전제 지식

이 책을 읽어 나가기에 앞서서 표 01과 같은 지식이 필요하다. 우선, HTML로 문서(콘텐츠)를 만들어 CSS로 모양을 정돈하고, JavaScript로 콘텐츠와 모양을 조작하는 트라이앵글과 같은 관계를 전체적인 이미지로 이해해두도록 하자(그림 01).

표 01 ▶ **이 책을 활용하기 위해서 필요한 지식**

기술	개요
HTML (HyperText Markup Language)	웹 페이지를 작성하기 위한 마크업 언어
CSS (Cascading StyleSheet)	웹 페이지를 디자인하기 위한 스타일 시트
JavaScript	브라우저 상에서 동작하는 스크립트 언어
jQuery	JavaScript 상에서 동작하는 범용 라이브러리

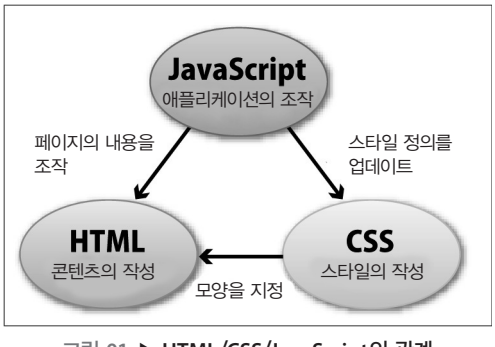


그림 01 ▶ **HTML/CSS/JavaScript의 관계**

jQuery는 JavaScript에서의 개발을 효율화하기 위한 라이브러리다.

앞으로 이것들 중 특히 HTML, jQuery에 대해서 최소한 파악해두어야 할 기본적인 문법과 구문에 대해 설명하겠다.

HTML5의 기본

이 책은 HTML의 최신 버전인 HTML5를 전제로 코드를 작성하였다.

HTML5는 아직도 기술적 사양의 변경이 진행 중이므로 현재 사용 중인 브라우저에서 충분히 대응하고 있다고 확실히 단정 지을 수는 없다. 하지만 HTML5는 기존 HTML4와의 호환을 강하게 의식하여 설계되었으므로, 올바른 구문만 잘 이해하고 있으면 HTML5를 지원하지 않는 브라우저에서도 페이지를 정상적으로 표시할 수 있다.

HTML5는 향후 분명히 널리 보급될 것이므로 HTML5의 기본적인 작성법에 대해 초반부터 익숙해지는 것은 상당히 중요하다.

그럼, 리스트 01의 예제를 통해 최소한 파악해야 할 큰 틀의 기법을 이해해보도록 하자.★

리스트 01 ▶ **HTML5 페이지의 기본(html5.html)**

```
<!DOCTYPE html> ──────────────────────────────────❶
<html>
<head>
<meta charset="UTF-8" /> ─────────────────────────❷❺
<title>HTML5의 기본</title>
</head>
<body>
<p>안녕하세요, 세계!<br /> ──❹──────────────────❸
안녕하세요, 여러분!</p> ──────────────────────────
</body>
</html>
```

HTML 페이지 선언하기(❶)

HTML 페이지에서는 우선 현재의 페이지가 HTML로 작성되어 있음을 브라우저에 통지할 필요가 있다. 이것을 실행하는 것이 ❶의 코드다. 문서의 형식을 나타내는 정보이니만큼 문서형 선언이라고도 불린다.

★ 여기서의 목적은 HTML5의 개별 태그를 이해하는 것이 아니다. 따라서 HTML5에서 이용 가능한 개별 태그에 대해서는 'HTML5 태그 레퍼런스(http://www.html5.jp/tag/elements/)★★' 등의 자료를 참조하기 바란다.

★★ **역주** HTML5의 한글 번역본은 다음과 같다. http://html5.clearboth.org/spec

HTML4 이전에는 다음과 같은 문서형 선언이 필요했다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

다시 비교해볼 것까지도 없이 HTML5에서는 선언이 확실히 단순해졌다.

문자 코드 선언하기(❷)

문자 코드란 컴퓨터에서 문자를 나타내기 위한 규칙을 말한다. 문자 코드에는 EUC-KR, Shift-JIS나 EUC-JP, ISO-2022-JP등 여러 종류가 있는데, HTML5에서는 UTF-8이라는 문자 코드를 사용할 것을 추천하고 있다.★

문자 코드의 선언은 리스트 01에서의 ❷부분이다. 이것도 HTML4에서는 다음과 같이 작성하지 않으면 안 되었으나 HTML5에서는 상당히 깔끔하게 된 것을 알 수 있을 것이다.

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

닫는 태그 생략 금지(❸)

HTML5에서는 정해진 조건 하에서는 닫는 태그를 생략할 수 있다. 예를 들어, 리스트 01의 ❸은 다음과 같이 작성해도 올바른 HTML5 페이지다.

```
<p>안녕하세요, 세계!<br />
안녕하세요, 여러분! … 닫는 태그 </p>가 없음!
```

그러나 닫는 태그는 언제라도 생략 가능한 것이 아니다. 생략해도 좋은 요소와 상황은 엄밀히 정해져 있다. 또한, 태그의 종료가 불명료하다는 것 자체가 코드의 가독성 면에서도 바람

★ 다른 문자 코드도 이용할 수 있지만, 사용하는 기능에 따라서는 미처 생각지 못한 버그의 원인이 된다. 특별한 이유가 없는 한 우선은 UTF-8을 이용하도록 하자.

직하지 못하다.

약간의 수고를 아까워하지 말고 닫는 태그를 생략하지 않는 것을 기본으로 삼자.

빈 요소는 '~/>'로 닫기❹

빈 요소란 요소의 아래에 텍스트를 갖지 않은 요소를 말한다. 구체적으로는 표 02와 같은 요소가 있다.

본체가 없으므로 닫는 태그도 작성해서는 안 된다. 예를 들어, 〈img〉〈/img〉와 같은 기술은 올바르지 않다. 대신에 〈img /〉라고 작성하기 바란다. 이로써 그 요소가 빈 요소인 것(=닫는 태그가 없다)을 명확하게 표현할 수 있다.

〈img〉라고 작성해도 틀린 것은 아니나, 〈img /〉 쪽이 봤을 때 코드의 가독성이 좋다라는 의미에서 필자는 이 작성법을 추천하고 있다.

표 02 ▶ **주요 빈 요소**

〈area〉	〈base〉	〈br〉	〈col〉	〈embed〉	〈hr〉
〈img〉	〈input〉	〈link〉	〈meta〉	〈param〉	〈source〉

속성 값은 큰따옴표로 묶는다❺

HTML5에서의 속성 값은 ["](큰따옴표), [']('작은따옴표)로 묶든가 아니면 그냥 값만으로도 표현할 수 있다. 따라서 리스트 02의 코드는 모두 다 올바른 HTML5 코드다.

단, 작은따옴표로 묶은 속성 값에 작은따옴표를 포함할 수는 없다(큰따옴표의 경우도 동일하다). 또한, 그냥 값만인 경우에는 ["], ['], [=], [<], [>], [`]를 포함해서는 안 된다.

기본적으로는 애플리케이션 안에서 통일해서 사용하고 있다면 어떠한 기술 방식을 사용해도 상관없으나, 적어도 필자는 제한이 많은 값만 사용하는 경우는 이용하지 말아야 한다고 생각하고 있다. 이 책에서는 큰따옴표를 우선적으로 이용한다.

