jQuery

1. jQuery 시작하기

1.1 jQuery 개요

jQuery는 브라우저 호환성(크로스 브라우징, Closs Browsing)을 지원하는 JavaScript 라이브러리로 전 세계에서 가장 많이 사용되고 있다. 또한 MIT 라이선스가 적용된 오픈소스 라이브러리로 저작권과 라이선스를 표시하면 누구나 배포와 변경도 가능하고 상용으로 이용하거나 수정한 라이브러리를 유료로 판매하는 것도 가능하다.
jQuery는 2006년 존 레식(John Resig)에 의해 처음 소개되었으며 현재는 jQuery Team 이라는 개발자 그룹이 개발과 유지보수를 담당하고 있다.

1.2 jQuery 장점

jQuery는 JavaScript를 이용한 개발을 쉽고 간편하게 구현하기 위해 고안된 라이브러리로 jQuery를 사용하면 클라이언트 사이드에서 보다 쉽게 문서 객체 모델(DOM, Document Object Model)을 탐색하고 조작할 수 있으며 애니메이션과 같은 다양한 효과를 간편하게 웹 페이지에서 연출할 수 있다.

▶ DOM 탐색과 변경이 쉽고 간편하다.

JavaScript에 비해 DOM 탐색과 변경이 매우 간단하며 무엇보다 CSS에서 사용하는 선택자를 그대로 사용해 간편하게 DOM에 접근할 수 있는 메소드를 지원한다.

▶ 크로스 브라우징을 지원 한다.

JavaScript에서는 브라우저 종류나 버전에 따라서 이벤트를 연결하는 방법이 달랐다. 하지만 jQuery는 일관된 이벤트 연결 방법을 제공하기 때문에 브라우저 종류나 버전에 관계없이 동일한 코드를 사용해 이벤트를 처리할 수 있다. 예전에 JavaScript Core에서 DOM 객체에 접근해 작업할 때 공백 문자 인식이나 브라우저 종류와 버전에 따라 발생하는 문제에 대해 일일이 대응하는 코드를 작성해야 했으나 jQuery를 이용하면 이런 문제에 대해 신경 쓰지 않아도 된다. 또한 JavaScript Core에서 ajax를 구현할 때도 위와 마찬가지로 브라우저 종류나 버전에 따라서 작성해야 하는 코드가 많았지만 jQuery는 그 내부에서 브라우저 호환성 문제를 해결했기 때문에 jQuery가 제공하는 ajax 지원 메소드를 사용해 보다 쉽고 간편하게 ajax를 구현할 수 있다.

▶ 애니메이션과 대화형 웹페이지를 쉽게 구현할 수 있다.

JavaScript를 사용해 애니메이션을 구현하려면 많은 양의 코드와 씨름해야 하고 더욱이 브라우저마다 다른 결과를 동일하게 맞추기 위해 크로스 브라우징 처리를 필히 해야 하지만 jQuery는 애니메이션을 쉽게 구현할 수 있도록 페이드인(Faid-in)이나 이징(easing) 등의 애니메이션을 위한 효과 메소드를 제공하고 있기 때문에 몇 줄의 코드만으로 애니메이션 효과를 연출할 수 있다. 물론 크로스 브라우징 처리도 jQuery가 내부적으로 처리해 준다. 또한 사용자 반응에 따라서 웹 페이지의 내

용을 바꾸는 대화형 처리도 jQuery를 이용하면 쉽고 간편하게 구현할 수 있다.

▶ 다양하고 풍부한 플러그인이 공개되어 있다.

jQuery 라이브러리의 기본 기능에는 포함되어 있지 않은 슬라이드, 메뉴, 애니메이션과 같은 다양한 기능을 제공하는 여러 가지 플러그인이 공개되어 있어 이들 플러그인을 사용해 jQuery 기능을 확장할 수 있다.

▶ 모바일 기기를 지원한다.

jQuery Mobile을 사용하면 PC 기반 웹 페이지를 제작할 때와 거의 비슷한 환경으로 스마트 폰 또는 태블릿 PC와 같은 모바일 기기에 대응하는 웹 페이지를 쉽고 빠르게 제작할 수 있다.

1.3 jQuery 사용하기

jQuery는 JavaScript 언어로 제작된 라이브러리로 js 파일 형태로 배포되며 http://www.jquery.com/download/ 에서 다운로드 받을 수 있다.
이 라이브러리는 각 버전별로 아래와 같이 두 가지 버전으로 배포된다. 아래는 jQuery 버전 3.3.1 버전을 예를 들어 배포되는 파일을 기술하였다.

- **jquery-3.3.1.js(274KB, 압축되지 않은 jQuery 소스 파일)**
- **jquery-3.3.1.min.js(87KB, 압축된 jQuery 소스 파일)**

위에서 () 안의 내용과 같이 압축되지 않은 jQuery 소스 파일은 개발버전(Development Version)이라고 하며 실제 jQuery 소스를 열어보면 읽기 쉽게 들여쓰기도 되어 있고 주석문도 달려있어 소스를 분석하기에 적합하다. 하지만 파일 용량이 크다는 단점이 있다. 그래서 실무에서는 대부분 압축된 jQuery 소스 파일을 사용한다. jquery-3.3.1.min.js 파일과 같이 jQuery 소스 파일의 이름에 min이 포함된 파일은 jQuery 소스 파일을 압축한 버전으로 제품버전(Production Version)이라고 하며 소스에서 주석이나 공백 등을 삭제해 파일 용량을 최대한 줄인 것이다.
jQuery는 크게 1.x.x 버전과 2.x.x 그리고 3.x.x 버전으로 나눌 수 있는데 1.x.x 버전은 구형 브라우저(IE 6/7/8)를 지원하고 있지만 2.x.x와 3.x.x 버전은 이를 지원하지 않는다. 그리고 1.x.x 버전과 2.x.x 버전은 이제 더 이상 버전 업(패치) 하지 않는다.
jQuery를 사용하기 위해서는 웹 페이지에서 <script> 태그를 사용해 jQuery 라이브러리를 참조할 수 있어야 한다. 웹 페이지에서 jQuery 라이브러리를 참조하는 방식은 직접 다운로드 받아 참조하는 방식과 CDN(Content Delivery Notwork)을 이용한 참조 방식이 있다.
아래는 많이 사용되는 jQuery 라이브러리 CDN 주소이다.

- **jQuery CDN : http://code.jquery.com/jquery-1.12.4.min.js**
- **jQuery CDN : http://code.jquery.com/jquery-3.3.1.min.js**
- **Google CDN : https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js**

- Google CDN : https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js

▶ 예제 1-1 웹 페이지에서 jQuery 사용하기

- jQueryStudyCh01/jquery01_01.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>웹 페이지에서 jQuery 사용하기</title>
<!-- 외부 자바스크립트 파일 jQuery 참조 -->
<!-- CDN(Content Delivery Network)을 이용해 jQuery를 참조할 수 있다.  -->
<!-- <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script> -->

<!-- 운영 서버의 특정 위치에 다운로드 받아서 jQuery를 참조할 수 있다. -->
<script src="js/jquery-3.3.1.min.js" ></script>
<script>
  /* jQuery()는 전역 함수로 인수를 분석해 jQuery 객체를 생성하는 역할을 한다.
   * 인수로 지정할 수 있는 데이터는 JavaScript에서 사용하는 객체나 함수,
   * DOM을 탐색하고 조작하기 위해 CSS에서 사용하는 선택자 등을 지정할 수 있다.
   * jQuery 소스 파일을 살펴보면 아래와 같이 전역함수로 선언되어 있고 window
   * 객체의 jQuery와 $ 속성으로 jQuery 함수가 등록되어 있는 것을 볼 수 있다.
   *
   * jQuery = function(selector, context ) { ... }
   * window.jQuery = window.$ = jQuery;
   **/
  jQuery(document).ready(function() {

    /* html() 메소드는 지정한 요소 안에 태그를 포함한 내용을 설정하거나 반환한다.
     * HTML 태그가 적용되어 화면에 표시되므로 아래는 div 요소의 하부에 h2 요소로
     * 추가된다.
     **/
    jQuery("div").html("<h2>안녕 jQuery - html()</h2>");
  });

  /* jQuery도 자바스크립트이므로 변수나 함수의 이름으로 $를 사용할 수 있다.
   * jQuery 소스에서 $는 jQuery의 또 다른 식별자 즉 별칭으로 선언되어 있으므로
   * 위에서 사용했던 jQuery() 함수를 아래와 같이 $() 함수로 함축하여 사용할 수 있다.
   **/
  $(document).ready(function() {

    /* text() 메소드는 지정한 요소 안에 텍스트를 설정하거나 반환한다.
     * HTML 태그가 적용되지 않으므로 아래는 h2 요소가 일반 텍스트 처럼 표시된다.
     **/
```

```
    $("#jquery2").text("<h2>Hi jQuery - text()</h2>");
  });

  // 위에서 사용한 $(document).read()를 아래와 같이 함축하여 사용할 수 있다.
  $(function() {

    //jQuery의 메소드는 jQuery 객체를 반환하므로 아래와 같이 메소드 체이닝이 가능하다.
    $("#jquery3").css("width", "500px").css("height", "250px")
      .css("border", "1px dotted blue")
      .text("안녕 제이쿼리");
  });
</script>
</head>
<body>
  <div id="jquery1"></div>
  <div id="jquery2"></div>
  <div id="jquery3"></div>
</body>
</html>
```

▶ 예제 1-2 DOM이 로드되었을 때 jQuery 실행하기

- jQueryStudyCh01/jquery01_02.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>DOM이 로드되었을 때 jQuery 실행하기</title>
<!-- 외부 자바스크립트 파일 jQuery 참조 -->
<!-- <script src="js/jquery-1.12.4.min.js" ></script> -->
<script src="js/jquery-3.3.1.min.js" ></script>
<script>
  /* window 객체의 load 이벤트에 함수를 연결하면 웹 페이지에 필요한 모든 자원
   * (DOM, css, image 등의 리소스)이 완전히 로드 되었을 때 load 이벤트가
   * 발생하여 매개변수로 지정한 함수가 실행되는 반면 jQuery의 ready() 메소드는
   * 웹 페이지의 문서 객체 모델(DOM, Document Object Model)이 로드되어
   * 이 DOM을 탐색하고 조작할 수 있을 때 매개변수로 지정한 함수가 실행된다.
   **/
  $(document).ready(function() {
    $('#img1').css('border', '5px solid #FF9898')
      .css("border-radius", "20px")
      .css("width", "300px").css("height", "240px");

    let imgWidth = $('#img1').width();
```

```
    let result = $('#resultReady').text() + imgWidth;
    $('#resultReady').text(result);
  });

  // window 객체의 load 이벤트 핸들러 등록 - 고전 이벤트 방식
  window.onload = function() {
    $('#img2').css('border', '5px solid #FF9898')
      .css("border-radius", "20px")
      .css("width", "400px").css("height", "320px");

    let imgWidth = $('#img2').width();
    let result = $('#resultOnload').text() + imgWidth;
    $('#resultOnload').text(result);
  };

  // window 객체의 load 이벤트 핸들러 등록 - jQuery 3.1 방식
  $(window).on("load", function() {
    alert($('#resultReady').text());
  });
</script>
</head>
<body>
  <div id="jQuery1"><img id="img1" src="images/myungjun05.jpg" /></div>
  <div id="jQuery2"><img id="img2" src="images/myungjun06.jpg" /></div>
  <div id="resultReady">ready : </div>
  <div id="resultOnload">onload : </div>
</body>
</html>
```

▶ 예제 1-3 html(), text() 메소드

- jQueryStudyCh01/jquery01_03.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>html()메서드와 text() 메소드</title>
<!-- 외부 자바스크립트 파일 jQuery 참조 -->
<script src="js/jquery-3.3.1.min.js"></script>
<script>
  $(document).ready(function() {

    /* text() 메소드는 지정한 요소 안에 텍스트를 설정하거나 반환하는
     * 메소드로 아래 코드는 문서에서 전체 p 태그의 텍스트를 반환한다.
```

```
     **/
    let result1 = $('p').text();

    /* html() 메소드는 지정한 요소 안에 태그를 포함한 내용을 설정하거나
     * 반환하는 메소드로 아래 코드는 문서에서 첫 번째 나오는 p 태그 안의
     * 내용을 태그를 포함해 반환한다.
     **/
    let result2 = $('p').html();

    $("*").css("font-size", "12px");

    /* text() 메서드로 문서 객체의 Text 내용을 출력할 수 있다. 이때 태그가
     * 포함되어 있어도 태그가 일반 텍스트로 인식되므로 태그가 적용되지 않는다.
     * css() 메서드는 현재 jQuery로 선택한 문서 객체에 스타일을 지정하거나
     * 이미 지정된 스타일 정보를 읽어 올 수 있다.
     **/
    $('#result1').text(result1)
      .css('margin', '15px')
      .css('border', '1px solid red')
      .css('padding', '15px')
      .css('width', '500px');

    $('#result2').html(result2)
      .css('margin', '15px')
      .css('border', '1px solid red')
      .css('padding', '15px')
      .css('width', '500px');
  })
</script>
</head>
<body>
  <div id="content">
    <p>jQuery는 <i>오픈 소스이고,</i> 이 프로젝트는 <b>MIT와 GNU 일반
    공개 라이선스(GPL)를</b></p><p>가지고 있다 즉 무료라는 이야기 이다.</p>
    <div><p>jQuery는 매우 작으며, 압축도 되어 있다.</p></div>
    <p>jQuery는 <b>세계에서 가장 인기 있는 자바스크립트 라이브러리</b> 이다.</p>
    <div>
      <p>jQuery는 웹 브라우저간의 차이를 표준화 했기 때문에
      크로스 브라우징 처리를 프로그래머가 직접 처리할 필요가 없다.</p>
    </div>
    <p>jQuery는 다양하고 풍부한 <i>플러그인</i>이 많이 공개되어 있다.</p>
  </div>
  <div id="result1"></div>
  <div id="result2"></div>
</body>
```

