Ajax

훈련교사 박원기
http://www.jobtc.kr
http://www.jhta.co.kr

1. Ajax 개론

1.1. Ajax의 특징

- Asynchronous JavaScript XML(비동기 방식의 자바스크립트와 XML) 의 약자이다.
- Ajax는 어떠한 ActiveX나 플러그인 프로그램을 설치하지 않아도 된다.
- XMLHttpRequest 객체 사용하여 서버와 통신한다.
- XMLHttpRequest 객체는 보안상의 이유로 도메인에 있는 같은 URL에만 접속할 수 있다.
- XMLHttpRequest 객체를 가져오는 방법은 IE계열과 표준계열(w3c)와 서로 다를 수 있다.

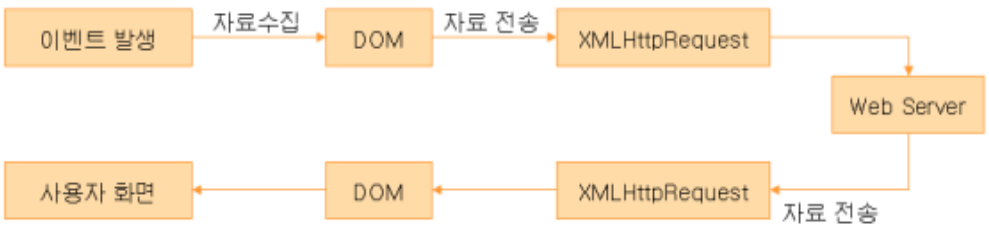
1.2. 기존 방식과 Ajax 방식의 비교

- 기존 웹 처리 방식 : 웹브라우저에 새로운 내용을 표시 할 때나, 표시된 내용을 변경하려고 할 때는 반드시 현재 화면을 갱신해야 한다.
- Ajax방식 : 바꾸거나 새롭게 내용을 추가할 위치만 변경할 수 있다.

1.3. Ajax의 구성 요소

XMLHttpRequest	웹서버와 사용자 웹브라우저와 통신을 담당한다. 사용자의 요청을 웹 서버에 전송하고, 웹서버로부터 받은 결과를 웹 브라우저에 전달한다.
DOM	문서의 구조를 나타낸다.
CSS	UI와 관계된 부분을 담당한다.(글자색, 바탕색, 화면구성)
JavaScript	사용자가 원하는 이벤트를 발생시키며, 자바스크립트를 통하여 XMLHttpRequest를 제어하여, 서버로 정보를 전달하거나 서버로 부터 수신된 정보를 처리한다.

1.4. Ajax의 처리철차



1.5. XMLHttpRequest 객체란

Ajax에서 가장 핵심적인 요소를 꼽자면 바로 XMLHttpRequest객체라 볼 수 있다. 이 객체의 특징을 살펴 보면 다음과 같다.

- Ajax가 서버와 데이터를 통신할 때 사용되는 객체이다.
- 현재로서는 웹표준에 속하지는 않는다.
- 거의 모든 웹 브라우저가 이를 지원하고 있으나 객체를 생성하거나 가져오는 방법이 W3C 표준안 방식과 IE 계열 방식이 다르다.

2. Ajax 주요 레퍼런스

2.1. XMLHTTP

IE계열의 브라우저에서 XMLHttpRequest 객체를 지정하는 프로퍼티

```
xhr = new ActiveXObject("Microsoft.XMLHTTP") // IE5 이하 버전
xhr = new ActiveXObject("MSXML2.XMLHTTP") // IE5~6버전
xhr = new XMLHttpRequest(); // 표준 브라이저 또는 IE7 버전 이상
```

* ie7 버전이상에서는 w3c 표준안을 따르고 있어 new XMLHttpRequest()를 사용하여 XMLHttpRequest 객체를 생성할 수 있다.

2.2. XMLHttpRequest

IE7 이상의 브라우저와 비IE 계열의 브라우저에서XMLHttpRequest 객체를 생성하는 메서드

```
xhr = new XMLHttpRequest();
```

[예] IE계열의 브라우저와 표준계열의 브라우저에서 XMLHttpRequest객체를 생성하고 어떤 브라우저에서 객체를 생성했는지 표시하라.

[소스]/section1/step1.jsp

```
<html>
<head>
<title>get XMLHttpRequest Test</title>
<meta http-equiv="Content-Type"
       content="text/html; charset=utf-8">
<script>
var xhr = null;
var message = "";

function get(){
  xhr = getXMLHttpRequest();
 rs = document.getElementById("result");
  rs.innerHTML = message;
}

function getXMLHttpRequest(){
  if(window.ActiveXObject){
    try{
      message = "IE 5.x 이상에서 객체 생성";
      return new ActiveXObject("Msxml2.XMLHTTP");
    }catch(e){
```

```
      try{
        message = "IE 5.x 이하에서 객체 생성";
        return new ActiveXObject("Microsoft.XMLHTTP");
      }catch(e1){
        return null;
      }

    }
  }else if(window.XMLHttpRequest){
    message = "W3C 에서 객체 생성";
    return new XMLHttpRequest();
  }else {
    return null;
  }
}

</script>
</head>

<body>
<input type="button" value="객체 생성"
        id="btn" onclick="get()"><br>
<div id="result"></div>
</body>
```

* IE11에서는 무조건 비IE에서 생성되었다고 표시됨.