리스트 02 ▶ 속성의 작성 방법

```
<meta charset="UTF-8" />
<meta charset='UTF-8' />
<meta charset=UTF-8 />
```

jQuery의 기본

jQuery는 존 레식(John Resig)에 의해서 개발된 JavaScript 경량 라이브러리다. [Write Less, Do More(보다 간단히, 보다 많은 것을)]가 모토이며, 지금까지 복잡한 코드를 필요로 했던 조작을 손쉽게 구현해낼 수 있는 편리함이 특징이다. 물론, 단순히 간편하다는 점 뿐만이 아니라 기본적인 HTML 페이지의 조작에서부터 애니메이션 기능, Ajax 통신, 이벤트 처리 등 JavaScript에 의한 웹 애플리케이션 개발의 범용적인 기능 거의 대부분을 폭넓게 지원하고 있다.

또한, jQuery 플러그인이라고 불리는 확장 라이브러리가 몇천에서 몇만 개씩 제공되고 있어, 이것들을 이용함으로써 jQuery 기능을 제한 없이 확장할 수 있다. 예를 들어 그림 02와 같은 차트 작성, 슬라이드 쇼, 엑셀과 같은 테이블도 jQuery와 그 플러그인을 이용함으로써 아주 간단한 코드로 구현 가능하다.

jQuery는 이러한 고기능성과 손쉽게 이용할 수 있는 간편함을 결합했기에 단기간에 JavaScript 라이브러리들 중 산업 표준으로서의 지위를 확립하였고, 이제는 JavaScript 개발에 없어서는 안 되는 존재가 되었다(또한, 이 책에서 소개하고 있는 라이브러리도 jQuery에 의존하고 있는 것이 많다).

그러므로 이후의 개별 라이브러리를 소개하기에 앞서 jQuery의 도입 방법, 그리고 기본적인 구문에 대해 이해해두도록 하자.

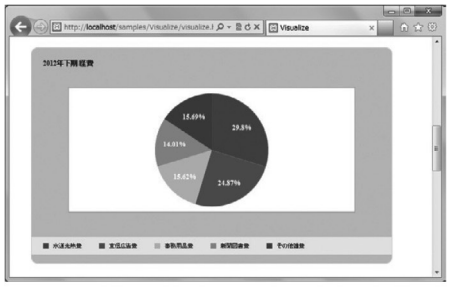


그림 02 ▶ **jQuery 플러그인을 이용한 웹 페이지의 예**

jQuery의 도입

jQuery의 동작에 필요한 라이브러리는 jQuery 공식 사이트에서 CDN★으로 제공되고 있다. 준비가 불필요하다는 점과 서버 측의 응답(Response)이 우수하다는 점에서도 이 방법을 이용하는 것이 편리하다. CDN을 이용하려면 리스트 03과 같은 코드를 웹 페이지에 포함시키기만 하면 된다.

★ Content Delivery Network(콘텐츠 전송 네트워크). 라이브러리나 스타일 시트, 이미지 리소스 등 콘텐츠의 전달에 특화된 전용 네트워크를 말한다.

만약 오프라인 환경에서 동작하고 싶은 경우에는 jQuery 공식 사이트(그림 03)에서 jquery-1.9.0.js를 다운로드하면 된다. 메인 페이지 우측 상단의 [Download jQuery] 버튼으로 다운로드 페이지로 이동하여 [Download the compressed, production jQuery 1.9.0] 링크를 클릭한다. 다운로드한 .js파일을 도큐먼트 폴더에 배치하고서 리스트 03과 동일한 방식으로 라이브러리를 임포트한다.

환경 준비가 되었다면 우선은 구체적인 예제를 동작시켜 보자. 리스트 04는 id 속성이 list인 〈ul〉 요소로부터 class 속성이 new인 이미지를 검색하여 이것을 확대시켜 애니메이션하는 예다. 이후로 이 예제를 전제로 jQuery의 기본적인 포인트를 파악해 나가겠다.

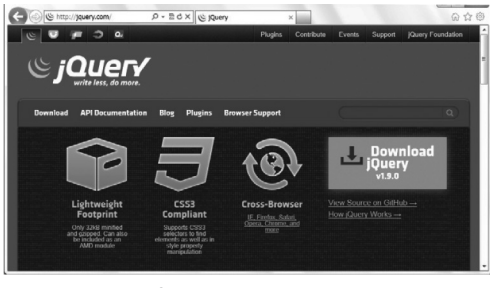


그림 03 ▶ http://jquery.com/

리스트 03 ▶ **jQuery를 이용하기 위한 코드**

```
<script type="text/javascript" src="http://code.jquery.com/jquery-1.9.0.js">
</script>
```

리스트 04 ▶ **basic.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>jQuery</title>
<script type="text/javascript"
  src="http://code.jquery.com/jquery-1.9.0.js">
</script>
<script type="text/javascript">
$(function( ) {
  $('#list.img.new').animate({
    height: '145px',
    width:'190px'
  }, 2000);
```

