AJAX

미래의 웹: 시맨틱 웹

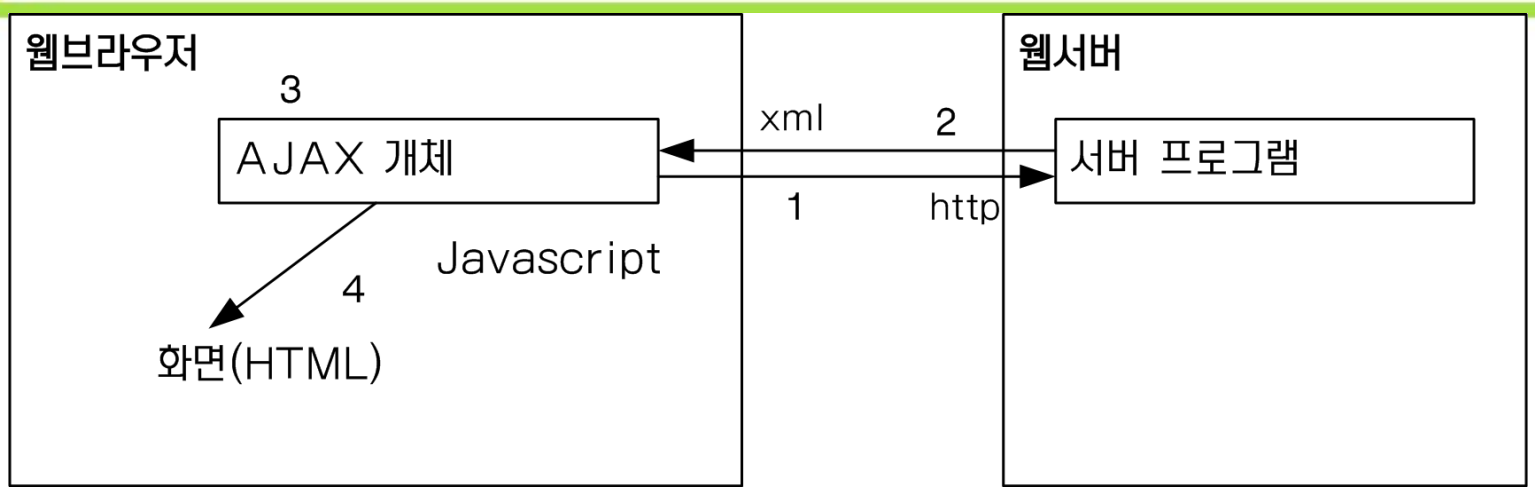
□ 미래의 웹은 시맨틱 웹

- XML을 활용하여 컴퓨터가 정보의 뜻을 이해하고 자동화된 처리를 할 수 있는 인공지능형 웹
- 웹 2.0 서비스로 주목 받는 많은 웹 서비스가 XML을 사용한 정보 교환에 기반을 두고 있다.
- 하이퍼링크로 연결된 단순한 거미줄 -> 의미로 연결된 아주 촘촘한 그물망

□ 시맨틱 웹을 위한 기본

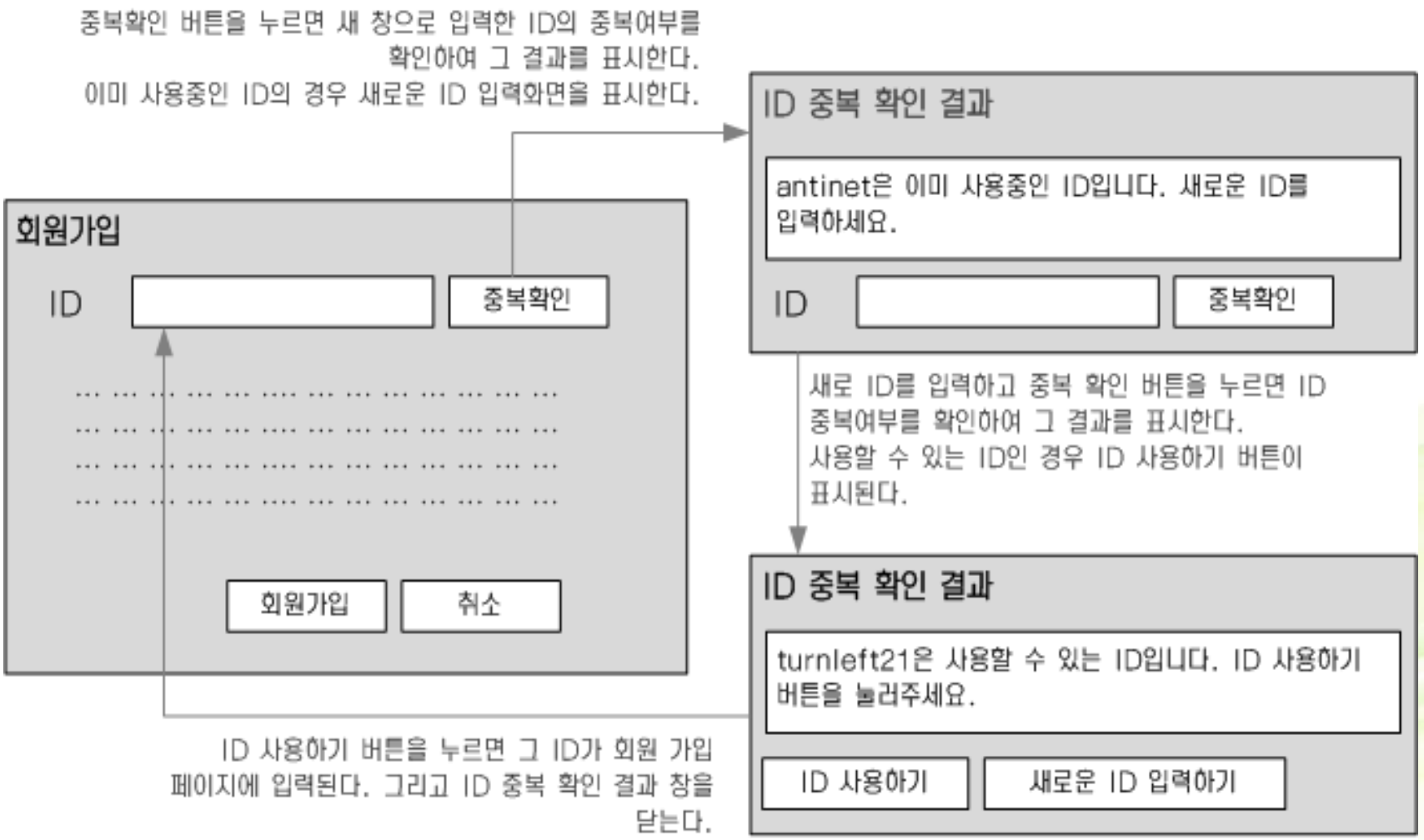
- 정보를 활용하기 위해서는 모양과 내용이 분리되어야 한다. 바이너리 형태의 정보는 HTML, XML 형태로 바꾸어야 한다.
- URI는 변하지 않아야한다 * URL(Uniform Resource Locator) -> URI(Uniform Resource Identifier
- 웹 페이지의 주소는 변하지 않고 항상 동일하게 유지되어 두려움 없이 연결할 수 있어야 한다. 프레임을 사용하고 있다면 프레임을 걷어내는 것이 좋다.

Ajax: Asynchronous JavaScript XML, XmlHttp



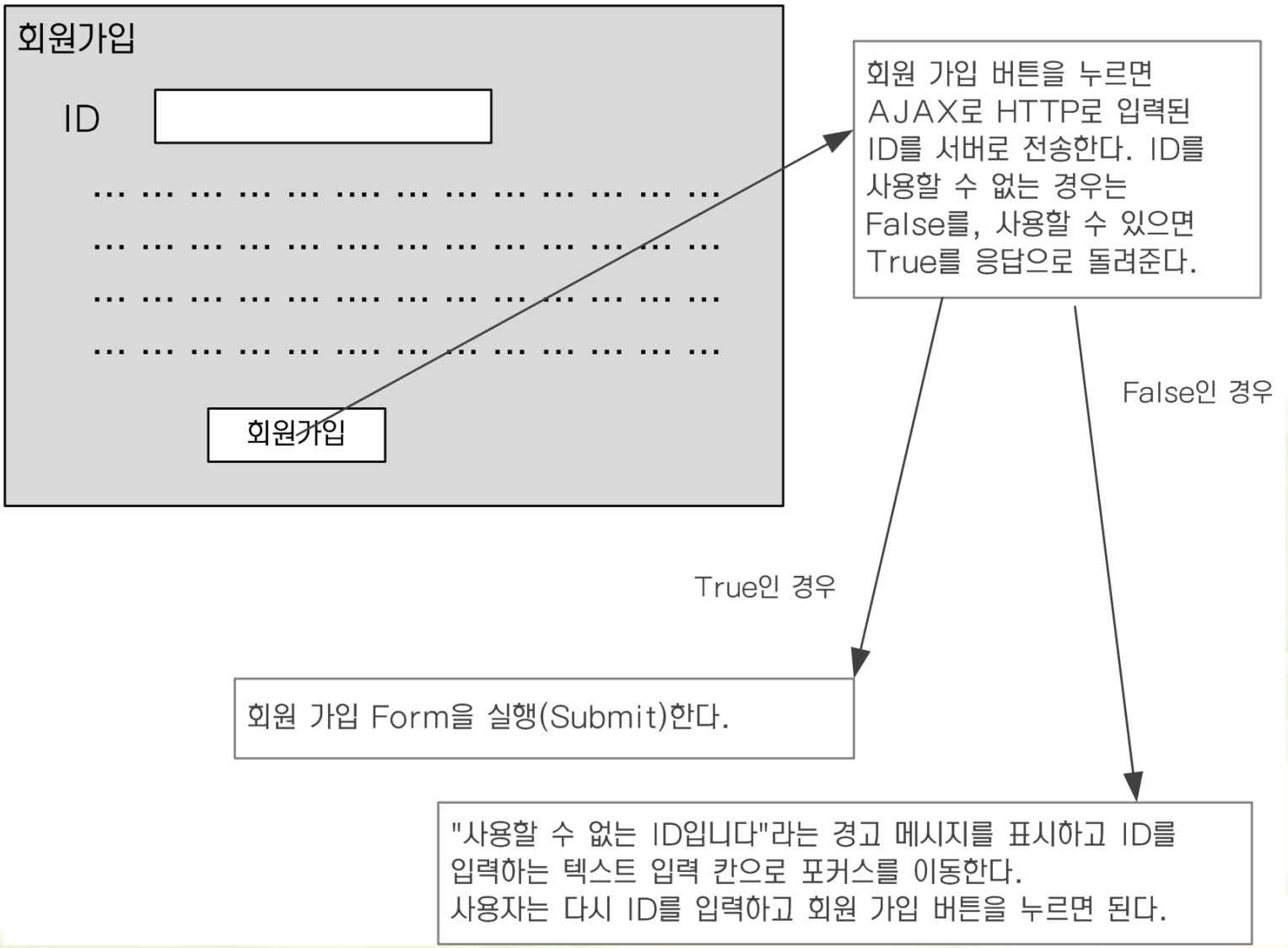
- 비동기 자바스크립트 XML(Asynchronous JavaScript + XML)
- 자바스크립트로 HTTP 요청을 보내서 XML 응답을 받아서 사용하는 기술
- HTTP 요청을 보냄 -> XML 문서를 응답으로 받음 -> 자동으로 XML 개체가 생성됨 -> 자바스크립트는 XML 개체에 접근하여 다양한 작업을 함.
- HTML+ CSS, DOM, XML, XMLHttpRequest, JavaScript을 합쳐서 사용하는 것
- Ajax는 자바스크립트에서 XML 문서를 불러와서 사용하는 기술.
- 관련 기술
  - HTML : 웹 페이지
  - CSS : 웹 페이지의 모양 지정
  - DOM : 자바스크립트로 웹 페이지에 접근하는 방법
  - XML : 정보를 표현하는 언어
  - XMLHttpRequest : 웹브라우저 개체. 자바스크립트로 실행함. 웹서버와 통신
  - 자바스크립트 : 웹 브라우저에서 실행되는 스크립트 언어, 사용자가 마우스를 드레그하거나 버튼을 클릭하면 XMLRequest객체를 사용해서 웹서버에 요청

