AJAX

개요

Asynchronous Javascripts + XML의 약자
Javascript에 의한 비동기적 통신
서버가 응답을 받을 준비가 되어있는지에 상관 없이 웹 브라우저가 서버로 정보를 전송하는 것을 의미

최근엔 XML을 사용하지 않음

AJAX의 장점

비동기통신이 가능 : 페이지의 이동과는 관계없이 서버측과 데이터 교환을 이루는 것 (작업의 결과를 비동기적으로 서버측에 요청해서 필요한 데이터를 빠르게 얻어냄)

서버측의 부담중 일부를 웹 클라이언트에서(화면 내에서 처리) 처리하므로 빠른 서버 어플리케이션을즐길 수 있음

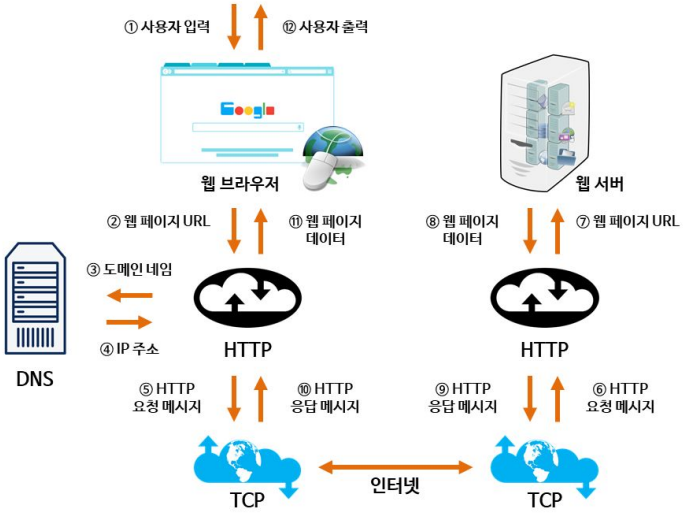
AJAX의 비동기 통신의 구조

Ajax의 특징

- 웹브라우저는 요청을 송신하면 응답을 기다리지 않는다
- 서버는 필요한 데이터만을 응답한다

동작 순서

1. 웹 브라우저에서 XLMHttpRequest로 서버에 요청을 송신
2. 서버측의 응답을 기다리지않고 웹 브라우저 안의 작업을 처리
3. 서버측에서는 송신받은 요청을 기본으로하여 각종 처리 작업/데이터 연동 작업 진행
4. 처리가 끝나면 웹 브라우저로 결과를 보냄
   (전체가 아닌 필요한 데이터만)



웹브라우저의 이벤트

Ajax 는 이벤트를 통해 서버측에 비동기로 요청을 송신하고 결과를 반영.

Ajax를 웹 어플리케이션에서 사용하기 위해서는 Ajax를 구현하는 Javascript의 Event Handler(이벤트가 발생했을 때 그것을 감지하고 처리하는것)를 그대로 사용

이벤트	설명
onMouseDown	마우스 단추를 눌렀을 때
onMouseUp	마우스 단추를 눌렀다 놓았을때
onMouseOver	마우스 커서가 브라우저상의 특정 객체 위에 있을 때
onMouseOut	마우스 커서가 브라우저상의 특정객체의 밖에 있을 때
onMouseMove	마우스 커서를 이동했을 때
onBlur	입력 폼등에서 포커스가 사라질때
onChange	입력 폼등에서 내용이 변경될때
onClick	단추를 클릭했을때
onDblClick	단추를 더블클릭했을때
onFocus	입력 폼등에서 포커스가 위치할때
onSelect	입력 폼등에서 문자데이터가 선택되었을때
onSubmit	“submit” 단추를 클릭했을때
onReset	“reset”단추를 클릭했을 때
onLoad	브라우저상에서 페이지가 읽혔을때
onUnload	다른 웹페이지로 이동할때
onAbort	이미지를 읽다가 중단시켰을 때
onMove	윈도/프레임 등이 움직였을 때

DHTML(Dynamic HTML)

동적인 웹페이지를 구현하는 기술
Ajax를 동적으로 동작하는 부분에 대한 기본 기술

DOM (Document Object Model)

문서의 객체화 - 계층적인 node라는 객체를 사용하고, 객체를 조작하기 위한 방법을 DOM으로 제공
DOM을 사용하면 태그형태의 언어로 정의되는 문서의 포맷은 모두 노드간에 계층관계의 구조를 가진 객체로 확인되어진다.