```
</html>
```

2. jQuery 선택자와 메소드를 이용한 DOM 제어

jQuery는 문서 객체 모델(DOM, Document Object Model)를 탐색하고 여러 조건을 지정해 선택할 수 있는 다양한 메서드를 지원하고 있다. 또한 문서 객체에 자식 요소나 형제 요소를 추가하거나 문서 객체의 크기 등을 바꿀 수 있는 다양한 문서 객체 조작 메서도 지원하고 있다. 이장에서 DOM을 탐색하고 조작할 수 있는 다양한 jQuery 메서드를 활용해 DOM을 탐색하고 조작하는 방법에 대해서 알아볼 것이다.

▶ 예제 2-1 jQuery의 다양한 선택자를 사용해 DOM 제어하기

- jQueryStudyCh02/jquery02_01.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>jQuery의 다양한 선택자를 사용해 DOM 제어하기</title>
<style type="text/css">
</style>
<!-- 외부 자바스크립트 파일 jQuery 참조 -->
<script src="js/jquery-3.3.1.min.js" ></script>
<script>
  $(function() {

    $('*').css('font-size', '12px');
    $('#nameOrder').css('width', '300px')
      .css('height', '150px')
      .css('padding', '10px 20px')
      .css('margin', '10px 0px')
      .css('border', '1px dotted blue')

    $('#result').css('width', '300px')
      .css('height', '150px')
      .css('padding', '10px 20px')
      .css('margin', '10px 0px')
      .css('border', '1px dotted red');

    $('#buttons').css('margin', '30px 0px');

    /* 각 버튼의 클릭 이벤트를 처리할 이벤트 핸들러 등록
     * jQuery의 클릭 이벤트 등록 메서드인 click()에 아래와 같이 익명 함수를 지정해서
     * 이벤트 핸들러를 등록할 수 있다. 선택자로 지정한 문서 객체에 클릭 이벤트가
```

```
 * 발생하면 아래에 등록한 익명 함수를 jQuery가 호출해 이벤트를 처리하게 된다.
 **/
$("#btnNameOrder").click(function() {
  let fruitList = [];

  /* $(selector).each() 메소드는 선택자로 지정한 DOM 요소의 집합을 반복하여
   * 처리할 때 사용하는 메소드로 배열이나 객체를 따로 만들어 적용하는 것이
   * 아니라 선택자를 사용해 여러 개의 DOM 요소를 선택하고 접근할 때 유용하게
   * 사용할 수 있는 메소드 이다. each() 메소드의 인수로 콜백 함수를 지정하는데
   * 이 함수의 첫 번째 매개변수로 웹 페이지에서 읽어온 요소 집합의 현재 index가
   * 넘어오고 두 번째 매개변수로 현재 index에 해당하는 DOM 객체가 넘어온다.
   *
   * id가 fruitList의 자식 요소들을 선택해 순차적으로 순회하면서
   * 요소의 텍스트를 읽어 fruit 객체를 생성해 배열에 담는다.
   *
   * children() 메소드는 지정한 요소의 바로 아래 자식 요소들을 선택하는 메소드
   **/
  $('#fruitList').children().each(function(index, value) {

    /* each() 메소드의 매개변수로 지정하는 콜백 함수 안에서 this는
     * 현재 index에 해당하는 문서 객체를 의미 한다.
     * 이 예제에서는 this가 li 요소를 의미한다.
     **/
    fruitList.push($(this).text());
  });

  fruitList.sort();

  /* id 값이 nameOrder인 요소 안의 모든 자손 요소를 지우고
   * 동적으로 문서 객체를 생성해 다시 추가 한다.
   * empty() 메소드는 지정한 요소 안의 자손 요소를 모두 제거하는 메소드
   * remove() 메소드는 지정한 요소 자신을 제거하는 메소드
   **/
  $('#nameOrder').empty();

  // id가 nameOrder인 문서 객체의 자식으로 h4 요소와 ol 요소를 추가한다.
  $('<h4>과일 이름순 정렬</h4><ol id="fruitName"></ol>')
    .appendTo('#nameOrder');

  /* $.each() 메소드는 jQuery 유틸리티 메소드로 JavaScript의
   * 객체나 배열을 반복하여 처리할 때 사용하는 메소드이다.
   * 첫 번째 인수로 객체나 배열을 지정하고 두 번째 인수로 콜백 함수를 지정하는데
   * 이 콜백 함수의 첫 번째 매개변수로 배열의 index가 넘어오고 두 번째
   * 매개변수로 현재 index에 해당하는 값이 넘어온다. $.each() 메소드의
   * 첫 번째 인수가 객체일 경우 콜백 함수의 첫 번째 매개변수로 객체의 프로퍼티
```

```
        * 이름이 넘어오고 두 번째 매개변수로 프로퍼티 값이 넘어온다.
        **/
       $.each(fruitList, function(index, value) {

         /* $.each() 메서드 안에서 this는 fruitList 배열에 들어있는 현재 index에
          * 해당 하는 요소를 의미한다. 아래는 위에서 동적으로 생성하여 id가
          * nameOrder인 문서 객체에 추가한 ol 요소 즉 id가 fruitName인 문서
          * 객체에 현재 index에 해당하는 배열 요소와 li 태그를 조합해 li 요소를
          * 동적으로 생성해 자식으로 추가 한다.
          **/
         $('#fruitName').append('<li>' + this + '</li>');
         //$('#fruitName').append('<li>' + fruitList[index] + '</li>');
       });
   });

   /* 아래와 같이 별도의 함수를 만들어서 이벤트 핸들러로 등록할 수 있다.
    * jQuery의 클릭 이벤트 등록 메서드인 click()에 아래와 같이 별도로 정의한 함수를
    * 이벤트 핸들러로 등록할 수 있다. 선택자로 지정한 문서 객체에 클릭 이벤트가
    * 발생하면 아래에 등록한 oddList 함수를 jQuery가 호출해 이벤트를 처리하게 된다.
    **/
   $("#btnOddList").click(oddList);
   $("#btnEvenList").click(evenList);
   $("#btnFirstList").click(firstList);
   $("#btnLastList").click(lastList);
});

function oddList() {
   let result = [];
   $('#result').empty();

   // id가 fruitList인 요소의 자식 요소 li 중에서 index가 홀수인 요소들을 선택
   $('#fruitList > li:odd').each(function(index, value) {
     result.push($(this).text());
   });
   $('#nameOrder li:odd').each(function(index, value) {
     result.push($(this).text());
   });

   $('<h3>index가 홀수(:odd) 번째 리스트</h3><ol></ol>').appendTo('#result');
   $.each(result, function(index, value) {
     $('<li>' + this + '</li>').appendTo('#result > ol');
   });
}

function evenList() {
```

```
  let result = [];
  $('#result').empty();

  // id가 fruitList인 요소의 자식 요소 li 중에서 index가 짝수인 요소들을 선택
  $('#fruitList > li:even').each(function(index, value) {
    result.push($(this).text());
  });
  $('#nameOrder li:even').each(function(index, value) {
    result.push($(this).text());
  });

  $('<h3>index가 짝수(:even) 번째 리스트</h3><ol></ol>').appendTo('#result');
  $.each(result, function(index, value) {
    $('#result > ol').append('<li>' + this + '</li>');
  });
}

function firstList() {
  let result = [];
  $('#result').empty();

  // id가 fruitList인 요소의 자식 요소 li 중에서 첫 번째 요소를 선택
  $('#fruitList > li:first').each(function(index, value) {
    result.push($(this).text());
  });
  $('#nameOrder li:first').each(function(index, value) {
    result.push($(this).text());
  });

  $('<h3>첫 번째(:first) 리스트</h3><ol></ol>').appendTo('#result');
  $.each(result, function(index, value) {
    $('#result > ol').append('<li>' + this + '</li>');
  });
}

function lastList() {
  let result = [];
  $('#result').empty();

  // id가 fruitList인 요소의 자식 요소 li 중에서 마지막 요소 선택
  $('#fruitList > li:last').each(function(index, value) {
    result.push($(this).text());
  });
  $('#nameOrder li:last').each(function(index, value) {
    result.push($(this).text());
```

```
    });

    /* 위의 3개의 each() 메소드를 아래와 같이 다중 선택자를
     * 사용하면 each() 메소드 하나로 구현이 가능하다.
     **/
    /*
    $('#fruitList > li:last, #nameOrder li:last')
    .each(function(index, value) {
      result.push($(this).text());
    });
    */

    $('<h3>마지막 번째(:last) 리스트</h3><ol></ol>').appendTo('#result');
    $.each(result, function(index, value) {
      $('#result > ol').append('<li>' + this + '</li>');
    });
  }
</script>
</head>
<body>
  <ul id="fruitList">
    <li>사과 : 500원</li>
    <li>배 : 700원</li>
    <li>바나나 : 300원</li>
    <li>한라봉 : 800원</li>
    <li>귤 : 400원</li>
  </ul>
  <div id="nameOrder">
  </div>
  <div id="result">
  </div>
  <div id="buttons">
    <button id="btnNameOrder">이름순으로 정렬하여 출력하기</button><br/>
    <button id="btnOddList">index가 홀수 번째만 가져오기</button>
    <button id="btnEvenList">index가 짝수 번째만 가져오기</button><br/>
    <button id="btnFirstList">첫 번째만 가져오기</button>
    <button id="btnLastList">마지막 번째만 가져오기</button><br/>
  </div>
</body>
</html>
```

▶ 예제 2-2 함수 필터 선택자

- jQueryStudyCh02/jquery02_02.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>함수필터 선택자</title>
<style type="text/css">
</style>
<!-- 외부 자바스크립트 파일 jQuery 참조 -->
<script src="js/jquery-3.3.1.min.js" ></script>
<script>
  $(function() {
    $('*').css('font-size', '12px');
    $('#list li').css('width', '150px')
      .css('list-style', 'none')
      .css('height', '30px')
      .css('text-align', 'center')
      .css('border', '1px solid blue')
      .css('line-height', '30px')
      .end().add('li:nth-child(2n)')
      .css('border-width', '0px 1px')
      .end().add('li:last')
      .css('border-bottom', '1px solid blue')
      .end().add($('div'))
      .css('float', 'left')
      .end().add($('div:eq(1)'))
      .css('width', '300px')
      .css('height', '300px')
      .css('border', '1px solid red')
      .css('margin', '15px 15px')
      .end().add('.btn')
      .css('margin', '5px 15px');

    //alert($('li:contains(Java)').size());

    // class가 btn인 버튼에 클릭 이벤트 핸들러 등록
    $(".btn").click(function() {

      // class가 btn인 버튼이 클릭되면 id가 result인 요소 안의 모든 내용을 비운다.
      $('#result').empty();
      let resultList = [];

      // jQuery의 이벤트 핸들러 안에서 this는 이벤트가 발생한 객체를 의미한다.
      let selector = $(this).attr("data-param");

      /* 함수 필터 선택자
```

```
        * 요소:contains(문자열) : 지정한 문자열을 포함하는 요소 선택
        * 요소:eq(n) : index가 n번째 위치하는 요소 선택
        * 요소:gt(n) : index가 n번째를 초과하는 위치의 요소 선택
        * 요소:lt(n) : index가 n번째 미만에 위치하는 요소 선택
        * 요소:nth-child(2n+1) : 2n+1번째 위치하는 요소 선택 (1, 3, 5 ...)
        * nth-child() 선택자는 index가 아니라 요소의 몇 번째 위치인지를 의미한다.
        **/
        $('#list li:' + selector).each(function(index, value) {
          resultList.push($(this).text());
        });
        $('<h4>함수 필터 선택자 검색결과</h4><ol id="resultList"></ol>')
          .appendTo('#result');

        $.each(resultList, function(index, value) {
          $('#resultList').append('<li>' + this + '</li>');
        });
        //alert($('#resultList li').size());
      });
    });
  </script>
  </head>
  <body>
    <div id="list">
      <ol>
        <li>Java</li>
        <li>Oracle</li>
        <li>JSP</li>
        <li>Spring</li>
        <li>MySql</li>
        <li>JavaScript</li>
        <li>jQuery</li>
        <li>Ajax</li>
        <li>VB.NET</li>
        <li>ASP.NET</li>
        <li>HTML5</li>
        <li>CSS3</li>
      </ol>
    </div>
    <div id="result"></div>
    <!-- HTML5에서는 "data-"로 시작하는 사용자 속성을 사용할 수 있다. -->
    <button class="btn" data-param="contains(Java)">contains(Java)</button><br/>
    <button class="btn" data-param="eq(3)">eq(3) - index</button><br/>
    <button class="btn" data-param="gt(7)">gt(7) - index</button><br/>
    <button class="btn" data-param="lt(7)">lt(7) - index</button><br/>
    <button class="btn" data-param="nth-child(2n+1)">nth-child(2n+1)-순번</button><br/>
```

```
</body>
</html>
```

▶ [연습문제 2-1] 배열을 문서에 출력하기

수업 시간에 제공한 jqueryExam02_01.html 문서에서 제시한 아래 배열을 요구사항에 따라서 html 문서에 출력하시오.

let members = ['홍길동', '강감찬', '이순신', '을지문덕', '임꺽정'];

1. 위의 배열을 div id=array1에 ‘,’로 구분하여 전체 배열 요소를 한 줄로 출력하시오
2. 위의 배열을 div id=array2에 배열 요소를 한 줄에 하나씩 출력하시오
3. 위의 배열을 div id=array2에 순서 있는 목록으로 출력하시오
4. 위의 배열을 div id=array3에 제목을 “회원 리스트”로 하여 배열 요소를 순서 있는 목록으로 출력하시오
5. div id=array3에 출력된 회원 목록을 가져와서 div id=array4에 제목을 “회원 리스트 역순”으로 하고 array3에 출력된 리스트의 역순으로 출력하시오.

▶ 예제 2-3 filter()와 find() 메소드를 이용한 DOM 탐색

- jQueryStudyCh02/jquery02_03.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>filter()와 find() 메서드를 이용한 DOM 탐색</title>
<style type="text/css">
</style>
<!-- 외부 자바스크립트 파일 jQuery 참조 -->
<script src="js/jquery-3.3.1.min.js" ></script>
<script>
  $(function() {

    $('#container')
      .css('position', 'relative')
      .css('width', '600px');

    $('#container > .imgs')
      .css('float', 'left')
      .css('width', '160px')
      .css('height', '150px')
```

```
      .css('margin', '10px 10px')
      .css('text-align', 'center');

  $('#result')
      .css('width', '600px')
      .css('height', '250px')
      .css('margin', '10px 10px')
      .css('border', '1px solid red');
});

function filterSearch(value) {

  // 지정한 요소의 style 속성을 지운다.
  $('#container img').removeAttr('style');

  /* $(selector).filter(selector) :요소 중에서 필터 조건에 맞는 요소만 선택
   * $(selector).filter(function(){}) : 필터 함수에서 true를 반환하는 요소만 선택
   *
   * end() 메소드는 문서 객체 선택을 한 단계 이전으로 돌린다.
   */
  if(value == 'selector') {

    /* id가 container 요소의 자손 요소 중 img 요소들을 선택하고 그중에서
     * src 속성의 값이 js 문자열을 포함하는 요소만 선택한 후 border를 설정한다.
     * 그리고 선택을 한 단계 앞으로 돌려 다시 id가 container 요소의 자손
     * 요소 중 img 요소들을 선택하고 그 중에서 src 속성의 값이 web 문자열을
     * 포함하는 요소만 선택해 border를 설정한다.
     **/
    $('#container img').filter('[src*=js]')
      .css('border', '3px dotted blue')
      .end().filter('[src*=web]')
      .css('border', '3px dotted red');

  } else if(value == 'function'){
    $('#container img')
      .filter(function(index) {
        return $(this)
          .attr('src') == 'images/modern_html5.jpg'
          || index > 5;
      }).css('border', '3px solid red');
  }
}

function findSearch(value) {
```

```
    // 선택한 요소를 지운다.
    $('#result > ol').remove();
    $('#result > div').remove();

    // $(selector).find(selector) : 요소 중 조건에 맞는 자손을 전체 선택
    if(value == 'img') {
      $('#result').append('<ol id="imgList"></ol>');
      $('#container').find(value).each(function(index, value) {
        $('#imgList')
          .append('<li>' + $(this).attr('src') + '</li>');
      });

    } else if(value == 'clone'){

      /* clone() 메소드는 선택한 요소를 복제하여 반환한다.
       * removeClass() 메소드는 선택한 요소의 class 속성을 지운다.
       **/
      let $clone = $('#container > .imgs:nth-child(3n+1)').clone()
        .removeClass('imgs');

      /* end() 메소드는 문서 객체 선택을 한 단계 이전으로 돌린다.
       * add() 메소드는 문서 객체를 추가로 더 선택한다.
       *
       * 아래는 id가 result에 $clone 요소를 추가하고 end() 메소드를 사용해
       * 현재 선택된 id가 result인 요소의 선택을 한 단계 앞으로 되돌리는데 여기서는
       * 현재 선택된 요소의 이전이 없으므로 add() 메서드에 지정한 요소만 선택되어
       * id가 result의 자손인 img 요소를 선택하여 border를 설정하고 있다.
       **/
      $('#result').append($clone)
        .end().add('#result img').css('border', 'none');

      // body 요소의 자손 요소 중에서 img 요소를 모두 선택한다.
      let $childs = $('body').find($('img'));
      $('<div><h3>문서의 이미지 수는 ' + $childs.length
        + '개 입니다.</h3></div>').appendTo($('#result'))
        .css('clear', 'both').css('text-align', 'center');
    }
  }
</script>
</head>
<body>
  <div id="container">
    <div class="imgs">
      <img src="images/js_coreguide.jpg" /><br/>
      <label>js_coreguide.jpg</label>
```

```
  </div>
  <div class="imgs">
    <img src="images/js_corepattern.jpg" /><br/>
    <label>js_corepattern.jpg</label>
  </div>
  <div class="imgs">
    <img src="images/web_publishing.jpg" /><br/>
    <label>web_publishing.jpg</label>
  </div>
  <div class="imgs">
    <img src="images/web_recipe.jpg" /><br/>
    <label>web_recipe.jpg</label>
  </div>
  <div class="imgs">
    <img src="images/modern_html5.jpg" /><br/>
    <label>modern_html5.jpg</label>
  </div>
  <div class="imgs">
    <img src="images/modern_javascript.jpg" /><br/>
    <label>modern_javascript.jpg</label>
  </div>
  <div class="imgs">
    <img src="images/make_html5.jpg" /><br/>
    <label>make_html5.jpg</label>
  </div>
  <div class="imgs">
    <img src="images/javascript_guide.jpg" /><br/>
    <label>javascript_guide.jpg</label>
  </div>
  <div class="imgs">
    <img src="images/js_webapplication.jpg" /><br/>
    <label>js_webapplication.jpg</label>
  </div>
  <div id="buttons">
    <button onclick="filterSearch('selector')">
      filter()로 탐색</button>
    <button onclick="filterSearch('function')">
      filter(function(){})로 탐색</button>
    <button onclick="findSearch('img')">
      find().each()로 탐색</button>
    <button onclick="findSearch('clone')">
      find()로 탐색</button>
  </div>
</div>
<div id="result"></div>
```

```
</body>
</html>
```