회원 가입 페이지 문제



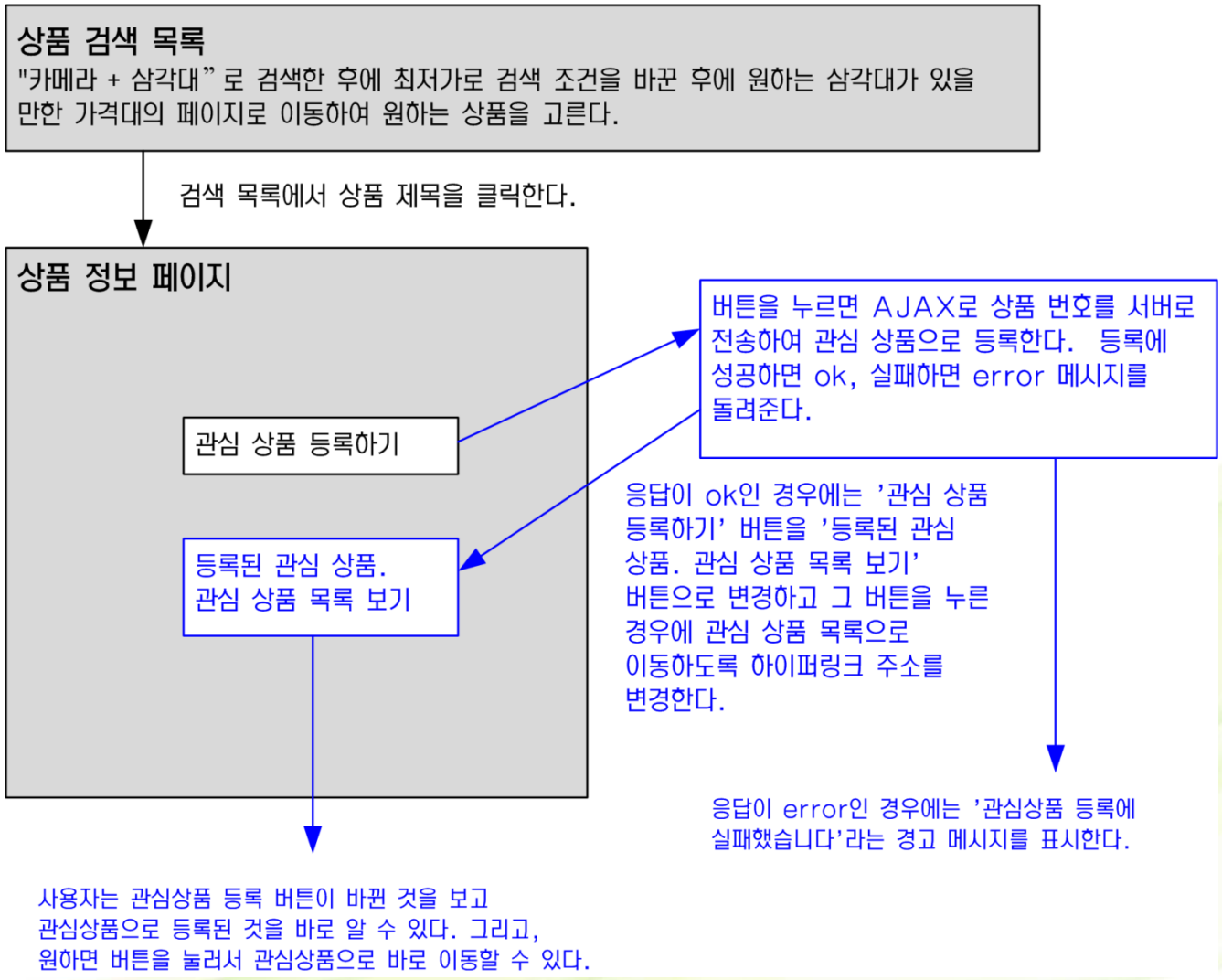
* 불필요하게 새 창이 뜸
* 사용자가 버튼을 두 번이나 더 눌러야 함.

Ajax를 사용한 해결책



불편한 관심 상품 등록하기

Ajax를 사용한 해결책



Ajax의 활용 분야

□ 기존 웹사이트에서 AJAX를 활용하면 효과가 있는 경우
- 웹 페이지를 바꾸지 않고 현재의 웹 페이지에서 어떤 동작을 하고 싶을 때
- 불필요한 팝업을 사용하여 처리하는 작업들.

□ AJAX 애플리케이션으로 개발할 필요가 있는 경우
- 여러 번 불필요하게 화면을 다시 출력할 때.
- 특정한 데이터를 반복해서 사용하면서 다양한 작업을 할 때.

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. async 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 설정한다.

XMLHttpRequest객체를 사용한 데이터 송수신

Ajax의 가장 큰 장점은 웹서버와 데이터를 주고받아 사용자가 웹 페이지 이동 없이 즉각적으로 원하는 기능을 수행할 수 있도록 하는 것

1. XMLHttpRequest 프로그래밍 순서
   1) XMLHttpRequest 객체 구하기
   2) 웹 서버에 요청 전송하기
   3) 웹 서버에서 응답이 도착하면 화면에 반응하기

2. XMLHttpRequest 객체 구하기

IE는 ActiveX 컴퍼넌트로 제공 나머지 브라우저는 기본적으로 제공

```
function getXMLHttpRequest() {
    if (window.ActiveXObject) {  //  IE에서 XMLHttpRequest 객체 구하기
            return new ActiveXObject("Msxml2.XMLHTTP
    } else if (window.XMLHttpRequest) {
            return new XMLHttpRequest();   // IE 이외의 브라우저에서 XmlHttpRequest
    } else {          return null;
    }
}
```

웹 서버에 요청 전송하기

- XMLHttpRequest 객체를 생성한 다음에는 웹서버에 요청 전송

open() 함수 : 요청의 초기화, GET/POST선택, 접속할 URL입력

send() 함수 : 홈 서버에 요청 전송

```
httpRequest = getXMLHttpRequest();
httpRequest.open(httpMethod, httpUrl, true);
httpRequest.send(null);
```

★. open의 인자

. HttpMethod ; get 또는 post

. httpUrl : 접속할 URL 웹페이지의 보안상 이유로 접속할 url은 현재 페이지와 같은 도메인에 있어야 함

. 세번재 인자 : 동기/비동기 지정

```
httpRequest.open("GET", "/test.jsp?id=mad&pw=1234", true);
httpRequest.open("GET", "/test.jsp", true);
httpRequest.send(null);

httpRequest.open("POST", "/test.jsp, true);
httpRequest.send("id=mad&pw=1234");
```

서버 응답처리하기 ; onreadystatechange 프로퍼티와 콜백함수

- XMLHttpRequest 객체는 웹 서버로부터 응답이 도착하면 특정한 자바스크립트 함수를 호출하는 기능이 있는데 이 프로퍼티가 onreadystatechange 임

```
function load(url) {
    httpRequest = getXMLHttpRequest();
    httpRequest.onreadystatechange = callbackFunction;
    httpRequest.open("GET", "/test.jsp", true);
    httpRequest.send(null);
}
```

코드가 실행되면 /test.jsp에 접속하여 응답이 도착하면 callbackFunction() 함수를 호출
onreadystatechange에 지정한 함수를 콜백 함수라고 부르며 콜백함수에서 화면을 변경하거나 경고창을 뛰우는 등 응답결과에 따라 수행

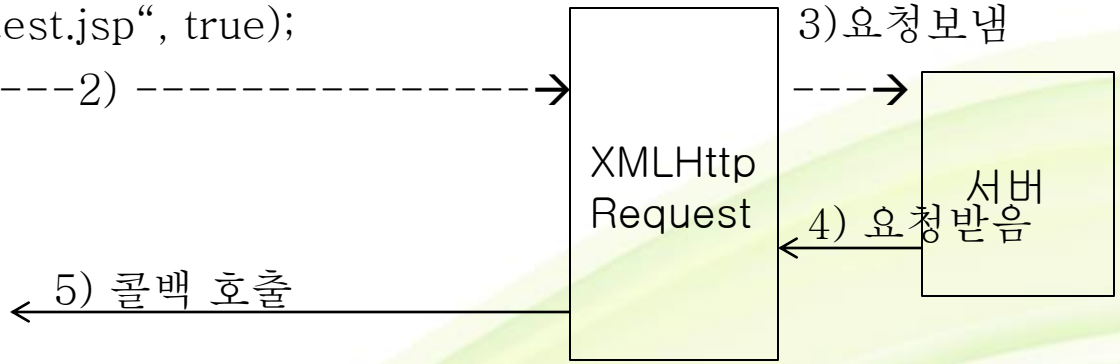
5. 코드가 실행하는 순서

. 사용자의 이벤트가 발생하면 이벤트 처리함수 호출
. 이벤트 처리함수에서는 XMLHttpRequest 객체의 send()함수 호출
. XMLHttpRequest객체의 send함수가 호출되면 웹 서버에 요청이 전송 된다
. XMLHttpRequest 객체에 응답이 도착하면 onreadystatechange 프로퍼티에 지정한 콜백함수 호출

```
<script type="text/html">
Function getXMLHttpRequest() {
  ...
}
var httpRequest = null;

Function processEvent() {
   httpRequest = getXMLHttpReauest();
   httpRequst.onreadystatechange = callBackFunction;
   httpRequest.open("GET", "/test.jsp", true);
   httpRequest.send(null);  ------2) ----------------->
}

function callBackFunction() {
    ...
}
</script>
...
<input type="button" onclick="processEvent()">
```