[결과]

2.3. open()

요청을 초기화해서 HTTP 메소드및 URL 등을 설정하는 메소드

```
xhr.open(method , url , 비동기방식,[사용자이름, 암호]);
```

- method : DELETE, GET, HEAD, OPTIONS, POST, PUT
- url : 요청 대상 URL
- 비동기 방식 : 비동기라면 true(기본값) , 동기라면 false

[간단예] post 형태로 값을 전달하려 할때

```
xhr.open("POST","/test.jsp");
send("data=itdocument");
```

[간단예] get형태로 값을 전달하려 할 때

```
xhr.open("GET","/test.jsp",true);
```

[간단예] 사용자이름과 암호를 지정할 때

```
xhr.open("GET","http://a.com",true, "jobtc","1111");
```

결과 : jobtc:1111@a.com 과 같은 내용으로 전송됨.

2.4. send()

요청된 내용을 송신하는 메소드

```
xhr.send("") / xhr.send(null)
xhr.send(body)
```

- body : POST의 문자열 또는 DOMDocument , InputStream
- POST 형태로 전달하려 할 때
  - xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
  - xhr.send("name=itjava"); // 와 같이 처리 한다.

2.5. status | statusText

HTTP 요청 결과 상태 코드를 저장하고 있는 프로퍼티

```
value = xhr.status
text = xhr.statusText
```

value의 대표적인 값

value	text	의 미
200	OK	클라이언트의 요청이 정상 처리됨.
403	Unauthorized	접근 거부됨.
404	Forbidden	페이지 없음.
500	Internal Server Error	서버 오류

- 요청 상태값을 숫자형으로 사용하든, 문자형으로 사용하던 상관없다.
- 간혹 statusText를 지원하지 않는 브라우저가 있으므로 status값을 사용하는 것이 보다 안정적일 수 있다.

2.6. readyState

요청의 처리에 대한 수신 결과를 나타내는 프로퍼티

```
value = xhr.readyState
```

value값의 의미

value	텍스트	내용
0	UNINITIALIZED	오브젝트는 작성되어 있으나 아직 초기화되어 있지 않다 (open 메소드는 불리지 않았다)
1	LOADING	오브젝트가 작성되었으나 아직 send 메소드가 불리지 않았다
2	LOADED	send 메소드가 불렸지만 status와 헤더가 아직 도착하지 않았다
3	INTERACTIVE	데이터 일부를 받았다
4	COMPLETED	데이터 전부를 받았다. 완전한 데이터가 이용 가능

2.7. onreadystatechange

서버의 응답 상태가 변화할 때 발생하며, 콜백함수를 지정하거나 이벤트 핸들러 함수를 실행하게 할 수 있다.

```
oj.onreadystatechange  = 핸들러 함수 이름(콜백함수);
oj.onreadystatechange  = function(oj){이벤트 발생시의 처리}
```

2.8. responseXML | reqsponseText | responseURL

요청에 대한 응답을 XML 또는 Text로 반환받기 위한 프로퍼티이다. 이는 위에서 살펴 본 status와 readyState의 상태값이 모두 정상일 때 처리해야 한다.

```
xml = xhr.responseXML
text = xhr.responseText
url = xhr.responseURL; // 응답 URL
```

[예1] 텍스트 문서를 읽어 문단에 표시한다.

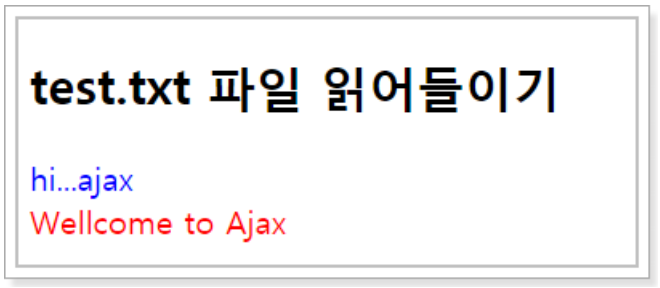
```
1.  <%@ page language="java" contentType="text/html; charset=UTF-8"
2.     pageEncoding="UTF-8"%>
3.  <!DOCTYPE html>
4.  <html>
5.  <head>
6.  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7.  <title>responseText</title>
8.  <script>
9.  var xhr;
10. function start(){
11.   xhr = new XMLHttpRequest();
12.   xhr.open('get', './test.txt', true);
13.   xhr.send();
14.   xhr.onreadystatechange = function(){
15.       if(xhr.readyState==4 && xhr.status==200){
16.           var rs = document.getElementById('result');
17.           rs.innerHTML = xhr.responseText;
```

```
        }
    }
}
</script>
</head>
<body>
<h2>test.txt 파일 읽어들이기</h2>
<div id='result'></div>
<script>start()</script>
</body>
</html>
```

test.txt 파일 내용

```
<font color='blue'>hi...ajax</font>
<br/>
<font color='red'>Wellcome to Ajax</font>
```

[실행결과]



[예2] xml 문서를 읽어 화면에 표시하기

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>responseXML</title>
<script>
var xhr;
function start(){
  xhr = new XMLHttpRequest();
  xhr.open('get', './test.xml', true);
  xhr.send();
  xhr.onreadystatechange = function(){
      if(xhr.readyState==4 && xhr.status==200){
          var rs = document.getElementById('result');
          var xml = xhr.responseXML;
          var id = xml.getElementsByTagName('id');
          var name = xml.getElementsByTagName('name');
```

