

# Effect

## Custom

### ◎ .queue()

- jQuery의 애니메이션은 기본적으로 fx라 불리는 큐(queue)에 의해서 관리가 된다.
- 이러한 큐를 확인하거나 수정하는데 사용된다.

### ◎ .dequeue()

- 큐에 있는 다음 함수를 실행하게 한다.

### ◎ .clearQueue()

- 실행되지 않은 큐에있는 함수들을 모두 제거한다.

# Effect

## Custom

- ◎ jQuery.fx.interval

- 초당 프레임 수를 조절한다.
- 기본값은 초당 13 프레임이다.

- ◎ jQuery.fx.off

- 모든 애니메이션이 종료되고 최종 상태를 보여준다.
- 이후에 실행되는 애니메이션도 건너뛰게 된다.

# jQuery

## AJAX

# AJAX

## Shorthand Methods

◎ `.load( url, [ data ], [ complete(responseText, textStatus, XMLHttpRequest) ] )`

```
$('#result').load('ajax/test.html');
```

- url에 공백으로 분리된 인수가 있을 경우에 이를 선택어로 인식하고 해당 콘텐츠를 반환한다.

```
$('#result').load('ajax/test.html #container');
```

- 객체로 data를 전송할 경우 POST, 다른 경우는 GET으로 작동한다.
- 콜백 함수를 지정할 수 있다.

```
$('#result').load('ajax/test.html', function() {  
    alert('Load was performed.');
```

```
});
```

# AJAX

## Shorthand Methods

- ◎ \$.get( url, [ data ], [ success(data, textStatus, jqXHR) ], [ dataType ] )

```
$.get('ajax/test.html', function(data) {  
    $('#result').html(data);  
    alert('Load was performed.');
```

- ◎ \$.post( url, [ data ], [ success(data, textStatus, jqXHR) ], [ dataType ] )

```
$.post('ajax/test.html', function(data) {  
    $('#result').html(data);  
});
```

- ◎ \$.getJSON( url, [ data ], [ success(data, textStatus, jqXHR) ] )
- ◎ \$.getScript( url, [ success(data, textStatus) ] )

# AJAX

## Global Ajax Event Handlers

- `.ajaxStart( handler() )`
- `.ajaxStop( handler() )`
- `.ajaxSend( handler(event, jqXHR, ajaxOptions) )`
- `.ajaxComplete( handler(event, XMLHttpRequest, ajaxOptions) )`
- `.ajaxSuccess()`
- `.ajaxError( handler(event, jqXHR, ajaxSettings, thrownError) )`



# jQuery

utilities

# Utilities

## Feature detection

### ◎ \$.support

- 브라우저의 기능을 탐지하여 기능이 제공되는지 판별한다.
- ajax, boxModel, changeBubbles, checkClone, checkOn, cors, cssFloat, hrefNormalized, htmlSerialize, leadingWhitespace, noCloneChecked, noCloneEvent, opacity, optDisabled, optSelected, scriptEval(), style, submitBubbles, tbody
  - <http://api.jquery.com/jquery.support/>

```
$.support.boxModel
```

# Utilities

## Feature detection

### ◎ \$.browser

- 1.3에서 폐지되었고 \$.support 사용이 권장된다.
- webkit, safari (deprecated), opera, msie, mozilla

```
if ( $.browser.msie ) {  
    $("#div ul li").css( "display","inline" );  
} else {  
    $("#div ul li").css( "display","inline-table" );  
}
```

```
if ( $.browser.msie ) {  
    alert( $.browser.version );  
}
```

# Utilities

## Array utility

- ◎ \$.isArray( obj )
  - 객체가 배열인지를 판단한다.
- ◎ \$.inArray( value, array )
  - 주어진 값과 일치하는 항목의 색인을 반환한다. 없으면 -1을 반환한다.
- ◎ \$.makeArray( obj )
  - 객체를 배열로 변환한다.

# Utilities

## Array utility

- ◎ `$.grep( array, function(elementOfArray, indexInArray), [invert] )`
  - 주어진 함수의 조건에 맞는 배열을 반환한다.
- ◎ `$.map( array, callback(elementOfArray, indexInArray) )`
  - 주어진 함수의 조건에 따라 새로운 배열을 생성한다.
- ◎ `$.merge( first, second )`
  - 두개의 배열을 첫번째 배열에 합친다.

# Utilities

## Object type

- ◎ \$.isEmptyObject( object )
  - 객체가 비었는지(속성이 없는지)를 판단한다.
- ◎ \$.isPlainObject( object )
  - 객체가 일반 객체({} 또는 new Object로 생성)인지를 판단한다.

# Utilities

## Object type

- ◎ \$.isFunction( obj )
  - 객체가 함수인지를 판단한다.
- ◎ \$.isWindow( obj )
  - 객체가 윈도우인지를 판단한다.
- ◎ \$.isXMLDoc()
  - 객체가 XML 문서인지를 판단한다.

# Utilities

## Object type

- ◎ \$.type( obj )
  - 객체의 형을 판단한다.

```
jQuery.type(true) === "boolean"  
jQuery.type(3) === "number"  
jQuery.type("test") === "string"  
jQuery.type(function(){} ) === "function"  
jQuery.type([]) === "array"  
jQuery.type(new Date()) === "date"  
jQuery.type(/test/) === "regexp"
```



# Utilities

## Parsing

- ◎ \$.parseJSON( json )
  - JSON 문자열을 자바스크립트 객체로 변환한다.
- ◎ \$.parseXML( data )
  - XML 문자열을 XML 문서로 변환한다.

# Utilities

**etc.**

- ◎ \$.contains( container, contained )
  - DOM 요소가 다른 DOM 요소 안에 있는지를 검사한다.
- ◎ \$.each( collection, callback(indexInArray, valueOfElement) )
  - 객체나 배열을 순서대로 탐색하면서 각각에 함수를 실행한다.

```
$.each([52, 97], function(index, value) {  
    alert(index + ': ' + value);  
});
```

- ◎ \$.unique( array )
  - 복제된 DOM 요소를 제외한 원래의 DOM 요소를 취한다.

# Utilities

**etc.**

- ◎ \$.extend( target, [object1,] [objectN] )
  - 두개 이상의 객체의 내용을 첫번째 객체에 합친다.
  - target 만 지정될 경우 제이쿼리 자체가 확장된다.
    - 플러그인 등을 만들 때 유용하다.
  - 재귀적으로 사용할 경우 첫번째 인자에 true를 지정한다.
    - 지정하지 않을 경우 같은 속성값은 덮어써진다.

```
var object = $.extend({}, object1, object2);
```

# Utilities

**etc.**

◎ \$.now()

- 현재시간을 나타내는 숫자를 반환한다. (= (new Date).getTime())

◎ \$.trim( str )

- 앞뒤 공백 문자를 없앤다.

◎ \$.globalEval( code )

- 자바스크립트 코드를 전역을 실행한다.

◎ \$.noop()

- 아무일도 하지 않는 함수이다.