Ajax

Ajax에 대해서

- Asynchronous(비동기) JavaScript and XML의 약어
- xml를 이용하여 비동기로 통신을 처리하기 위한 기술
- XMLHttpRequest(XHR) 객체로 구현

Ajax 개체, XMLHttpRequest(XHR) 의 속성

속성	설명	읽기/쓰기
readyState	AJAX 개체의 상태를 나타내는 숫자. 처음 AJAX 개체를 생성하면 0이다. get() 메소드로 요청할 페이지 정보를 설정하면 1이 되고, send() 메소드로 요청을 보내면 2가 되고, 서버에서 응답이 오기 시작하면 3이 되고, 서버 응답이 완료되면 4가 된다.	읽기 전용
status	서버로부터 받은 응답의 상태를 나타내는 숫자. 정상적으로 응답을 받은 경우 200이고, 페이지를 찾지 못한 경우 404가 된다.	읽기 전용
statusText	서버로부터 받은 응답의 상태를 나타내는 문자열. 정상적으로 응답을 받으면 'OK'가 되고 파일을 찾지 못하면 'Not Found'가 된다.	읽기 전용
responseText	서버 응답 내용을 나타내는 문자열.	읽기 전용
responseXML	서버 응답 내용을 나타내는 XML 개체.	읽기 전용
onreadystatechange	readyState 속성이 바뀌었을 때 실행할 이벤트 핸들러를 지정한다.	읽기/쓰기

Ajax 개체, XMLHttpRequest(XHR) 의 메서드

메소드	설명
open()	open(method, url [, async]); AJAX 요청을 초기화하면서 요청 방식, 주소, 동기화 여부를 지정한다. method 인자는 http 요청 방식을 나타내며 "get" 또는 "post" 방식을 사용한다. url 인자는 요청할 페이지의 주소를 지정한다. 마지막으로 aysnc 인자는 비동기 통신 여부를 나타내며 true 또는 false로 지정한다. aysnc 인자를 지정하지 않으면 true를 기본값으로 사용한다.
send()	send(body); AJAX 요청을 보낸다. Body 인자에는 요청과 함께 서버로 보낼 내용을 지정한다.
abort()	abort() send() 메소드로 보낸 요청을 취소한다.

Ajax 개체, XMLHttpRequest(XHR) 의 메서드

메소드	설명
getAllResponseHeaders()	getAllResponseHeaders(); 응답을 받은 경우 응답의 모든 헤더 정보를 문자열로 돌려준다.
getResponseHeader()	getResponseHeader(header) 응답을 받은 경우 header 인자로 지정한 이름의 헤더 정보 값을 문자열로 돌려준다.
setRequestHeader()	setResonseHeader(header, value) 요청을 보내기 전에 header 헤더 정보의 값을 value로 설정한다.

자바스크립트로 Ajax 구현하기

```
<script type="text/javascript">
window.onload = function(){
    var xhr;
    if(window.XMLHttpRequest){
        xhr=new XMLHttpRequest();
    }else if(window.ActiveXObject){
        xhr=new ActiveXObject("Msxmi2.XMLHTTP");
    }else{
        throw new Error("Ajax가 지원하지 않는 브라우저입니다.");
    }
 }
</script>
```

서버로 요청 보내기

1. POST나 GET으로 HTTP 메소드를 명시한다.
2. 연결하려는 서버 측 자원의 URL을 제공한다.
3. XHR 인스턴스에 진행 과정의 전달 방법을 알린다.
4. POST 요청인 경우 본문 콘텐츠를 제공한다.

```
xhr.open('get', ‘resource');
xhr.send('');
```

- send() 메소드로 요청을 보내기 전에 open() 메소드를 실행
- AJAX에서 GET 방식으로 요청하기
  - ◉ open() 메소드의 첫 번째 인자를 'get'으로 지정하고 두 번째 인자는 URL을 지정한다.

진행 상황 추적하기

```
xhr.onreadystatechange = function(){

}
```

- XHR 인스턴스는 준비 핸들러를 통해 진행상황을 추적 onreadystatechange 프로퍼티에 함수 참조를 할당하여 설정
- 요청이 send() 메소드로 전송되면 이 함수는 요청이 다양한 상태 변화를 겪으면서 변할 때마다 호출된다.