1) 사용자 이벤트

XMLHttpRequest 객체의 상태 ; readyState

- onreadystatechange 에서 명시한 콜백함수는 readyState라는 프로퍼티의 값이 변경될 때 마다 호출됨

readyState 프로퍼티의 값

값	의 미	설 명
0	UNINITIALIZED	객체만 생성되고 초기화되지 않은 상태(open메서드가 호출되지 않음)
1	LOADING	Open메서드가 호출되고 아직 send메세지가 불리지 않은 상태
2	LOADED	send메세지가 불렸지만 status와 헤더는 도착하지 않은 상태
3	INTERACTIVE	데이터의 일부를 받은 상태
4	COMPLETED	데이터를 전부 받은 상태, 완전한 데이터 이용 가능

보통 readyState 값이 4인 경우에 기능 구현

```
function callbackFunction() {
  if  (httpRequest.readyState == 4) {
   // 서버에서 완전하게 응답이 도착한 경우
  }
}
```

처리 시간이 오래 걸릴 경우 1, 2, 3일 때 작업이 진행 중이라는 메시지 출력하는 것이 좋음

서버로부터의 응답상태 : status/statusTest

웹 서버로부터 응답이 도착하면 웹 서버에서의 처리가 올바르게 수행되었는지 확인

주요 상태코드

값	텍 스 트	설 명
200	정상	요청성공
403	Forbidden	접근 거부
404	Not Found	페이지 없음
500	Internal Server Error	서버 오류 발생

서버로부터 응답이 오면 status프로퍼티를 사용해서 요청이 성공적으로 수행되었는지 확인

```
function callbackFunction() {
    if ( httpRequest.readyState == 4 {
        if ( httpRequest.status == 200 {
            // 정상적으로 수행
        } else {
            alert("문제 발생 : ' +  httpRequest.ststus);
        }
}
```

응답 데이터 사용하기

서버로 부터 응답이 도착한 것을 확인하고 (readyStatus 프로퍼티의 값이 4이고) 서버가 요청을 올바르게 수행했다면(status 프로퍼티 값이 200이면) 전송된 데이터 사용

웹 서버는 단순 텍스트 또는 XML의 두가지 형식으로 데이터를 전송

단순 응답테스트 참조

```
function callbackFunction()  {
      if ( httpRequest.readyState == 4 {
            if ( httpRequest.status == 200 {
                var txt = httpRequest.responseText;
               // txt를 사용해서 알맞은 작업 수행
            }
      } .
```

서버의 응답텍스트를 alert로 출력하는 예제

simpleAjaxApp.http ----------

```
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns=http://www.w3.org/1999/xhtml> <head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr" />
<title>간단한 Ajax 어플리케이션!</title>
<script type="text/javascript">
var httpRequest = null;
function getXMLHttpRequest() {
    if (window.ActiveXObject) {
            return new ActiveXObject("Msxml2.XMLHTTP");
    } else if (window.XMLHttpRequest) {  return new XMLHttpRequest();
    } else { return null;   }
 }
```

```
function load(url) {
	httpRequest = getXMLHttpRequest();
	httpRequest.onreadystatechange = viewMessage;
	httpRequest.open("GET", url, true);		httpRequest.send(null);
}
function viewMessage() {
	if (httpRequest.readyState == 4) {
		if (httpRequest.status == 200) { alert(httpRequest.responseText);
		} else {  alert("실패: "+ httpRequest.status);  }
	}
}
</script>
</head>
<body>
	<input type="button" value="simple.txt" onclick="load('simple.txt')"/>
	<input type="button" value="simple2.txt" onclick="load('simple2.txt')"/>
	<input type="button" value="simple.jsp" onclick="load('simple.jsp')"/>
	<input type="button" value="simple2.jsp" onclick="load('simple2.jsp')"/>
</body>
</html>
```

simple.txt ---------- 파일 엔코딩 ; utf-8
simple.txt의 응답 텍스트

simple2.txt ---------- 파일 엔코딩 ; euc-kr
simple2.txt의 응답 텍스트

simple.jsp ---------- 파일 엔코딩 ; utf-8
<%@ page contentType="text/plain; charset=utf-8" %>
simple.jsp의 응답 텍스트

simple2.jsp ---------- 파일 엔코딩 ; euc-kr
<%@ page contentType="text/plain; charset=euc-kr" %>
simple2.jsp의 응답 텍스트

택스트인 경우에는 utf-8일 경우에만 올바르게 출력되나
Jsp인 경우에는 두 가지 경우 다 제대로 출력

% 단순 캐릭터는 응답캐릭터 셋을 마음대로 지정할 수 없으나
jsp는 응답 캐릭터 셋을 설정 할 수 있다

동기방식과 비동기 방식의 실행차이

Open의 세번째 인수가 true이면 비동기방식 호출임

httpRequest.open(“GET”,”/test.txt”, true);

httpRequest.send(null);

. 비동기 방식으로 실행한 경우는 send()함수가 호출된 뒤에 곧바로 다음 코드 실행

. 동기 방식으로 호출한 경우에는 서버와의 완전히 종료된 후에 다음코드 실행

세번째 인수가 false인 경우에 동기 방식

파라미터의 한글 처리 방법

자바스크립트는 문자열을 utf-8로 인코딩 해주는 함수인 encodURIComponent() 함수 제공

이 함수가 요청 파라미터의 값을 UTF_8로 인코딩해 줄 수 있다

var params = “name = “ + encodeURLCommand(“홍길동”);

httpRequest.open(“GET”, “/hello.jsp?”+ params, true);

UTF-*로 인코딩하여 전송하므로 JSP같은 웹 서버 프로그램은 반대로 UTF-8로 디코딩하여

파라미터 값을 읽는다.

HttpServletRequest.setCharacterEncoding()메서드를 사용해서 읽어올 파라미터의 캐릭터

셋을 명시

```
<% request.setCharacterEncoding(“utf-8”);
    String name = request.getParameter(“name”);
%>
```

XMLHttpRequest모듈만들기 : httpRequest.js

```
var httpRequest = null;
function getXMLHttpRequest() {
    if (window.ActiveXObject) {
            return new ActiveXObject("Msxml2.XMLHTTP");
    } else if (window.XMLHttpRequest) {
            return new XMLHttpRequest();
     } else {
                return null;
     }
}
```

```
function sendRequest(url, params, callback, method) {
    httpRequest = getXMLHttpRequest();
    var httpMethod = method ? method : 'get';
    if (httpMethod != 'get' && httpMethod != 'get') {
        httpMethod = 'get';
    }
    var httpParams = (params == null || params == '') ? null : params;
    var httpUrl = url;
    if (httpMethod == 'get' && httpParams != null) {
        httpUrl = httpUrl + "?" + httpParams;
    }
    httpRequest.open(httpMethod, httpUrl, true);
    httpRequest.setRequestHeader(
        'Content-Type', 'application/x-www-form-urlencoded');
    httpRequest.onreadystatechange = callback;
    httpRequest.send(httpMethod == 'post' ? httpParams : null);
}
```

content-type, MIME, application/x-www-form-urlencoded

http 프로토콜에서 request 나 response 메세지가 발생할 때 그 메세지는 헤더(header)를 가지고 있다.
이 헤더는 http 트랜젝션이 동작하는 것에 대해서 기술한다. http header 의 속성 중에 content-type 이라는 것이 있다. 이는 request 나 response 메세지 모두에서 해당하는 값이다.

request 는 서버에 무언가를 요청한다는 의미이고, 이 때 서버에 뭔가를 요청할 때, 무엇을 요청할 것인지에 대한 정보 정도는 보내줘야 한다. request 메세지에 포함되어야 하는 정보가 있을 때, 그 데이터 타입이 어떠해야 하는지를 나타낸다. 모든 request 메세지에 다 지정해 줄 필요가 있는 것은 아니고, post 나 put 의 경우 content-type 을 지정해줘야 한다. content-type 이 가질 수 있는 값은 MIME type 의 값 들이다.

인터넷 메디아 타입(internet media type) 이라고도 불리는 이 타입은, Multipurpose Internet Mail Extension 의 약자이다. 처음에 MIME 타입은, SMTP(인터넷 프로토콜(IP) 에서 e-mail 을 보낼 때 사용되던 프로토콜) 를 통해 e-메일을 보낼 때 사용되는 것이 지금은 다른 프로토콜에서도 확장되어 사용되고 있다.

MIME 타입은 아스키코드(ASCII code) 로 기술 될 수 없는 메세지를 인코딩 하여 보내는 것을 가능하게 해준다.
즉, 영어 외의 일본어 한국어 같은 것들도 표현하여 메세지를 보낼 수 있다는 의미이다.
또한 그림, 음악, 영화, 컴퓨터 프로그램과 같은 8비트 바이너리 파일을 전자우편으로 보낼 수 있도록 한다.

MIME type 은 크게 두 부분으로 나누어 진다. type 과 subtype 으로 나누어 지는데, subtype 에 따라 추가적으로 파라미터를 가질 수 있다.
<%@ page contentType="text/html; charset=utf-8" pageEncoding="utf-8" %>

type 에 해당하는 부분이 text, subtype 에 해당하는 부분이 html 이며, charset=utf-8 은 subtype 이 html 일 때 가질 수 있는 파라미터에 해당하는 값이다.
request 메세지를 post 방식으로 서버에 보낼 때, MIME type 이 필요하다. 웹 브라우저에서 web form 엘리먼트로 부터 post 방식으로 데이터를 보낼 때, 표준 MIME type 이

application/x-www-form-urlencoded 이다.

helloApp.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns=http://www.w3.org/1999/xhtml> <head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr" />
<script type="text/javascript" src="httpRequest.js"></script>
<script type="text/javascript">
   function helloToServer() {
      var params = "name="+ encodeURIComponent(document.f.name.value);
      sendRequest("hello.jsp", params, helloFromServer, "post");
   }
   function helloFromServer() {
     if (httpRequest.readyState == 4) {
        if (httpRequest.status == 200) {  alert("서버응답:"+ httpRequest.responseText); }
      }
   }
</script></head>
```

```
<body>
<form name="f">
<input type="text" name="name" />
<input type="button" value="입력"
   onclick="helloToServer()" />
</form>
</body>
</html>
```

```
var start = text1.indexOf('<body>');
var end   = text1.indexOf('</body>');
var text2 = text1.substring((start+6), end);
```

hello.jsp

```
<%@ page contentType="text/plain; charset=euc-kr" %>
<%
    request.setCharacterEncoding("utf-8");
    String name = request.getParameter("name");
%>
안녕하세요, <%= name %> 회원님!
```

Ajax요소

. 사용자의 이벤트 발생할 경우 XMLHttpRequest 객체를 사용해서 서버에 작업 수행 요구

. 작업수행을 요청할 때 필요한 정보를 파라 미터로 전송

. 서버가 생성한 결과를 바탕으로 화면에 알맞은 결과 화면 출력

innerHTML

innerHTML 속성을 사용한 화면 변경

XMLHttpRequest

. 사용자가 이벤트를 발생시키면 웹 서버에 데이터 전송

. 웹 서버가 생성한 응답결과를 바탕으로 화면을 조작

사용결과 화면의 내요이 변경되거나 새로운 내용 추가

. HTML 요소의 innerHTML 속성에 HTML 코드 지정하기

. DOM(Document Object Model) API 사용하기

1. innerHTML 속성을 사용하는 기본 코드
   . 특정한 HTML 태그안에 들어갈 HTML 코드를 포함

---------- viewinnerHTML.html ----------