▶ 예제 2-4 클래스 조작 메서드

- jQueryStudyCh02/jquery02_04.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>클래스 조작 메소드</title>
<style>
  #img2 {
    width: 400px;
    height: 320px;
    border-radius: 20px;
  }
  #img2.hover {
    border-radius: 160px;
  }
</style>
<!-- 외부 자바스크립트 파일 jQuery 참조 -->
<script src="js/jquery-3.3.1.min.js" ></script>
<script>
  $(function() {

    $('#img2').css('border', '5px solid #FF9898')
    .css("width", "400px").css("height", "320px");

    /* hover() 메서드는 첫 번째 매개변수의 콜백 함수에 마우스 커서가
     * 문서 객체에 올라갔을 때 실행할 코드를 기술하고 두 번째 매개변수의
     * 콜백 함수에 마우스 커서가 문서 객체를 떠날 때 실행할 코드를 기술하면 된다.
     *
     * addClass() : 문서 객체에 클래스 속성을 추가하는 메서드
     * removeClass() : 문서 객체에서 클래스 속성을 삭제하는 메서드
     * toggleClass() : 문서 객체에 클래스 속성을 추가/삭제 동작을 전환하는 메서드
     **/
    $('#img2').hover(function() {
      $(this).addClass("hover")
    }, function() {
      $(this).removeClass("hover");
    });

  });
```

```
</script>
</head>
<body>
  <div><img id="img2" src="images/myungjun06.jpg" /></div>
</body>
</html>
```

3. jQuery 이벤트

웹 문서에서 사용자가 버튼을 클릭 하거나, 마우스 포인터를 특정 요소위에 올리거나, 텍스트 박스의 내용을 변경 할 때와 같이 사용자가 웹 문서에서 취하는 일련의 동작으로 발생하는 사건을 이벤트라고 한다. 브라우저에서 발생하는 이벤트에는 사용자 동작에 의해 발생하는 이벤트가 있고 특정 상황에 따라 자동으로 발생되는 이벤트가 있다. 특정 상황이란 웹 문서 로드가 완료되었을 때 또는 폼이 Submit 될 때 등등의 상황을 말한다.

프로그래밍에서 이벤트가 발생하면 각각의 이벤트에 따라 이를 처리하기 위해 함수와 같은 코드를 연결(등록)하여 이벤트를 처리하는 프로그래밍 모델을 이벤트 드리븐 모델(Event Driven Model 또는 이벤트 모델) 이라고 하며 이벤트가 발생했을 때 이를 처리하기 위해 이벤트에 연결(등록)한 함수를 이벤트 핸들러(Event Handler)라 부른다.

예전에 JavaScript 만으로 이벤트를 처리할 때는 브라우저 버전이나 종류에 따라 이벤트 객체의 속성이 다르기 때문에 각각의 브라우저에 대응하는 코드를 기술해야 했다. 하지만 jQuery는 내부적으로 이벤트 처리를 정형화 시켰기 때문에 jQuery 이벤트 객체는 동일한 사용방법과 동일한 이름의 속성을 갖는다. 그래서 jQuery를 사용해 이벤트를 처리하게 되면 각각의 브라우저에 대응하는 코드를 따로 기술하지 않아도 된다.

jQuery는 다음과 같은 이벤트 등록 메소드를 제공하고 있으며 jQuery가 제공하고 있는 이벤트 등록 메소드는 하나의 이벤트 종류만 등록할 수 있는 단일 이벤트 등록 메소드와 하나 이상의 이벤트를 등록할 수 있는 다중 이벤트 등록 메소드를 지원하고 있다.

▶ jQuery의 단일 이벤트 등록 메소드

구 분	이벤트 타입 (이벤트 이름)	이벤트 등록 메서드	발생 상황
문서 읽기	load	load()	HTML 문서(iframe, 이미지, 동영상)의 모든 구성요소가 로딩이 완료 되었을 때 jQuery 3.1부터 deprecated 되어 있으므로 on() 메서드를 이용해 이벤트를 처리해야 함
	ready	ready()	HTML 문서 객체(DOM)가 로딩이 완료 되었을 때
	unload	unload()	문서를 닫거나 다른 문서로 이동할 때 브라우저를 새로 고침 해도 발생함.
윈도우	resize	resize()	윈도우 크기가 변경 될 때 강제로 resize 이벤트를 발생시키거나 이벤트 처리 핸들러를 등록할 수 있다.
	scroll	scroll()	스크롤 했을 때
	error	error()	에러가 발생할 때
포커스	blur	blur()	요소에서 포커스가 사라질 때
	focus	focus()	요소가 포커스를 받을 때
	focusin	focusin()	요소가 포커스를 받을 때 focus() 메소드 보다 먼저 실행됨
	focusout	focusout()	요소에서 포커스가 사라질 때 blur() 메소드 보다 먼저 실행됨

구 분	이벤트타입 (이벤트 이름)	이벤트 등록 메서드	발생 상황
폼 요소	submit	submit()	서브밋 버튼을 눌렀을 때
	reset	reset()	리셋 버튼을 눌렀을 때
	change	change()	입력 양식의 내용이나 항목이 변경될 때
	select	select()	입력 양식의 항목이 선택될 때 intput type=text, textrea 제외
키보드	keydown	keydown()	키보드의 키를 눌렀을 때
	keypress	keypress()	키보드의 키가 눌렸다 떼어질 때
	keyup	keyup()	키보드의 키가 눌렸다 올라올 때 keypress() 메소드 다음으로 실행됨
마우스	click	click()	마우스가 클릭되었을 때
	dblclick	dblclick()	마우스가 더블클릭 되었을 때
	mousedown	mousedown()	마우스 버튼이 눌려질 때
	mouseup	mouseup()	마우스 버튼이 눌렸다 떼어질 때
	mouseover	mouseover()	마우스 커서가 요소에 올라올 때 mouseenter() 보다 먼저 실행됨
	mouseenter	mouseenter()	마우스 커서가 요소의 외부에서 내부로 들어올 때
	mouseout	mouseout()	마우스 커서가 요소를 벗어날 때 mouseleave() 보다 먼저 실행됨
	mouseleave	mouseleave()	마우스 커서가 요소의 내부에서 외부로 나갈 때
	mouseenter mouseleave	hover (mouseenter, mouseleave)	마우스 커서가 요소에 올라올 때와 벗어날 때 각각 실행됨 mouseenter 이벤트와 mouseleave 이벤트를 동시에 등록할 때 사용
	mousemove	mousemove()	요소 내에서 마우스 커서가 움직일 때
	contextmenu	contextmenu()	컨텍스트 메뉴가 표시될 때

jQuery의 단일 이벤트 등록 메소드

▶ 이벤트 객체의 메서드와 속성

jQuery에서는 이벤트가 발생하면 이벤트 핸들러의 첫 번째 인수로 이벤트 객체가 전달되기 때문에 이벤트 핸들러 안에서 이벤트 객체에 접근하려면 첫 번째 인수를 통해 접근할 수 있다. 이벤트 객체는 이벤트에 대한 여러 가지 정보를 저장하고 있는 객체로 다음과 같은 메서드와 속성을 제공하고 있다.

구 분	메서드와 속성	설 명
일반	target	이벤트가 발생한 마지막 요소
	type	이벤트 종류 - click, mousedown 등
	cancelBubble	이벤트 전파 속성, 기본 값은 false이며 true를 설정하면 이벤트 전파를 차단한다.
	stopPropagation()	이벤트 전파를 차단하는 메서드
	preventDefault()	기본 동작을 차단하는 메서드, <a> 태그는 클릭되면 href 속성에 저장한 곳으로 문서가 이동되는 기본 동작을 가지고 있다. 이 요소에 click 이벤트를 적용하고 이벤트 핸들러 안에서 preventDefault() 메서드를 호출하면 <a> 태그의 기본 동작을 차단할 수 있다.
좌표	clientX	문서를 기준으로 이벤트가 발생한 X좌표(스크롤 미포함)
	clientY	문서를 기준으로 이벤트가 발생한 Y좌표(스크롤 미포함)
	screenX	모니터를 기준으로 이벤트가 발생한 X좌표
	screenY	모니터를 기준으로 이벤트가 발생한 Y좌표
	pageX	문서를 기준으로 이벤트가 발생한 X좌표(스크롤 포함)
	pageY	문서를 기준으로 이벤트가 발생한 Y좌표(스크롤 포함)
키보드 /마우스	button	마우스 버튼 종류(왼쪽 : 0, 휠 : 1, 오른쪽 : 2)
	keyCode	키보드에서 현재 눌린 키의 아스키 코드 값
	altKey	alt 키의 눌린 상태(눌린 상태 true, 아니면 false)
	ctrlKey	ctrl 키의 눌린 상태(눌린 상태 true 아니면 false)
	shiftKey	shift 키의 눌린 상태(눌린 상태 true 아니면 false)

이벤트 객체의 메서드와 속성

▶ 예제 3-1 마우스관련 단일 이벤트 등록 메소드와 이벤트 객체

- jQueryStudyCh03/jquery03_01.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>마우스관련 단일 이벤트 등록 메소드와 이벤트 객체</title>
<style type="text/css">
  .mouseover {
    border: 2px dotted blue;
    opacity: 0.5;
  }
  .mouseout {
    border: none;
  }
</style>
```

```
<!-- 외부 자바스크립트 파일 jQuery 참조 -->
<script src="js/jquery-3.3.1.min.js"></script>
<script>
  $(function() {

    /* jQuery는 다양한 이벤트에 대응하는 이벤트 핸들러를 등록할 수 있도록
     * 각각의 이벤트 이름과 동일한 이벤트 핸들러 등록 메서드를 제공하고 있다.
     *
     * $(selector).이벤트 등록 메서드(function(event) { });
     **/
    /* 마우스 커서가 img 요소의 외부에서 내부로 들어올 때 mouseenter 이벤트가 발생하며
     * 이 이벤트를 처리하기 위해서 아래와 같이 익명 함수를 메서드의 인수로 지정하면
     * mouseenter 이벤트가 발생할 때 마다 jQuery가 이 함수를 호출한다. 이렇게 이벤트에
     * 대응하는 함수를 이벤트를 핸들링 하는 함수라는 의미로 이벤트 핸들러라고 부른다.
     *
     * 이벤트 핸들러 안에서 이벤트 객체에 접근하려면 아래와 같이 첫 번째 매개변수를
     * 지정하고 그 매개변수로 이벤트 객체의 속성이나 메서드를 사용할 수 있다.
     **/
    $('img').mouseenter(function(e) {

      // 이벤트 핸들러 함수 안에서 this는 이벤트가 발생한 문서 객체를 의미한다.
      $(this).removeClass('mouseout')
        .addClass('mouseover');

      /* is(selector) 메소드는 지정한 객체가 selector와
       * 일치하는지 판단해 boolean 값으로 반환하는 메소드
       *
       * 이벤트 속성에서 target은 이벤트가 발생한 문서 객체를 의미한다.
       **/
      if($(e.target).is('#img3')) {
        alert('세 번째 이미지로 마우스가 들어 왔네요');
      }
    });

    /* 마우스 커서가 img 요소의 내부에서 외부로 나갈 때 mouseleave 이벤트가 발생하며
     * 이 이벤트를 처리하기 위해서 아래와 같이 익명 함수를 메서드의 인수로 지정하면
     * mouseleave 이벤트가 발생할 때 마다 jQuery가 이 함수를 호출한다. 이렇게 이벤트에
     * 대응하는 함수를 이벤트를 핸들링 하는 함수라는 의미로 이벤트 핸들러라고 부른다.
     **/
    $('img').mouseleave(function(e) {
      $(this).removeClass('mouseover')
        .addClass('mouseout');
    });

    /* hover() 메소드는 mouseentter와 mouseleave 이벤트를 동시에 등록하는 메소드로
```

```
     * 첫 번째 인수로 지정하는 함수는 mouseenter 이벤트를 처리할 핸들러이며
     * 두 번째 인수로 지정하는 함수는 mouseleave 이벤트를 처리할 핸들러이다.
     *
     * $(selector).hover(function(event) { }, function(event) { });
     **/
    /* $("img").hover(function(e) {
      // 이벤트 핸들러 함수 안에서 this는 이벤트가 발생한 문서 객체를 의미한다.
      $(this).removeClass('mouseout')
        .addClass('mouseover');

      // 이벤트 속성에서 target은 이벤트가 발생한 문서 객체를 의미한다.
      if($(e.target).is('#img3')) {
        alert('세 번째 이미지로 마우스가 들어왔네요');
      }
    }, function() {
      $(this).removeClass('mouseover')
      .addClass('mouseout');
    }); */

    // img 요소에서 더블 클릭 이벤트가 발생하면 처리할 이벤트 핸들러 등록
    $('img').dblclick(function(e) {
      if($(this).is('#img2')) {
        alert('두 번째 이미지를 더블 클릭하셨네요111'
          + "\n이벤트 종류 : " + e.type + ", clientX : " + e.clientX
          + "\nscreenX : " + e.screenX + ", pageX : " + e.pageX)
      }
    });

    // img 요소에서 클릭 이벤트가 발생하면 처리할 이벤트 핸들러 등록
    $('img').click(function(e) {
      if($(this).is('img:eq(0)')) {
        alert('첫 번째 이미지를 클릭 하셨네요'
          + "\n이벤트 종류 : " + e.type + ", 눌린 버튼 : "
          + (e.button == 0 ? "왼쪽 버튼" : e.button == 1 ? "휠" : "오른쪽 버튼"));
      }
    })
  });
</script>
</head>
<body>
  <div id="container1">
    <img id="img1" src="images/modern_javascript.jpg" />
  </div>
  <div id="container2">
    <img id="img2" src="images/make_html5.jpg" />
```

```
  </div>
  <div id="container3">
    <img id="img3" src="images/web_recipe.jpg" />
  </div>
</body>
</html>
```