```
            var address = xml.getElementsByTagName('address');

            var str = '<ul>';
            str += "<li>ID: " + id.item(0).firstChild.nodeValue;
            str += "<li>Name: " + name.item(0).firstChild.nodeValue;
            str += "<li>Address: " + address.item(0).firstChild.nodeValue;
            str += '</ul>';
            rs.innerHTML = str;
        }
    }
}
</script>
</head>
<body>
<h2>test.xml 파일 읽어들이기</h2>
<div id='result'></div>
<script>start()</script>
</body>
</html>
```

[text.xml 문서 내용]

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <id>park</id>
    <name>박원기</name>
    <address>서울 종로</address>
</root>
```

[실행결과]

2.9. onload

응답 헤더의 로딩을 완료 했을 때 발생하여 핸들러 함수를 실행하는 이벤트

```
oj.onload = 핸들러 함수 이름
oj.onload = function(oj){처리내용}
```

2.10. abort()

요청을 취소하는 메소드

```
xhr.abort();
```

2.11. getAllResponseHeaders()

HTTP 요청에 대한 모든 응답 헤더를 돌려주는 메소드

```
value = xhr.getAllResponseHeaders()
```

[예]

```
<script src="getXML.js"></script>
<script>
var xhr = null;
window.onload=function(){
  xhr = getXMLHttpRequest();
  processEvent();
}

function processEvent(){
  url = "gugu.jsp?dan=5";
  xhr = getXMLHttpRequest();
  xhr.onreadystatechange = myFunc;
  xhr.open("HEAD",url, true);
  xhr.send(null);
}

function myFunc(){
  if(xhr.readyState == 4){
    if(xhr.status == 200){
      result.innerHTML += xhr.getAllResponseHeaders();
      result.innerHTML += "<p>";

    }else{
      alert("fail");
    }
  }
}
</script>
```

[실행결과]

2.12. getResponseHeader(인자값)

인수로 지정한 응답 헤더를 돌려주는 메소드

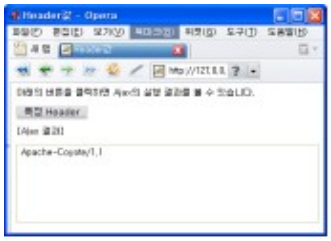
```
value = xhr.getResponseHeader(header);
```

* header : 얻고싶은 정보를 입력

[간단예]

```
<script src="getXML.js"></script>
<script>
var xhr = null;

window.onload=function(){
  xhr = getXMLHttpRequest();
  processEvent();
}

function processEvent(){
  url = "test.txt";
  xhr = getXMLHttpRequest();
  xhr.onreadystatechange = myFunc;
  xhr.open("HEAD",url, true);
  xhr.send(null);
}

function myFunc(){
  if(xhr.readyState == 4){
    if(xhr.status == 200){
      result.innerHTML += xhr.getResponseHeader("Server");
      result.innerHTML += "<p>";

    }else{
      alert("fail");
    }
  }
}
```

[실행결과]



2.13. setRequestHeader()

- 서버로 전송할 헤더값을 지정한다.

```
xhr.setRequestHeader(header,값)
```

예) xhr.setRequestHeader("user-Agent","itdocument")
xhr.setRequestHeader('content-type','application/x-www-form-urlencoded;charset=utf-8');
xhr.setRequestHeader('content-type','text/html;charset=utf-8');

[참고] 인터넷 캐시 처리

캐시란 일종의 임시기억 장치이다. 빠른 중앙처리 장치나 주기억장치에 비해 상대적으로 느린 주변 입출력장치간의 속도차를 해결하기 위해 나온 기술이다.
인터넷 캐시 역시 컴퓨터 자체의 처리 능력 보다 훨신 느린 인터넷 속도를 보상하기 위해 한번 읽어 들인 정보를 컴퓨터의 기억장치에 저장한 뒤 다시 읽어 들여야 할 정보가 동일한 정보라 판단되면 다시 읽어 들이지 않고 컴퓨터에 저장된 정보를 사용하도록 되어 있다.

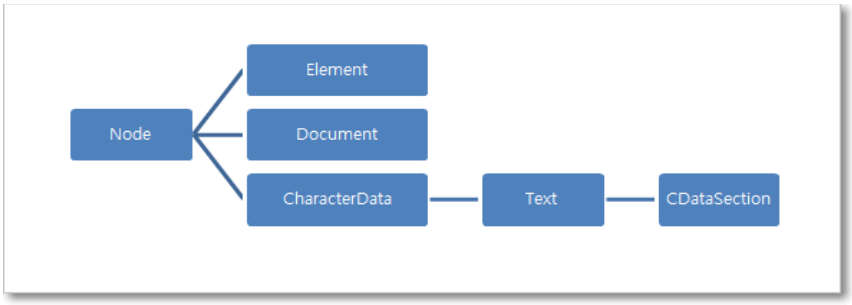
이런 기능은 빠른 처리 결과를 얻는 대신 서버의 정보가 새롭게 갱신되었음에도 불구하고 동일한 정보로 인식되어 읽어 들이지 못 할 때가 발생한다. 이를 방지하고 캐시정보를 사용하지 않도록 아래의 코드 처럼 임시로 인터넷 환경을 변경할 수 있다.

```
<%
response.setHeader("Cache-Control", "no-cache"); // 캐시 사용 여부
response.setHeader("Pragma","no-cache"); // 자동 연결(auto-linking)
response.setHeader("Expires", "0");// 캐시에 문서가 남아있을 시간
%>
```

3. DOM API

DOM(Document Object Model)은 문서의 내용을 트리구조로 구조화 하여 메모리에 매핑한 후 이를 접근할 수 있도록 만들어진 API를 의미한다.