리치클라이언트

유저인터페이스의 화면표시 형태가 좋고, 펀리한 조작이라는 특징을 가진 클라이언트
Ajax는 웹브라우저에서 리치클라이언트를 구현시켰다.

작업환경

Ajax는 서버쪽의 기술도 함께 사용
java, .net, Ruby, PHP 중 하나의 기술을 사용해야함

--- java / tomcat 다운로드..

Javascript의 역할

개요

javascript는 Ajax에서 입출력에 관한 동작을 처리한다.
사용자들의 요구(이벤트)를 처리하는 메소드(이벤트 핸들러)를 통해서 사용자의 요구 수용 후,
처리를 위해 로직을 가지고있는 서버에 처리하는 자바 클래스의 처리를 전송하여,
결과를 받아 표시할 HTML 태그에 표시함

자바스크립트 프레임워크 (Dojo, Rico, TIBET, AjaxSLT) 를 사용하면 편하지만 근본적으로 js를 사용해야함

이벤트 Event	특정 작업을 수행하기 위한 클릭/키 입력 등의 동작
이벤트 핸들러 Event Handler	이벤트가 발생하면 발생된 이벤트에 반응하여 처리를 수행하는 것
이벤트 드리븐 프로그램	동작에 응답하는 방식으로 처리되는 프로그램

웹브라우저의 객체

window 객체

속성명	하는일
self	자기자신을 가리키는 객체
opener	현재 window가 열려진 경우 현재 window를 연 특정 윈도우

메서드	하는일
open()	윈도우를 새로 연다
close()	닫는다
find()	윈도우안의 특정문자열 검색

이벤트명	발생조건
onLoad	브라우저로 문서가 로드될때 발생
onUnload	문서가 모두 제거될때 발생
onBlur	현재의 브라우저가 focus를 잃었을때 발생
onFocus	현재의 브라우저가 focus를 잃었을때 발생
onMove	윈도우를 이동했을때 발생

document 객체

웹브라우저 내용이 출력되는 부분
window.document로 사용하지만 그냥 document로 사용해도 무관(둘 다 같은 객체 지칭)

메소드명	하는일
clear()	브라우저에 출력된 내용을 지움
wirte()	브라우저에 특정 문자 출력
getSelection()	현재 선택한 문자열을 얻어냄

이벤트명	하는일
onClick	클릭했을때 발생
onDblClick	더블클릭했을 때 발생
onKeyDown	키보드의 키를 누르는 순간 발생
onKeyPress	키가 눌려져있는 순간 발생
onKeyUp	키가 눌려져있다가 키를 놓는 순간 발생
onMouseDown	마우스의 단추를 누르는 순간 발생
onMouseUp	눌려진 마우스의 단추를 놓는 순간 발생

폼(Form) 에서 발생하는 이벤트

이벤트명	발생되는 폼 요소	발생조건
onClick	type=”button”	버튼을클릭하는경우
	type=”submit”	버튼을클릭하는경우
	type=”checkbox”	체크선택,해제
	type=”radio”	라디오단추 선택
onChange	type=”text”	텍스트상자에 값이변경될 경우
	<textarea>	값이 변경된 경우
	<select>	포커스가 텍스트 상자안에 들어왔을 경우
onFocus	<textarea>	포커스가 textarea에
	<select>	포커스가 select에
onBlur	type=”text”	포커스가 나갔을 경우
	<textarea>	