진행 상황 추적하기

```
xhr.onreadystatechange = function(){
    if(xhr.readyState == 4){
        if(xhr.status >= 200 && xhr.status < 300){
        }
    }
}
```

- readyState로 현재 상태를 파악하여 완료된 상태이면 status로 요청의 성공여부를 확인한다.
- status가 200에서 299사이이면 성공한 것이다.

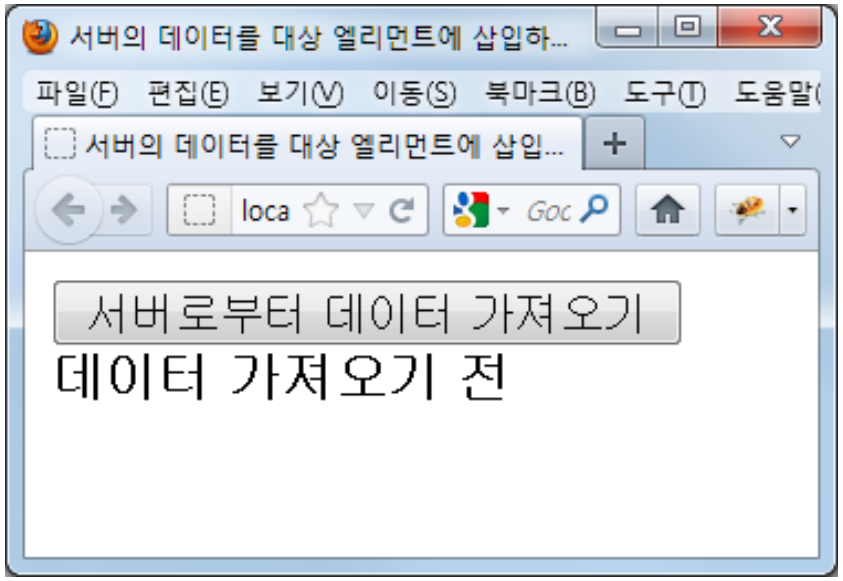
진행 상황 추적하기

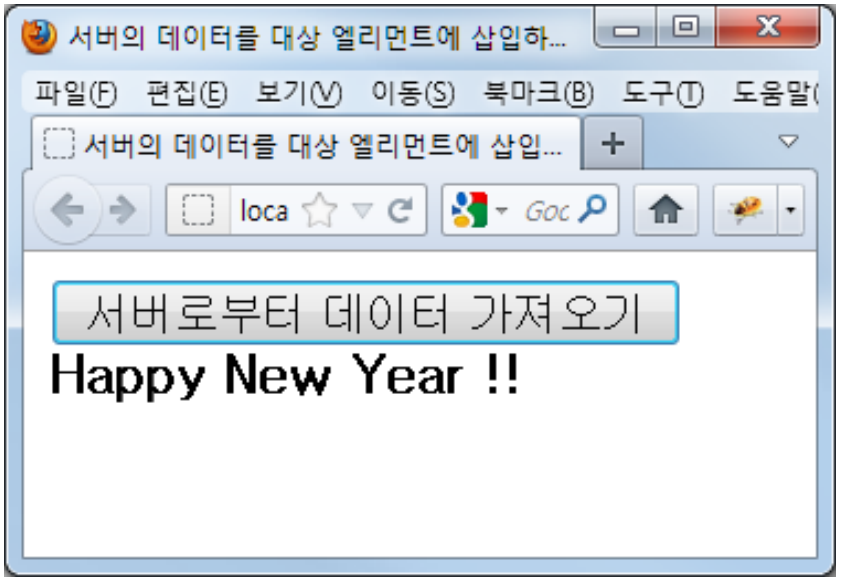
```
xhr.onreadystatechange = function(){
    if(xhr.readyState == 4){
        if(xhr.status >= 200 && xhr.status < 300){
            document.getElementById('container')
                        .innerHTML = xhr.responseText;
        }
    }
}
```

- 페이지가 로드할때 resource라는 자원을 이용해 서버에서 HTML 코드를 가져와서 id가 container인 엘리먼트의 콘텐츠로 설정한다.

서버의 데이터를 대상 엘리먼트에 삽입하기-load()

페이지를 동적으로 로드하기 위해서는 load() 메소드를 호출한다.





서버의 데이터를 대상 엘리먼트에 삽입하기- load()

```
load(url[, data][, success])
```

- url은 요청할 URL 주소이고, data는 요청할 때 서버에 보낼 자바스크립트 객체 맵이나 문자열 형식의 데이터이다.
- success는 요청이 성공했을 때 호출할 콜백 함수이다.
- load() 메소드는 완료된 응답을 일치하는 집합에 있는 엘리먼트의 콘텐츠로 삽입한다.