DOM은 트리구조를 구성하기 위해 몇가지 인테페이스를 사용하는데 주요 인터페이스에 대한 구조는 아래와 같다.



- Document – 전체 문서를 나타낸다.
- Element – 각 태그들을 의미한다.
- Text – 문자열 데이터를 나타낸다.
- CDataSection – XML 문서의 CDATA 영역의 문자열 값을 저장한다.

3.1. Document interface

3.1.1. 주요 프로퍼티

타입	이름	내용
Element	documentElement	문서의 루트 노드를 나타낸다.

HTML 문서에서 document.documentElement.nodeName을 스크립트로 실행하면 'HTML' 이 표시된다. 루트 엘리먼트가 'HTML'인 것이다.

```
1. <script>
2. var rs = '';
3. rs = document.documentElement.nodeName;
4. document.write(rs);
5. </script>
```

[실행결과]

HTML

3.1.2. 주요 함수

함수명	의미
NodeList getElementsByTagName("태그명")	지정한 태그명을 NodeList로 반환한다.
Element getElementById("id")	id에 해당하는 태그를 구한다.
NodeList getElementsByClassName("클래스명")	지정된 클래스명을 NodeList로 반환한다.

태그명을 기준으로 요소를 가져오면 기본적으로 배열 처리된다. 동일한 태그가 2개 이상 있을 가능성이 높기 때문에 이렇게 만들어졌으리라...^^

[예]getElementsByTagName

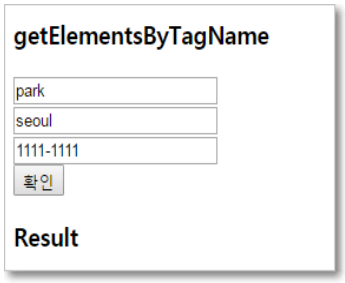
가장 많이 사용하게 되는 form태그의 input 태그들을 여러 개 만들어 getElementsByTagName 함수를 사용하여 제어해 보자.

step 1. 간단한 UI를 작성한다.

```
<body>
<h3>getElementsByTagName</h3>

<form name='frm'>
  <input type='text' name='irum' value='park' id='id_irum' class='class_irum '/><br/>
  <input type='text' name='address' value='seoul' id='id_address' class='class_address'/><br/>
  <input type='text' name='phone' value='1111-1111' id='id_phone' class='class_phone'/><br/>
</form>
<input type='button' id='btn' value='확인'/>
<h3>Result</h3>
<div id='result'></div>

<script>getEleInit()</script>
</body>
```

window.onload() 함수 대신 맨 하단에서 getEleInit()함수를 호출하도록 구성하였다.

[결과화면]



step 2. 스크립트를 작성한다.

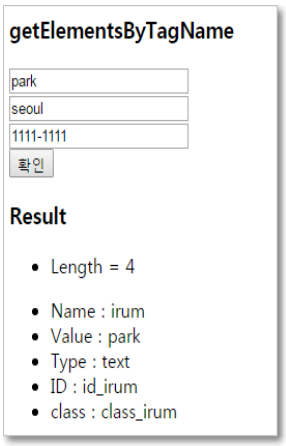
```
<script>
function getEleInit(){
  // 결과를 출력하기 위해
  var rs = document.getElementById('result');
  // 버튼를 id로 가져옴.
  var btn = document.getElementById('btn');

  // 버튼이 클릭되면
  btn.onclick=function(){
      // input 태그를 모두 가져옴.
      var input = document.getElementsByTagName('input');

      var str = "<ul><li>Length = " + input.length + "</li></ul>";
      for(i=0 ; i<input.length; i++){
          str += '<ul><li> Name : ' + input[i].name;
          str += '<li>Value : ' + input[i].value;
          str += '<li>Type : ' + input[i].type;
          str += '<li>ID : ' + input[i].id;
          str += '<li>class : ' + input[i].className;
          str += "</ul>";
      }
      rs.innerHTML = str;
  }
}
</script>
```

[전체소스]

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script>
function getEleInit(){
  // 결과를 출력하기 위해
  var rs = document.getElementById('result');
  // 버튼를 id로 가져옴.
  var btn = document.getElementById('btn');

  // 버튼이 클릭되면
  btn.onclick=function(){
      // input 태그를 모두 가져옴.
      var input = document.getElementsByTagName('input');

```

```
        var str = "<ul><li>Length = " + input.length + "</li></ul>";
        for(i=0 ; i<input.length; i++){
            str += '<ul><li> Name : ' + input[i].name;
            str += '<li>Value : ' + input[i].value;
            str += '<li>Type : ' + input[i].type;
            str += '<li>ID : ' + input[i].id;
            str += '<li>class : ' + input[i].className;
            str += "</ul>";
        }
        rs.innerHTML = str;
    }
}
</script>
</head>
<body>
<h3>getElementsByTagName</h3>