	<select>	
onSubmit	<form>	폼에 입력한 내용을 action 속성의 값으로 지정한 페이지로 보내는 명령을 실행했을때. 입력한 내용을 체크하는 메소드를 기술
onMouseDown	type="button"	마우스 단추로 누른경우
onMouseUp	type="button"	눌려진 마우스 단추를 놓는 경우
onKeyDown	<textarea>	키를 누르는경우
onKeyUp	<textarea>	눌려진 키를 놓는경우

DOM

Dom ...(90-167pg)

XMLHttpRequest(XHR)객체

XMLHttpRequest(XHR)객체는 Ajax에서 서버와 클라이언트 간에 통신을 담당
W3C의 표준이 아니기 때문에 모든 웹브라우저에서 이 객체를 지원하는 것은 아님

XMLHttpRequest 객체 생성

XHR객체는 서버와 클라이언트간의 요청과 응답을 처리하기 위해 필요한 객체. XHR객체는 요청작업과 응답작업이 필요하기 때문에, 작업을 처리하기전에 반드시 생성해야함.

IE5.0 / IE 6.0 인경우 : ActiveXObject 객체를 생성
(요즘 7도 안쓰는데..)

```
var xmlHttpObject =
            new ActiveXObject("Microsoft.XMLHTTP");
# "Microsoft.XMLHTTP" 와 "Msxml2.XMLHTTP" 두개 객체 제공
# "Msxml2.XMLHTTP" > IE 5.0 이상부터 사용가능
# "Microsoft.XMLHTTP" = IE 전 버전 사용 가능
```

IE7.0, firefox, safari, opera 인경우 : JS 내장객체 사용

```
var xmlHttpObject = new XMLHttpRequest();
```

객체 생성의 차이점 때문에 XHR 객체를 생성하는 문장을 자바스크립트에 기술할 경우, ActiveX를 사용하거나 자바스크립트 내장객체를 사용해서 생성하는 부분에 반드시 모두 기술되는 것이 좋음(크로스브라우징)

```
var xmlHttpObject;

function createXMLHttpRequest(){
   if(window.ActiveXObject){ // IE5.0, IE6.0
     xmlHttpObject =
           new ActiveXObject("Microsoft.XMLHTTP");
   } else if(window.XMLHttpRequest){ // IE7.0
     xmlHttpObject = new XMLHttpRequest();
   }
}
```

XHR 객체의 메소드와 프로퍼티

메서드	하는일
void abort()	현재요청 취소(중단)한다
`var xmlHttpObject.abort() #xhr 객체`	
String getAllResponseHeaders()	HTTP요청에 대한 모든 응답 헤더들을 키/값의 문자열 쌍으로 리턴
`var str = xmlHttpObject.getAllResponseHeaders()`	
String getResponseHeader(headername)	매개변수로 주어진 HTTP 헤더의 값을 리턴
`var str = xmlHttpObject.getResponseHeader("Content-Type");`	
void open(method, url) void open(method, url, async) void open(method, url, async, userName) void open(method, url, async, userName, password)	사용자의 요청을 설정하는 메소드로 필수 매개변수인 method와 url 요청에 대한 선택적 매개변수를 갖고있다. method 매개변수는 "GET", "POST", "HEAD", "PUT", "DELETE" 중 하나의 값을 가짐 url 매개변수는 상대/절대경로를 가짐 async는 비동기로 전송할지 여부(생략:true-비동기) userName 사용자의 아이디명 password 비밀번호
`xmlHttpObject.open("GET","test.xml",true)`	
void send(content)	요청을 서버로 보낸다

`xmlHttpObject.send(null); // GET방식 사용한경우`
`xmlHttpObject.send(str); // POST방식 사용한경우`	