▶ 예제 3-2 통합 이벤트 등록 메서드와 제거 메서드 1

- jQueryStudyCh03/jquery03_02.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>통합 이벤트 등록 메서드와 제거 메소드 1</title>
<style type="text/css">
  .mouseover {
    border: 2px dotted blue;
    opacity: 0.5;
  }
  .mouseout {
    border: none;
  }
</style>
<!-- 외부 자바스크립트 파일 jQuery 참조 -->
<script src="js/jquery-3.3.1.min.js"></script>
<script>
  $(function() {

    /* jQuery 1.7부터 통합 이벤트 등록을 위해서 on() 메소드를 제공하고 있으며
     * 이 메서드를 사용하면 단일 이벤트와 다중 이벤트에 대응하는 이벤트 핸들러를 등록할
     * 수 있다. on() 메소드를 사용하면 현재 문서에 존재하는 문서 객체와 동적으로 생성되어
     * 문서에 추가되는 문서 객체에 이벤트 핸들러를 등록할 수 있다.
     **/
     // $("선택자").on("이벤트명", function(e) {}) 단일 이벤트 핸들러 등록
    $('img').on('mouseenter', function(e) {

      // 이벤트 핸들러 안에서 this는 이벤트가 발생한 문서 객체를 의미한다.
      $(this).removeClass('mouseout')
        .addClass('mouseover');
    });

    $('img').on('mouseleave', function(e) {
      $(this).removeClass('mouseover')
```

```
            .addClass('mouseout');
        });

        $('#img1').on('click', function(e) {
            alert($('div#container1').next().attr('id'));
        });

        // $("선택자").on({"이벤트명" : function(e){}, ...}) 다중 이벤트 핸들러 등록
        $('img').on({

            /* is(selector) 메소드는 지정한 객체가 selector와
             * 일치하는지 판단해 boolean 값으로 반환하는 메소드
             **/
            dblclick: function(e) {
                if($(this).is('#img2')) {
                    alert('두 번째 이미지를 더블 클릭 하셨네요');
                }
            },
            mouseenter: function(e) {
                if($(e.target).is('#img3')) {
                    console.log('세 번째 이미지로 마우스가 들어 왔네요');
                }
            },
            mouseleave: function(e) {
                if($(this).is('#img1')) {
                    $(this).css('border', '2px solid red');
                }
            }
        });

        // 버튼이 클릭되면 처리할 이벤트 핸들러 등록
        $("#btn").on("click", function() {

            // trigger() 메서드는 인수로 지정한 이벤트를 강제로 발생시키는 메서드
            $("#img2").trigger("dblclick");
        });
        // 이벤트 제거 - 인수에 제거할 이벤트 타입을 지정한다.
        //$('img').off('dblclick');
        //$('img').off('dblclick mouseenter');
    });
</script>
</head>
<body>
    <div id="container1">
        <img id="img1" src="images/modern_javascript.jpg" />
```

```
  </div>
  <div id="container2">
    <img id="img2" src="images/make_html5.jpg" />
  </div>
  <div id="container3">
    <img id="img3" src="images/web_recipe.jpg" />
  </div>
  <div id="container4">
    <button id="btn">두 번째 이미지 더블 클릭 발생시키기</button>
  </div>
</body>
</html>
```

▶ 예제 3-3 통합 이벤트 등록 메서드와 제거 메서드 2

- jQueryStudyCh03/jquery03_03.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>통합 이벤트 등록 메서드와 제거 메소드 2</title>
<style type="text/css">
  .mouseover {
    border: 2px dotted blue;
    opacity: 0.5;
  }
  .mouseout {
    border: none;
  }
</style>
<!-- 외부 자바스크립트 파일 jQuery 참조 -->
<script src="js/jquery-3.3.1.min.js"></script>
<script>
  $(function() {

    /* 이벤트 등록 메소드
     * one() : 한 번만 이벤트 등록
     *
     * 통합 이벤트 등록 메소드와 제거 메소드 - jQuery 1.7부터 지원
     * on() : 기본 이벤트 핸들러 및 이벤트 위임 방식 이벤트 핸들러 등록
     * off() : on() 메서드로 등록한 이벤트 핸들러를 제거하는 메소드
     **/
    // on() 메소드를 이용해 click 이벤트에 대한 단일 이벤트 등록
    $('#img1').on('click', function(e) {
```

```
        $(this).parent().clone() .appendTo('body');

        alert($('img').length + ' 번째에 이미지를 추가하였습니다.\n'
          + '마우스의 X좌표 : ' + e.pageX
          + ', Y좌표 : ' + e.pageY
          + ', 눌린버튼 : ' + e.which);
      });

      /* one() 메소드를 이용한 click 이벤트에 대한 단일 이벤트 등록
       * 이벤트를 한 번만 실행하려면 one() 메소드를 이용하면 편리하다.
       **/
      $('#img2').one('click', function(e) {
        $(this).parent().clone() .appendTo('body');
      });

      /* 기본 이벤트 핸들러 등록
       * 현재 존재하는 문서 객체만 이벤트 핸들러를 등록할 수 있으며
       * 동적으로 추가되는 문서 객체에는 이벤트 핸들러가 등록되지 않는다.
       **/
      $('#img2').on('mouseenter', function(e) {
        $(this).parent().prev().clone().appendTo('body');
      });

      /* 이벤트 위임 방식 이벤트 핸들러 등록
       * 이벤트 위임 방식은 상위 요소에 이벤트 핸들러를 등록하는 방식으로
       * 이벤트 버블링을 활용하기 때문에 동적 생성해 추가되는 문서 객체나
       * 기존 객체를 복제한 문서 객체 등에 이벤트 핸들러를 등록할 수 있다.
       **/
      $(document).on('mouseenter', '#img3', function(e) {
        $(this).parent().clone().appendTo('body');
      })
    });
</script>
</head>
<body>
  <div id="container1">
    <img id="img1" src="images/modern_javascript.jpg" />
    modern_javascript.jpg
  </div>
  <div id="container2">
    <img id="img2" src="images/make_html5.jpg" />
    make_html5.jpg
  </div>
  <div id="container3">
    <img id="img3" src="images/web_recipe.jpg" />
```

```
    web_recipe.jpg
  </div>
</body>
</html>
```

▶ 예제 3-4 이벤트 전파와 기본 이벤트 제어

- jQueryStudyCh03/jquery03_04.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>이벤트 전파와 기본 이벤트 제어</title>
<style type="text/css">
  .mouseover {
    border: 2px dotted blue;
    opacity: 0.5;
  }
  .mouseout {
    border: none;
  }
</style>
<!-- 외부 자바스크립트 파일 jQuery 참조 -->
<script src="js/jquery-3.3.1.min.js"></script>
<script>
  /* 이벤트 전파(Event Propagation)
   * 이벤트는 보통 하나의 문서 객체에 대해 발생하지만 서로 중첩된
   * 문서 객체에서 상위 또는 하위로 이벤트가 전파되는 특징을 가진다.
   * 이벤트 전파는 다음과 같이 이벤트 버블링과 이벤트 캡처링이 있다.
   *
   * - 이벤트 버블링 :
   *   이벤트가 발생한 객체로부터 순서대로 상위 객체로 이벤트가 전파되는 것을
   *   의미하며 이벤트가 발생한 객체부터 document 객체까지 상위로 전파된다.
   *
   * - 이벤트 캡처링 :
     *   document 객체에서 시작해 이벤트가 발생한 객체까지 하위 객체로 이벤트가
   *   전파되는 것을 의미한다.
     *
   * - 이벤트 전파 막기
   *   이벤트 객체의 stopPropagation() 메서드를 사용하면 이벤트 전파를 막을 수 있다.
   *   e.stopPropagation();
   *
   * 기본 이벤트(Default Event)
   * HTML 문서에서 어떤 요소는 기본적으로 연결된 이벤트 핸들러를 가지고 있어서
```

```
 * 별도의 이벤트 핸들러를 등록하지 않아도 특정 이벤트가 발생하면 기본적인
 * 동작을 하는데 이런 기본 동작을 기본 이벤트(Default Event)라고 부른다.
 * 예를 들면 HTML 문서에서 a 요소가 클릭되면 href 속성에 연결된 주소로 이동하는
 * 기본 동작을 하는데 이런 기본 동작을 기본 이벤트라고 한다. 이렇게 a 요소와
 * 같이 기본 동작이 연결되어 있어서 특정 이벤트가 발생하면 무조건 기본 동작을
 * 수행하는 요소들이 있다. 대표적으로 form 요소와 관련있는 입력 컨트롤이 이런
 * 기본 동작을 수행하며 checkbox, radio, text 입력 상자와 같이 input 요소가
 * 기본 동작을 가지고 있다. checkbox를 예를 들면 사용자가 클릭할 때 마다 체크
 * 박스의 항목이 체크 되거나 해제되는 기본 동작을 한다. 이외에도 submit 버튼
 * 있으며 submit 버튼이 클릭되면 폼 안에 있는 입력 컨트롤에 입력된 정보를
 * form 요소의 action 속성에 지정된 url로 전송하는 기본 동작을 수행한다.
 * 만약 폼이 전송될 때 폼 유효성 검사를 수행해서 폼 전송을 취소해야 된다면
 * submit 되는 기본 동작을 멈춰야 하는데 이렇게 요소가 가진 기본 이벤트를
 * 취소해야 할 때 종종 발생한다.
 *
 * - 기본 이벤트 취소
 *  이벤트 객체의 preventDefault() 메서드를 사용하면 기본 이벤트를 멈출 수 있다.
 *  e.preventDefault();
 **/
$(function() {

  $('#container').css('width', '350px')
    .css('height', '350px')
    .css('border', '1px solid red')
    .children().css('width', '250px')
    .css('height', '250px')
    .css('border', '1px solid blue')
    .css('margin', '50px auto')
    .css('line-height', '350px');

  $('img').on('click', function(e) {
    $(this).css('border', '3px dotted red');

    // 이벤트 전파 제거
    e.stopPropagation();
  });

  $('#inner').on('click', function(e) {
    $(this).css('background', '#ffaaaa');
  });

  $('#container').on('click', function(e) {
    $(this).css('background', '#aaffaa');
  });
```

```
    $('#naver').click(function(e) {

      // 기본 이벤트 제거
      e.preventDefault();

      /* jQuery를 사용할 때 이벤트 핸들러 안에서 false를 반환하면
       * 이벤트 전파와 기본 이벤트 두 가지 모두를 제거할 수 있다.
       **/
      //return false;
    })
  });
</script>
</head>
<body>
  <div id="container">
    <div id='inner'>
      <img src="images/modern_javascript.jpg"></img>
    </div>
  </div>
  <div>
    <a id="naver" href="http://www.naver.com">네이버 가기</a>
    <a id="daum" href="http://www.daum.net">다음 가기</a>
  </div>
</body>
</html>
```

▶ 예제 3-5 체크박스 다루기

- jQueryStudyCh03/jquery03_05.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>체크박스 다루기</title>
<style>
</style>
<!-- 외부 자바스크립트 파일 jQuery 참조 -->
<script src="js/jquery-3.3.1.min.js"></script>
<script>
  $(function() {

    $('.checkWrap').css('padding', '5px');

    $('form').find('[type=checkbox]').on('click', function() {
```

```
// "전체 선택하기" 체크박스가 클릭되었으면
if($(this).is('#checkall')) {

  /* 폼 안의 모든 checkbox를 "전체 선택하기" checkbox와 체크 상태를
   * 동일하게 설정한다. 요소의 속성을 설정하거나 값을 읽어올 때는 일반적으로
   * attr() 메서드를 이용하지만 아래와 같이 checked 속성과 같이 별도로
   * 지정하는 값이 없고 속성만 지정하는 경우에는 prop() 메서드를 사용해야
   * 한다. 아래에서 attr() 메소드를 이용하면 한 번은 제대로 동작하나 이후는
   * 오동작 한다. checked 속성은 별도로 지정하는 값이 없고 속성만 지정하기
   * 때문에 prop() 메서드를 사용해야 한다. disabled, readonly와 같은
   * 속성도 prop() 메소드를 사용해야 제대로 동작한다.
   **/
  $('input[type=checkbox]')
    .prop('checked', $(this).is(':checked'));

  console.log("chkall - " + $('input[type=checkbox]').length
      + ", " + $(this).is(':checked'));
}

let amount = 0;
let checkCount = 0;

/* div 태그중에서 index 0 보다 큰 div 태그의 자식에서 type 속성이
 * checkbox인 문서 객체를 선택하여 if 문과 is() 메서드를 이용해
 * checked된 문서 객체인지 조사하여 합계 금액과 체크된 수를 계산한다.
 **/
$('div').filter(':gt(0)').find(':checkbox').each(function(index, value) {

  // 현재 checkbox가 checked 상태면
  if($(this).is(':checked')) {
    amount = amount + parseInt($(this).val());
    ++checkCount;
  }
});
console.log(checkCount);

// 상품 선택 체크박스가 모두 선택되었는지 조사
if(checkCount == 4) {
  // attr() 메서들 이용하면 제대로 동작하지 않는다.
  //$('input').first().attr('checked', true);
  $('#checkall').prop('checked', true);

  console.log("true checkCount = " + checkCount
      + " : "+ $('#checkall').attr('checked'));
```