```
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"  xml:lang="ko" lang="ko">
<head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR"
   /><title>Insert title here</title>
<script type="text/javascript">
   function view() {
         var p1 = document.getElementById("p1");
         p1.innerHTML="기초 프로그래밍!!, <strong>Ajax</strong>"
         alert(p1.innerHTML);
   }
</script></head><body>
   <p id="p1"><strong>Ajax</strong>프로<br/>기초</p>
   <input type="button" value=보기 onclick="view()" />
</body></html>
```

. Document.getElementById()함수는 id 속성의 값을 사용하여 원하는 html태그를 사용할 수 있도록 해주므로 p1변수는 id 속성값이 p1인 <P>태그와 일치하게 된다

. innerHTML 속성을 사용하면 화면의 내요을 변경할 수 있다. 즉, innerHTML 속성에 원하는 HTML 코드만 값으로 주면 해당 내용 변경

1초마다 현재 시간을 출력해주는 예(clock.html)

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"> <head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR" /><title>시
   계</title>
<script type="text/javascript">
    function printTime() {
        var clock = document.getElementById("clock");
        var now = new Date();
        clock.innerHTML = now.getFullYear() + "년 “+ (now.getMonth()+ 1) + "월 "+
              now.getDate() + "일 “+ now.getHours() + "시 “+ now.getMinutes() + "분 "+
              now.getSeconds() + "초";
        setTimeout("printTime()", 1000);
    }
    window.onload = function() { printTime();  }
</script> </head>
<body>
        현재 시간은 <span id="clock"></span> 입니다.
</body>
</html>
```

웹 서버 응답결과를 innerHTML을 사용하여 반영

웹 서버의 응답결과와 innerHTML 을 사용하여 동적으로 변경하는 방법

. 웹 서버의 응답 텍스트의 값이 HTML 코드이고 그 응답 텍스트를 innerHTML을 사용하여 그 대로 화면에 반영

. 웹 서버의 응답텍스트가 임의의 포맷에 맞춰져 있고 자바 스크립트는 그 응답 텍스트를 분석해서 HTML 코드를 생성한 뒤 innerHTML을 사용해서 화면에 반영

1. 응답텍스트로 생성한 HTML코드를 그대로 반영하기(getNewsTitles.jsp)

```
<%@ page contentType = "text/plain; charset=euc-kr" %>
<%
    String[] titles = {
        "서재응 완벽투구… 게레로 3구 삼진",
        "리틀 감독 '서재응 12일부터 고정 선발'",
        "박찬호 김선우 시범경기 등판 호투"
    };
    for (int i = 0 ; i < titles.length ; i++) {
%>
  <% if (i == 0) { %><strong><% } %>
  <%= titles[i] %>
  <% if (i == 0) { %></strong><% } %>
<br/>
<%  }  %>
```

News.html

```
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko“><head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr" /><title>뉴스
    </title>
<script type="text/javascript" src="httpRequest.js"></script>
<script type="text/javascript">
   function loadNews() {  sendRequest("getNewsTitles.jsp", null, loadedNews, “get");    }
   function loadedNews() {
        if (httpRequest.readyState == 4) {
            if (httpRequest.status == 200) {
                var newsList = document.getElementById("newsList");
                newsList.innerHTML = httpRequest.responseText;
            }
        }
    }
    window.onload = function() {  loadNews();  }
</script> </head>
<body>
       <div id="newsList"></div>
</body></html>
```

CSV양식의 응답 텍스트를 분석해서 화면에 출력하기

- 응답 텍스트를 특정 포맷으로 생성하고 자바 스크립트에서 응답 텍스트를 분석해서 원하는 화면을 직접 생성하는 방법

% CSV(Comma Separated Value) 값을 콤마로 구분해서 표시하는 데이터 양식

---------- getMaxTemperature.jsp ----------

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%><!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title></head><body>
<%    double[] temp = { 9.8, 12.2, 24.7, 23.7, 23.1 };
        for (int i = 0 ; i < temp.length; i++ ) {
              out.println(temp[i]);
              if (i != (temp.length -1)) {    out.println(",");   }
        }
%>
</body></html>
```

Javascript의 split()함수 ; 문자열을 지정한 문자로 구분해서 배열로 생성하는 기능

문자형을 숫자로 변경하기 위해서 parseInt() 또는 parseFloat() 함수 사용

--- maxTemperature.html ----------

```
<!DOCTYPE html><html><head><meta charset="EUC-KR">
<title>Insert title here</title>
<script type="text/javascript" src="httpRequest.js"></script>
<script type="text/javascript">
    function maxView() {
        sendRequest("getMaxTemperature.jsp", null, maxResult, "GET");
    }
    function maxResult() {
        if (httpRequest.readyState == 4) {
            if (httpRequest.status = 200) {
                var csvStr1 = httpRequest.responseText;
                var temp    = document.getElementById("temp");
                temp.innerHTML = csvStr1;
                var start = csvStr1.indexOf('<body>');
                var end   = csvStr1.lastIndexOf('</body>');
                var csvStr   = csvStr1.substring(start+ 6, end);
                var tempCsv = csvStr.split(",");
                var arrTemp = new Array(tempCsv.length);
```

```
                for(var i = 0 ; i < tempCsv.length; i++) {
                    arrTemp[i] = parseFloat(tempCsv[i]);
                }
                var max = 0;
                for (var i = 0; i < arrTemp.length; i++) {
                    if ( max < arrTemp[i]) { max = arrTemp[i]; }
                }
                var tempMax = document.getElementById("max");
                tempMax.innerHTML = max;
            }
        }
    }
</script></head><body>
<div id = "temp"></div><p>
최고 기온은 <span id="max"></span>입니다<p>
<input type="button" value="최고온도 보기" onclick="maxView()">
</body>
</html>
```

자바스크립트 디버깅 콘솔 만들기

. innerHTML 속성을 사용하면 alert() 함수로 출력할 메시지를 innerHTML 속성을 사용해서 특정 HTML 요소에 덧붙여 출력하는 방식을 사용하면 된다

---------- log.js -----

```
function log(msg) {
    var console = document.getElementById("debugConsole");
    if (console != null) {
        console.innerHTML + = msg + "<br/>";
    }
}
```

---

log()함수는 innerHTML을 사용해서 “debugConsole” 요소에 인자로 전달 받은 msg를 덧붙여 준다.

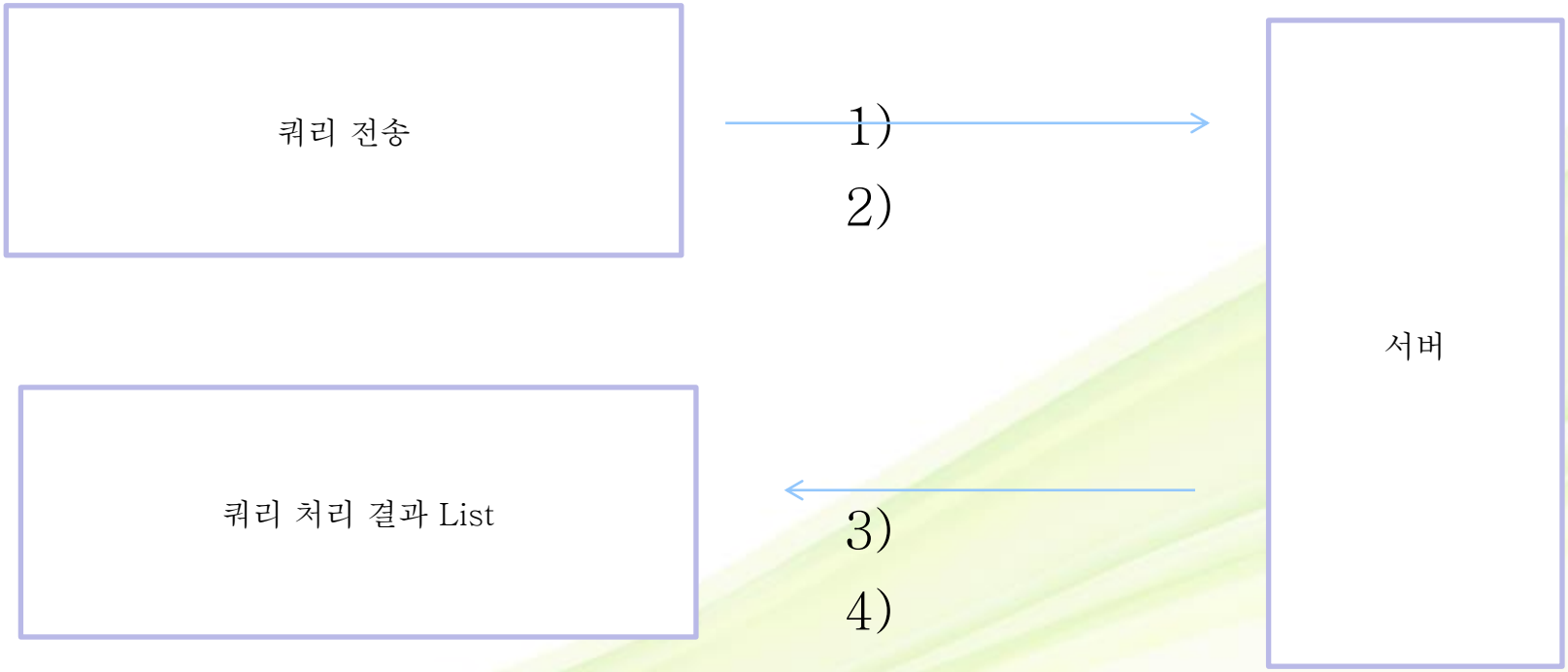
```
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr" />
<title>log 테스트</title>
<script type="text/javascript" src="log.js"></script>
<script type="text/javascript">
   function loadList() {
       log("loadList 시작");
       // 알맞은 작업
       log("loadList 종료");
   }
   window.onload = function() {
       log("window.onload 호출");
       loadList();
   }
</script></head>
<body>
       <div id="debugConsole" style="border: 1px solid #000"></div>
</body>
</html>
```

제시어 기능 구현하기

검색어를 입력하면 유사한 검색어 목록을 출력하는 기능

제시어 기능 흐름

1) 입력한 제시어를 서버에 전송
2) 전달받은 검색어를 사용하여 제시어 목록을 추출
3) 제시어 목록을 지정한 양식으로 클라이언트에 전송
4) 응답 텍스트를 분석한 후에 제시어 목록을 화면에 출력



서버 측 제시어 기능 구현

여기서는 배열로 되어 있지만 DB에서 읽어오는 것
search() ; keyword를 사용하여 제시어 목록을 추출해 주는 메서드

---------- suggest.jsp ----------

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="java.util.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title></head><body>
<%!
      String[] keywords = {"AJAX", "AJAX 실전 프로그래밍", "자바",
            "자바 프로그래밍","자바서버 페이지", "자바스터디", "자바서비스", "자바캔"} ;
      public List<String> search(String keyword) {
            List<String> result = new ArrayList<String>();
            keyword = keyword.toUpperCase();
            if (keyword == null || keyword.equals(""))
                return java.util.Collections.EMPTY_LIST;
            for(int i = 0; i < keywords.length; i++ ) {
                 if (((String)keywords[i]).startsWith(keyword)) {
                       result.add(keywords[i]);
                  }
            }
             return result;
        }
%>
```

```
<%
	request.setCharacterEncoding("utf-8");
	String keyword = request.getParameter("keyword");
	List<String> keywordList = search(keyword);

	out.print(keywordList.size());
	out.print("|");
	for (int i = 0 ; i < keywordList.size() ; i++) {
		String key = (String)keywordList.get(i);
		out.print(key);
		if (i < keywordList.size() - 1) {
			out.print(",");
		}
	}
%>
</body>
</html>
```