```
  });
</script>
</head>
<body>
<ul id-="list">
  <li><img src="images/01.jpg" class="new" /></li>
  <li><img src="images/02.jpg" /></li>
  <li><img src="images/03.jpg" /></li>
  <li><img src="images/04.jpg" class="new" /></li>
</ul>
</body>
</html>
```

모든 코드는 '$(function() {...});'로 묶는다

jQuery를 사용한 코드는 우선 '$(function() {...});'로 묶는 것이라 기억해두길 바란다. '$(function() {...});'란 페이지 내의 코드를 모두 읽어들인 후에 실행하라는 의미다.

이 규칙을 지키지 않으면 조작 내용에 따라 스크립트가 올바로 동작하지 않으므로 많은 주의가 필요하다. 예를 들어, 그림 04에서는 JavaScript를 실행하는 시점에서 아직 조작 대상의 〈ul〉 요소가 읽어들여지지 않았으므로, 코드가 기대한대로 동작하지 않는다.

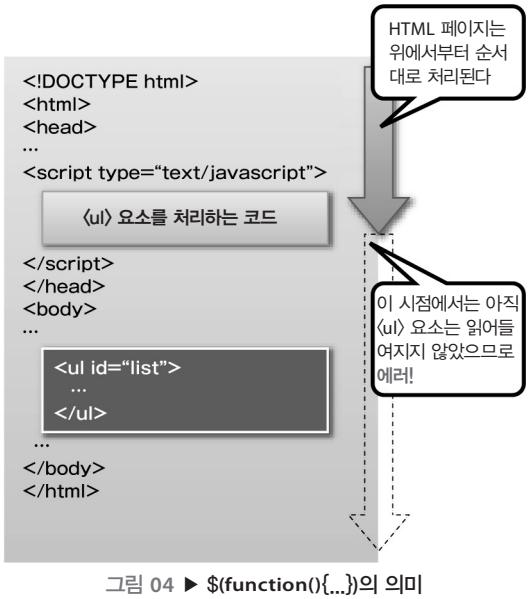


그림 04 ▶ $(function(){...})의 의미

jQuery의 기본은 '무엇'을 '어떻게' 하는가다

jQuery의 코드는 '무엇'을 '어떻게' 하는가가 기본이다(그림 05). 그리고 그중에서도 '무엇'을 나타내는 $() 함수는 jQuery에 있어서 코드의 시작점이기도 하여 jQuery 학습의 기본 열쇠라고도 말할 수 있다.

$() 함수의 인수-셀렉터(Selector)란 HTML 페이지로부터 특정의 요소(여러 요소)를 추출하기 위한 기법이다(표 03). 여기서는 '#list img .new'라는 셀렉터로 'id 속성이 list인 요소 밑의 〈img〉 요소에서 class 속성이 new인 것'을 추출하고 있다. '#list'가 'id="list"인 요소'를, 'img .new'는 'class 속성이 new인 〈img〉 요소'를, 양쪽을 공백으로 연결함으로써 '그 아래'를 나타내는 것이다.

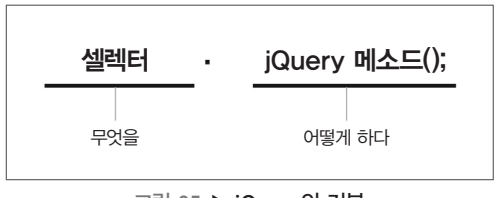


그림 05 ▶ jQuery의 기본

표 03 ▶ jQuery의 주요 셀렉터

셀렉터	개요	예
element	지정한 요소를 취득	$('a') //모든 〈a〉 요소
#id	지정한 id 값을 갖는 요소	$('#list') // id="list"인 요소
.class	지정된 class 속성을 갖는 요소	$('.menu') // class="menu"인 요소
ancestor descendant	ancestor에서 지정한 요소의 모든 자식 요소 descendant	$('#list li') // id="list"인 요소 아래의 모든 〈li〉 요소
parent 〉 child	Parent로 지정한 요소 아래의 자식 요소 child	$('#main 〉 a') // id="main"인 요소 바로 아래의 모든 〈a〉 요소

참고 jQuery Migrate

jQuery 1.9에서는 이젠 그다지 사용되지 않는 기능을 중심으로 정리하고 있다. 그 결과 jQuery 1.8 이전에 만들어진 라이브러리나 애플리케이션은 jQuery 1.9에서는 동작하지 않을 수도 있다. 이러한 경우에는 jQuery와 함께 jQuery Migrate를 임포트하기 바란다. 이것은 jQuery 1.8과의 호환성을 유지하기 위한 라이브러리로, 오래된 애플리케이션을 jQuery 1.9에 손쉽게 대응시킬 때 유용하다.

이러한 조건을 코드로 표현하려고 한다면 다음과 같은 처리가 필요할 것이다.

- 웹 페이지로부터 'id 속성이 "list"인 요소'를 검색하기
- 그 밑에 〈img〉 요소만을 추출하기
- 각각의 〈img〉 요소를 체크하여 class 속성이 "new"인 것을 추출하기

하지만 셀렉터를 이용함으로써 이러한 복잡한 처리를 겨우 13문자로 표현할 수 있게 되었다.

게다가 셀렉터 구문은 jQuery에서만 사용하는 특유의 기법이 아니라 원래는 CSS(Cascading Style Sheet)에서 이용되고 있던 셀렉터 구문을 jQuery에서 거의 그대로 이용할 수 있도록 한 것이다. 엄밀하게는 CSS에다가 약간의 독자 사양을 추가하고 있지만, 그렇다고는 해도 CSS를 이해하고 있는 사람이라면 직감적으로 다룰 수 있을 것이다.

jQuery의 실체는 jQuery 객체

$() 함수의 반환 값은 jQuery 객체다. jQuery 객체란, 셀렉터로 일치한 요소(여러 요소)를 보관함과 동시에, 이것들을 조작하기 위한 메소드를 제공하고 있는 객체를 말한다. 표준적인 요소 객체를 보다 사용하기 쉽게 만든 것으로 생각해도 좋다.★

참고 $() 함수의 그 외 사용법

$() 함수의 역할은 기존의 웹 페이지로부터 요소를 추출하는 것만이 아니다. HTML 문자열로부터 새로운 jQuery 객체를 생성하거나, 표준적인 요소 객체를 jQuery 객체로 변환하는 용도로도 사용할 수 있다(리스트 A).

특히, 두 번째의 기법은 매우 중요하다. 스크립트를 작성하다 보면, 다른 라이브러리 등으로부터 표준적인 요소 객체를 취득하는 일도 자주 있다. 그러나 본문에서도 언급했듯이, 표준적인 요소 객체와 jQuery 객체는 별개의 것이다. 그래서 $() 함수로 요소 객체에 대해 jQuery 객체로서의 성질을 부여하곤 한다. 이 기술 방식은 나중에 등장할 이벤트 리스너에서도 자주 이용하고 있으므로 제대로 이해해두어야 한다.

★ 단, 요소 객체라는 것은 별개다. 예를 들어, jQuery 객체 경유로 innerHTML 프로퍼티나 appendChild 메소드 등을 호출하는 것은 불가능하므로 주의하기 바란다.

리스트 A ▶ $() 함수의 그 외의 사용법

```
// 신규로 작성한 링크를 <body> 요소 아래에 추가
$('<a href="index.html">トップ</a>').appendTo('body');
// 웹 페이지의 텍스트를 검정색(#000)으로 설정
$(document.body).css('color', '#000');
```

반복하지만 jQuery에서는 $() 함수에서 조작 대상을 추출(=무엇을)하여 취득한 jQuery 객체로 조작(=어떻게 한다)하는 것이 기본이다. 예를 들어, 예제에서는 $()에서 추출한 요소에 대해 애니메이션 효과를 적용하는 animate 메소드를 실행하고 있다. animate(prop, sec) 메소드는 요소가 sec 밀리초 후에 파라미터 prop의 상태로 되도록 애니메이션을 실행한다(이 예에서라면 높이가 145px, 폭이 190px이 되도록 서서히 리사이즈한다).

파라미터를 설정하는 데 이용하고 있는 다음의 내용은 JavaScript의 해쉬(연상 배열)를 나타내는 기법이다.