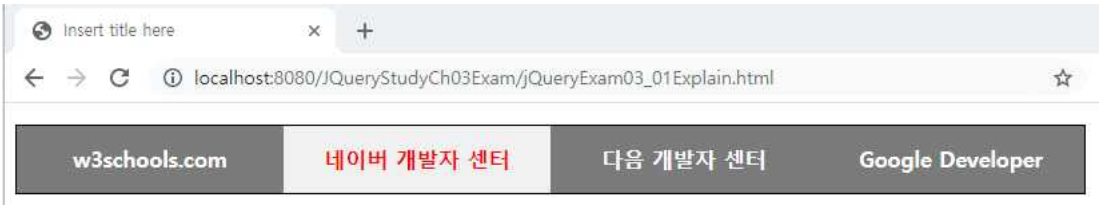
```
        } else {
          // attr(), removeAttr() 메서드를 이용하면 제대로 동작하지 않는다.
          //$('input').first().attr('checked', false);
          //$('#checkall').removeAttr('checked');
          $('#checkall').prop('checked', false);

          console.log("flase checkCount = " + checkCount
              + " : " + $('#checkall').attr('checked'));
        }

        /* 기존에 p 태그를 삭제하고 체크박스에서 선택된 상품의 합계 금액을
         * 출력하기 위해 다시 p 태그를 생성해 마지막 div 요소 다음에 추가한다.
         **/
        $('form p').remove();
        $('<p></p>').insertAfter('div:eq(4)');
        $('p').text('합계 금액 : ' + amount + '원 입니다.');
      });
    });
</script>
</head>
<body>
  <form>
    <div class="checkWrap">
      <label><input type="checkbox" id="checkall">전체 선택하기</label>
    </div>
    <div class="checkWrap">
      <label><input type="checkbox" id="pizza"
        name="pizza" value="3500">조각피자 3500원</label>
    </div>
    <div class="checkWrap">
      <label><input type="checkbox" id="hotdog"
        name="hotdog" value="2500">핫도그 2500원</label>
    </div>
    <div class="checkWrap">
      <label><input type="checkbox" id="coke"
        name="coke" value="3000">조각케익 3000원</label>
    </div>
    <div class="checkWrap">
      <label><input type="checkbox" id="fries"
        name="fries" value="3000">튀김 3000원</label>
    </div>
  </form>
</body>
</html>
```

▶ [연습문제 3-1] 메뉴 구성하고 이벤트 처리하기

다음과 같은 html 문서가 있다. 스타일을 적용해 코드 아래쪽에 있는 실행 화면과 같이 실행되도록 작성하고 이 문서에서 메뉴를 클릭하면 urls 배열에 있는 url로 이동할 수 있도록 jQuery를 활용해 이벤트를 처리하는 프로그램을 작성하시오.




```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>

<script>
  $(function() {

    let urls = [
        "http://www.w3schools.com",
        "http://dev.naver.com",
        "http://developers.daum.net",
        "http://developers.google.com"
    ]


  });
</script>
</head>
<body>
  <nav>
    <ul>
      <li><a href="#">w3schools.com</a></li>
      <li><a href="#">네이버 개발자 센터</a></li>
      <li><a href="#">다음 개발자 센터</a></li>
      <li><a href="#">Google Developer</a></li>
    </ul>
```

```
  </nav>
</body>
</html>
```

4. jQuery 효과와 애니메이션

이전에는 웹 페이지에서 애니메이션과 같은 시각적 효과를 주기위해 주로 플래시를 많이 사용했었다. 하지만 지금은 HTML5와 CSS3를 사용해 간단한 애니메이션을 구현할 수 있고 jQuery를 사용하면 보다 멋스러운 여러 가지 시각적 효과를 표현할 수 있다.

jQuery는 기본 애니메이션을 위한 효과(Effect) 메소드를 제공하고 있으며 사용자가 직접 애니메이션을 지정할 수 있는 애니메이션 메소드를 제공하고 있다. jQuery에서 제공하는 기본적인 애니메이션 메소드를 효과(Effect) 메소드라고 부르며 이 메소드들은 문서 객체의 크기를 변경하거나 투명도를 조절해 애니메이션을 구현한다.

메소드	설 명
show()	지정한 문서 객체를 점점 선명하게 하면서 크게 표시한다.
hide()	지정한 문서 객체를 점점 투명하게 하면서 작게 숨긴다.
toggle()	show()와 hide()를 번갈아 실행한다.
slideDown()	지정한 문서 객체를 미끄러지듯이 크게 표시한다.
slideUp()	지정한 문서 객체를 미끄러지듯이 작게 숨긴다.
slideToggle()	slideDown()과 slideUp()을 번갈아 실행한다.
fadeIn()	지정한 문서 객체를 투명한 상태에서 선명하게 표시한다.
fadeOut()	지정한 문서 객체를 선명한 상태에서 투명하게 숨긴다.
fadeTo()	지정한 문서 객체를 지정한 투명도 까지 투명도를 조절한다.
fadeToggle()	fadeIn(), fadeOut()을 번갈아 실행한다.

jQuery 효과(Effect) 메소드

▶ 기본 효과(Effect) 메소드 사용법

```
$(selector).method()
$(selector).method(duration)
$(selector).method(duration, callback)
$(selector).method(duration, easing, callback)
$(selector).method({duration, easing, complete})
```

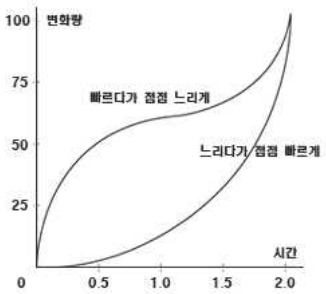


그림 5-1 easing 속성

duration : 애니메이션 진행 시간, 밀리 초 단위의 숫자
또는 slow(600), normal(400), fast(200) 문자열

callback : 애니메이션을 완료한 후에 실행할 함수

easing : easing은 애니메이션 움직임 패턴을 지정하는 속성으로 시간에 따른 애니메이션의 변화와 속도 패턴의 명칭이 미리 정의되어 있다. 별도의 추가적인 플러그인을 사용하지 않을 때에는 linear, swing 두 가지 문자열만 지정할 수 있다.

jQuery Easing Plugin 참고사이트 : http://easings.net

jQuery Easing Plugin 다운로드 : http://gsgd.co.uk/sandbox/jquery/easing/

▶ 예제 4-1 jQuery 효과(Effect) 메소드

- jQueryStudyCh04/jquery04_01.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>jQuery 효과(Effect) 메소드</title>
<style type="text/css">
  #container {
    width: 600px;
    height: 480px;
    border: 1px dotted #EAEAEA;
  }
  img {
    width: 600px;
    height: 480px;
  }
</style>
<!-- <script src="js/jquery-1.12.4.min.js"></script> -->
<script src="js/jquery-3.3.1.min.js"></script>
<script src="js/jquery.easing.1.3.min.js"></script>
<script>

  /* ◈ jQuery 효과(Effect) 메소드 ◈
   *
   * show() : 지정한 문서 객체를 점점 선명하게 하면서 크게 표시한다.
   * hide() : 지정한 문서 객체를 점점 투명하게 하면서 작게 숨긴다.
   * toggle() : show()와 hide()를 번갈아 실행한다.
   * slideDown() : 지정한 문서 객체를 미끄러지듯이 크게 표시한다.
   * slideUp() : 지정한 문서 객체를 미끄러지듯이 작게 숨긴다.
   * slideToggle() : slideDown()과 slideUp()을 번갈아 실행한다.
   * slide로 시작하는 메소드는 문서 객체의 width와 height 속성을
   * 지정했을 때와 지정하지 않았을 때 동작이 달라진다. 문서 객체에 크기를
   * 지정했을 때 slideDown() 메소드는 아래로 펼쳐지고 slideUp()
   * 메소드는 위쪽으로 접혀지는 형태로 동작하게 된다.
   * fadeIn() : 지정한 문서 객체를 투명한 상태에서 선명하게 표시한다.
   * fadeOut() : 지정한 문서 객체를 선명한 상태에서 투명하게 숨긴다.
   * fadeTo() : 지정한 문서 객체를 지정한 투명도 까지 투명도를 조절한다.
   * fadeToggle() : fadeIn(), fadeOut()을 번갈아 실행한다.
   *
   *  ◈ jQuery 효과(Effect) 메소드 사용법 ◈
   *
   * $(selector).method()
   * $(selector).method(duration)
```

```
 * $(selector).method(duration, callback)
 * $(selector).method(duration, easing, callback)
 * $(selector).method({duration, easing, complete})
 *
 * duration : 애니메이션 진행 시간, 밀리 초 단위의 숫자
 *                  또는 slow(600), normal(400), fast(200) 문자열
 * callback : 애니메이션을 완료한 후에 실행할 함수
 * easing : easing(애니메이션 움직임 패턴 - 시간에 따른 속성 값의 변화 속도)
 *       속성을 지정, 별도에 플러그인을 사용하지 않을 때 linear, swing
 *       두 가지 문자열만 지정할 수 있다.
 *
 * jQuery Easing Plugin 참고사이트 : http://easings.net
 * jQuery Easing Plugin 다운로드 : http://gsgd.co.uk/sandbox/jquery/easing/
 **/
$(function() {

   $("#btn1").click(function() {
      /* 선택한 이미지를 보이지 않은 상태에서 점점 선명하고 크게
       * 좌 상단에서 부터 이미지 크기만큼 펼쳐 표시한다.
       **/
      $("#img").show(1000, "swing");
   });

   $("#btn2").click(function() {
      // 선택한 이미지를 보이는 상태에서 점점 투명하고 작게 좌 상단으로 숨긴다.
      $("#img").hide(1000);
   });

   $("#btn3").click(function() {
      /* 선택한 이미지를 보이지 않은 상태에서 점점 크게 좌 상단에서 부터 이미지
       * 크기만큼 펼쳐 표시한다. 위의 css에서 img 태그의 주석을 풀고 실행하면
       * 이미지가 미끄러지듯이 아래쪽으로 펼쳐진다.
       **/
      $("#img").slideDown("slow");
   });

   $("#btn4").click(function() {
      /* 선택한 이미지를 보이는 상태에서 점점 작게 좌 상단으로 숨긴다. 위의 css에서
       * img 태그의 주석을 풀고 실행하면 이미지가 접히듯이 위쪽으로 사라진다.
       **/
      $("#img").slideUp("slow");
   });

   $("#btn5").click(function() {
      // 선택한 이미지를 보이지 않는 상태에서 점점 선명하게 표시한다.
```

```
    $("#img").fadeIn(1000);
});

$("#btn6").click(function() {
    // 선택한 이미지를 보이는 상태에서 점점 투명하게 숨긴다.
    $("#img").fadeOut(1000);
});

$("#btn7").click(function() {
    /* 한 번 클릭할 때마다 show() 메소드와 hide() 메소드를 번갈아 실행한다.
     * 객체가 화면에 보이는 상태면 hide() 메소드가 호출되고 보이지 않는 상태면
     * show() 메소드가 호출된다.
     **/
    $("#img").toggle(1000, function() {
        /* #btn7 요소의 click 이벤트에 연결된 핸들러를 호출한다.
         * 이벤트 핸들러 안에서 자기 자신을 호출하는 것이므로
         * 이미지가 작게 사라졌다 펼쳐지며 나타났다를 무한 반복한다.
         * jQuery의 이벤트 등록 메소드는 $("#btn7").click(function() { });와
         * 같이 함수를 인수로 지정하여 호출하면 인수로 지정한 함수를 이벤트
         * 핸들러로 등록하게 되며 아래와 같이 인수 없이 호출하면 이벤트에
         * 등록된 핸들러를 호출하는 역할을 한다. 아래는 click 이벤트를 강제로
         * 발생시켜 click 이벤트에 등록된 이벤트 핸들러를 호출하게 된다.
         **/
        $("#btn7").click();
    });
});

$("#btn8").click(function() {
    /* 한 번 클릭할 때 마다 slideUp() 메소드와 slideDown() 메소드를
     * 번갈아 실행한다. 객체가 화면에 보이는 상태면 slideDown() 메소드가
     * 호출되고 보이지 않는 상태면 slideUp() 메소드가 호출된다.
     **/
    $("#img").slideToggle(1000);
});

$("#btn9").click(function() {
    // 속도와 투명도를 인수로 지정한다.
    $("#img").fadeTo("slow", 0.5);
});

$("#btn10").click(function() {
    /* 기본 애니메이션 메소드의 인수로 객체를 지정하는 경우
     * 한 번 클릭할 때 마다 fadeIn() 메소드와 fadeOut() 메소드를
     * 번갈아 실행한다. 객체가 화면에 보이는 상태면 fadeOut() 메소드가
     * 호출되고 보이지 않는 상태면 fadeIn() 메소드가 호출된다.
```

```
        **/
        $("#img").fadeToggle({
          duration: 3000,
          easing: "easeInQuart",
          complete: function() {

            /* jQuery의 trigger() 메소드를 사용해 아래와 같이 #btn10
             * 요소의 click 이벤트를 강제로 발생 시킬 수 있다. 이벤트 핸들러
             * 안에서 자기 자신을 호출하는 것이므로 이미지가 투명해졌다
             * 선명해졌다를 무한 반복한다.
             **/
            $("#btn10").trigger("click");
          }
        });
      });
    });
</script>
</head>
<body>
  <div id="container">
    <img id="img" src="images/myungjun06.jpg" />
  </div>
  <div id="buttons">
    <button id="btn1">show</button>
    <button id="btn2">hide</button>
    <button id="btn3">slideDown</button>
    <button id="btn4">slideUp</button>
    <button id="btn5">fadeIn</button>
    <button id="btn6">fadeOut</button><br/>
    <button id="btn7">toggle</button>
    <button id="btn8">slideToggle</button>
    <button id="btn9">fadeTo</button>
    <button id="btn10">fadeToggle</button>
  </div>
</body>
</html>
```

▶ 예제 4-2 fadeIn(), fadeOut() 메소드를 이용한 모달 이미지 뷰어

- jQueryStudyCh04/jquery04_02.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
```

```
<title>fadeIn(), fadeOut() 메소드를 이용한 모달 이미지 뷰어</title>
<style>
  /*  ul 요소의 넓이 지정 및 중앙 배치, 마커제거 */
  ul {
    width: 700px;
    margin: auto;
    list-style: none;
  }

  /* li 요소를 가로 배치 */
  ul li { float: left; }

  /* 이미지에 대한 테두리 제거 및 여백 지정 */
  ul li img {
    border: 0;
    margin: 10px;
  }

  /* 큰 이미지가 화면에 표시되면 이미지 뒤로 반투명을 표현할 배경 레이아웃 - div */
  div#backgroundLayer {

    /* 배경에 사용할 레이어(div)를 화면에서 숨긴다. */
    display: none;

    /* 배경에 사용할 레이어(div)를 body 크기로 설정한다. */
    position: fixed;
    left: 0; top: 0;
    height: 100%; width: 100%;
    background: black;

    /* 배경 레이어(div)가 화면에 보일 때 반투명이 적용된다. */
    opacity: 0.60;
  }

  /* 이미지를 표시할 이미지 레이아웃 - div */
  #modalLayer {

    /*  큰 이미지를 표시할 div 요소를 화면에서 숨긴다.
     * display: none은 화면에 보이지 않으며 자리도 차지하지 않는다.
     * visibility: hidden은 화면에 보이지 않지만 자리는 차지한다.
     **/
    display: none;

    /* 큰 이미지를 표시할 div 요소를 화면 중앙에 배치 */
    position: fixed;
```

```
    top: 50%; left: 50%;
    margin-left: -325px; margin-top: -244px;
  }
</style>
<script src="js/jquery-3.3.1.min.js"></script>
<script>
  $(function() {
    $("a.modal").click(function(e) {

      // 배경 레이어를 CSS에서 지정한 투명도로 표시한다.
      $("#backgroundLayer").fadeIn(300);

      // CSS에서 불투명도 지정없이 같은 효과를 줄 수 있다.
      //$("#backgroundLayer").fadeTo(300, 0.6);

      // 이미지 레이어를 화면에 표시한다.
      $("#modalLayer").fadeIn(200);

      // a 요소의 href 속성을 <img>태그에 설정하고 이미지 레이어에 출력한다.
      $("#modalLayer").html("<img src='" + $(this).attr("href") + "' />");

      /* 원본 이미지의 크기가 모두 다르다면 이미지를 중앙에 배치하기 위해서
       * 아래와 같이 실행시에 이미지의 출력 위치를 조정해 줘야 한다.
       * 크롬에서는 첫 번째 클릭시 이미지의 크기가 정해지지 않아 크기가 0이
       * 되어 화면 중앙에 출력되지 않고 두 번째 클릭시 화면 중앙에 출력된다.
       **/
      //let width = $("#modalLayer > img").outerWidth();
      //let height = $("#modalLayer > img").outerHeight();
      //$("#modalLayer").css("margin-left", (-width / 2));
      //$("#modalLayer").css("margin-top", (-height / 2));
      //console.log(width + ", " + height)

      /* a 요소의 기본 이벤트 진행을 취소한다.
       * 아래와 같이 두 가지 방법으로 기본 이벤트 진행을 취소할 수 있다.
       **/
      //return false;
      e.preventDefault();
    });

    // 화면에 표시된 큰 이미지를 클릭한 경우
    $("#modalLayer").click(function() {

      // 이미지 레이어를 숨긴다.
      $(this).fadeOut(200);
```

```
        // 배경 레이어를 숨긴다.
        $("#backgroundLayer").fadeOut(300);
      });
    });
</script>
</head>
<body>
  <!--
    a 요소가 클릭되면 큰 이미지가 화면에 표시될 영역 - 기본적으로 숨겨진 상태
   -->
  <div id='backgroundLayer'></div>
  <div id='modalLayer'></div>
  <ul>
    <li>
      <a href="images/photo1.jpg" class="modal">
      <img src="images/photo1_thum.jpg" alt="샹드리아" /></a></li>
    <li>
      <a href="images/photo2.jpg" class="modal">
      <img src="images/photo2_thum.jpg" alt="장미" /></a></li>
    <li>
      <a href="images/photo3.jpg" class="modal">
      <img src="images/photo3_thum.jpg" alt="바다" /></a></li>
    <li>
      <a href="images/photo4.jpg" class="modal">
      <img src="images/photo4_thum.jpg" alt="문" /></a></li>
    <li>
      <a href="images/photo5.jpg" class="modal">
      <img src="images/photo5_thum.jpg" alt="바다" /></a></li>
    <li>
      <a href="images/photo6.jpg" class="modal">
      <img src="images/photo6_thum.jpg" alt="꽃" /></a></li>
    <li>
      <a href="images/photo7.jpg" class="modal">
      <img src="images/photo7_thum.jpg" alt="하늘" /></a></li>
    <li>
      <a href="images/photo8.jpg" class="modal">
      <img src="images/photo8_thum.jpg" alt="건물" /></a></li>
  </ul>
</body>
</html>
```