제시어 목록을 CSV양식으로 출력
n|제시어1,제시어2, …. ,제시어n
n은 제시어 갯수

클라이언트 측 제시어 기능 구현

클라이언트에서 처리할 일

- 사용자가 검색어를 입력하면 해당 검색어를 서버에 전송하기
- 서버로 부터 받은 응답 텍스트를 사용해서 제시어 목록을 출력하기

1. 입력한 검색어 서버에 전송하기

input 태그에 onkeydown이벤트 처리

```
<form name="search">
    <input type="text" name="keyword" id="keyword" onkeydown="startSuggest()" />
    <input type="button" value="검색" />
    <div id="suggest" style="display:; postion: absolute; left: 0px; top: 30px;">
      <div id="suggestList"></div>
    </div>
```

서버가 전송한 제시어 목록이 추력될 곳

검색어를 입력하면 onkeydown에 명시된 startSuggest()자바스크립트 함수 호출

style 속성의 값에 display는 화면에 div영역을 표시하지 않는다는 뜻

-- startSuggest() 및 sendKeyword() 함수 ---

```
var checkFirst = false;
var lastKeyword = '';  // 검색어가 변경 되었는지 여부 (문자열 비교)
var loopSendKeyword = false;   // 사용자가 제시어를 선택했는지 유무
function startSuggest() {
        if (checkFirst == false) {   // 한번도 호출하지안았다면
                setTimeout("sendKeyword();", 500);  // 0.5ch 후에 sendKeyword()함수 실행
                loopSendKeyword = true;
        }
        checkFirst = true;  // startSuggest()가 한번 호출이 되면 true
}
function sendKeyword() {
        if (loopSendKeyword == false) return;
        var keyword = document.search.keyword.value;
        if (keyword == '') {
                lastKeyword = '';
                hide('suggest');
        } else if (keyword != lastKeyword) {
                lastKeyword = keyword;
                if (keyword != '') {  // 서버에 입력한 검색어를 전송
                        var params = "keyword="+ encodeURIComponent(keyword);
                        sendRequest("suggest.jsp", params, displayResult, 'POST');
                } else {     hide('suggest');              // post방식으로 전송
                }
        }
        setTimeout("sendKeyword();", 500);  // 0.5초 후에 sendKeyword()함수를 호출
}
```

. startSuggest() 함수는 0.5초 후에 sendKeyword()함수를 호출하고 다른 작업은 없음
. sendKeyword() 함수의 실행이 끝나면 0.5ch 후에 또 sendKeyword() 함수를 실행
% 0.5ch마다 함수를 실행하는 이유는 IE는 영움 한글 관계없이 키보드 누를 때마다 startSuggest()함수를 실행하지만 firefox는 영문은 키보드 누룰 때마다 호출되지만 한글은 호출되지 않기 때문임

키보드에 입력할 때만 서버에 요청 보내기 위한 방법(0.5초마다 호출하지 않고)

```
var checkFirst = false
  처음 startSuggest() 함수를 호출할 때 checkFirst를 true로 변경하여 이후에 startSuggest() 함 수가 호출
    되면 sendKeyword() 함수를 호출하지 않음
if (checkFirst == false) {   // 한번도 호출하지안았다면
            setTimeout("sendKeyword();", 500);  // 0.5초 후에 sendKeyword()함수 실행
            loopSendKeyword = true;
 }
            checkFirst = true;  // startSuggest()가 한번 호출이 되면 true

var lastKeyword 는 현재 사용자가 입력한 검색어가 다른 경우에만 서버에 요청 0.5초 후에 재비교
} else if (keyword != lastKeyword) {
                        lastKeyword = keyword;
```

사용자가 제시된 검색어를 선택한 후에는 서버 요청 종료

```
var loopSendKeyword의 값이 false이면 sendKeyword() 함수를 호출하지 않음
if (loopSendKeyword == false) return;
```

서버에서 받은 제시어 목록 출력하기

sendKeywor()에서 제시어를 전송하는 코드

```
var params = "keyword="+ encodeURIComponent(keyword);
sendRequest("suggest.jsp", params, displayResult, 'POST');
```

서버에서 응답텍스트가 도착하면 displayResult(_함수가 호출되어 제시어 목록을 화면에 출력

```
function displayResult() {
    if (httpRequest.readyState == 4) {
        if (httpRequest.status == 200) {
            var resultText = httpRequest.responseText;   var result = resultText.split('|');
            var count = parseInt(result[0]);                    var keywordList = null;
            if (count > 0) {
                keywordList = result[1].split(',');   var html = '';
                for (var i = 0 ; i < keywordList.length ; i++ ) {
                    html + = "<a href=₩"javascript:select('“+  keywordList[i]+ "')₩">"+
                     keywordList[i]+ "</a><br/>";
                 }
                 var listView = document.getElementById('suggestList');
                  listView.innerHTML = html;   show('suggest');
            } else {  hide('suggest'); }
        } else { alert("에러 발생: "+ httpRequest.status); }
    }
}
```

Suggst.jsp는 제시어 목록을 제시하는 양식 : n|자바 서버페이지,자바프로그래밍
제시어 개수와 제시어 목록부분을 구별하는 작업

var resultText = httpRequest.responseText; var result = resultText.split('|');
따라서 작업 결과는 {“2”,”자바 서버페이지,자바프로그래밍”}

keywordList에 저장

```
var count = parseInt(result[0]);
 var keywordList = null;
            if (count > 0) {
               keywordList = result[1].split(',');
```

위 프로그램 수행 결과
{”자바 서버페이지”,”자바프로그래밍”}

화면에 출력하는 코드

```
var html = '';
for (var i = 0 ; i < keywordList.length ; i++) {
      html += "<a href=₩"javascript:select('“+  keywordList[i]+ "')₩">"+
       keywordList[i]+ "</a><br/>";
}
var listView = document.getElementById('suggestList');
listView.innerHTML = html;   show('suggest');
```

제시어를 선택한 경우에 처리

제시어가 출력될 때 사용된 html코드

자바서버 페이지</br>

자바 프로그래밍

Select함수

```
function select(selectedKeyword) {
        document.search.keyword.value = selectedKeyword;
        loopSendKeyword = false;
        checkFirst = false;
        hide('suggest');
}
```

선택한 제시어가 입력받는 input값으로 할당

제시어 목록을 hide함수를 써서 감추고 선택한 제시어 만 보이게 하

완전한 suggestclient.html 코드

```
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr" />
<title>서제스트</title>
<script type="text/javascript" src="httpRequest.js"></script>
<script type="text/javascript">
var checkFirst = false;
var lastKeyword = '';
var loopSendKeyword = false;

function startSuggest() {
    if (checkFirst == false) {
        setTimeout("sendKeyword();", 500);
        loopSendKeyword = true;
    }
    checkFirst = true;
}
```

```
function sendKeyword() {
    if (loopSendKeyword == false) return;
    var keyword = document.search.keyword.value;
    if (keyword == '') {
        lastKeyword = '';
        hide('suggest');
     } else if (keyword != lastKeyword) {
      lastKeyword = keyword;

          if (keyword != '') {
               var params = "keyword="+ encodeURIComponent(keyword);
               sendRequest("suggest.jsp", params, displayResult, 'POST');
           } else {
               hide('suggest');
          }
     }
    setTimeout("sendKeyword();", 500);
}
```

```
function displayResult() {
    if (httpRequest.readyState == 4) {
        if (httpRequest.status == 200) {
            var resultText1 = httpRequest.responseText;
            var start = resultText1.indexOf('<body>');
            var end = resultText1.lastIndexOf('</body>');
            var resultText = resultText1.substring(start+ 6, end);
            var result = resultText.split('|');      var count = parseInt(result[0]);
            var keywordList = null;
            if (count > 0) {
                keywordList = result[1].split(',');
                var html = '';
                for (var i = 0 ; i < keywordList.length ; i++ ) {
                    html += "<a href=₩"javascript:select('“+  keywordList[i]+ "')₩">"+
                            keywordList[i]+ "</a><br/>";
                }
                var listView = document.getElementById('suggestList');
                listView.innerHTML = html;
                show('suggest');
            } else {  hide('suggest'); }
        } else {  alert("에러 발생: "+ httpRequest.status);  }
    }
}
```

```
function select(selectedKeyword) {
    document.search.keyword.value = selectedKeyword;
    loopSendKeyword = false;  checkFirst = false;   hide('suggest');
}
function show(elementId) {
      var element = document.getElementById(elementId);
      if (element) {  element.style.display = '';  }
}
function hide(elementId) {
     var element = document.getElementById(elementId);
     if (element) {  element.style.display = 'none'; }
}
</script>
<style>    #view {  border: 1px solid #999;  }</style>
 </head><body>
   <form name="search">
       <input type="text" name="keyword" id="keyword" onkeydown="startSuggest()" />
       <input type="button" value="검색" />
       <div id="suggest" style="display:; postion: absolute; left: 0px; top: 30px;">
       <div id="suggestList"></div>
</div></form>
</body></html>
```