```
{ 파라미터명: 값,... }
```

이후에도 라이브러리의 파라미터를 설정하는 데에 자주 이용하므로 여기서 제대로 기억해 두자.

animate 메소드 이외에도 jQuery에서는 많은 메소드를 제공하고 있다(표 06). 이것들은 jQuery 기능의 일부분에 지나지 않지만, 웹 애플리케이션 개발에서 자주 사용하는 기능이 폭넓게 지원되고 있음을 알 수 있을 것이다.

표 06 ▶ **jQuery 객체의 주요 메소드**

	메소드	개요
속성 & 스타일	attr(name[,value])	속성 name의 값 value를 설정. value 생략 시는 속성 name의 값을 취득
	attr(props)	[속성명: 값]의 해시 형식으로 여러 속성을 같이 정리하여 설정
	removeAttr(name)	속성 name을 삭제
	css(name[,value])	요소에 스타일(이름은 name, 값은 value)를 설정. value 생략 시는 스타일 name의 값을 취득
	css(props)	[스타일명: 값]의 해시 형식으로 여러 속성을 함께 정리하여 설정
	addClass(clazz)	class 속성으로서 clazz를 추가
	removeClass(clazz)	class 속성으로서 clazz를 삭제
	toggleClass(clazz)	class 속성에 clazz가 없는 경우는 추가, 있는 경우는 삭제
요소	append(c)	요소 c를 현재 요소의 자식 요소 끝에 추가
	prepend(c)	요소 c를 현재 요소의 자식 요소 앞에 추가
	after(c)	요소 c를 현재 요소의 뒤에 추가
	before(c)	요소 c를 현재 요소의 앞에 추가
	empty()	현재 요소 아래에 있는 모든 요소를 삭제
	html(val)	요소 아래에 HTML 문자열을 설정
	text(val)	요소 아래에 평문 텍스트를 설정(HTML 예약 문자는 이스케이프 처리)
이펙트(효과)	show(speed)	비표시의 요소를 speed 밀리초 후에 표시
	hide(speed)	표시된 요소를 speed 밀리초 후에 비표시
	slideDown(speed)	요소를 speed 밀리초 후에 슬라이드 다운
	slideUp(speed)	요소를 speed 밀리초 후에 슬라이드 업
	fadeIn(speed)	요소를 speed 밀리초 후에 페이드 인
	fadeOut(speed)	요소를 speed 밀리초 후에 페이드 아웃

jQuery에서는 요소의 수를 의식하지 않는다

jQuery 객체는 그 기능의 풍부함뿐만 아니라, $() 함수에서 추출한 요소의 수를 의식하지 않고 동일하게 메소드를 호출할 수 있다는 특징도 있다.

예를 들어, 리스트 04에서는 $('#list img.new')로 두 개의 〈img〉 요소를 추출하고 있는데, 만일 그냥 JavaScript라면 for 명령 등을 사용하여 하나씩 요소를 추출하여 개별 처리할 필요가 있다. 그러나 jQuery는 이러한 작업이 필요치 않다.

jQuery 객체는 자신이 관리하고 있는 요소(현재 요소) 모두에 대해서 자동적으로 후속 처리를 적용해준다. 이에 따라 jQuery의 코드에서는 중복이 발생하기 쉬운 반복 처리를 별도로 작성할 필요가 거의 없다.

여러 메소드를 연이어 호출하기 – 메소드 체인

jQuery 객체에는 또 하나의 중요한 성질이 있는데, 그것은 바로 거의 대부분의 메소드 반환 값이 jQuery 객체 자신이라는 점이다. 이 성질을 이용함으로써, jQuery에서는 복수의 메소드를 도트 연산자로 계속해서 호출할 수 있다(그림 06). 이러한 기법을 메소드 체인이라고 부른다.

예를 들어, 리스트 05는 리스트 04를 (단순히 이미지를 리사이즈하는 것만이 아니라) '페이드 아웃 → 페이드 인한 후에 리사이즈'하는 애니메이션으로 바꿔 쓴 코드다. 만일 메소드 체인의 성질을 이용하지 않았다면 리스트 06과 같이 코드를 작성할 필요가 있다. 하나의 객체에 대해서 복수의 조작을 시행할 경우 메소드 체인은 매우 값진 기능 중에 하나다.

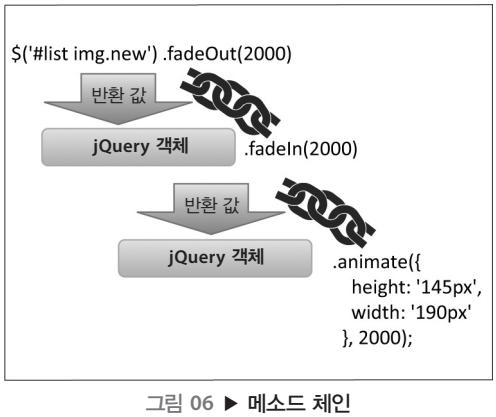


그림 06 ▶ 메소드 체인

리스트 05 ▶ **basic2.html**

```
$(function( ) {
  $('#list.img.new')
  .fadeOut(2000)
  .fadeIn(2000)
  .animate({
    height: '145px',
    width:'190px'
  }, 2000);
});
```

리스트 06 ▶ **메소드 체인을 이용하지 않은 경우**

```
$(function( ) {
  $('#list.img.new').fadeOut(2000);
  $('#list.img.new').fadeIn(2000)
  $('#list.img.new').animate({
    height: '145px',
    width:'190px'
  }, 2000);
});
```

jQuery에서의 이벤트 처리

jQuery를 이용할 때 반드시 파악해두어야 할 또 다른 한 가지가, 바로 이벤트 처리의 기술 방식이다. jQuery에서는 정교한 이벤트 처리 구문이 준비되어 있어 크로스 브라우저에 대응한 이벤트 리스너를 직감적인 코드로 기술할 수 있다.

예를 들어, 리스트 07은 리스트 05를 수정하여 '이미지를 클릭한 타이밍에 해당 이미지를 리사이즈하도록' 한 것이다. 이벤트 처리는 다음과 같이 표현할 수 있다.

```
$(...).이벤트명(function(){
  ...이벤트 처리...
});
```

리스트 07 ▶ **basic3.html**

```
$(function( ) {
  $('img').click(function(){
    $(this) .animate({
      height: '145px',
      width:'190px'
    }, 2000);
  });
});
```

jQuery에서 이용할 수 있는 이벤트는 표 07과 같다. 'function() {...}'은 JavaScript에서 한 덩어리의 처리로 표현하기 위한 기법으로 익명 함수라고도 불린다. 라이브러리를 이용하다 보면, 예를 들어 '라이브러리에서 어떠한 처리를 끝낸 후에 실행해야 할 처리'를 나타낼 때의 용도로 자주 사용하는 기술 방식이므로 빠른 시일 내에 익숙해지길 바란다.

이벤트 처리('function() {...}'의 안)에서는 $(this)로 이벤트를 발생시킨 객체에 액세스할 수 있다. 여기서의 경우라면 클릭된 이미지 객체가 이에 해당한다. 일반적인 JavaScript에 익숙해져 있다면 단순히 'this.animate~'와 같이 작성하게 되겠지만, this 그 자체로는 jQuery의 메소드에 액세스할 수 없으므로 주의가 필요하다.

표 07 ▶ **jQuery에서 이용할 수 있는 주요 이벤트**

카테고리	이벤트명	발생 타이밍
마우스/키	focusin	마우스나 탭 키 등으로 요소를 포커스했을 때
	focusout	마우스나 탭 키 등으로 요소의 포커스로부터 벗어났을 때
마우스	click	요소의 클릭 시
	dbclick	요소의 더블 클릭 시
	mousemove	요소 내에서 마우스 포인터가 이동했을 때
	mouseout	요소로부터 마우스 포인터가 벗어났을 때
	mouseover	요소에 마우스 포인터가 위치했을 때
	mouseup	마우스 버튼에서 손을 떼었을 때

키	keydown	키를 눌렀을 때
	keypress	키를 누르고 있을 때
	keyup	키를 눌렀다 떼었을 때
폼	blur	요소로부터 포커스를 벗어났을 때
	change	요소의 값을 변경했을 때(input, select, textarea 등)
	focus	요소에 포커스를 이동했을 때
	select	텍스트 박스나 텍스트 에리어의 텍스트를 선택했을 때
	submit	폼을 송신했을 때
그 외	load	페이지, 이미지, 폼 등의 리소스를 읽어들였을 때
	resize	윈도우 사이즈를 변경했을 때
	scroll	페이지나 요소를 스크롤했을 때
	unload	페이지가 언로드되었을 때
	error	이미지 로드에 실패했을 때

참고 그 외의 이벤트를 이용하는 방법

물론, 표 07 이외의 이벤트를 jQuery에서 취급하는 것도 가능하다. 그러한 경우는 on 메소드를 이용한다. 예를 들어, 리스트 B는 contextmenu 이벤트★를 이용하여 콘텍스트 메뉴의 표시를 억제하는 예다. on 메소드의 인수에는 '이벤트명', '이벤트 리스너'의 순서로 지정하여 이벤트와 대응한 처리를 연관지을 수 있다.

리스트 B ▶ 콘텍스트 메뉴를 무효화