▶ 예제 4-3 사용자 지정 애니메이션 메소드

- jQueryStudyCh04/jquery04_03.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>사용자 지정 애니메이션 메소드</title>
<style type="text/css">
  div {
    border: 1px solid red;
  }
  #img {
    position: relative;
  }
</style>
<script src="js/jquery-3.3.1.min.js"></script>
<script src="js/jquery.easing.1.3.min.js"></script>
<script>

  /* ◈ 사용자 지정 애니메이션 메소드 ◈
   * $(selector).amimate(object)
   * $(selector).amimate(object, duration)
   * $(selector).amimate(object, duration, callback)
   * $(selector).amimate(object, duration, easing)
   * $(selector).amimate(object, duration, easing, callback)

   * object : 요소의 움직임을 만들 수 있는 위치나 불투명도 같은 css 속성을
   *       객체 형태로 지정한다. opacity, width, height, top,
   *       left, right, bottom, margin, padding 등을 지정한다.
   *       각 속성에는 단위를 지정해야 하지만 생략하면 px가 적용된다.
   * duration : 애니메이션 진행 시간, 밀리 초 단위의 숫자
   *       또는 slow(600), normal(400), fast(200) 문자열
   * callback : 애니메이션을 완료한 후에 실행할 함수
   * easing : easing(애니메이션 움직임 패턴 - 시간에 따른 속성 값의 변화 속도)
   *       속성을 지정, 별도에 플러그인을 사용하지 않을 때 linear, swing
   *       두 가지 문자열만 지정할 수 있다.
   *
   * jQuery Easing Plugin 참고사이트 : http://easings.net
   * jQuery Easing Plugin 다운로드 : http://gsgd.co.uk/sandbox/jquery/easing/
   **/
  $(function() {

    $("#btn1").click(function() {

      $("#img").animate({ left: "-=50" }, "normal", function() {

        // 속성 값이 120px 형태의 문자열을 유효한 숫자만 숫치 데이터로 변환
```

```
            let left = parseInt($(this).css("left"));

            if(left <= 0) {
                $(this).animate(
                    {left: 0},
                    500, "easeOutBounce");
            }
        });
    });

    $("#btn2").click(function() {

        $("#img").animate({ left: "+=100" }, "normal", function() {

            // 속성 값이 120px 형태의 문자열을 유효한 숫자만 숫치 데이터로 변환
            let parentWidth = parseInt($(this).parent().css("width"));
            console.log("parentWidth : " +$(this).parent().css("width"));
            let width = parseInt($(this).width());
            let left = parseInt($(this).css("left"));

            if((width + left) >= parentWidth) {
                $(this).animate(
                    {left: parentWidth - width},
                    500, "easeOutBounce");
            }
        });
    });
  });
</script>
</head>
<body>
  <div><img id="img" src="images/html5_game.jpg" /></div>
  <button type="button" id="btn1">◀◀◀</button>
  <button id="btn2">▶▶▶</button>
</body>
</html>
```

▶ 예제 4-4 이미지 슬라이딩 배너

- jQueryStudyCh04/jquery04_04.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
```

```
<title>이미지 슬라이딩 배너</title>
<style>
  * {
    margin: 0px 0px;
    padding: 0px 0px;
  }
  body {
    width: 100%;
    height: 100%;
  }
  #imgs {
    margin: 0px auto;
    width: 650px;
    height: 500px;
    border: 1px solid red;
    overflow: hidden;
  }
  ul { list-style: none; }
  #imgs ul {
    width: 100%; height: 488px; position: relative;
  }
  #imgs ul li {
    width: 650px; height: 488px; position: absolute;
  }
  #imgs .visual_01 {
    left: 0%;
    background: url('images/photo1.jpg') 0 0 no-repeat;
  }
  #imgs .visual_02 {
    left: 100%;
    background: url('images/photo2.jpg') 0 0 no-repeat;
  }
  #imgs .visual_03 {
    left: 200%;
    background: url('images/photo3.jpg') 0 0 no-repeat;
  }
  #btnList li {
    background: url('images/btnVisual.png') 0px -16px no-repeat;
    width: 14px;
    height: 15px;
    float: left;
    left: 50%;
    top: -470px;
    position: relative;
    margin: 0px 5px;
```

```
      cursor: pointer;
    }
    #btnList li.on { background-position: 0 0; }
    #btnList li > a{ display: block; text-indent: -9999px; }
</style>
<script src="js/jquery-3.3.1.min.js"></script>
<script>
$(function(){
      let buttons = $('#btnList > li');
      let imgs = $('#imgs > ul > li');
      let current = 0;

      buttons.click(function(){
        let btnLi = $(this);
        let i = btnLi.index();

        buttons.removeClass('on');
        btnLi.addClass('on');

        move(i);
        return false;
      });

      function move(i){
        // 현재 진행 중인 애니메이션을 즉시 멈추고 다음 애니메이션을 시작한다.
      imgs.eq(current).css('left', 0).stop().animate({left:'-100%'});
        imgs.eq(i).css('left', '100%').stop().animate({left:0});

        current = i;
      }

      setInterval(function(){
        let n = current + 1;
        if(n == buttons.length) n = 0;
        buttons.eq(n).trigger('click');
      }, 7000);
});
</script>
</head>
<body>
  <div id="wrap">
    <div id="imgs">
      <ul>
        <li class="visual_01"></li>
        <li class="visual_02"></li>
```

```
        <li class="visual_03"></li>
      </ul>
    </div>
    <ul id="btnList">
      <li class="on"><a href="#">이미지1</a></li>
      <li><a href="#">이미지2</a></li>
      <li><a href="#">이미지3</a></li>
    </ul>
  </div>
</body>
</html>
```

5. Ajax(Asynchronous Javascript and XML)

Ajax(Asynchronous JavaScript and XML)란 비동기 방식의 JavaScript와 XML을 의미한다. Ajax란 이름만 보면 자바스크립트와 XML만 사용해 구현하는 것이 Ajax의 전부일 것 같지만 실제로는 보다 많은 기술들이 복합적으로 사용되어 Ajax를 구현하게 된다.
Ajax에서 서버와 클라이언트 간의 요청과 응답에 대한 처리는 웹 브라우저가 직접 담당하는 것이 아니라 XMLHttpRequest 객체를 사용한다. 그렇기 때문에 Ajax에서 XMLHttpRequest 객체는 서버와 비동기 통신을 위한 핵심 객체이다. 예전에 JavaScript Core를 사용해 Ajax를 구현할 때는 XMLHttpRequest 객체를 직접 사용해 Ajax를 구현했지만 요즘에는 jQuery를 사용하기 때문에 XMLHttpRequest 객체를 직접 사용하는 경우는 거의 없고 주로 jQuery가 제공하는 Ajax 지원 메소드를 통해 Ajax를 구현한다.

▶ 동기 방식과 비동기 방식

웹브라우저가 서버에 요청을 보내고 응답이 올 때까지 다른 부분은 할 수 없고 무한정 기다리면서 응답이 도착했을 때 현재 페이지 전체를 응답 데이터로 갱신하는 것이 동기 방식이다. 이에 반해 현재 페이지의 다른 부분을 처리하면서 특정 요청에 대한 응답이 도착하면 이벤트를 발생시켜 웹 페이지의 일부분만 갱신 하는 것이 비동기 방식이다. 다시 말하자면 한 번의 요청으로 전체 페이지를 갱신하는 것이 동기 방식이라면 웹 페이지에서 필요한 특정 부분만 갱신하는 것을 비동기 방식이라고 한다.

- 동기 방식

1. 웹브라우저가 웹 서버로의 요청을 담당한다.
2. 웹 서버는 서버 사이드 기술(jsp/php/asp 등)을 사용해 요청을 처리한 후
   그 결과를 HTML 문서로 생성하여 웹브라우저로 응답 데이터를 전송한다.
3. 웹브라우저는 서버로부터 받은 응답 데이터 즉 HTML 문서를 해석해 화면에 표시 한다.
   웹브라우저가 웹 서버에 요청하면 서버에서는 요청을 처리한 결과를 HTML 문서로 생성하여
   웹브라우저로 전송하게 되는데 이 때 사용자 입장에서 페이지 이동이 발생하게 된다.

- 비동기(Ajax) 방식

1. 사용자의 요청이 발생하면(댓글 추가 요청 등) 자바스크립트의 DOM 기술을 사용해
   요청 정보를 구하고 XMLHttpRequest 객체를 통해 웹 서버에 요청한다.
2. 웹 서버는 XMLHttpRequest 객체로 부터 받은 요청을 처리한 후 그 결과를 XML, HTML, Text, CSV(Comma Separated Value), JSON(JavaScript Object Notation) 등의 다양한 데이터 포맷으로 응답 데이터를 생성해 XMLHttpRequest 객체에 전송한다.
3. 서버로부터 응답을 받은 XMLHttpRequest 객체는 응답 데이터의 도착을 알리기 위해 이벤트를 발생시키고 이 이벤트에 연결된 콜백 함수가 실행되어 응답 데이터가 화면에 반영된다.
4. 앞에서 XMLHttpRequest 객체가 언급되고 있지만 우리는 jQuery를 사용해 Ajax를 구현할 것이므로 지금 당장은 크게 신경 쓸 필요는 없다. 다만 jQuery 내부에서 XMLHttpRequest 객체가 사용된다는 것과 jQuery가 제공하는 Ajax 지원 메소드 사용법을 익히면 된다.

5.1 jQuery로 Ajax관련 메소드

jQuery는 JavaScript Core를 이용해 Ajax를 구현할 때 필요한 여러 단계의 작업을 내부적으로 처리해 주는 Ajax 작업을 위한 다양한 메소드를 제공하고 있다.
우리는 jQuery가 제공하는 Ajax 지원 메소드를 호출하기만 하면 jQuery 내부에서 Ajax 통신에 대한 여러 가지 작업이 이루어지며 클라이언트와 서버간의 통신이 완료된 후 jQuery가 서버에 요청하여 받은 결과 값을 우리가 등록한 콜백 함수에 넣어 준다. 우리는 jQuery가 콜백 함수를 통해 넣어 주는 데이터를 받아서 화면에 출력하는 방법에 대한 부분만 구현하면 된다.

▶ ajax()

Ajax 통합 지원 메소드로 POST, GET 방식 요청 모두를 지원하며 HTML, XML, JSON, Text 형식의 데이터로 응답 받을 수 있다. 아래의 get(), post(), getJSON() 메소드의 기능을 모두 수행할 수 있는 메소드 이다.
아래는 ajax() 메소드의 사용법이며 optionsObject는 JSON 형태의 옵션을 의미한다.

- **$.ajax(optionsObject);**
- **$.ajax(url, optionsObject);**

구분	옵 션	설 명
기본 옵션	url	데이터를 전송할 페이지 지정, 기본 값은 현재 페이지
	type	요청 전송방식(GET, POST)을 지정, 기본 값은 GET
	dataType	응답 받을 데이터 형식(HTML, XML, JSON, JSONP, TEXT)을 지정
	data	서버로 전송할 요청 데이터
구분	옵 션	설 명
콜백 함수	beforeSend	요청을 전송하기 전에 실행할 콜백 함수를 지정 **beforeSend: function(xhrJ);**
	complete	Ajax 통신이 완료 되었을 때 실행할 콜백 함수를 지정 **complete: function(xhr, textStatus);**
	success	Ajax 통신이 정상적으로 완료 되었을 때 실행할 콜백 함수를 지정 **success: function(data, textStatus, xhr);**
	error	Ajax 통신이 실패하면 실행할 콜백 함수를 지정 **error: function(xhr, textStatus, errorThrown);**
기타 옵션	async	Ajax 통신을 동기, 비동기로 할지 지정, 기본 값은 비동기 방식(true)
	cache	요청한 페이지를 캐시로 저장할지 지정, 기본 값은 true;
	contentType	서버로 전송할 데이터의 Content-Type 지정 기본 값은 application/x-www-form-urlencoded
기타 옵션	jsonp	JSONP(JSON with Padding 또는 JSON-P)에 대한 요청을 할 때 사용
	dataFilter	응답 데이터를 필터링 하는 함수 지정

	timeout	Ajax 통신 타임아웃 시간 지정, 밀리 초로 지정
	username	HTTP 액세스를 할 때 사용자 인증이 필요한 경우에 지정

$.ajax() 메소드의 옵션

▶ get()

GET 방식 요청을 위한 Ajax 지원 메소드 이다.