DOM(Document Object Model)과 XML

DOM(Document Object Model)과 HTML/XML

DOM문서를 객체로 표현하기 위한 표준으로서 HTML이나 XML 등의 문서를 객체로 표현할 때 사용되는 API

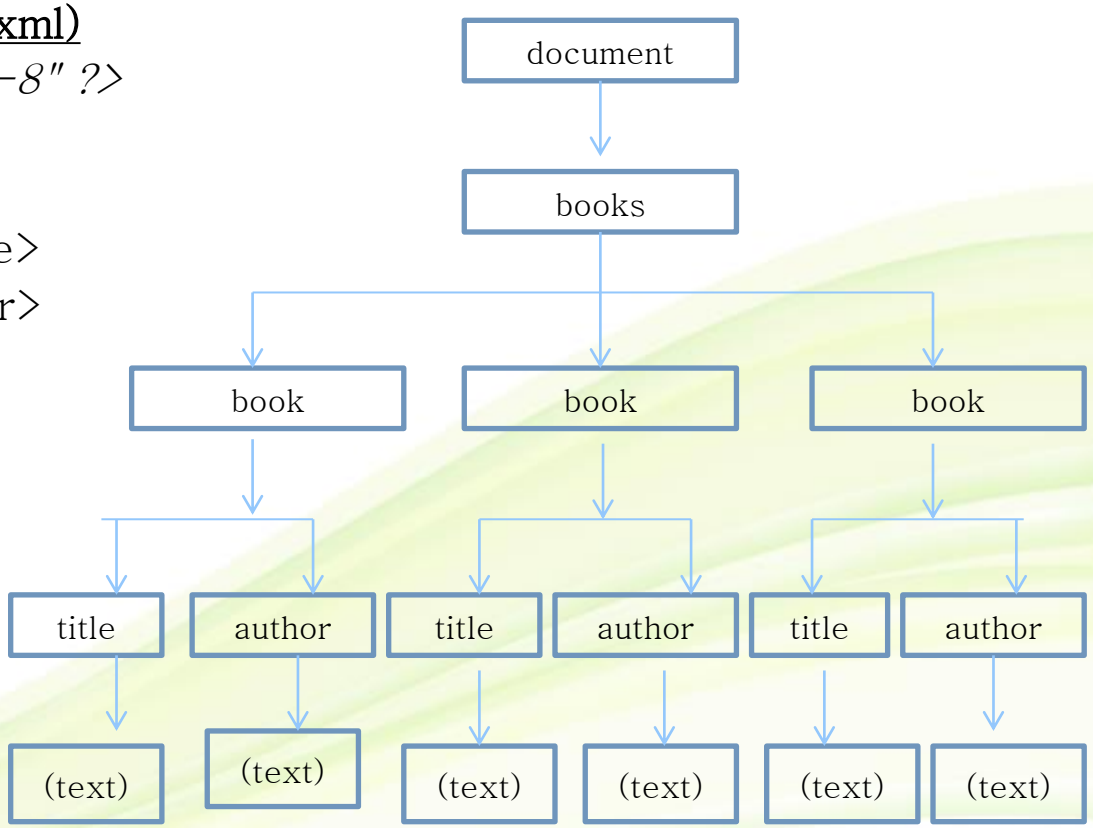
DOM API는 문서를 Tree구조로 표현하여 이해하기 쉬움

XMLHttpRequest객체는 응답텍스트 대신 XML응답 결과를 사용할 수 있음

이때 DOM API를 사용해서 서버가 생성한 XML로부터 데이터를 추출

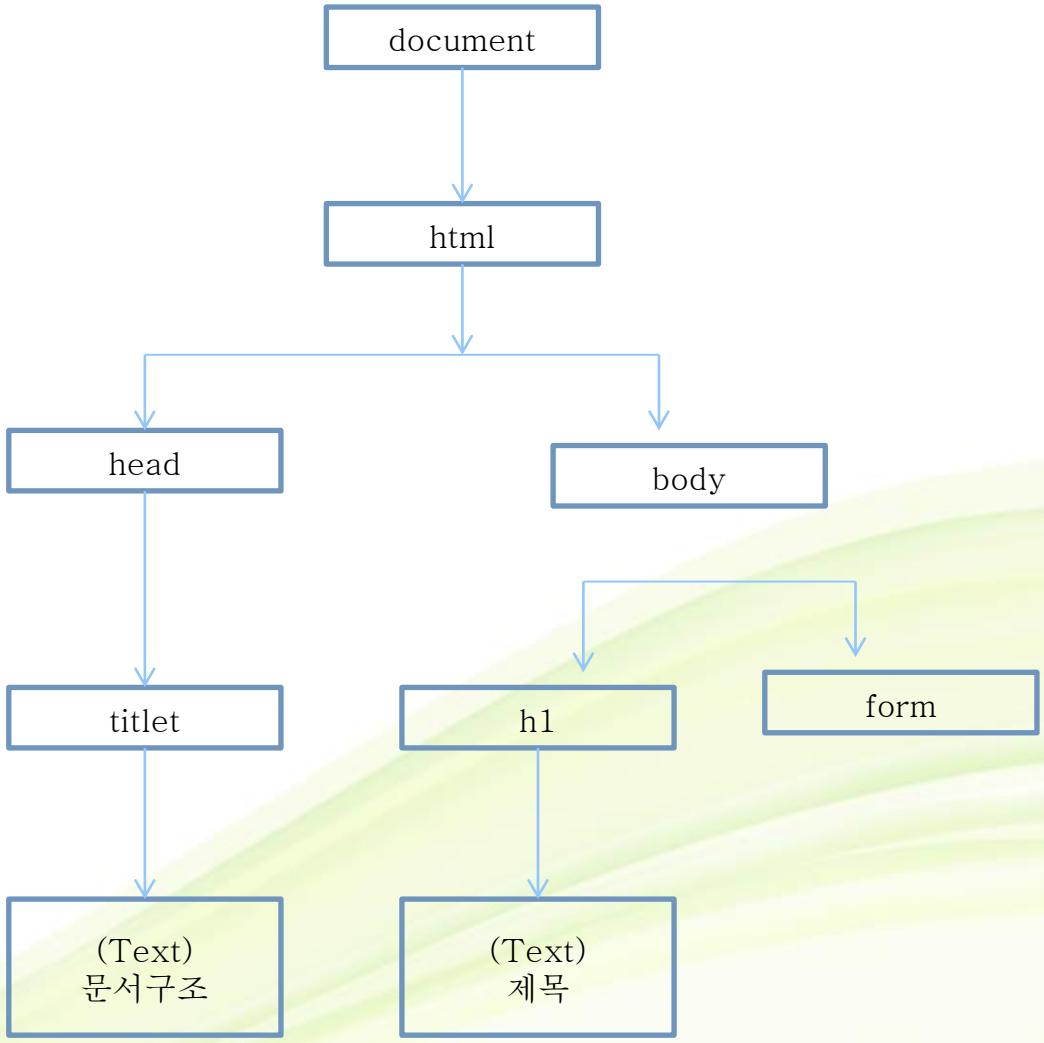
1. XML 문서와 DOM트리구조(books.xml)

```
<?xml version="1.0" encoding="utf-8" ?>
<books>
<book>
        <title>프로젝트 생존 전략</title>
        <author>스티브 맥코넬</author>
</book>
<book>
        <title>JSP  프로그래밍</title>
        <author>주니</author>
</book>
<book>
        <title>웹 표준</title>
        <author>댄 씨더홈</author>
</book>
</books>
```

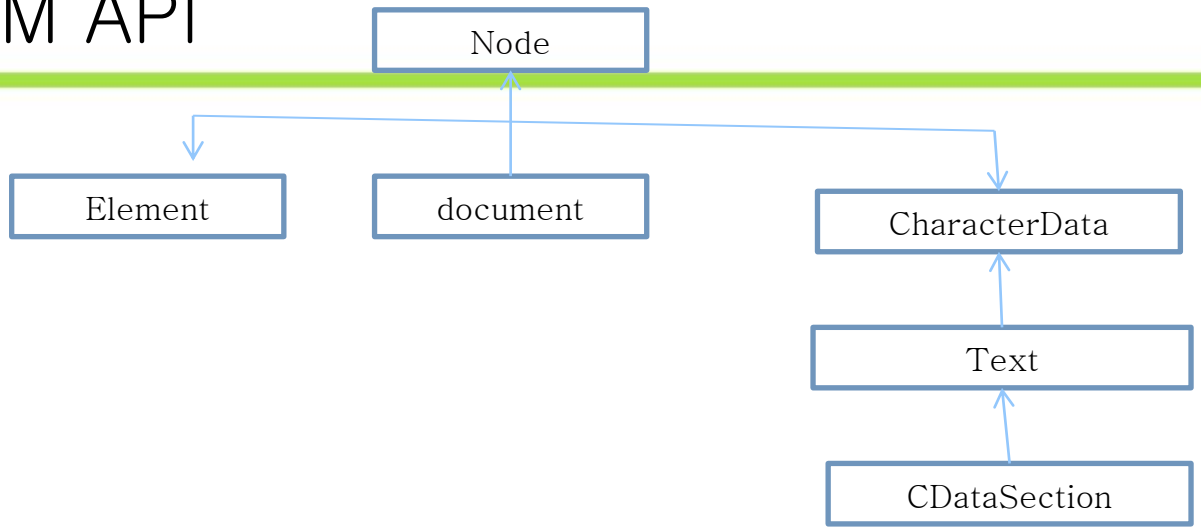


HTML문서와 DOM 트리구조

```
<html>
<head>
<title>문서구조</title>
</head>
<body>
<h1>제목</h1>
<form></form>
</body>
</html>
```



주요 DOM API



DOM API에서 모든 것은 Node로 표현하며 문서의 구성요소들은 모두 Node또는 하위 인터페이스로 매핑

Document ; 문서 전체

Element : 각 태그

Text ; 문자열 데이터가 Text 노드로 표현

CDataSection : Cdata 영역의 문자열 값 저장

1) Node 인터페이스의 주요 프로퍼티

nodeName, nodeValue, nodeType, parentNode,
childNodes(자식노드 목록), firstChild, lastChild, ownerDocument
previousSibling ; 현재 노드와 같은 부모를 갖는 자식 노드 중 현재 노드 이전의 자식노드
nextSibling : 현재 노드와 같은 부모를 갖는 자식 노드 중 현재 노드 다음의 자식노드

NodeList ; childNodes로 구한 자식 노드에 접근할 수 있도록 하기 위해
- length : NodeList에 저장된 노드의 개수
- item(i) ; index i에 저장된 노드를 구함, 0부터 시작

nodeName – div
nodeValue – null

오늘의 하루
nodeName – #text
nodeValue – 오늘의 하루

< Node에 정의된 nodeType관련 상수 값>

멤버 필드	상수값	멤버 필드	상수값
ELEMENT_NODE	**1**	**PROCESSING_INSTRUCTION_NODE**	**7**
ATTRIBUTE_NODE	**2**	**COMMENT_NODE**	**8**
TEXT_NODE	**3**	**DOCUMENT_NODE**	**9**
CDATA_SECTION_NODE	**4**	**DOCUMENT_TYPE_NODE**	**10**
ENTITY_REFERENCE_NODE	**5**	**DOCUMENT_FRAGMENT_NODE**	**11**
ENTITY_NODE	**6**	**NOTATION_NODE**	**12**

Document 인터페이스의 주요 프로퍼티 및 함수

Document 인터페이스가 제공하는 Element 노드 구하기 함수

- NodeList getElementsByTagName(String tagName) : 지정한 이름의 태그에 해당하는 모든 Element의 목록을 구한다
- Element getElementById(String elementId) : id속성의 값이 elementId인 태그를 구한다

% span태그의 목록을 가져올 때

```
var spanList = document.getElementsByTagName("span")
for (var I = 0; i<spanList.length; i++) {
    var span = spanList.item(i);
    …..
}
```

% var bSpan = document.getElementById("b");