```
$(document).on('contextmenu',function(e) {
return false; // 이벤트를 취소
});
```

★ 오른쪽 마우스 버튼을 클릭해서 컨텍스트 메뉴가 표시될 때에 발생하는 이벤트다.

PART

UI

파트 2에서는 UI(유저 인터페이스)편으로 화면을 화려하고 멋지게 꾸며,
유저의 조작성을 향상시키는 편리한 라이브러리를 소개한다.

1 <ul> 리스트를 고급스런 메뉴로 꾸며주기

◆ 야마다 요시히로

Apycom Menu는 〈ul〉 요소에서 정의된 글머리 기호 리스트를 짧은 스크립트 코딩을 가지고 고급스런 메뉴로 예쁘게 꾸며주는 라이브러리다. jQuery를 기반으로 동작하는 라이브러리이지만, jQuery의 코드조차 기술할 필요가 없이 미리 정해진 룰에 따라 리스트를 마크업 언어로 작성해두면 그것만으로도 메뉴를 생성할 수 있다는 것이 커다란 장점이다.

명칭	Apycom Menu
분류	UI
URL	http://apycom.com/
관련 파일	jquery-1.9.0.js, menu.js, menu.css, /images 폴더

Apycom Menu로 생성한 메뉴

리스트 001-01은 Apycom Menu를 사용하여 작성한 메뉴다. Apycom Menu를 이용하려면 메뉴의 기반이 되는 리스트를 아래의 룰에 따라 마크업해두어야 한다.

❶ 메뉴는 〈ui class="menu"〉~〈/ul〉 요소로 정의
❷ 서브 메뉴를 갖게 하려면 〈li〉 요소의 아래에 〈ul〉 요소를 중첩해서 추가
❸ 서브 메뉴의 부모가 되는 메뉴의 〈a〉 요소에는 'class="parent"' 속성을 지정
❹ 복수 열의 메뉴를 생성하고 싶은 경우에는 〈ul〉 요소를 병렬로 나열
❺ 메뉴는 전체를 〈div id="menu"〉~〈/div〉 요소로 둘러싼다.

각각의 번호는 리스트 001-01의 해당 부분에 대응하고 있으므로, 전반적으로 확인해두기 바란다.

참고로, Apycom Menu는 표준으로 100가지가 넘는 풍부한 스타일을 제공하고 있어 사이트 디자인에 맞춰 원하는 것을 선택할 수 있다(그림 001-01). 나중에 디자인을 바꿀 경우에도 다운로드해온 패키지 안에서 /images 폴더와 menu.css를 교체하는 것만으로도 작업이 끝나므로 매우 간단하게 대응할 수 있다.

리스트 001-01 ▶ **apycomMenu.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Apycom Menu</title>
<link rel="stylesheet" type="text/css" href="css/menu.css" />
<script type="text/javascript"
  src="http://code.jquery.com/jquery-1.9.0.js"></script>
<script type="text/javascript" src="js/menu.js"></script>
</head>
<body>
<div id="menu">                                                    ❺
  <ul class="menu">                                                ❶
    <li><a href="top.html"><span>메인 페이지</span></a></li>
    <li><a href="foods.html" class="parent"><span>식품</span></a>
      <div><ul>
        <li><a href="vegetables.html" class="parent"><span>야채</span></a>
          <div><ul>
            <li><a href="roots.html" class="parent"><span>근채</span></a>
              <div><ul>
                <li><a href="burdock.html"><span>우엉</span></a></li>
                <li><a href="radish.html"><span>무우</span></a></li>
                <li><a href="carrot.html"><span>당근</span></a></li>
              </ul></div>
            </li>
```

```
    <li><a href="greens.html" class="parent"><span>엽채류</span></a>
      <div><ul>
        <li><a href="cabbage.html"><span>양배추</span></a></li>
        <li><a href="lettuce.html"><span>상추</span></a></li>
        <li><a href="napa.html"><span>배추</span></a></li>
      </ul></div>
    </li>
    <li><a href="potato.html" class="parent"><span>감자류</span></a>
      <div><ul>
        <li><a href="potatoes.html"><span>감자</span></a></li>
        <li><a href="sweet_potato"><span>고구마</span></a></li>
        <li><a href="taro.html"><span>토란</span></a></li>
        <li><a href="yam.html"><span>참마</span></a></li>
      </ul></div>
    </li>
    <li><a href="general.html" class="parent"><span>일반야채</span></a>
      <div><ul>
        <li><a href="cucumber.html"><span>오이</span></a></li>
        <li><a href="tomato.html"><span>토마토</span></a>
        <li><a href="eggplant.html"><span>가지</span></a></li>
        <li><a href="onion.html"><span>양파</span></a></li>
        <li><a href="pimiento.html"><span>피망</span></a></li>
      </ul></div>
    </li>
    <li><a href="watermelon.html"><span>수박</span></a>
    </li>
  </ul></div>
</li>
<li><a href="dairy_goods.html" class="parent"><span>유제품</span></a>
  <div><ul>
    <li><a href="milk.html"><span>우유</span></a></li>
    <li><a href="butter.html"><span>버터</span></a></li>
    <li><a href="yogurt.html"><span>요구르트</span></a></li>
  </ul></div>
</li>
<li><a href="fish.html" class="parent"><span>생선</span></a>
  <div><ul>
    <li><a href="whole.html" class="parent"><span>생물 생선</span></a>
      <div><ul>
        <li><a href="mackerel.html"><span>고등어</span></a></li>
        <li><a href="sardine.html"><span>멸치</span></a></li>
        <li><a href="saury.html"><span>꽁치</span></a></li>
      </ul></div>
    </li>
    <li><a href="dried_fish" class="parent"><span>말린 생선</span></a>
      <div><ul>
        <li><a href="horse_mackerel.html"><span>전갱이 말림</span></a></li>
        <li><a href="atka_mackerel .html"><span>조기 말림</span></a></li>
      </ul></div>
    </li>
    <li><a href="sashimi.html" class="parent"><span>생선회</span></a>
      <div><ul>
        <li><a href="sea_bream.html"><span>도미</span></a></li>
        <li><a href="tuna.html"><span>참치</span></a></li>
        <li><a href="salmon.html"><span>연어</span></a></li>
      </ul></div>
```

❶ ❺

```
          </li>
        </ul></div>
      </li>
    </ul></div>
  </li>
  <li><a href="#" class="parent"><span>생활잡화</span></a>
    <div class="columns two">
      <ul class="one">
        <li><a href="cleanser.html"><span>세제</span></a></li>
        <li><a href="softener.html"><span>유연제</span></a></li>
        <li><a href="bleach.html"><span>표백제</span></a></li>
      </ul>
      <ul class="two">
        <li><a href="toilet_paper.html"><span>화장실 휴지</span></a></li>
        <li><a href="box_tissue.html"><span>각 티슈</span></a></li>
        <li><a href="pocket_tissue.html"><span>휴대용 티슈</span></a></li>
        <li><a href="diaper.html"><span>기저귀</span></a></li>
      </ul>
    </div>
  </li>
  <li><a href="#" class="parent"><span>문의</span></a>
    <div class="contact">
      <ul>
        <li><a href="tel.html"><span>전화</span></a></li>
        <li><a href="mail.html"><span>메일</span></a></li>
      </ul>
    </div>
  </li>
  <li class="last"><a href="access.html"><span>오시는 길</span></a></li>
 </ul>
</div>
<a href="http://apycom.com/">Apycom jQuery Menus</a>
</body>
</html>
```