- $.get(url, function(responseData, textStatus, xhr));
- $.get(url, requestData, function(responseData, textStatus, xhr));
- $.get(url, requestData, function(responseData, textStatus, xhr), dataType);

▶ post()

POST 방식 요청을 위한 Ajax 지원 메소드 이다.

- $.post(url, function(responseData, textStatus, xhr));
- $.post(url, requestData, function(responseData, textStatus, xhr));
- $.post(url, requestData, function(responseData, textStatus, xhr), dataType);

▶ $(Selector).load()

Ajax 작업이 완료된 후 Selector에 지정한 요소에 응답 받은 데이터를 출력해 주는 메소드 이다. 주로 Text나 HTML 형식의 데이터를 응답 받아 특정 요소에 추가할 때 사용한다. 다시 말해 지정한 url로 데이터를 전송하고 외부 콘텐츠(Tetxt, HTML)를 가져와 현재 페이지의 일부분으로 사용할 때 load() 메소드를 사용할 수 있다.

- $(Selector).load(url);
- $(Selector).load(url, requestData);
- $(Selector).load(url, requestData, function(responseData, textStatus) { });

▶ getJSON()

GET 방식으로 요청하고 JSON 형식의 데이터로 응답 받을 수 있는 메소드 이다.

- $.getJSON(url, function(responseData, textStatus, xhr));
- $.getJSON(url, requestData, function(responseData, textStatus, xhr));

▶ getScript()

Ajax를 이용해 외부 자바스크립트를 읽어 올 때 사용되는 메소드 이다.

- $.getScript("scriptURL.js");

▶ 예제 5-1 ajax() 메소드(POST 방식 요청과 JSON 응답)

- webapp/jquery05_01.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>$.ajax() 메소드(POST 방식 요청과 JSON 응답)</title>
<script src="js/jquery-3.3.1.min.js"></script>
<script>
  $(function() {
    $("#form1").on("submit", function(e) {

      if($("#id").val().length <= 0 || $("#pass").val().length <= 0) {
        alert("id 또는 비밀번호가 입력되지 않았습니다.");
        return false;
      }

      /* 폼 양식에 입력된 데이터를 다음과 같은 파라미터 문자열(쿼리스트링)로 읽어온다.
       * "id=admin&pass=1234"
       **/
      let param = $(this).serialize();

      /* jQuery의 Ajax 통합지원 메소드
       * type은 생략할 수 있으며 생략되면 기본 값은 GET 이다.
       * dataType도 생략할 수 있으며 기본 값은 text 이다.
       * 서버로 보내는 데이터가 없다면 data 속성도 생략할 수 있다.
       **/
      $.ajax({
        url: "ajax/ajaxLoginJson.jsp",
        type: "POST",
        data: param,
        dataType: "json",
        /* 정상적으로 통신이 완료(응답 코드가 200일 때)되고 jQuery가
         * 응답 데이터를 dataType에 지정한 문서 형식으로 제대로 해석(parsing)
         * 했을 때 실행되는 콜백 함수
         **/
        success: function(resData, statusText, xhr) {

          /* 서버에서 JSON 형식의 문자열로 응답되었으므로 자바스크립트에서는
           * 객체가 되므로 객체 접근 방식으로 데이터에 접근할 수 있다.
```

```
            *
            * 서버에서 데이터를 JSON 형식의 문자열로 직렬화 할 경우에
            * 아래에서 설명하는 것과 같이 작성되어야 제대로 해석할 수 있다.
            *
            * "{"속성 이름": "속성의 값"}"와 같이 안쪽의 속성 이름과 속성의 값을
            * 반드시 쌍 따옴표("")로 감싸서 작성해야 제대로 해석(parsing)할 수 있다.
            **/
            console.log("data : " + resData);
             console.log(resData.message);
            $("#result").html(resData.message + "<br/>" + resData.greeting);
        },
        /* 응답 코드가 200 이외이거나 jQuery가 응답 데이터를 파싱할 때
         * 에러가 발생하면 실행되는 함수
         **/
        error: function(xhr, statusText, error) {
            alert("error : " + statusText + ", " + xhr.status);
        }
    });

    // 폼이 submit 되는 기본 이벤트를 제거
    return false;
  });
});
</script>
</head>
<body>
  <form name="form1" id="form1">
    아이디 : <input type="text" name="id" id="id"/><br/>
    비밀번호 : <input type="password" name="pass" id="pass"/><br/>
    <input type="submit" value="로그인" />
  </form>
  <div id="result"></div>
</body>
</html>
```

- **webapp/ajax/ajaxLoginJson.jsp**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
 <%
  //request.setCharacterEncoding("UTF-8");

  //Ajax 통신으로 클라이언트가 서버에 요청한 데이터 즉 요청 파라미터를 읽어온다.
  String id = request.getParameter("id");
```

```
  String pass = request.getParameter("pass");
  String message = null;
  String greeting = null;

  System.out.println(id + ", " + pass);

  if(id.equals("admin") && pass.equals("1234")) {
    message = "로그인 성공";
    greeting = "안녕 하세요 " +  id + "님!";

  } else {
    message = "로그인 실패";

    if(! id.equals("admin")) {
      greeting = "아이디가 맞지 않습니다.";
    } else {
      greeting = "비밀번호가 맞지 않습니다.";
    }
  }

  /* 클라이언트(jQuery)에서 JSON 데이터로 파싱하기 위해서는 JSON 형식의
   * 문자열 데이터를 만들 때 아래와 같이 작성되어야 제대로 해석 할 수 있다.
   *
   * "{"속성 이름": "속성의 값"}"와 같이 안쪽의 속성 이름과 속성의 값을
   * 반드시 쌍 따옴표("")로 감싸서 작성해야 제대로 해석(parsing)할 수 있다.
   **/
  String result = "{"
      + "\"message\": \"" + message + "\", "
      + "\"greeting\": \""+ greeting + "\""
      + "}";

  out.println(result);
%>
<%-- <%= result %> --%>
```

▶ 예제 5-2 get() 메소드(HTML 응답)

- webapp/jquery05_02.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>$.get() 메소드(HTML 응답)</title>
<script src="js/jquery-3.3.1.min.js"></script>
```

```
<!-- <script src="js/jquery-1.12.4.min.js"></script> -->
<script>
  $(function() {
    $("#form1").on("submit", function(e) {

      if($("#id").val().length <= 0 || $("#pass").val().length <= 0) {
        alert("id 또는 비밀번호가 입력되지 않았습니다.");
        return false;
      }

      /* 폼 양식에 입력된 데이터를 다음과 같은 파라미터 문자열(쿼리스트링)로 읽어온다.
       * "id=admin&pass=1234"
       **/
      let param = $(this).serialize();

      // GET 방식으로 Ajax 요청을 보내는 메소드
      $.get(
          "ajax/ajaxLoginHtml.jsp",
          param,
          // 정상적으로 통신이 완료(응답 코드가 200일 때)되면 실행되는  콜백 함수
          function(responseData, statusText, xhr) {
            $("#result").html(responseData);
      });

      // 폼이 submit 되는 기본 이벤트를 제거
      return false;
    });
  });
</script>
</head>
<body>
  <form name="form1" id="form1">
    아이디 : <input type="text" name="id" id="id"/><br/>
    비밀번호 : <input type="password" name="pass" id="pass"/><br/>
    <input type="submit" value="로그인" />
  </form>
  <div id="result"></div>
</body>
</html>
```

- webapp/ajax/ajaxLoginHtml.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
```

```
<%
  //request.setCharacterEncoding("UTF-8");

  //Ajax 통신으로 클라이언트가 서버에 요청한 데이터 즉 요청 파라미터를 읽어온다.
  String id = request.getParameter("id");
  String pass = request.getParameter("pass");
  String message = null;

  System.out.println(id + ", " + pass);

  if(id.equals("admin") && pass.equals("1234")) {
    message = "<h2>로그인 성공</h2>" + "안녕 하세요 " +  id + "님!";

  } else {
    message = "<h2>로그인 실패</h2>";

    if(! id.equals("admin")) {
      message += "아이디가 맞지 않습니다.";
    } else {
      message += "비밀번호가 맞지 않습니다.";
    }
  }
%>
<%--
  응답 객체에 출력하면 Ajax 통신의 응답 데이터가 된다.
--%>
<%= message %>
```

▶ 예제 5-3 post() 메소드(XML 응답)

- webapp/jquery05_03.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>$.post() 메소드(XML 응답)</title>
<script src="js/jquery-3.3.1.min.js"></script>
<script>
  $(function() {
    $("#form1").on("submit", function(e) {

      if($("#id").val().length <= 0 || $("#pass").val().length <= 0) {
        alert("id 또는 비밀번호가 입력되지 않았습니다.");
        return false;
```

```
        }

        /* 폼 양식에 입력된 데이터를 다음과 같은 파라미터 문자열(쿼리스트링)로 읽어온다.
         * "id=admin&pass=1234"
         **/
        let param = $(this).serialize();

        // POST 방식으로 Ajax 요청을 보내는 메소드
        $.post("ajax/ajaxLoginXml.jsp",
          param,
          // 정상적으로 통신이 완료(응답 코드가 200일 때)되면 실행되는  콜백 함수
          function(responseData, statusText, xhr) {

            /* 서버로부터 들어오 응답 데이터가 xml 형식이므로 jQuery의
             * DOM관련 메소드를 이용해 아래와 같이 탐색 하거나 조작할 수 있다.
             **/
            let message = $(responseData).find("message").eq(0).text();
            $("#result").html(message);

            let greeting = $(responseData).find("greeting").eq(0).text();
            $("#result").html($("#result").html() + "<br/>" + greeting);

        });

        // 폼이 submit 되는 기본 이벤트를 제거
        return false;
      });
    });
</script>
</head>
<body>
  <form name="form1" id="form1">
    아이디 : <input type="text" name="id" id="id"/><br/>
    비밀번호 : <input type="password" name="pass" id="pass"/><br/>
    <input type="submit" value="로그인" />
  </form>
  <div id="result"></div>
</body>
</html>
```

- **webapp/ajax/ajaxLoginXml.jsp**

```
<?xml version="1.0" encoding="utf-8" ?>
<%@ page language="java" contentType="text/xml; charset=UTF-8"
```

```
    pageEncoding="UTF-8"%>
<%
  //request.setCharacterEncoding("UTF-8");

  // Ajax 통신으로 클라이언트가 서버에 요청한 데이터 즉 요청 파라미터를 읽어온다.
  String id = request.getParameter("id");
  String pass = request.getParameter("pass");
  String message = null;
  String greeting = null;

  System.out.println(id + ", " + pass);

  if(id.equals("admin") && pass.equals("1234")) {
    message = "로그인 성공";
    greeting = "안녕하세요 " +  id + "님!";

  } else {
    message = "로그인 실패 ";

    if(! id.equals("admin")) {
      greeting = "아이디가 맞지 않습니다.";
    } else {
      greeting = "비밀번호가 맞지 않습니다.";
    }
  }
%>
<%--
  응답 객체에 출력하면 Ajax 통신의 응답 데이터가 된다.
  아래는 xml 형식으로 응답 데이터를 작성하였다.
--%>
<result>
  <success>1</success>
  <greeting><%= greeting %></greeting>
  <message><%= message %></message>
</result>
```

▶ 예제 5-4 $(Selector).load() 메소드를 이용해 외부 문서 가져오기

- webapp/jquery05_04.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>load() 메소드를 이용해 외부 문서 가져오기</title>
```

```
<style>
</style>
<script src="js/jquery-3.3.1.min.js"></script>
<script>
  $(document).ready(function() {

    $("button").on("click", function(e) {

      $("#result > div").empty();

      /* load() 메소드를 사용하면 URL에 위치한 외부 문서에 요청을 보내고
       * 그 결과에 대한 응답 데이터를 jQuery로 선택한 문서 객체의 자식으로
       * 추가할 수 있다. 아래는 id가 result인 요소 바로 아래 자식인 div 요소에
       * load() 메서드의 첫 번째 인수로 지정한 URL의 HTML 문서가 추가된다.
       **/
      $("#result").load("ajax/load.jsp?id=jQuery");
    })
  });
</script>
</head>
<body>
  <button>jQuery 참고 도서</button><br/>
  <div id="result">
  </div>
</body>
</html>
```

- webapp/ajax/load.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
  String id = request.getParameter("id");
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>load() 메소드를 이용해 외부문서 가져오기</title>
</head>
<body>
<% if(id.equals("jQuery"))  { %>
  <h2>jQuery 참고도서</h2>
  <ul>
```

```
    <li>jQuery를 활용한 인터렉티브 웹</li>
    <li>사전처럼 바로 찾아 쓰는 jQuery</li>
    <li>jQuery Cookbook</li>
  </ul>
<%} else { %>
  <h2>자바스크립트 참고도서</h2>
  <ul>
    <li>자바스크립트 완벽가이드</li>
    <li>모던 웹을 위한 자바스크립트</li>
    <li>자바스크립트 마스터북</li>
  </ul>
<%} %>
</body>
</html>
```

▶ 예제 5-5 Gson라이브러리를 이용한 JSON 데이터 만들기 1

- webapp/jquery05_05.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Gson 라이브러리를 이용한 JSON 데이터 만들기</title>
<script src="js/jquery-3.3.1.min.js"></script>
<script>
  $(function() {
    $("#form1").on("submit", function(e) {

      if($("#id").val().length <= 0 || $("#pass").val().length <= 0) {
        alert("id 또는 비밀번호가 입력되지 않았습니다.");
        return false;
      }

      /* 폼 양식에 입력된 데이터를 다음과 같은 파라미터 문자열(쿼리스트링)로 읽어온다.
       * "id=admin&pass=1234"
       **/
      let param = $(this).serialize();

      /* $.ajax() 메소드를 이용해 POST 방식으로 요청하고 JSON 데이터 받기
       * type은 생략할 수 있으며 생략되면 기본 값은 GET 이다.
       * dataType도 생략할 수 있으며 기본 값은 text 이다.
       * 서버로 보내는 데이터가 없다면 data 속성도 생략할 수 있다.
       **/
      $.ajax({
```

```
            url: "ajax/ajaxLoginGson.jsp",
            type: "POST",
            data: param,
            dataType: "json",
            success: function(responseData, statusText, xhr) {

              /* 서버에서 응답 받은 데이터가 JSON 형식이면 responseData는
               * 자바스크립트 객체이므로 message 라는 속성이 없다면 undefined
               * 되어 아래의 if 문에서 false가 되고 message라는 속성은 존재하지만
               * 데이터가 공백 문자열("")인 경우에도 false가 된다.
               **/
              if(responseData.message) {
                $("#result").html("<h2>JSON 데이터 응답</h2>"
                    + responseData.message + "<br/>"
                    + responseData.greeting);

              // 응답 데이터가 JSON 형식이 아닌 경우
              } else {
                $("#result").html("<h2>배열 데이터 응답</h2>"
                    + responseData[0] + "<br/>"
                    + responseData[1]);
              }
            },
            error: function(xhr, statusText, error) {
              alert("error : " + statusText + ", " + xhr.status);
            }
          });

          // 폼이 submit 되는 기본 이벤트를 제거
          return false;
        });
      });
</script>
</head>
<body>
  <form name="form1" id="form1">
    아이디 : <input type="text" name="id" id="id"/><br/>
    비밀번호 : <input type="password" name="pass" id="pass"/><br/>
    <input type="submit" value="로그인" />
  </form>
  <div id="result"></div>
</body>
</html>
```