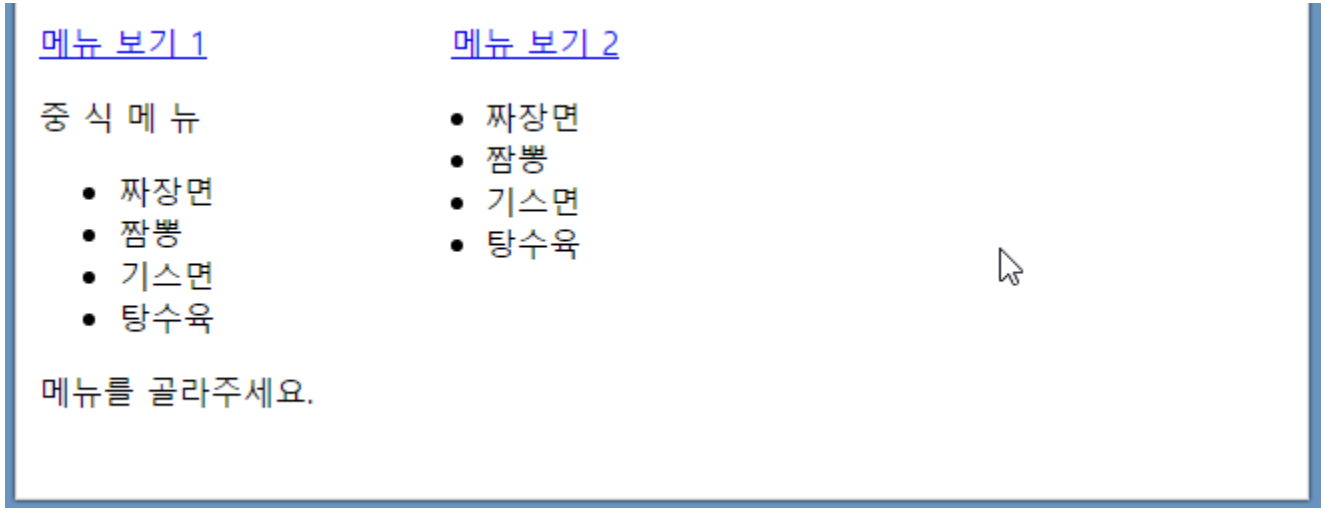
```
$("#container").load(“resource");
```

Ajax 요청 직접 제어하기

- jQuery에서는 $.ajax()라는 유틸리티 함수로 Ajax 요청을 설정 제어한다.

$.ajax(options)

- 요청의 생성 방법과 통보 받을 콜백을 제어하고자 전달된 options 를 사용하여 Ajax 요청을 전송한다. 반환값은 XHR 인스턴스이다.



Ajax 요청 직접 제어하기

- url (String)
  - 요청 URL
- type (String)
  - 사용할 HTTP 메서드, 일반적으로 POST나 GET을 사용한다. 생략하면 기본값으로 GET을 사용한다.
- data (Object)
  - 요청에 전달되는 프로퍼티를 가진 객체. GET 요청이면 데이터는 쿼리 문자열로 제공된다. POST 요청이면 데이터는 요청의 본문으로 제공된다. 두 경우 모두 $.ajax() 유틸리티 함수가 값의 인코딩을 처리한다.

Ajax 요청 직접 제어하기

- dataType (String)
  - 응답의 결과로 반환되는 데이터의 종류를 식별하는 키워드
    - xml - 응답 텍스트는 XML 문서로 파싱되며, 콜백에 결과로 생성된 XML DOM을 전달한다.
    - html - 응답 텍스트는 처리 과정 없이 콜백 함수로 전달된다. 반환된 HTML 코드에 있는 모든 <script> 블록이 평가된다.
    - json - 응답 텍스트는 JSON 문자열로 평가되며, 생성된 객체는 콜백에 전달된다.
    - jsonp - 원격 스크립트를 허용한다는 점을 제외하고는 json 과 유사하다. 원격 서버가 이와 같은 방식을 지원한다고 가정한다.
    - script - 응답 텍스트는 콜백에 전달된다. 응답은 모든 콜백의 호출보다 먼저 자바스크립트 구문으로 처리된다.
    - text - 응답 텍스트는 일반 텍스트다.