❶ ❷ ❸ ❹ ❺

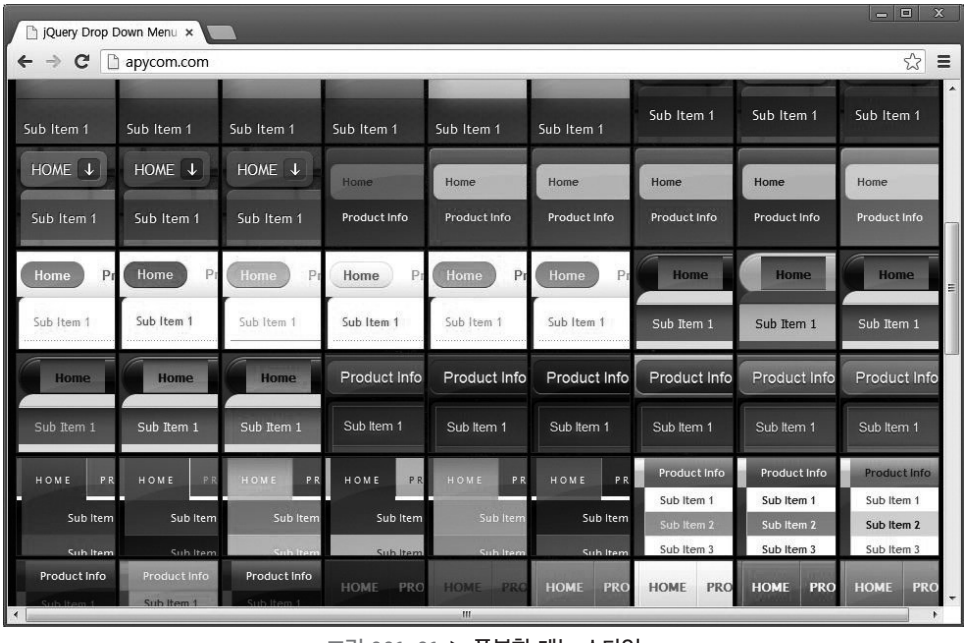


그림 001-01 ▶ 풍부한 메뉴 스타일

Apycom Menu는 원칙적으로 무상 이용이 가능하지만, 그 대신 이용한 사이트에서는 웹 페이지의 적당한 곳에 공식 사이트로의 링크를 넣어 둘 필요가 있다.

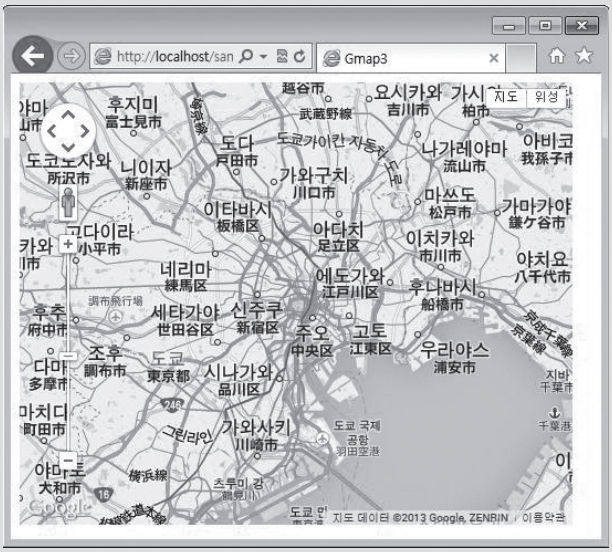
```
<a href="http://apycom.com/">jQuery Menu by Apycom</a>
```

2 구글 맵과 연계한 페이지 손쉽게 작성하기

◆ 야마다 요시히로

Gmap3는 jQuery 플러그인의 일종으로, 구글 맵(Google Map)의 기능을 간단히 호출하기 위한 기능을 제공하고 있다. 구글 맵의 표준 API인 Google Maps API의 경량 래퍼와 같은 라이브러리다. 물론, Google Maps API를 직접 이용해도 상관없으나 Gmap3를 이용함으로써, jQuery의 코드로부터 보다 심플한 코드로 구글 맵을 조작할 수 있다.

명칭	Gmap3
분류	UI
URL	http://gmap3.net/
관련 파일	jquery-1.9.0.js, gmap3.min.js



Gmap3로 생성한 지도

예를 들어, 리스트 002-01은 Gmap3로 구글 맵을 표시한 예다. Gmap3를 이용하려면 ❶과 같이 구글 맵을 표시하기 위한 영역을 준비해둔다. Style 속성에 height/width 프로퍼티를 지정함으로써, 이 값이 그대로 지도의 높이/폭이 된다.

리스트 002-01 ▶ **gmap3_basic.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Gmap3</title>
<script type="text/javascript"
  src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script type="text/javascript"
  src="http://code.jquery.com/jquery-1.9.0.js"></script>
<script type="text/javascript" src="js/gmap3.min.js"></script>
<script type="text/javascript">
$(function(){
  // Geolocation API를 이용할 수 있는지를 체크
  if (navigator.geolocation) {]
    // Geolocation API로 현재 위치를 취득
    navigator.geolocation.getCurrentPosition(
      function (pos) {
        // 현재 위치를 취득
        var current = [ pos.coords.latitude, pos.coords.longitude ];
        // 구글 맵을 표시
        $('#map').gmap3({
          map:{
            options: {
              center: current, // 지도의 중심점(현재 위치를 세트)
              zoom: 10 // 줌 비율
            }
          }
        });
      }
    );
  } else {
    alert('브라우저가 Geolocation API를 서포트하고 있지 않습니다.');
  }
});
</script>
</head>
<body>
<!-- 구글 맵을 표시하기 위한 영역을 준비 -->
<div id="map" style="width:500px;height:400px;"></div> ❶
</body>
</html>
```

그런 다음, 이 영역에 대하여 gmap3 메소드를 호출하기만 하면 된다. 구문은 다음과 같다.

```
$(맵을 나타내는 셀렉터).gmap3(
  {파라미터명: 값,... });
```

지정할 수 있는 파라미터에는 표 002-01과 같은 것이 있다. 이러한 파라미터에 대해, 중첩하여 Google Maps API에 건네야 하는 파라미터를 지정함으로써 지도의 표시를 설정하거나, 마커나 도형을 추가해 나가는 것이 Gmap3의 기본이다. 우선, 최소한의 지도를 표시하려면 map 파라미터를 설정한다.

표 002-01 ▶ **Gmap3 플러그인의 주요 파라미터**

명칭파라미터	개요
map	구글 맵의 기본 설정
circle	원 그리기
rectangle	사각형 그리기
polyline	선 그리기
polygon	다각형 그리기
marker	마커 작성

map 파라미터에는 중첩해서 Google.maps.MapOption 객체★의 프로퍼티를 지정한다. 샘플에서는 중심 좌표(위도, 경도)와 줌 비율(0~21. 큰 값일수록 확대)을 최소한으로 지정하고 있다. 중심 좌표의 위도, 경도에는 HTML5의 Geolocation API★★로부터 취득한 현재 위치의 좌표를 설정한다.

지도에 선분 그리기

계속해서 지도에 선분을 그린다(리스트 002-02. 그림 002-01). 표 002-01에서도 본 것처럼 Gmap3에서는 도형 그리기를 위한 circle, polyline, polygon, renctangle의 파라미터를 준비해놓고 있다. 선 그리기를 하고자 한다면 polyline을 이용하자.

★ Google Maps API의 객체다. 상세는 'Google Maps Javascript API V3 Reference'(https://developers.google.com/maps/documentation/javascript/reference)를 참조하기 바란다.

★★ **역주** 현재 위치의 좌표를 취득하기 위한 API다.

polyline 파라미터에는 중첩해서 google.maps.PolylineOptions 객체의 프로퍼티를 지정한다. 샘플에서 이용하고 있는 프로퍼티에 대해서는 코드 안의 주석도 같이 참조하기 바란다. Path 파라미터에는 [[위도, 경도], ...]의 배열 형식으로 필요한 수만큼 선의 좌표를 지정한다.

리스트 002-02 ▶ **gmap3_polyline.html(변경 부분만)**

```
$('#map').gmap3({
  map:{

    ...중략...