<form name='frm'>
    <input type='text' name='irum' value='park' id='id_irum' class='class_irum '/><br/>
    <input type='text' name='address' value='seoul' id='id_address' class='class_address'/><br/>
    <input type='text' name='phone' value='1111-1111' id='id_phone' class='class_phone'/><br/>
</form>
<input type='button' id='btn' value='확인'/>
<h3>Result</h3>
<div id='result'></div>

<script>getEleInit()</script>
</body>
</html>
```

[실행결과] 실행결과는 출력의 일부분만을 캡쳐하였다.



입력상자 3개와 버튼까지 전체 4개가 인식되었으며 각각의 요소에 지정된 name, value, id, class 등과의 정보를 얻어와 표시 하였다.

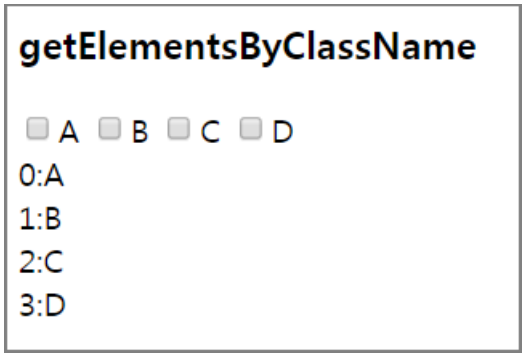
[예2] getElementsByClassName

클래스명으로 요소를 가져옴.

getElementsByClassName.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h3>getElementsByClassName</h3>

<div id='data'>
  <label><input type='checkbox' value='A' class='alpha'>A</label>
  <label><input type='checkbox' value='B' class='alpha'>B</label>
  <label><input type='checkbox' value='C' class='alpha'>C</label>
  <label><input type='checkbox' value='D' class='alpha'>D</label>
</div>

<script>
var data = document.getElementById('data');
var nodes = data.getElementsByClassName('alpha');

for(i = 0 ; i<nodes.length ; i++){
  document.write(i + ":" + nodes[i].value + '<br/>');
}

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

[결과]



3.2. Node interface

3.2.1. 주요 프로퍼티

타입	이름	내용

String	nodeName	노드의 이름
String	nodeValue	노드의 값
unsigned short	nodeType	노드 타입
Node	parentNode	부모 노드
NodeList	childNodes	자식 노드
Node	firstChild	첫 번째 자식 노드
Node	lastChild	마지막 자식 노드
Node	previousSibling	자신의 이전 노드(좌측 노드)
Node	nextSibling	자신의 이후 노드(우측 노드)
Document	ownerDocument	자신을 포함하고 있는 Document 객체

3.2.2. nodeName 과 nodeValue의 비교

노드 타입	nodeName	nodeValue
Document	#document	null
Element	태그명	null
Text	#text	문자열

노드타입에 따라 nodeName과 nodeValue가 서로 다른 값을 갖고 있다.

3.2.3. 노드 타입

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

3.2.4. 노드 접근

속성	의미
length	NodeList 의 개수
item(n)	NodeList에서 n번째의 노드를 나타냄

간단한 HTML 문서를 사용하여 node들을 접근해 보자.

```
<div id='main'>
  <section>
        <h2>타이틀1</h2>
        <p>문단1</p>
    </section>

    <section>
        <h2>타이틀2</h2>
        <p>문단2</p>
    </section>
</div>
```

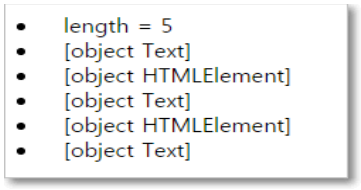
id='main' 안에 있는 자식 노드들의 개수를 세어보자. 그런데 <section/> 태그 2개가 나올것 같지만 5개가 나온다.

```
var main = document.getElementById('main');
var childs = main.childNodes;
var str = '<li>length = ' + childs.length;
```

그 이유를 살펴 보기 위해 childs.length 만큼 루프를 돌려서 Node들을 찍어보자.

```
var main = document.getElementById('main');
var childs = main.childNodes;
var cnt = 0;
var str ='';
for(i=0 ; i<childs.length ; i++){
  var nodeName = childs.item(i);
  if(nodeName.nodeType != 1) continue;
  str += "<li>" + nodeName.nodeName ;
  cnt++;
}
document.write('<li>length = ' + cnt);
document.write(str);
```

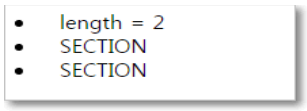
[실행결과]



2개가 나와야 정상인데 5개가 나온다. 비밀은 태그와 태그사이에 있는 줄바꿈 때문이다. DOM은 이런 줄바꿈 조차도 [object Text] 로 인식하기 때문이다. 따라서 하위에 있는 자식 테그들만 가져 오기위해 소스를 수정했다.

```
...
for(i=0 ; i<childs.length ; i++){
        var nodeName = childs.item(i);
        if(nodeName.nodeType != 1) continue;
        str += "<li>" + nodeName;
         cnt++;
}
document.write('<li>length = ' + cnt);
document.write(str);
```

[실행결과]



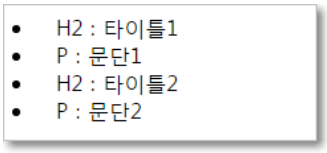
nodeType 이 1인 경우가 하위 Element인 것이다.

다음 단계로 특정 태그의 하위 노드들을 가져오는 부분을 응용하여 <section/>안에 있는 태그들의 태그명과 문자열을 표시해 보자.

위의 예제에서는 id='main'안에 있는 모든 태그들을 가져와 사용하였는데 이러면 너무 복잡하므로 바로 <section/>안에 있는 태그들만 가져와 사용하도록 하겠다.