Ajax 요청 직접 제어하기

- timeout (Number)
  - Ajax 요청의 제한 시간을 밀리초 단위로 설정한다. 제한 시간 안에 요청이 완료되지 않으면 요청을 취소하거나, error 콜백이 정의되어 있다면 호출된다.
- global (Boolean)
  - true나 flase에 따라 전역 함수(global function)를 활성화하거나 비활성화한다. 전역 함수는 엘리먼트에 덧붙일 수 있으며 Ajax 호출 동안 다양한 위치나 조건에서 실행된다. 값이 생략되면 기본 값으로 전역 함수를 활성화한다.
- contentTypeString
  - 요청에 명시되는 콘텐츠 타입. 생략하면 'application/x-www-form-urlencoded'가 기본으로 설정되며, 이는 폼 전송이 기본으로 사용하는 타입과 동일하다.

Ajax 요청 직접 제어하기

- success (Function)
  - 응답이 성공 사태 코드를 반환하면 호출되는 함수. 응답 본문은 이 함수의 첫 번째 매개변수로 전달되며, dataType 프로퍼티에 명시한 형태로 구성된다. 두 번째 매개변수는 상태 값을 나타내는 문자열이며, 이번 경우에는 항상 'success' 다.
- error (Function)
  - 응답이 에러 상태 코드를 반환하면 호출되는 함수. 매개변수가 세 개 전달되는데, 각각 XHR 인스턴스, 상태 값이 항상 'error' 인 메시지 문자열, 선택사항으로 XHR 인스턴스가 반환한 예외 객체다.

Ajax 요청 직접 제어하기

- complete (Function)
  - 요청이 완료되면 호출되는 함수, 매개변수 두 개가 전달되는데, 각각 XHR 인스턴스와 'success' 혹은 'error'를 나타내는 상태 메시지 문자열이다. success' 혹은 'error'를 나타내는 상태 메시지 문자열이다. success 나 error 롤백을 명시했다면, 이 함수는 해당 콜백이 호출된 후에 실행된다.
- beforeSend (Function)
  - 요청이 전송되기에 앞서 먼저 호출되는 함수. 이 함수는 XHR 인스턴스를 전달 받으며, 사용자 정의 헤더를 설정하거나 요청 전에 필요한 연산을 수행하는 데 사용할 수 있다.

Ajax 요청 직접 제어하기

- async (Boolean)
  - false 이면 요청이 동기 호출로 전송된다. 기본은 비동기 요청이다.
- processDataBoolean
  - false로 설정되면, URL 인코딩된 형태로 처리되어 전달된 데이터를 금지한다. 기본 값은 데이터가 'application/x-www-form-urlencoded' 타입의 요청에 사용하는 형태의 URL로 인코딩된다.
- ifModified Boolen
  - true 일 때 Last-Modified 헤더를 확인하여 마지막 요청 이후에 응답 콘텐츠가 변경되지 않았다면 요청이 성공한다. 만일 생략하면 헤더를 확인하지 않는다.

JSON 이용하기

- JavaScript Object Notation의 약어
- XML 데이터를 대신하기 위해서 사용한다.
- 키와 값을 쌍으로 가지는 구조
- 배열을 사용할 경우에는 대괄호 안에 중괄호를 사용하여 조합

JSON 이용하기

- **JSON**에 저장되는 정보의 형태
  - 배열
    - 대괄호 안에 값을 콤마로 구분하여 나열하며, 대괄호 안에 나오는 순서대로 배열 요소의 순서가 매겨진다.
    - [1, 2, 3]
  - 객체
    - 중괄호 안에 있는 이름: 값의 형태로 멤버 하나를 표현
    - 각 멤버는 콤마로 구분
    - 순서가 아닌 이름으로 읽기 때문에 멤버의 순서는 의미가 없다.
    - {"name": "레몬" }

JSON 이용하기


```
[
  {
    "id": "1",
    "name": "레몬",
    "price": " 3000",
    "description": "레몬에 포함되어 있는 쿠엔산은 피로회복에 좋다. "
  },
  {
    "id": "2",
    "name": "키위",
    "price": " 2000",
    "description": "비타민C가 매우 풍부하다."
  }
]
```

JSON 이용하기

id	name	price	description
1	레몬	3000	레몬에 포함되어 있는 쿠엔산은 피로회복에 좋다. 비타민C도 풍부하다.
2	키위	2000	비타민C가 매우 풍부하다. 다이에트와 미용에도 매우 좋다.
3	블루베리	5000	블루베리에 포함된 anthocyanin(안토시아닌)은 눈피로에 효과가 있다.
4	체리	5000	체리는 맛이 단 성분이 많고 피로회복에 잘 듣는다.
5	메론	5000	메론에는 비타민A와 칼륨이 많이 포함되어 있다.
6	수박	15000	수분이 풍부한 과일이다.