- webapp/ajax/ajaxLoginGson.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="com.google.gson.*, java.util.*" %>
 <%
  //request.setCharacterEncoding("UTF-8");
  String id = request.getParameter("id");
  String pass = request.getParameter("pass");
  String message = null;
  String greeting = null;

  System.out.println("파라미터 : " + id + ", " + pass);

  if(id.equals("admin") && pass.equals("1234")) {
    message = "로그인 성공";
    greeting = "안녕 하세요 " +  id + "님!";

  } else {
    message = "로그인 실패 ";

    if(! id.equals("admin")) {
      greeting = "아이디가 맞지 않습니다.";
    } else {
      greeting = "비밀번호가 맞지 않습니다.";
    }
  }

  /* gson 다운로드 : https://github.com/google/gson
   * 참고사이트 : http://emflant.tistory.com/47
   * http://1004lucifer.blogspot.kr/2015/04/javagson-gson-java-json.html
   * https://sites.google.com/site/gson/gson-user-guide
   **/
  Gson gson = new Gson();
  HashMap<String, String> map = new HashMap<String, String>();
  map.put("message", message);
  map.put("greeting", greeting);

  /* HashMap에 저장된 데이터를 아래와 같은 JSON 형식으로 직렬화 한다.
   * {"message": "로그인 성공", "greeting": "안녕하세요 admin님!"}
   **/
  String result = gson.toJson(map);

  /* 자바 배열을 아래와 같은 자바스크립트 배열 형식의 JSON 데이터로 직렬화 한다.
   * ["로그인 성공", "안녕하세요 admin님!"]
```

```
     **/
    String[] messages = new String[]{ message, greeting };
    String result1 = new Gson().toJson(messages);

    System.out.println("result : " + result);
    System.out.println("result1 : " + result1);

    /* 아래와 같이 자바 객체를 JSON 형식으로 직렬화한 데이터를 응답 스트림으로
     * 출력하면 클라이언트로 응답되어 자바스크립트에서 JSON 형식의 데이터를 받기
     * 때문에 별도의 처리 없이 바로 자바스크립트 객체로 받아 처리할 수 있다.
     **/
    out.println(result);
%>
<%-- <%= result %> --%>
```

▶ 예제 5-6 Gson 라이브러리를 이용한 JSON 데이터 만들기 2

- webapp/jquery05_06.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
    #searchDate, #searchId {
        width: 300px;
        height: 60px;
    }
    #searchId {
        display: none;
    }
    table {
        border: 1px solid blue;
        border-collapse: collapse;
        font-size: 12px;
    }
    table, h2 {
        width: 700px;
        margin: 10px auto;
    }
    h2 {
        text-align: center;
    }
    th {
        border: 1px solid blue;
```

```
    background: #EAEAFF;
    height: 20px;
    line-height: 20px;
    height: 16px;
  }
  th:nth-child(5n), th:nth-child(6n) {
    width: 150px;
  }
  td {
    border: 1px dotted blue;
    width: 100px;
    line-height: 16px;
  }
  tr:hover {
    background: #EAEAEA;
  }
  #resultNum, #requestDate {
    display: none;
  }
</style>
<title>Gson 라이브러리를 이용한 JSON 데이터 만들기 2</title>
<script src="js/jquery-3.3.1.min.js"></script>
<script>
  $(function() {
    $("#form1").on("submit", function(e) {

      let choice = $("input[type=radio]:checked").val();

      /* 폼 양식에 입력된 데이터를 다음과 같은 파라미터 문자열(쿼리스트링)로 읽어온다.
       * "searchOption=1&id=midas
       **/
      let param = $(this).serialize();
      let result;

      // 검색할 아이디가 입력되었는지 체크
      if(choice == 2) {
        if($("#id").val().length <= 0) {
          alert("검색할 회원의 아이디가 입력되지 않았습니다.");
          return false;
        }
      }

      // $.ajax() 메소드를 이용해 POST 방식으로 요청하고 JSON 데이터 받기
      $.ajax({
        url: "ajax/ajaxObjectGson.jsp",
```

```
type: "POST",
data: param,
dataType: "json",
success: function(data) {

  // 결과를 출력하기 전에 #result의 자식 요소를 모두 지운다.
  $("#result").empty();

  // 전체 회원 리스트라면 배열이므로 length 속성이 존재하므로 true
  if(data.length <= 0 && choice == 1) {
    alert("회원 리스트가 존재하지 않습니다.");
    return false;

  } else if(! data.id && choice == 2) {
    /* 입력한 회원 아이디가 존재하지 않으면 아이디
     * 입력 상자에서 아이디를 읽어와 경고 창을 띄운다.
     **/
    alert($("#id").val() + "은 존재하지 않는 아이디 입니다.");
    return false;
  }

  $("<h2>회원 리스트</h2>").appendTo("#result");
  $("<table id='memberTable'></table>").appendTo("#result");
  $("<tr><th>아이디</th><th>이름</th><th>비밀번호</th>"
    + "<th>나이</th><th>이메일</th><th>등록일</th></tr>")
    .appendTo("#memberTable");

  /* $(data).length은 객체의 개수를 반환 한다.
   * 콜백 함수의 파라미터로 넘어 오는 응답 데이터 data는 검색 옵션에
   * 따라서 회원 한 명의 정보 또는 여러 명의 정보를 JSON 형식으로
   * 담고 있다. 요청을 처리하는 서버에서 searchOption 파라미터 값이
   * 1일 때 전체 회원 리스트를 ArrayList에 담아 Gson 라이브러리를
   * 사용해 JSON 데이터로 변환하게 되는데 이 데이터를 응답 데이터로
   * 받으면 객체 배열 형태가 되지만 searchOption 파라미터 값이 2일
   * 때는 한 명의 회원 정보를 Member 객체에 담아 Gson 라이브러리를
   * 사용해 JSON 데이터로 변환하게 되는데 이 데이터를 응답 데이터로
   * 받으면 하나의 객체가 되기 때문에 length 속성의 값은 1이 된다.
   **/
  console.log("datalength: " + $(data).length);
  for(let i = 0; i < $(data).length; i++) {

    let regDate = new Date($(data).get(i).regDate);
    let strDate = regDate.getFullYear() + "년 "
        + (regDate.getMonth() + 1) + "월 "
        + regDate.getDate() + "일";
```

```
                $("<tr><td>" + $(data)[i].id + "</td>"
                  + "<td>" + $(data).get(i).name + "</td>"
                  + "<td>" + $(data)[i].pass + "</td>"
                  + "<td>" + $(data)[i].age + "</td>"
                  + "<td>" + $(data)[i].email + "</td>"
                  + "<td>" + strDate + "</td>"
                  + "</tr>").appendTo("#memberTable");
              }
            },
            error: function(xhr, statusText, error) {
              alert("error : " + statusText + ", " + xhr.status);
            }
          });

          // 기본 이벤트 제거
          return false;
        });

        // 라디오 버튼이 클릭되면 애니메이션 설정
        $("input[type=radio]").click(function() {
          //console.log($("input[type=radio]:checked").val());
          let val = $(this).val();

          /* 숨겨진 input 요소의 required 속성이 적용되어 있으면 폼이 전송될 때
           * 유효성 검사를 수행하기 때문에 숨겨진 input 요소의 required 속성은
           * 삭제하고 화면에 표시되는 input 요소의 required 속성만 적용한다.
           **/
          if(val == 1) {
            $("#searchId").slideUp(1000, function() {
              $(this).find("input").removeAttr("required");
              $(this).find("input").val("");
            });

          } else {
            $("#searchId").slideDown(1000, function() {
              /* 텍스트 입력 상자의 입력 속성을 required로 설정해 입력이
               * 없는 상태에서 검색하기 버튼이 클릭되면 submit 되지 않도록 함
               **/
              $(this).find("input").prop("required", true);
            });
          }
        });
      });
    </script>
```

```
</head>
<body>
  <form name="form1" id="form1">
    <div>
      <label>
        <input type="radio"
          name="searchOption" value="1" checked/>전체 회원리스트 
      </label>
      <label>
        <input type="radio" name="searchOption" value="2" />아이디로 검색
      </label>
    </div>
    <div id="searchId">
      아이디 : <input type="text" name="id" id="id"
        placeholder="검색할 아이디 입력" />
    </div>
    <input type="submit" value="검색하기" />
  </form>
  <div id="result"></div>
</body>
</html>
```

▶ 회원 한 명의 정보를 저장하는 VO(Value Object)

- src/com.jquery.ajax.ch05.Member

```
public class Member {

  private String id;
  private String name;
  private String pass;
  private int age;
  private String email;
  private Timestamp regDate;
  public String getId() {
    return id;
  }
  public void setId(String id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
```

```
  }
  public String getPass() {
    return pass;
  }
  public void setPass(String pass) {
    this.pass = pass;
  }
  public int getAge() {
    return age;
  }
  public void setAge(int age) {
    this.age = age;
  }
  public String getEmail() {
    return email;
  }
  public void setEmail(String email) {
    this.email = email;
  }
  public Timestamp getRegDate() {
    return regDate;
  }
  public void setRegDate(Timestamp regDate) {
    this.regDate = regDate;
  }
}
```

▶ Oracle과 MySQL 접속 정보를 저장하는 클래스

- src/com.jquery.ajax.ch05.DBConnectionInfo

```
public class DBConnectionInfo {
  public static final String ORACLE_URL = "jdbc:oracle:thin:@localhost:1521:xe";
  public static final String ORACLE_USER = "hr";
  public static final String ORACLE_PASS = "12345678";

  public static final String MYSQL_URL = "jdbc:mysql://localhost:3306/ajax";
  public static final String MYSQL_USER = "root";
  public static final String MYSQL_PASS = "12345678";
}
```

▶ DB 작업을 전담하는 DAO(Data Access Object) 클래스

- src/com.jquery.ajax.ch05.MemberDao

```
public class MemberDao {
```

```
private String url;
private String user;
private String pass;
private Connection conn;
private PreparedStatement pstmt;
private ResultSet rs;

public MemberDao(String dbName) {
  if(dbName.equals("oracle")) {
    this.url = DBConnectionInfo.ORACLE_URL;
    this.user = DBConnectionInfo.ORACLE_USER;
    this.pass = DBConnectionInfo.ORACLE_PASS;

    try {
      Class.forName("oracle.jdbc.driver.OracleDriver");

    } catch(ClassNotFoundException e) {
      System.out.println(e.getClass().getSimpleName());
      e.printStackTrace();
    }

  } else if(dbName.equals("mysql")) {
    this.url = DBConnectionInfo.MYSQL_URL;
    this.user = DBConnectionInfo.MYSQL_USER;
    this.pass = DBConnectionInfo.MYSQL_PASS;

    try {
      Class.forName("com.mysql.jdbc.Driver");

    } catch(ClassNotFoundException e) {
      System.out.println(e.getClass().getSimpleName());
      e.printStackTrace();
    }
  }
}

// DB에서 전체 회원 리스트를 읽어오는 메서드
public ArrayList<Member> getMemberList() {
  ArrayList<Member> memberList = null;
  String query = "SELECT * FROM ajax_member";
  try {
    memberList = new ArrayList<Member>();
    conn = DriverManager.getConnection(url, user, pass);
    pstmt = conn.prepareStatement(query);
```

```
            rs = pstmt.executeQuery();

            while(rs.next()) {
                Member m = new Member();
                m.setId(rs.getString("id"));
                m.setName(rs.getString("name"));
                m.setPass(rs.getString("pass"));
                m.setAge(rs.getInt("age"));
                m.setEmail(rs.getString("email"));
                m.setRegDate(rs.getTimestamp("reg_date"));

                memberList.add(m);
            }
            return memberList;
        } catch(SQLException e) {
            System.out.println(e.getClass().getSimpleName());
            e.printStackTrace();
        } finally {
            try {
                if(rs != null) rs.close();
                if(pstmt != null) pstmt.close();
                if(conn != null) conn.close();
            } catch(SQLException e) {}
        }
        return memberList;
    }

    // DB에서 id에 해당하는 회원 정보를 읽어오는 메서드
    public Member getMember(String id) {
        Member member = null;
        String query = "SELECT * FROM ajax_member WHERE id = ?";
        try {
            conn = DriverManager.getConnection(url, user, pass);
            pstmt = conn.prepareStatement(query);
            pstmt.setString(1, id);
            rs = pstmt.executeQuery();
            member = new Member();

            if(rs.next()) {
                member.setId(rs.getString("id"));
                member.setName(rs.getString("name"));
                member.setPass(rs.getString("pass"));
                member.setAge(rs.getInt("age"));
                member.setEmail(rs.getString("email"));
                member.setRegDate(rs.getTimestamp("reg_date"));
```

```
        }
      } catch(SQLException e) {
        System.out.println(e.getClass().getSimpleName());
        e.printStackTrace();
      } finally {
        try {
          if(rs != null) rs.close();
          if(pstmt != null) pstmt.close();
          if(conn != null) conn.close();
        } catch(SQLException e) {}
      }
      return member;
    }
}
```

▶ Ajax 요청을 받아 처리하고 JSON 데이터로 응답하는 JSP 페이지

- WebContent/ajax/ajaxObjectGson.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ page import="java.sql.*, java.util.*,com.jquery.ajax.ch05.*" %>
<%@ page import="com.google.gson.*, java.text.*, java.util.Date" %>
<%
  // 요청 파라미터를 읽어온다. 1=전체 회원 리스트, 2=아이디에 해당하는 회원 정보
  String searchOption = request.getParameter("searchOption");

  // Gson 생성자를 이용해 Gson 객체를 생성하는 경우
  Gson gson = new Gson();

  // DAO 객체 생성 - Oracle DBMS 사용
  MemberDao dao = new MemberDao("oracle");

  // DAO 객체 생성 - MySQL DBMS 사용
  //MemberDao dao = new MemberDao("mysql");
  String responseData = null;

  if(searchOption.equals("1")) { // 전체 회원리스트 요청 처리

    /* DB로부터 회원 리스트를 읽어와 Gson 라이브러리를 이용해 JSON 형식으로
     * 직렬화 한다. 아래와 같이 객체가 들어간 배열이나 ArrayList를 JSON 형식으로
     * 직렬화 하면 아래과 같은 자바스크립트 객체 배열 형태로 직열화 된다.
     *
```

```
        * [{"id": "midas", "name": "홍길동", "age": 25, ...}, {}, {}, ...]
        **/
        ArrayList<Member> memberList = dao.getMemberList();
        responseData = gson.toJson(memberList);

    } else if(searchOption.equals("2")) { // 아이디에 해당 하는 회원 정보 요청 처리
        String id = request.getParameter("id");
        Member member = dao.getMember(id);

        /* Member 객체를 직렬화 하면 아래와 같은 자바스크립트 객체가 된다.
        *
        * {"id": "midas", "name": "홍길동", "age": 25, ...}
        **/
        responseData = gson.toJson(member);
    }
    System.out.println(responseData);

    // OutStream에 출력하면 클라이언트로 응답 데이터가 전송된다.
    out.println(responseData);
%>
```