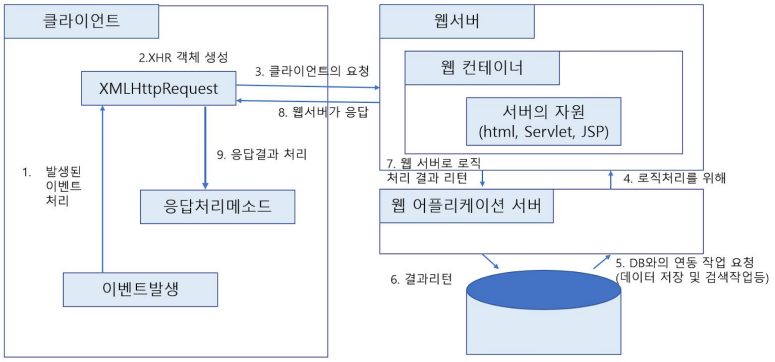
void setRequestHeader(header, value)	헤더의 값을 설정하는것 헤더명이 header인 값을 value로 설정함
`xmlHttpObject.setRequestHeader("content-type","application/x-www-form-unlencoded");`	

프로퍼티	하는일
onreadystatechange	변경이 발생했을 때 이벤트를 처리하기위한 JS의 이벤트 핸들러를 기술
`var xmlHttpObject.onreadystatechange= changeState;`	
readyState	요청객체의 상태 리턴 - 0 : open()메소드 호출 X 상태 - 1 : send() 메소드 호출X로 로드중 - 2 : send() 메소드의 호출이 끝나서, 헤더와 status의 사용이 가능한 상태 로드완료 - 3 : 일부 데이터를 받을 수 있는 상태 - 4 : 모든데이터를 받을 수 있는 상태 완료
`if (xmlHttpObject == 4) {...}`	
responseText	문자열로 이루어진 서버의 응답을받음
`document.getElementById('area').innerHTML = xmlHttpObject.responseText;` `// 'area' 요소에 응답을 입력함`	
responseXML	XML로 이루어진 서버의 응답을 받음 파싱이 가능한 W3C DOM 노드트리 구조의 메소드와 프로퍼티를 XML 문서객체로 리턴받음
responseBody	2진코드의 문자열로 서버의 응답을 받음. ActiveX 컴포넌트를 사용해서 생성한 XMLHttpRequest 객체에서만 사용

`xmlHttpObject = new ActiveXObject("Microsoft.XMLHTTP");`
`...`
`alert (xmlHttpObject.responseBody);`	

status	서버로부터 응답받는 HTTP 상태코드 ex. 200, 400, 500 ...
statusText	서버로부터 HTTP 상태를 문자열로 받음

XHR 객체를 사용한 서버에 요기



동작 순서

1. 이벤트가 발생하면 이벤트처리 메소드 실행
2. XHR 객체를 생성해서 서버와 클라이언트간의 통신준비 open()메소드를 호출해서 요청작업 설정

```
var id = document.getElementById('userId');
var str = "http://127.0.0.1:8080..." + id.value;
xhrObject = new XMLHttpRequest();
// xmlhttpRequest 객체 생성
xhrObject.open("GET", str);
// 요청작업 설정, HTTP 메소드를 get방식으로 지정
xhrObject.onreadystatechange = result;
// 요청의 처리결과인 응답된 내용을 처리할
result() 메소드 지정
xhrObject.send(null); // 서버에 요청
```

3. 클라이언트 요청을 서버로 전달 : send()메소드 호출

```
xhrObject.send(null)
// http 메소드가 "get"방식일경우
xhrObject.send(query)
// http 메소드가 "post"방식일 경우
```

4. 클라이언트의 요청을 받은 서버가 로직 처리를 위해 웹어플리케이션 서버(WAS)에 처리작업을 보낸다
5. 웹 어플리케이션 서버에서는 DB와의 연결이 필요한경우 작업을 처리한 후 연동한다.
6. 처리한 작업을 웹 어플리케이션 서버로 보내 처리한다.
7. 로직과 DB작업에 대한 처리가 끝나면 처리결과를 웹 서버로 보낸다.
8. 웹 서버에서는 이 작업처리 결과를 클라이언트에게로 응답을 보낸다.
9. 클라이언트는 서버로부터 응답이 오게되면 받아서 처리할 메소드로 작업을 일임한다.
   주로 onreadystatechange 속성