3) Element 인터페이스의 주요 프로퍼티 함수

Element 인터페이스는 태그를 나타내는 인터페이스다. 즉 html, body, p, div 태그 등을 사용할 때 사용되는 인터페이스가 Element

Element 인터페이스의 속성관련 함수

- String getAttribute(String name) : name에 해당하는 속성의 값
- setAttribute(String name, String value) : 이름이 name인 속성의 값을 value로 저장
- removeAttribute(String name) : 이름이 name인 속성을 제거

DOM API를 사용해서 HTML 문서의 정보를 탐색하는 예제 (usingDOM.html)

```
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN“
   http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>
 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko“><head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr" />
<title>DOM API를 사용한 HTML 문서 접근</title>
<script type="text/javascript" src="log.js"></script>
<script type="text/javascript">
  window.onload = function() {
      var roodNode = document.documentElement;   log("root 태그: "+ roodNode.tagName);
      var bodyNode = document.getElementsByTagName("body").item(0);
      log("body 태그: "+ bodyNode.tagName);
      var spanList = document.getElementsByTagName("span"); log("span 태그의 개수:
    "+ spanList.length);
      for (var i = 0 ; i < spanList.length ; i++ ) {
            var span = spanList.item(i);   log((i+ 1)+ "번째 span의 id : "+ span.getAttribute("id"));
     }
     var debugConsoleDiv = document.getElementById("debugConsole");
     Log("debugConsole 요소: "+ debugConsoleDiv.tagName);
     var bodyLastChild = bodyNode.lastChild;
    Log("body의 마지막 자식 노드: "+ bodyLastChild.nodeName);
}
</script></head>
```

```
<body>
      <SPAN id="a">a</SPAN>      <p>test<span id="b">b</span></p>
      <div><p>p</p><span id="c">c</span></div>
      <div id="debugConsole" style="border: 1px solid #000"></div>
</body></html>
```

실행결과

A

testb

p

c

root 태그: HTML

body 태그: BODY

span 태그의 개수: 3

1번째 span의 id : a

2번째 span의 id : b

3번째 span의 id : c

debugConsole 요소: DIV

body의 마지막 자식 노드: #text

% 마지막 text노드는 와 </dody> 사이의 “₩n”처리를 text 노드로 취급

DOM API를 사용하여 문서 구조를 변경하기

1) Document 인터페이스의 Element 노드 생성 함수

. Element createElement(String tagName) : 지정한 태그 이름을 갖는 Element 노드를 생성

. Text createTextNode(String text) ; text를 값으로 갖는 Text노드를 생성

<p>태그 생성

```
var pNode = document.createElement("p");
var textNode = document.createTextNode("테스트");
pNode.appendChild(textNode);
```

2) Node 인터페이스의 DOM 트리 변경 관련 함수

Node inserBefore(Node newChild, Node refChild) 현재 노드의 자식 노드인 refNode노드의 previousSibling자리에 newChild노드를 삽입한다.

Node replaceChild(Node newChild, Node refChild) 현재 노드의 oldChild노드를 새로운 newNode로 교체한다

Node removeChild(Node oldChild) 현재 노드의 자식노드인 oldChild 노드를 현재노드에서 제거

Node appendChild(Node newChild) newChild를 현재 노드의 마지막 자식노드로 추가

DOM API를 사용하여 HTML 화면 변경하는 예제

. DOM트리를 변경할 수 있다는 것은 자바스크립트를 사용해서 HTML 화면을 원하는 대로 변경할 수 있다는 것을 의미한다.

---- changeUsingDom.html ----------

```
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN“
   http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko“><head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr" />
<title>DOM을 사용한 변경</title><script type="text/javascript">
  var count = 0;
  function appendItem() {
     count++;     var newItem = document.createElement("div");
     newItem.setAttribute("id", "item_" + count);
     var html = '새로 추가된 아이템['+ count+ ']'+ '<input type="button" value="삭제"
          onclick="removeItem(' + count + ')"/>';
     newItem.innerHTML = html;  var itemListNode = document.getElementById('itemList');
    itemListNode.appendChild(newItem);
  }
  function removeItem(idCount) {
     var item = document.getElementById("item_"+ idCount);
     if (item != null) {  item.parentNode.removeChild(item);  }
  }
</script> </head>
<body>
    <input type='button' value='추가' onclick='appendItem()' />
    <div id="itemList"></div>
</body>  </html>
```

XML 응답 사용하기

- XMLHttpRequest객체는 서버로부터 XML문서를 응답으로 읽어올 ㅅ 있으며 서버에서 읽어온 XML문서는 DOM트리로 반환되어 저장된다

1. 서버에서 XML응답 생성하기

   XMLHttpRequest객체가 XML 문서를 응답으로 받으려면 서버에서 응답을 XML로 생성

```
<?xml version="1.0" encoding="euc-kr" ?>
<%@ page contentType="text/xml; charset=euc-kr" %>
<books>
    <book>
        <title>프로젝트 생존 전략</title>
        <author>스티브 맥코넬</author>
    </book>
    <book>
        <title>JSP 2.0 프로그래밍</title>
        <author>고녀석</author>
    </book>
    <book>
        <title>웹 표준</title>
        <author>댄 씨더홈</author>
    </book>
</books>
```

responseXML로 XML응답 읽어오기

. XMLHttpRequest 객체는 XML문서를 응답데이터로 사용할 수 있다. XML 응답을 사용할 때는 responseXML을 사용하면 된다.

----- viewBooks.html -----

```
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr" />
<title>책정보 보기</title>
<script type="text/javascript" src="httpRequest.js"></script>
<script type="text/javascript">
function loadBooks() {
   sendRequest("books.jsp", null, loadedBooks, "GET");
}
function loadedBooks() {
   if (httpRequest.readyState == 4) {
       if (httpRequest.status == 200) {
           var xmlDoc = httpRequest.responseXML;
           var bookList = xmlDoc.getElementsByTagName("book");
           var message = "책 개수 : "+ bookList.length+ "권₩n";
```

```
        for (var i = 0 ; i < bookList.length ; i++) {
            var book = bookList.item(i);
            var titleValue = book.getElementsByTagName("title").item(0)
                    .firstChild.nodeValue;
            var authorValue = book.getElementsByTagName("author").item(0)
                    .firstChild.nodeValue;
            message += titleValue + "(" + authorValue + ")₩n";
        }
        alert(message);
    }
  }
}
window.onload = function() {
   loadBooks();
}
</script>
</head>
<body>
   책 정보를 alert 으로 출력
</body></html>
```

XSL/T를 사용하여 XML을 HTML로 변환하기

. XML문서의 각 노드로부터 원하는 정보를 하나씩 읽어와 HTML 코드를 생성한 뒤 innerHTML을 사용하서 원하는 정보를 생성할 수 있다

. XSL/T는 XML문서를 XSL을 사용해서 원하는 문서를 변환 시킬 때 사용하는 기술 XML문서를 HTML 문서로 변환할 때 많이 사용한다

-------- books.xsl ---------------

```
<?xml version="1.0" encoding="euc-kr" ?>

<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method = "html" indent="yes" encoding="euc-kr" />
    <xsl:template match="books">
    <ul>
        <xsl:for-each select="book">
        <li><b><xsl:value-of select="title" /></b>
        (<i><xsl:value-of select="author" /></i>)
        </li>
        </xsl:for-each>
    </ul>
    </xsl:template>
</xsl:stylesheet>
```

```
-------  bookList.html  -------------
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr" />
<title>책 목록</title>
<script type="text/javascript" src="httpRequest.js"></script>
<script type="text/javascript">
   var xmlDoc = null;    // 서버에서 읽어온 XML문서의 XSL문서를 저장한다
   var xslDoc = null;
   function loadBooks() {
        sendRequest("books.jsp", null, loadedBooksXML, "GET");  // XML문서 로딩
    }
    function loadedBooksXML() {
        if (httpRequest.readyState == 4) {
            if (httpRequest.status == 200) {
                xmlDoc = httpRequest.responseXML;
                sendRequest("books.xsl", null, loadedBooksXSL, "GET");
             } // XML읽은 뒤 X니 무서 로딩
        }
}
```

```
    function loadedBooksXSL() {
        if (httpRequest.readyState == 4) {
            if (httpRequest.status == 200) {  xslDoc = httpRequest.responseXML;
                doXSLT();              }        }  // XSL을 읽은 뒤 변환 실행
    }
    function doXSLT() {
        if (xmlDoc == null || xslDoc == null) return;
            var bookList = document.getElementById("bookList");
            if (window.ActiveXObject) {
                    bookList.innerHTML = xmlDoc.transformNode(xslDoc);  // IE인 경우
   변환
            } else {  var xsltProc = new XSLTProcessor();
                    xsltProc.importStylesheet(xslDoc);  // 기타 브라우저 변환 처리
                    var fragment = xsltProc.transformToFragment(xmlDoc, document);
                    bookList.appendChild(fragment);                }
        }
        window.onload = function() {  loadBooks();
    }
</script>  </head>
<body>
      <h1>신규 책 목록</h1>
      <div id="bookList"></div>
</body> </html>
```

자바스크립트 객체.JSON 표기법

Prototype을 사용한 자바스크립트 클래스 만들기

. 자바스크립트에는 Date, RegExp, String 등의 클래스가 잇고, 이 객체를 사용하여 새로운 객체를 생성할 수 있다.

```
var now = new Date();  var year = now.getFullYear(); var month = now.getMonth();
```

. 자바스크립트가 기본적으로 제공하고 있는 클래스에 추가적으로 개발자가 직접 새로운 클래스를 정의할 수 있고, 객체기반으로 자바스크립트 프로그램을 할 수 있다.