JSON 이용하기

```
$.getJSON('item.json', function(data, textStatus) {
        $.each(data, function() {
                $("#treeData").append("<tr>" + "<td>"
                        + this.id + "</td>" + "<td>"
                        + this.name + "</td>" + "<td align='right'>"
                        + this.price + "</td>" + "<td>"
                        + this.description + "</td>" + "</tr>");
        });
});
```

- $.getJSON(url[, data][, success])
  - JSON으로 표현한 데이터를 파일에 저장해 두었다가 필요할 경우 이를 로드 하는 전역 메소드

$.getJSON으로 Flickr에서 이미지 얻기

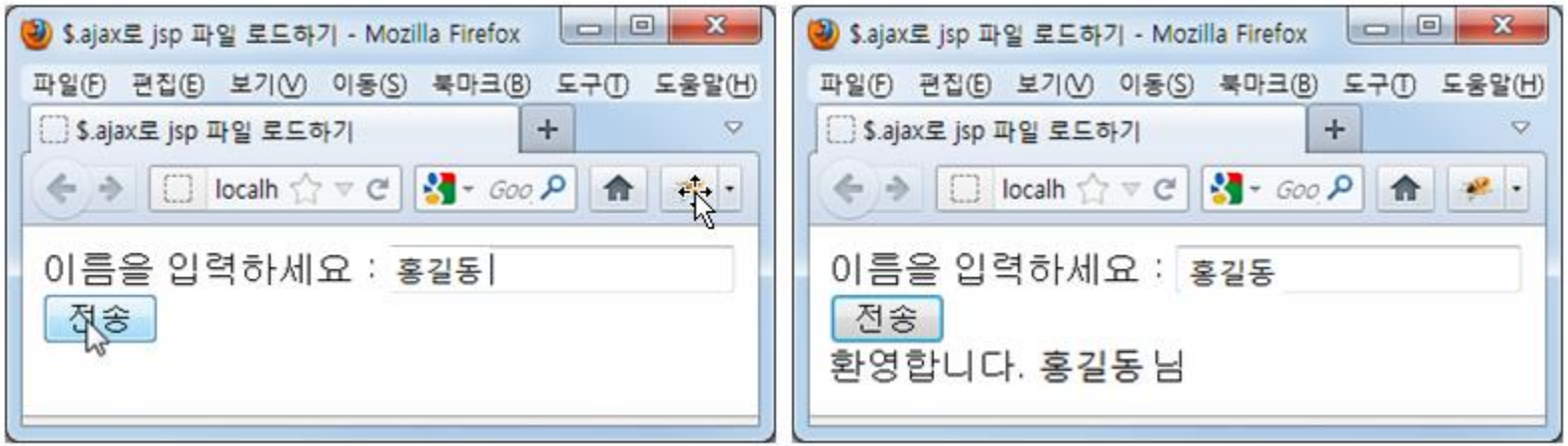
```
$.getJSON("http://api.flickr.com/services/feeds/photos_public.gne?"
      + "tags=raccoon&tagmode=any&format=json&jsoncallback=?",
      function(data) {
        $.each(data.items, function(i, item) {
          $("<img/>").attr("src", item.media.m).appendTo("#images");
          if (i == 4)
            return false;
        });
  });
```

XML 파일을 GET 방식으로 로드하기- $.get()

```
$.get('item.xml', function(data) {
        $(data).find('item').each(function() {
         var $item = $(this);
         $("#treeData").append("<tr>" + "<td>"
            + $item.attr("id") + "</td>" + "<td>"
            + $item.attr("name") + "</td>" + "<td align='right'>"
            + $item.find("price").text() + "</td>" + "<td>"
            + $item.find("description").text() + "</td>" + "</tr>");
      });
    });
```

- $.get(URL, fn)
  - GET 방식으로 서버와 통신하는 jQuery Ajax 함수
  - URL로 지정한 파일을 로드해서 그 데이터를 텍스트 형식으로 콜백 함수(fn)에게 넘겨주는 기능을 한다.