```
// section 태그들만 가져옴.
var section = document.getElementsByTagName('section');
var rs = document.getElementById('result');
var str ='';
for(i=0 ; i<section.length ; i++){
  var childs = section[i].childNodes; //section 태그의 자식들
  for(j=0; j<childs.length ; j++){
      var childNode = childs.item(j);
      if(childNode.nodeType != 1) continue;
      str += '<li>' + childNode.nodeName + ' : ' // 태그명을 가져옴.

      // j번째 테그의 자식 태그가 [obejct text] 타입이므로
      // 자식 태그를 얻어와서 텍스트값을 가져옴.
      str += childNode.firstChild.nodeValue;
  }
}
document.write(str);
```

[실행결과]



3.3. Element interface

Element 인터페이스는 Node 인터페이스를 상속받아 만들어졌다. 일반적으로 element나 html tag로 불려지는 요소들의 속성들을 제어할 수 있다.

3.3.1. Attributes

속성	이름
readonly DOMString	tagName

- 읽기 전용이며 태그의 이름을 대문자로 변환하여 반환한다.

3.3.2. Methods

속성	이름
DOMString	getAttribute(in DOMStrinn name) element의 속성중 name에 해당하는 값을 가져온다.
void	setAttribute(in DOMString name, in DOMString value) name속성에 해당하는 값을 value로 지정한다.
void	removeAttribute(in DOMString name) name속성를 제거한다.
Attr	getAttributeNode(in DOMString name) name 속성에 해당하는 값을 가져온다.
Attr	setAttributeNode(in Attr newAttr) 해당 element에 새로운 속성 newAttr을 지정한다.
Attr	removeAttributeNode(in Attr oldAttr) oldAttr로 지정된 속성을 제거한다.
NodeList	getElementsByTagName(in DOMString name) name에 해당하는 element들을 가져온다.

[예1] 특정 태그의 속성값 가져오기

파일명 : getAttribute.html

```
<html>
<head>
<title>getAttribute </title>

<script>
window.onload=function(){
  window.title = "getAttribute";
  node = document.getElementById("p3");

  result.innerText = "\n text : " + node.innerText;
  result.innerText += "\n color : " + node.getAttribute("color");
}

</script>
</head>

<body>
<font color=red id="p3" class="c1">수월한 IT FIGHTING</font>
<hr>
<b>Ajax Result</b>
<div id="result"></div>
</body>
</html>
```

[결과]

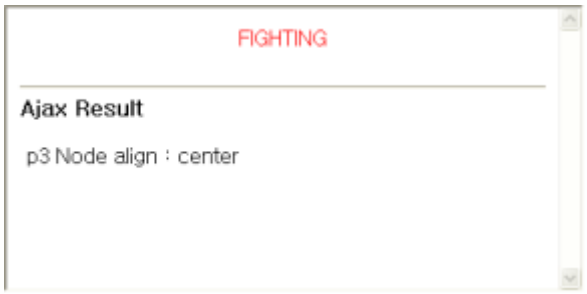
[예2] setAttributeNode 예제

파일명 : setAttributeNode.html

```
<!-- setAttributeNode test ajax ---------------- -->

<html>
```

```
4.  <head>
5.  <title> element test </title>
6.
7.  <script>
8.  window.onload=function(){
9.    window.title = "setAttributeNode test for ajax";
10.   node = document.getElementById("p3");
11.
12.   // setAttributeNode() sample
13.   addAtt = document.createAttribute("align");
14.   addAtt.nodeValue="center";
15.   node.setAttributeNode(addAtt);
16.
17.   result.innerText = "\n p3 Node align : " + node.getAttribute("align");
18.
19. }
20.
21. </script>
22. </head>
23.
24. <body>
25. <p id="p3" bgcolor="#FFCC00">
26. <font color=red>FIGHTING</font>
27. </p>
28.
29. <hr>
30. <b>Ajax Result</b>
31. <div id="result"></div>
32. </body>
33. </html>
```

[결과]



3.4. Element 추가 및 삭제

특정 요소(Element)를 생성하여 추가하고 삭제해 보도록 하자. 이와 관련된 주요 메서드는 아래와 같다.

메서드	설명
Element createElement(태그명)	태그명에 해당하는 요소를 생성한다.
appendChild(요소)	지정된 요소를 추가한다.
removeChild(요소)	지정된 요소을 제거한다.

3.4.1. 요소 생성

var br = document.createElement('태그명')

으로 태그를 생성할 수 있다. 이 때 '태그명'은 <> 기호를 생략한다. 아래의 예는
, <input/>를 만드는 방법이다.

- var br = document.createElement('br');
- var element = document.createElement('input');

이렇게 만들어진 요소에 각종 속성들을 추가해야 하는데 이때는 setAttribute()메서드를 사용하여 속성들을 추가 한다. 만약,

<input type='text' name='irum' value='jobtc'/>

와 같은 태그를 생성한다면 아래와 같은 코드들이 필요하다.