HTTP 메소드

- GET
  : 간단하고 중요도가 떨어지는 데이터를 보낼때
  : 여러개의 요청이 동일한 결과를 응답받아야 할 경우
  (요청에 의해 서버의 상태가 변하지 않는다면)
- POST
  : 중요도가 높고 많은 데이터를 보낼때
  : 서버의 상태를 변경하고자 할 때
  (동일한 결과값을 응답한다는 보장 X)

서버와의 연동

데이터베이스와의 연동을 위해서는 서버쪽의 기술을 사용해야함.
모든 데이터는 체계적인 관리를 위해서 데이터베이스에서 관리

GET/POST 방식으로 서버에 요청전송

비동기 통신을 위한 XHR 객체를 이용해서 사용자의 요청을 보냄

GET/POST 방식으로 요청보내기

GET방식으로 요청보내기 단계

1. XHR 객체 생성
2. 서버로부터 응답시, 응답결과를 처리할 메소드 설정

```
xhrObject.onreadystatechange = resultProcess;
```

3. 서버에 요청을 설정

```
xhrObject.open("GET", "xhrTest.xml", "true");
```

4. 서버로 요청을 전송

```
xhrObject.send(null); // 실제로 서버로 요청을 보냄
```

5. 서버로부터 응답받은 결과 처리

```
function resultProcess(){
  if(xhrObject.readyState == 4({
      // 상태가 모든 데이터를 받을수 있는 상태
    if(xhrObject.status == 200){
      // 서버로부터 응답받는 HTTP상태가 정상인 경우
      // 처리할 내용
    }
  }
}
```

GET방식으로 요청 파라미터 보내기

일반적으로 GET방식으로 요청 파라미터를 보낼때는 URL 다음에 파라미터의 이름과 값을 기술

```
http://127.0.0.1:8080/ParameterGet?num=8190&su=10
```

GET방식은 중요한 값이 주소의 URL로 표시되므로 보안에 문제가 있다. 그러나 Ajax에서는 GET방식이라 해도 주소에 표현되지 않는다.
요청을 조금더 안정적으로 보낼 때는 타임스탬프(timestamp)를 URL에 붙이는 것이 좋음. 타임스탬프를 보내는 방식은 GET/POST 모두에게 좋다.

```
"http://127.0.0.1:8080/ParameterGet?" + parameters
+ "timestamp=" + newDate().getTime();
```

이후는 java로 작업

POST 방식으로 요청보내기 단계

1. XHR객체 생성
2. 서버로부터 응답시, 응답결과를 처리할 메소드 설정
3. 서버에 요청을 설정

```
xhrObject.open("POST", "xhrTest.xml", "true");
```

4. 서버로 요청 전송

```
xhrObject.send(str);
// POST일때는 매개변수 값을 서버로 보낼 요청
파라미터를 가지고 있는 변수로 설정한다
```

5. 서버로부터 응답받은 응답결과 처리

GET방식으로 요청 파라미터 보내기

일반적으로 POST 방식으로 요청 파라미터를 보낼때는 그 내용이 URL에 노출되지 않는다.

```
http://127.0.0.1:8080/ParameterPost
```

Ajax에서 POST방식의 경우 파라미터를 보내는 부분과 URL을 별도로 분리해서 보낸다.

```
// URL
var url= "http://127.0.0.1:8080/ParameterPost"

//요청 파라미터
var query = getParameterValues();

xhrObject.open("POST", url, "true");
xhrObject.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded;charset=utf8")
// post 방식으로 파라미터를 보낼 때 반드시 이 형식으로
// 입력해야함.

// Content-Type 설정
xhrObject.send(query);
```

open() 메소드를 설정하고 나면 Content-Type을 설정해야 하는데, 반드시 이 순서로 지정해야함. (순서가 틀릴경우 요청 파라미터값을 서블릿에서 받지못함)

서버로부터 응답결과 처리

XHR 객체로 보낸 서버의 응답결과는 responseText 프로퍼티(문자열)와 responseXML 프로퍼티(문서객체)로 받는다. responseText로는 간단한 내용을 수신할 때 좋으며, responseXML로는 브라우저의 종류에 상관없이 DOM 구조로 표시된 동일한 결과를 유지해준다. 그리고 JSON은 복잡하게 생성되는 XML의 구조를 대체하기 위해 제안된 방식.