POST 방식으로 서버와 통신하기- $.post()



- $.post(URL[, data][,fn][,dataType])
  - POST 방식으로 서버와 통신하는 jQuery Ajax 함수이다. $.post()는 URL로 지정한 파일을 로드해서 그 데이터를 텍스트 형식으로 콜백 함수(fn)에게 넘겨주는 기능을 한다.

POST 방식으로 서버와 통신하기- $.post()

```
var username = $('.username').val();
var sendData = 'username=' + username;

$.post("welcome.jsp",
        sendData,
        function (data) {
                $('#message').html(data);
        });
```

Ajax에 대한 global 옵션 설정- $.ajaxSetup()

```
$.ajaxSetup({
            type:"POST",
            url:"logincheck.jsp",
            dataType : "text",
            success: function (msg) {
                        $('#message').html(msg);
                        }
});
$.ajax({
          data: sendData
});
```

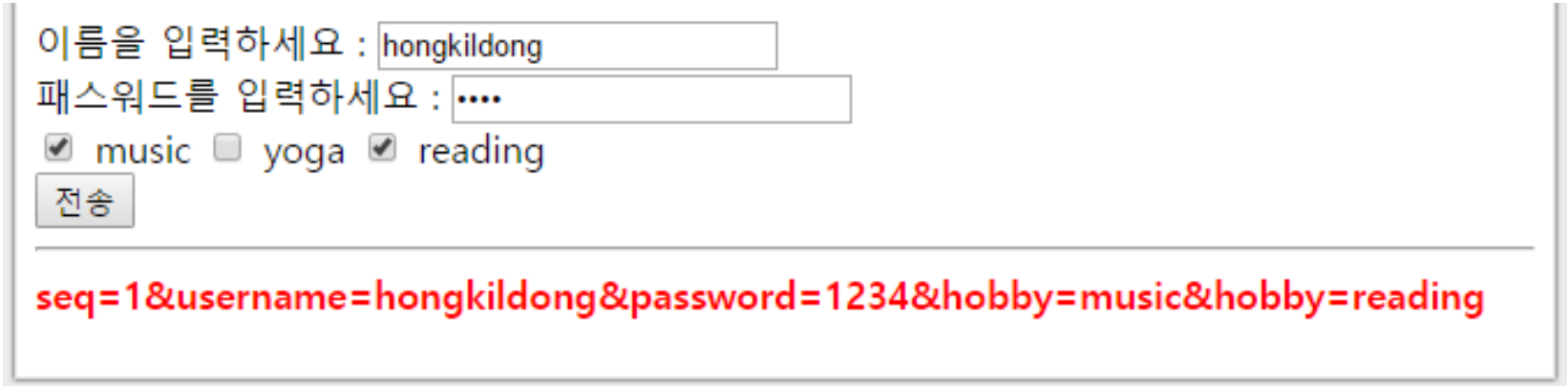
- $.ajaxSetup()
  - jQuery Ajax에 대한 공통적인 기본 설정들을 지정하기 위해서 사용된다.

스크립트 로드하기 - $.getScript()

```
$.getScript("test.js");
```

- $.getScript(url[, success])
  - 자바 스크립트 파일을 로드한다.
  - url : 요청할 서버 측 자원의 URL
  - success : 요청이 성공했을 때 호출할 함수

폼 데이터를 쿼리 스트링으로 변환- serialize()



폼 데이터를 쿼리 스트링으로 변환-serialize()

```
var form_data=$('form').serialize();
```

- serialize()
  - 폼 엘리먼트의 값을 쿼리 스트링으로 변환한다.
  - 키/값의 쌍 형태의 문자열로 구성되는데 이를 쿼리 스트링이라고 한다.

```
seq=1&username=pinksung&password=1234&hobby=music&hobby=reading
```

엘리먼트 형식의 값을 배열 형태로 변환- serializeArray()

```
var form_data=$('form').serializeArray();
```

- serializeArray()
  - 폼 엘리먼트의 값을 객체 배열로 변환한다.
  - 객체 배열은 {name=value}의 Array 형태

```
[ Object { name="seq", value="1"},
  Object { name="username", value="subean"},
  Object { name="password", value="4321"},
  Object { name="hobby", value="yoga"}  ]
```