```
<script>
//요소 생성
var input = document.createElement('input');
// 속성 추가
input.setAttribute('type', 'text');
input.setAttribute('name','irum');
input.setAttribute('value','jobtc');
</script>
```

[예] 버튼을 클릭하면 입력상자와 br를 추가한다.

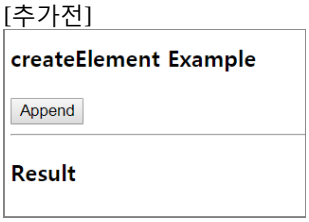
createElement.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h3>createElement Example</h3>
```

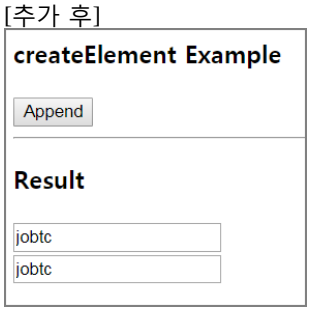
```
<input type='button' value='Append' onclick='append()' />
<hr/>
<h3>Result</h3>
<div id='result'></div>

<script>
function append(){
 var rs = document.getElementById('result');

 //요소 생성
 var input = document.createElement('input');
 var br = document.createElement("br");
 // 속성 추가
 input.setAttribute('type', 'text');
 input.setAttribute('name','irum');
 input.setAttribute('value','jobtc');

 rs.appendChild(input);
 rs.appendChild(br);
}
</script>
</body>
</html>
```

[결과]

[추가전]



[추가 후]



3.4.2. 요소삭제

태그명이나 id, class 속성등을 사용하여 삭제하고자 하는 요소를 선택한 후 removeChild(요소)를 사용하여 삭제할 수 있다.

[예]

removeChild.html

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

```
</head>
<body>
<h3>removeChild</h3>

<input type='button' value='remove' onclick='remove()' />
<p/>
<div id='parents'>
 <input type='button' value='button A' >
 <input type='button' value='button B' >
 <input type='button' value='button C' >
 <input type='button' value='button D' >
</div>

<script>
function remove(){
 var p = document.getElementById('parents');
 var c = p.getElementsByTagName("input");

 p.removeChild(c[0]);//항상 첫번째 삭제
}
</script>

</body>
</html>
```

4. xml응답 처리

responseXML메서드를 사용하면 서버에서 전송한 XML문서를 XML 유형으로 처리할 수 있다. 이때 읽어온 XML 문서는 DOM 트리 구조로 저장되기 때문에 DOM API를 사용하여 문서의 내용중 원하는 정보를 얻을 수 있을 것이다.

responseXML은 문서 전체를 나타내는 Document 객체를 리턴 받는다.

서버로부터 XML문서를 읽어 들이는 프로그램을 만들어 보자.

4.1. 관련 파일

- xmlDocText.jsp : xml 응답에 사용될 xml 파일
- getXMLHttpRequest.js : XMLHttpRequest 객체를 생성하는 자바 스크립트 파일
- xmlLoad.html : 메인 파일

4.2. 작업절차

- XML 응답 문서 정의
- GUI작성
- XMLHttpRequest 객체 생성 코드 작성
- XMLHttpRequest 상태변화에 따른 callback 함수 정의
- XML 문서 처리 함수 정의

4.2.1. xml 응답 문서 정의

XMLHttpRequest 의 요청에 의해 읽어 들일 응답 문서를 정의해 보자.

파일명 : xmlDocText.jsp(전체)

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

<members>
  <member>
    <name>김씨</name>
    <addr>서울</addr>
    <phone>1111</phone>
  </member>

  <member>
    <name>이씨</name>
    <addr>부산</addr>
    <phone>2222</phone>
  </member>

  <member>
    <name>디씨</name>
```

```
    <addr>대구</addr>
    <phone>none</phone>
  </member>

</members>
```

- 2행 : contentType='text/xml'로 바꾸어주어야 한다. 이는 문서의 종류가 xml임을 나타냄.

4.2.2. GUI작성

파일명 : xmlLoad.html(body 부분)

```
<body>
<input type="button" value="XML Load" id="btn" onclick="loadFunc()">
<hr>
<div id="result"></div>
</body>
```

- 41행 : 번튼을 클릭하면 loadFunc()함수를 호출하여 XMLHttpRequest 객체를 생성하고 상태 변화에 따른 callback함수를 지정한다.
- 43행 : xml문서를 읽어 처리된 결과가 표시되는 영역.

4.2.3. XMLHttpRequest 상태변화에 따른 callback 함수 정의

loadFunc()함수 내에서 상태변화에 따른 callback함수를 지정한다.

파일명 : xmlLoad.html(loadFunc() 부분)

```
function loadFunc(){
  xhr = getXMLHttpRequest();
  xhr.onreadystatechange=xmlLoad;
  xhr.open("get","xmlDocText.jsp",true);
  xhr.send("");
}
```

12행 : XMLHttpRequest객체를 생성하는 함수 호출
13행 : XMLHttpRequest객체의 상태가 변화되면 호출되는 함수 지정(callback함수)
14행 : xmlDocText.jsp파일을 비동기 방식으로 열도록 지정한다.
15행 : 요청 정보를 전송한다.

4.2.4. XMLHttpRequest 객체 생성 코드 작성

getXMLHttpRequest.js은 별도의 파일로 작성한후 <script src="getXMLHttpRequest.js"></script>를 사용하여 현재 문서에 포함시켜 사용한다.

파일명 : getXMLHttpRequest.js