응답결과를 XML로 처리

기존의 웹 어플리케이션은 어떠한 처리를 하기위해서 기존 페이지를 다시 읽어들이는 구조이며, 해당 작업은 리소스가 낭비되는 작업.
이에 대응하여 기존 콘텐츠를 재사용 하고, 변경된 부분만을 갱신하는 것이고, 여기에 DOM 구조가 사용됨

자바스크립트로 DOM 구조를 이용해 변경된 부분만 동적으로 생성할 수 있기때문에 응답결과를 XML 문서객체로 받아서 처리하는것이 사용됨

응답결과를 XMLHttpRequest 객체의 responseXML 프로퍼티를 사용해서 XML 문서 객체로 받아서 해결해야 할 경우, DOM 구조로 이루어진 문서에 대한 기본 프로퍼티의 개념이 필요

DOM의 기본 인터페이스

1. Node 인터페이스

- node 인터페이스의 주요 프로퍼티		
리턴타입	프로퍼티명	설명
Node	parentNode	현재노드의 부모노드를 리턴
NodeList	childNodes	자식노드들을 배열로 리턴
Node	firstChild	가장 첫번째 자식노드 리턴
	lastChild	가장 마지막 자식노드 리턴
	previousSibling	이전 형제노드 리턴
	nextSibling	다음 형제노드 리턴
DOMString	nodeName	노드의 이름 리턴
	nodeValue	노드의 값을 리턴

node 인터페이스의 주요 메서드		
리턴타입	프로퍼티명	설명
Node	insertBefore (newChild, refChild)	자식 노드인 refChild 노드 앞에 newChild 노드를 삽입
	replaceChild (newChild, oldChild)	자식노드인 oldChild 노드를 newChild 노드로 대체
	removeChild (oldChild)	자식노드인 oldChild를 제거한다
	appendChild (newChild)	현재노드의 자식 노드인 newChild를 추가한다
boolean	hasChildNodes()	현재의 자식노드가 있으면 true, 없으면 false

2. Document 인터페이스

Document 인터페이스는 HTML또는 XML 문서 전체를 구현하는 인터페이스 = 문서에 대한 접근을 하는 인터페이스
Node 인터페이스를 상속받아 만든것으로, Node 인터페이스가 제공하는 프로퍼티와 메소드 모두 사용 가능

Document 인터페이스의 주요 메서드		
리턴타입	프로퍼티명	설명
DOMString	createElement()	tagName 요소 생성
Text	createTextNode(data)	data 텍스트 노드 생성
NodeList	getElementsByTagName()	tagname요소 나열
element	getElementById()	id가일치하는 요소 리턴

3. Element 인터페이스

Node 인터페이스를 상속받아 만든것으로, Node 인터페이스가 제공하는 프로퍼티와 메소드 모두 사용 가능

Element 인터페이스의 주요 메서드		
리턴타입	프로퍼티명	설명
DOMString	getAttribute(name)	현재 요소 name속성 가져옴
Void	setAttribute(name, value)	name 속성 을 value값으로 설정
Void	removeAttribute(name)	현재 속성의 name 속성 제거
Nodelist	getElementsByTagName(name)	자식요소중 tagname요소를 모두 배열로 리턴

응답결과를 XML로 처리

응답결과를 XML로 처리할 경우, responseXML 프로퍼티 사용

```
// 서버응답을 변수에 저장
var xmlDoc = xhrObject.responseXML
var xmlNode = xmlDoc.getElementsByTagName('test');
```

JSON(Javascript Object Notation)

개요

json은 저중량 데이터 교환 형식으로, 사용자가 읽고 쓰기 쉬운 형식으로 이루어져있음 & 컴퓨터 시스템이 파싱하고 생성하기 쉽게 되어있음. json은 자바스크립트 기반으로 만들어져있으나 완벽하게 특정언어에 독립적인 텍스트 형식 보유.