. 새로운 클래스를 정의하기 위해서는 먼저 function을 사용해서 클래스을 사용해서 클래스를 추가한다.

```
클래스이름 = function(파라미터) {    }
Member = function(id, name) {
    this.id = id;
    this.name = name;
}
```

함수 정의에서는 'this프로퍼티'을 사용해서 객체의 프로퍼티 값에 파라미터로 전달 받은 값을 할당해 주고 있다.

```
var mem = new Member('aaa', '주니');
클래스이름.prototype.함수이름 = function(파라미터);
Member.prototype.setValue = function(newId, newName) {
  this.id = newId;
  this.name = newName;
}
var mem = new Member('aaa', '주니');   mem.setValue ('aaa', '주니');
```

--------- member.js ----------

```
Member = function(name, id, securityNo) {
    this.name = name;
    this.id = id;
    this.securityNo = securityNo;
}
Member.prototype.setValue = function(newName, newId, newSecurityNo) {
    this.name = newName;
    this.id = newId;
    this.securityNo = newSecurityNo;
}
Member.prototype.getAge = function() {
    var birthYear = parseInt(this.securityNo.substring(0, 2));
    var code = this.securityNo.substring(6,7);
    if (code == '1' || code == '2') {       birthYear += 1900;
    } else if (code == '3' || code == '4') {       birthYear += 2000;  }
    var today = new Date();
    return today.getFullYear() - birthYear;
}
Member.prototype.toString = function() {
     return this.name +  "[" +  this.id +  "]";
}
```

```
---------    useMember.html   ----------
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko“><head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr" />
<title>Member 클래스 사용</title>
<script type="text/javascript" src="log.js"></script>
<script type="text/javascript" src="member.js"></script>
<script type="text/javascript">
   window.onload = function() {
       var mem = new Member("biginner", "주니", "7700001000000");
       log('변경전');
       log('회원 : ' + mem.toString());   log('나이 : ' + mem.getAge());
       mem.setValue("excellent", "길동", "8000001000000");
       log('변경후');
       log('회원 : ' + mem.toString());  log('나이 : ' + mem.getAge());
   }
</script> </head>
<body>
      <div id="debugConsole"></div>
</body></html>
```

Object를 사용한 개별 객체에 프로퍼티 확정

클래스를 정의하지 않은 상태에서 객체에 직접적으로 프로퍼티나 함수를 추가할 수 있는 방법

```
var mem = new Object();
mem.id = 'juni';
mem.name = '주니';
mem.secuityNo = 7700001000000;
```

새로운 함수를 추가하는 방법 : toString()함수 추가

```
var mem = new Object();
mem.id = 'juni';
mem.name = '주니';
mem.toString = function() {
    return this.name + "[" + this.id + "]";
}
alert(mem.toString();
```

객체를 생성한 뒤 직접프로퍼티나 함수를 추가하는 방식은 Member 클래스와 같은 별도의 클래스를 정의하지 않고 그때그때 원하는 객체를 만들고 싶을 때 사용

JSON 표기법

- JSON(JavaScript Object Notation) 표기법은 서로 다른 프로그래밍 언어간에 데이터를 교환하기 위한 표기법

. 이름/값 쌍(자바의 Map이나 Hashtable과 같은 방식)

{이름1:값1, 이름2:값2, 이름3:값3}

```
var countries = {ko:'대한민국', fr:'프랑스', uk:'영국'}
var koName = countries.ko;
var frName = countries['fr'];
```

. 배열을 표시할 때 – '객체[인덱스]' 형식으로 접근

[값0, 값1, 값2, 값3]

```
var countryCodes = ['ko', 'fr', 'uk', 'us']
var idx0 = countryCodes[0]; // 'ko'
var idx2 = countryCodes[2]; // 'uk';
```

% http://www.json.org/ 사이트에서 참조

1. JSON표기법을 사용한 클래스 정의

JSON표기법의 이름/값 부분에서 함수 이름이 '이름'에 들어가도 함수의 정의가 '값' 들어감

---------- member_json.js ----------

```
Member = function(name, id, securityNo) {
    this.name = name;
    this.id = id;
    this.securityNo = securityNo;
}
Member.prototype = {
    setValue: function(newName, newId, newSecurityNo) {
         this.name = newName;
         this.id = newId;
          this.securityNo = newSecurityNo;
    },
    getAge: function() {
        var birthYear = parseInt(this.securityNo.substring(0, 2));
        var code = this.securityNo.substring(6,7);
        if (code == '1' || code == '2') {  birthYear += 1900;
        }  else if (code == '3' || code == '4') {  birthYear += 2000; }
        var today = new Date();
       return today.getFullYear() - birthYear;
   },
   toString: function() {    return this.name +  "[" +  this.id +  "]";  }
}
```

자바스크립트에서 패키지 정의하기

- 다른 사람이 만든 모듈 이름과 이름이 겹치는 것을 방지 하기 위해서 객체의 프로퍼티를 사용해서 패키지처럼 사용함

```
  var ajax = new Object();
  ajax.Reauest = function()  {  // Request 클래스 정의
  ….
}
ajax.Request.prototype = {  // Request 클래스에 함수 추가
   someFunction:function()  {
       ….
   },
   ….
}

var req = new ajax.Request();
req.someFunction();

 var ajax = { }; //  var ajax = new Object();와 같은 의미
 ajax.xhr = {};
 ajax.xhr.Request = function() {
   ……
}
```

클래스로 XMLHttpRequest모듈 만들기

. XMLHttpRequest.js 모듈은 한번에 여러 번의 요청을 보낼 경우 콜백함수에서 알맞은 XMLHttpRequest객체를 사용하지 못함

. 클래스를 사용해서 ajax요청을 전송하는 클래스 모델 (ajax.js)

```
var ajax = {};
ajax.xhr = {};
ajax.xhr.Request = function(url, params, callback, method) {
    this.url = url;          this.params = params;    // Request 클래스의 생성자
    this.callback = callback;                         // 객체 생성과 동시에 send()함수 호출
    this.method = method;
    this.send();
}
ajax.xhr.Request.prototype = {
    getXMLHttpRequest: function() {
    if (window.ActiveXObject) {
        return new ActiveXObject("Msxml2.XMLHTTP");
    } else if (window.XMLHttpRequest) {  return new XMLHttpRequest();
    } else {  return null;  }
},
```

```
  send: function() {
      this.req = this.getXMLHttpRequest();
      var httpMethod = this.method ? this.method : 'get';
      if (httpMethod != 'get' && httpMethod != 'post') {
          httpMethod = 'get';
      }
      var httpParams = (this.params == null || this.params == '') ? null : this.params;
      var httpUrl = this.url;
      if (httpMethod == 'get' && httpParams != null) {
          httpUrl = httpUrl + "?" + httpParams;
      }
      this.req.open(httpMethod, httpUrl, true);
      this.req.setRequestHeader(
              'Content-Type', 'application/x-www-form-urlencoded');
      var request = this;                                  // XMLHttpRequest 객체의 readyState
      this.req.onreadystatechange = function() {   // 값이 바뀔 때 이 객체(Request 객체)의
              request.onStateChange.call(request);      // onStateChange 함수 호출
       }
       this.req.send(httpMethod == 'post' ? httpParams : null);
  },
   onStateChange: function() {  // 이 과제의 callback 프로퍼티에 할당된 함수를 호출한다
       this.callback(this.req);    // 이때 인자로 this.req 객체를(XMLHttpRequest객체) 전달
  }
}
```

--- bookList.html ------

```
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko“><head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr" />
<title>책 목록</title>
<script type="text/javascript" src="ajax.js"></script>
<script type="text/javascript">
  var xmlDoc = null;
  var xslDoc = null;
  function loadBooks() {
     new ajax.xhr.Request("books.jsp", null, loadedBooksXML, "GET");
     new ajax.xhr.Request("books.xsl", null, loadedBooksXSL, "GET");
  }
  function loadedBooksXML(req) {
      if (req.readyState == 4) {
          if (req.status == 200) {
              xmlDoc = req.responseXML;
              doXSLT();
           }
      }
   }
```

```
    function loadedBooksXSL(req) {
        if (req.readyState == 4) {
            if (req.status == 200) {
                xslDoc = req.responseXML;
                doXSLT();
            }
        }
    }
    function doXSLT(req) {
        if (xmlDoc == null || xslDoc == null) return;
        var bookList = document.getElementById("bookList");
        if (window.ActiveXObject) {
            bookList.innerHTML = xmlDoc.transformNode(xslDoc);
        } else {
            var xsltProc = new XSLTProcessor();
            xsltProc.importStylesheet(xslDoc);
            var fragment = xsltProc.transformToFragment(xmlDoc, document);
            bookList.appendChild(fragment);    }
    }
    window.onload = function() {  loadBooks();  }
</script> </head>
<body>
        <h1>신규 책 목록</h1>          <div id="bookList"></div>
</body></html>
```

응답 결과를 자바스크립트 객체로 저장하기

- 서버에서 생성한 응답결과를 자바스크립트 객체에 저장하는 방법
  - XML 응답에서 값을 추출하여 객체에 저장
  - JSON 응답을 객체로 변환

1. XML응답을 객체로 변환

- 서버에서 생성한 XML응답을 자바스크립트 객체로 저장하기 위해서 XML응답으로부터 일일이 데이터를 추출해야 한다.

----- member_xml.jsp -----

```
<?xml version="1.0" encoding="euc-kr" ?>
<%@ page contentType="text/xml; charset=euc-kr" %>
<result>
    <code>success</code>
    <data>
        <member>
            <name>주니</name>
            <id>jusi</id>
            <sno>7700001000000</sno>
        </member>
    </data>
</result>
```

---- loadMemberFromXML.html ----

```
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr" />
<title>XML로부터 객체 생성</title>
<script type="text/javascript" src="ajax.js"></script>
<script type="text/javascript" src="member.js"></script>
<script type="text/javascript">
      window.onload = function() {
          new ajax.xhr.Request("member_xml.jsp", "", viewInfo, “get");
      }
```

```
    function viewInfo(req) {
        if (req.readyState == 4) {
            if (req.status == 200) {
                var docXML = req.responseXML;
                var code = docXML.getElementsByTagName("code“)
                        .item(0).firstChild.nodeValue;
                if (code == 'success') {    // XML DOM으로부터 일일히 정보를 추출
                    var name = docXML.getElementsByTagName("name“)
                            .item(0).firstChild.nodeValue;
                    var id = docXML.getElementsByTagName("id“)
                        .item(0).firstChild.nodeValue;
                    var sno = docXML.getElementsByTagName("sno")
                                .item(0).firstChild.nodeValue;
                    var mem = new Member(name, id, sno);
                    alert(mem.toString() + ", " + mem.getAge() + "세");
                } else { alert("실패");  }
            } else { alert("에러 발생:"+ req.status); }
        }
    }
</script>
</head>
<body>
</body>
</html>
```

2. JSON 응답을 객체로 변환

응답 결과가 XML이 아닌 텍스트이기 때문에 응답컨텐츠 타입을 text/plain으로 지정

```
---------  member_json.jsp  --------------------
<%@ page contentType="text/plain; charset=euc-kr" %>
{
    code: 'success',
    data: {
        member: {
            name: '주니',
            id: 'madvirus',
            sno: '7700001000000'
        }
    }
}
------------------ loadMemberFromJSON.html  -------------
<?xml version="1.0" encoding="EUC-KR" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
    <meta http-equiv="content-type" content="text/html; charset=euc-kr" />
    <title>JSON으로부터 객체 생성</title>
```

```
<script type="text/javascript" src="ajax.js"></script>
<script type="text/javascript">
    window.onload = function() {
        new ajax.xhr.Request("member_json.jsp", "", viewInfo, "GET");
    }
    function viewInfo(req) {
      if (req.readyState == 4) {
          if (req.status == 200) {
    // EVAL 함수는 자바스크립트 코드가 맞는지  틀렸는지 검정하고 문자열을 자바스크립트 코드로  수행하는 함수
              var result = eval("(" +  req.responseText +  ")");
              if (result.code == 'success') {
                  var m = result.data.member;
                   alert(m.id +  "[" +  m.name +  "]");
               } else {  alert("실패"); }
          } else {
               alert("에러 발생:"+ req.status);
          }
       }
    }
</script></head><body>
</body></html>
```

---

req.responseText에는 member_json.jsp가 생성한 데이터가 저장됨
서버에서 생성한 JSON 데이터를 eval() 함수를 사용해서 자바스크립트 객체로 변환