  },
  // 지도 상에 꺾은 선을 표시
  polyline: {
    options: {
      strokeColor: '#f00', // 색
      strokeOpacity: 1.0, // 투명도
      strokeWeight: 3, // 선의 굵기
      path: [
        current,
        [35.71007, 139.80948],
        [35.634629, 139.881408]
      ] // 꺾은 선의 좌표
    }
  }
});
```

그림 002-01 ▶ **지도의 지정 좌표에 선분 그리기**

지도에 마커 추가하기

계속해서 지도에 마커를 배치한다. 또한, 마커에 마우스 포인터를 갖다 대면 팝업 윈도우를 표시하도록 해보자(리스트 002-03, 그림 002-02).

마커의 설정에는 marker 파라미터를 이용한다. marker 파라미터에는 중첩해서 google.maps.MarkerOptions 객체의 프로퍼티를 지정한다. 예제에서 이용하고 있는 프로퍼티에 대해서는 코드 안의 주석도 함께 참조하기 바란다.

리스트 002-03 ▶ **gmap3_marker.html(변경 부분만)**

```
$('#map').gmap3({
  map:{

    ...중략...

  },
  marker: {
    // 마커의 표시 위치
    values: [                                                          ❶
      { latLng: current, data: '현재 위치' },
      { latLng: [35.71007, 139.80948], data: '스카이트리' },
      { latLng: [35.634629, 139.881408], data: '디즈니랜드' }
    ],
    events:{
      // 마커에 마우스를 갖다 대었을 때의 동작
      mouseover: function(marker, event, context){*                    ❷
        // 현재의 지도, 윈도우를 취득
        var map = $(this).gmap3('get');
        var win = $(this).gmap3({ get: {name: 'infowindow' }});
        // 윈도우가 존재하는 경우에는 그대로 오픈
        if (win){
          win.open(map, marker);
          win.setContent(context.data);
        } else {
          //윈도우가 존재하지 않는 경우에는 신규로 작성
          $(this).gmap3({                                              ❸
            infowindow: {
              anchor: marker,  // 기본 베이스가 되는 마커
              options: { content: context.data }  // 윈도우 옵션
            }
          });
        }
      }
    }
  },
  polyline: {

    ...중략...

  }
});
```

그림 002-02 ▶ 지도에 마커 추가

마커의 표시 위치는 values 파라미터로 다음과 같은 객체 배열의 형식으로 지정한다(❶).

```
[{latLng: [위도, 경도],
  data: '현재 위치' },... ]
```

또한, 마커에 마우스 포인터가 위치했을 경우의 동작을 events-mouseover 파라미터로 지정한다(❷). 이벤트 리스너를 지정하는 작성법은 [이벤트명: 리스너]다.

리스너 안에서는 현재의 마커에 관련된 윈도우가 존재하는지를 체크하여 존재할 경우에는 그 상태에서 윈도우를 열고, 존재하지 않는 경우에는 윈도우를 새로 열어 작성한다. 윈도우를 작성하려면 다음과 같이 기술한다(❸).

```
$(this).gmap3({ infowindow:
  { 파라미터명: 값,... } });
```

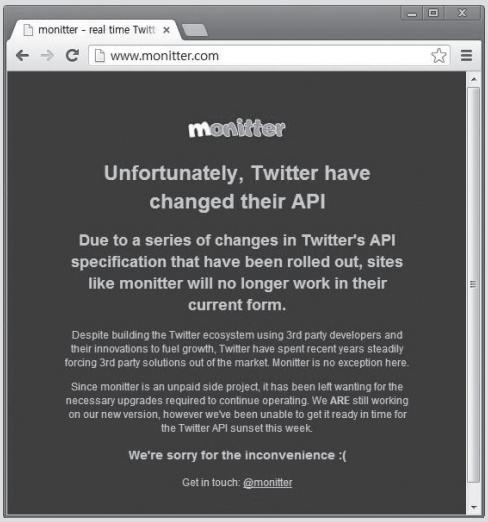
anchor 파라미터는 윈도우와 연관시킬 마커를, options 파라미터에는 윈도우 옵션을 각각 설정한다. 여기에서는 content 파라미터(윈도우에 표시하는 텍스트)에 대하여 마커에 연관시킨 데이터(context.data)를 대입하고 있다.

3 트위터에서의 트윗을 키워드 검색하기

◆ 야마다 요시히로

monitter widgets은 트위터(Twitter)에서의 트윗을 키워드로 검색하여 그 결과를 리스트 표시한다. jQuery 플러그인의 한 종류로, 정형적인 처리라면 플러그인이 내부적으로 조달해주기 때문에 코드를 일절 작성하지 않고도 가능하다는 점이 장점이다.

명칭	monitter widgets
분류	UI
URL	http://www.monitter.com/widget/
관련 파일	jquery-1.9.0.js, monitter.min.js, monitter.css



키워드 'JavaScript'로 트윗 검색

※출판사 안내 – 현재 트위터 API가 변경되어 해당 플러그인이 동작하지 않습니다. 참고하시기 바랍니다.

★ 역주 현재 이 라이브러리는 트위터 API 변경에 따라 서비스가 되지 않고 있다. 현재 해당 사이트에서는 다음과 같은 메시지가 표시된다.

리스트 003-01은 JavaScript에 관한 트윗을 리스트로 표시하는 예다. monitter widgets을 이용하려면 다음과 같은 룰에 따라 태그를 준비해둘 필요가 있다.

- 리스트를 표시하기 위한 영역에는 'class="monitter"'를 지정
- 동일한 title 속성에는 검색 문자열을 지정
- 검색 대상의 언어를 필터링하려면 lang 속성을 지정(일본어라면, ja)

리스트 003-01 ▶ **monitter.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Monitter Widgets</title>
<link rel="stylesheet" type="text/css" href="css/monitter.css" />

<style type="text/css">
#tweets p { font-size: 12px; color: #efefef; }
.monitter { width: 250px; height: 400px; }
</style>

<script type="text/javascript"
  src="http://code.jquery.com/jquery-1.9.0.js"></script>
<script type="text/javascript" src="js/monitter.min.js"></script>

</head>
<body>

<!--검색 결과를 표시하기 위한 영역을 준비 -->
<div id="tweets" class="monitter" title="JavaScript" lang="ko"></div>

</body>
</html>
```

title 속성에 지정할 수 있는 검색 문자열의 포맷은 'Using the Twitter Search API'(https://dev.twitter.com/docs/using-search)도 같이 참조하기 바란다. 예를 들어, "JavaScript jQuery"(공백으로 구분)이라면 'JavaScript, jQuery라는 문자열을 둘 다 포함하는 트윗'을 검색할 수 있으며, "JavaScript OR jQuery"(OR 연산자)라면 'JavaScript 또는 jQuery를 포함하는 트윗'을 의미한다.

"JavaScript – jQuery"(마이너스 연산자)라면 'JavaScript를 포함하나 jQuery는 포함하지 않는 트윗'의 검색도 가능하다. 또한 "to:yyamada", "from:yyamada"로 함으로써 '@yyamada

로의 트윗', '@yyamada로부터의 트윗'을 나타내는 것도 가능하다.