json의 두가지 구조

1. 이름/값의 쌍으로 이루어진 데이터의 집합
객체(object), 레코드(record), 스트럭트(struct), 딕셔너리(dictionary), 해시테이블(hash table), 리스트(list), 또는 배열(associative) 등으로 사용

2. 순서가 있는 값들의 목록
배열(array), 벡터(vector), 리스트(list) 로 사용

json의 데이터구조는 다른 언어의 데이터 교환에 사용할 수 있음. json은 js기반으로 만들어졌기 때문에 대부분 브라우저 호환.

JSON 직렬화/역직렬화

1. **객체의 직렬화 :**
브라우저가 도착한 데이터(문자열 등)를 자바스크립트 객체로 변환하는 과정
2. 객체의 역직렬화 :
브라우저에서 객체를 JSON 표현식을 이용한 문자열로 변경하는 과정

json 진행 방식

서버로부터 웹브라우저로 JSON 데이터가 전송되면 이 데이터는 문자열로 전송된다.
데이터가 브라우저로 도착하면 스크립트는 이문자열을 자바스크립트 객체로 변환해야한다.(역직렬화)
역직렬화는 json이라는 내장 객체의 parse() 메서드를 이용하면 된다. 이 객체는 전역 객체이므로 인스턴스를 만들 필요 없이 사용하면 된다.

stringify() 메서드

json 객체는 stringify() 라는 메서드도 제공하는데, 이 메서드는 객체를 json 표현식을 이용한 문자열로 변경하여 브라우저가 서버로 객체를 전송할 수 있도록 한다. 이 과정을 객체의 직렬화

라고 한다. 이 메서드는 사용자가 페이지에서 데이터를 가지고 있는 자바스크립트 객체를 수정한 후, 이를 서버에 전달하여 서버에 저장되어있는 정보를 수정하고자 할 때 사용할 수 있다.

jQuery를 이용한 Ajax

```
// 서버응답을 변수에 저장
$.ajax ({
  // URL은 필수 요소이므로 반드시 구현
  url   : "url", // 요청이 전송될 URL 주소
  type  : "GET", // http 요청 방식 (default: 'GET')
  async : true,
  // 요청 시 동기화 여부.
  // 기본은 비동기(asynchronous) 요청 (default: true)
  cache : true,  // 캐시 여부
  timeout : 3000,
  // 요청 제한 시간 안에 완료되지 않으면
  // 요청을 취소하거나 error 콜백을 호출.(단위: ms)
  data  : {key : value}, // 요청 시 포함되어질 데이터
  processData : true,
   // 데이터를 컨텐츠 타입에 맞게 변환 여부
  contentType : "application/json",
   // 요청 컨텐트 타입
  dataType    : "json",
   // 응답 데이터 형식 (명시하지 않을 경우 자동으로 추측)
  beforeSend  : function () {
   // XHR Header를 포함해서
   // HTTP Request를 하기전에 호출됩니다.
  },
  success : function(data, status, xhr) {
   // 정상적으로 응답 받았을 경우에는
   // success 콜백이 호출되게 됩니다.
   // 이 콜백 함수의 파라미터에서는 응답 바디,
   // 응답 코드 그리고 XHR 헤더를 확인할 수 있습니다.
  },
  error: function(xhr, status, error) {
   // 응답을 받지 못하였다거나 정상적인 응답이지만
   // 데이터 형식을 확인할 수 없기 때문에
   // error 콜백이 호출될 수 있습니다.
   // 예를 들어, dataType을 지정해서
   // 응답 받을 데이터 형식을 지정하였지만,
   // 서버에서는 다른 데이터형식으로 응답하면
   // error 콜백이 호출되게 됩니다.
  },
  complete : function(xhr, status) {
   // success와 error 콜백이 호출된 후에 반드시 호출됩니다.
   // try - catch - finally의 finally 구문과 동일합니다.
  }
});
```