검색 문자열 동적으로 전환하기

입력 값으로부터 monitter widgets에서의 검색 내용을 동적으로 전환할 수도 있다. 리스트 003-02는 텍스트 박스에서 검색 문자열을 지정하는 예다(그림 003-01). 여기서는 추가 부분만을 나타내고 있다.

앞서 기술하였다시피 monitter widgets에서는 title 속성으로 검색 문자열을 지정할 수 있었다. 따라서 예제에서도 ❶과 같이 텍스트 박스의 값을 title 속성에 반영시킴으로써 동적으로 monitter widgets이 인식하여 대응하는 검색 결과를 표시해준다.

리스트 003-02 ▶ **monitter2.html**

```
<script type="text/javascript">
$(function(){
  // [검색] 버튼을 클릭하면 키워드를 title 속성에 반영한다
  $('#search').click(function() {
    var keywd = $('#keywd').val();
    $('#tweets').attr('title', keywd); ──────────────────────────────❶
  });
});
</script>

...중략...

<div>
  <input id="keywd" type="text" value="JavaScript" />
  <input id="search" type="button" value="검색" />
</div>
<div id="tweets" class="monitter" title="JavaScript" lang="ko"></div>
```

그림 003-01 ▶ 입력 값에 따라서 검색 결과를 변화

4 HTML 테이블로부터 막대 그래프나 꺾은 선 그래프 등 생성하기

◆ 야마다 요시히로

Visualize는 jQuery 그래픽의 일종으로, HTML 테이블에서 미리 준비된 데이터를 바탕으로 막대 그래프, 꺾은 선 그래프, 에리어 차트, 원 그래프 등의 차트 이미지를 생성한다. 차트를 변경하는 데에도 데이터 그 자체를 변경할 필요가 없이 Visualize 플러그인의 옵션만을 변경하면 되므로 매우 간편하다.

명칭	Visualize
분류	UI
URL	http://filamentgroup.com/lab/update_to_jquery_visualize_accessible_charts_with_html5_from_designing_with/
관련 파일	jquery-1.9.0.js, visualize.jQuery.js, visualize.css, visualize-light.css

테이블 데이터를 기반으로 여러 차트를 생성

리스트 004-01은 2012년 하기 경비를 각각 꺾은 선 그래프, 원 그래프, 에리어 차트, 막대 그래프로 나타낸 예다.

리스트 004-01 ▶ visualize.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Visualize</title>
<link rel="stylesheet" type="text/css"
 href="css/visualize.css" />
<link rel="stylesheet" type="text/css"
 href="css/visualize-light.css" />
<script type="text/javascript"
  src="http://code.jquery.com/jquery-1.9.0.js"></script>
<script type="text/javascript" src="js/visualize.jQuery.js"></script>
<script type="text/javascript">
$(function () {
  // 막대 그래프, 에리어 차트, 원 그래프, 꺾은 선 그래프를 작성
  $('#data').visualize({type: 'bar'});
  $('#data').visualize({type: 'area', width: '400px' });
  $('#data').visualize({type: 'pie', pieLabelPos: 'inside' });
  $('#data').visualize({type: 'line', lineWeight: 2});
});
</script>
</head>
<body>
<!-- 차트용 데이터를 테이블로 준비 -->
<table id="data">
  <caption>2012년 하기 경비</caption>
  <thead>
    <tr>
      <td></td><th>7月</th><th>8月</th><th>9月</th>
      <th>10月</th><th>11月</th><th>12月</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>수도 광열비</th><td>213</td><td>224</td><td>205</td>
      <td>187</td><td>190</td><td>202</td>
    </tr>
    <tr>
      <th>선전 광고비</th><td>211</td><td>175</td><td>110</td>
      <td>199</td><td>123</td><td>201</td>
    </tr>
    <tr>
      <th>사무용품비</th><td>111</td><td>101</td><td>99</td>
      <td>121</td><td>98</td><td>110</td>
    </tr>
    <tr>
      <th>신문 도서비</th><td>101</td><td>99</td><td>87</td>
      <td>96</td><td>100</td><td>91</td>
```

❶

❷

```
    </tr>
    <tr>
      <th>그 외 잡비</th><td>124</td><td>108</td><td>99</td>
      <td>101</td><td>112</td><td>99</td>
    </tr>
  </tbody>
</table>
</body>
</html>
```

Visualize 플러그인을 작성하려면 우선 ❶과 같이 원래의 데이터를 테이블로서 준비한다. 기본적으로는 각각의 행이 하나의 차트로서 그려진다. 단, 원 그래프만은 행의 합계가 하나의 데이터 항목으로 인식된다. 또는 범례와 데이터 행을 Visualize 플러그인이 인식할 수 있도록 선두 행은 〈thread〉 요소로, 이후의 행은 〈tbody〉 요소로 각각 묶어둔다. 또한, 차트 타이틀로 〈caption〉 요소에 테이블 캡션도 준비해둔다.

데이터 준비가 되면, 이후에는 이것에 visualize 메소드로 차트 기능을 적용하기만 하면 된다(❷). visualize 메소드의 구문은 다음과 같다.

```
$(테이블을 나타내는 셀렉터).visualize(
  { 파라미터명: 값,... })
```

주요 파라미터는 표 004-01에 정리해두었다.

표 004-01 ▶ Visualize 플러그인의 주요 파라미터

파라미터	개요	디폴트 값
type	차트의 종류(bar | pie | line | area)	bar
height	차트의 높이	(테이블의 높이)
width	차트의 폭	(테이블의 폭)
parseDirection	테이블의 데이터를 분석하는 방향(x | y)	x
colors	그리기 색	['#be1e2d','#666699','#92d5ea','#ee8310', '#8d10ee','#5a3b16','#26a4ed','#f45a90','#e9e744']
textColors	텍스트 색	[]
appendTitle	차트에 타이틀을 부여할 것인가?	true
title	차트의 타이틀	(테이블의 캡션)

pieLablePos	원 그래프에서의 레이블 표시 위치 (inside | outside)	inside
lineWeight	꺾은 선 그래프, 에리어 차트의 선 굵기	4
pieMargin	원 그래프의 마진	20
barGroupMargin	막대 그래프(그룹 단위)의 마진	10
barMargin	막대 그래프의 마진	1

차트의 종류는 대부분 정해져 있지만, 자주 사용하는 차트를 직감적으로 알기 쉬운 파라미터 설정만으로 손쉽게 생성할 수 있다는 점이 Visualize 플러그인의 장점이다.

참고로, 예제에서는 스타일 시트로 visualize-light.css(밝은 색 디자인)를 이용하고 있지만 또 하나의 visualize-dark.css를 이용함으로써 어두운 분위기의 차트로 변환하는 것도 가능하다(그림 004-01).

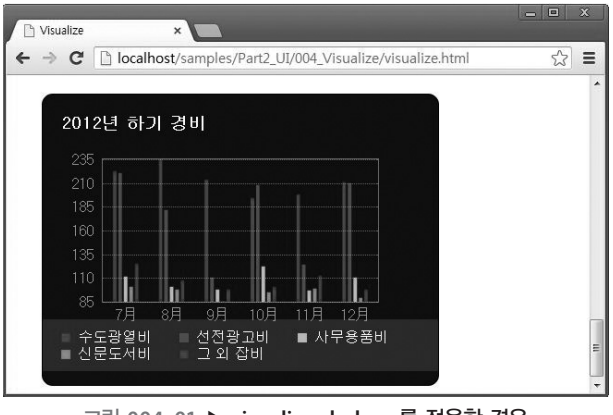


그림 004-01 ▶ **visualize-dark.css를 적용한 경우**