```
1.  var xhr = null;
2.
3.  function getXMLHttpRequest(){
4.      if(window.ActiveXObject){
5.          try{
6.              return new ActiveXObject("Msxml2.XMLHTTP");
7.          }catch(e){
8.              try{
9.                  return new ActiveXObject("Microsoft.XMLHTTP");
10.             }catch(e1){
11.                 return null;
12.         }
13.
14.      }
15.     }else if(window.XMLHttpRequest){
16.         return new XMLHttpRequest();
17.     }else {
18.         return null;
19.   }
20. }
```

4.2.5. XML 문서 처리 함수 정의

open()과 send()함수에 의해 정보 요청이 완료되면 요청된 정보의 정상적인 수신인지를 판단하여 수신된 정보를 적당한 양식으로 변경하여 result 영역에 표시한다.

파일명 : xmlLoad.html(xmlload() 부분)

```
1.  function xmlLoad(){
2.    var rs = document.getElementById("result");
3.    var str = "";
4.    if(xhr.readyState == 4 && xhr.status==200){
5.      var xmlDoc = xhr.responseXML;
6.      var list = xmlDoc.getElementsByTagName("member");
7.      var len = list.length;
8.
9.      for(i=0 ; i<len ; i++){
```

```
10.       arry = list.item(i);
11.       ele1 = arry.getElementsByTagName("name").item(0).firstChild.nodeValue;
12.       ele2 = arry.getElementsByTagName("addr").item(0).firstChild.nodeValue;
13.       ele3 = arry.getElementsByTagName("phone").item(0).firstChild.nodeValue;
14.
15.       str += (ele1 + "," + ele2 + "," + ele3 + "<br>") ;
16.     }
17.     rs.innerHTML += "length : " + len + "<p>" + str;
18.   }
19. }
```

[처리결과]

[xmlLoad.html 전체 소스]

파일명 : xmlLoad.html(전체)

```
1.  <!--
2.  서버로 부터 XML 문서를 읽어 들여 DOM API를
3.  사용하여 표시하는 프로그램
4.  -->
5.  <html>
6.  <head>
7.  <meta http-equiv="Content-Type" content="text/html;charset=utf-8">
8.  <title>XML Load</title>
9.  <script src="getXML.js"></script>
10. <script>
11. function loadFunc(){
12.   xhr = getXMLHttpRequest();
13.   xhr.onreadystatechange=xmlLoad;
14.   xhr.open("get","xmlDocText.jsp",true);
15.   xhr.send("");
16. }
17.
18. function xmlLoad(){
19.   var rs = document.getElementById("result");
20.   var str = "";
21.   if(xhr.readyState == 4 && xhr.status==200){
```

```
22.     var xmlDoc = xhr.responseXML;
23.     var list = xmlDoc.getElementsByTagName("member");
24.     var len = list.length;
25.
26.     for(i=0 ; i<len ; i++){
27.       arry = list.item(i);
28.       ele1 = arry.getElementsByTagName("name").item(0).firstChild.nodeValue;
29.       ele2 = arry.getElementsByTagName("addr").item(0).firstChild.nodeValue;
30.       ele3 = arry.getElementsByTagName("phone").item(0).firstChild.nodeValue;
31.
32.       str += (ele1 + "," + ele2 + "," + ele3 + "<br>") ;
33.     }
34.     rs.innerHTML += "length : " + len + "<p>" + str;
35.   }
36. }
37. </script>
38.
39. </head>
40. <body>
41. <input type="button" value="XML Load" id="btn" onclick="loadFunc()">
42. <hr>
43. <div id="result"></div>
44. </body>
45. </html>
```

4.3. xml 응답을 xsl/t로 처리

Ajax를 통하여 xml문서를 읽어와 text, html등과 같은 문서를 표현하는 것처럼 xsl형태로 변형하여 처리할 수 있다. 이장의 내용을 충분히 이해하고 활용하려면 xsl의 기본 지식이 있어야 하겠다.

xsl이란 xml문서를 다른 구조의 xml문서로 변환하는 방법 중 하나이며, CSS의 단점을 보완하여 W3C에서 표준화한 새로운 스타일 시트 언어이다.

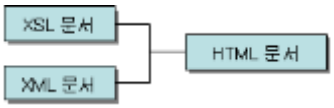
그러나 이장에서는 xml의 응답처리 방식을 다루는 방법에 관한 내용임으로 임으로 xsl 파일을 만들거나 의미하는 바는 설명하지 않고 넘어 가겠다.

4.3.1. 관련 파일

- xslDoc.xsl : 새로운 구조를 나타내는 스타일 시트(xsl파일)
- xmlDocText.jsp : xml 응답에 사용될 xml 파일
- getXML.js : XMLHttpRequest 객체를 생성하는 자바 스크립트 파일

- xslLoad.html : 메인 파일

xmlDocText.jsp, getXMLHttpRequest.js파일은 앞장에서 사용했던 파일을 그대로 사용할 것이다.



위의 그림처럼 문서의 구조를 나타내는 XSL문서와 문서의 내용이 들어 있는 XML문서를 각각 읽어들여 두파일을 조합하여 새로운 문서 형태인 HTML 문서 형태로 만들게 될것이다.

4.3.2. 작업절차

- 데이터 파일(문서 내용) 작성 : 윗 단원에서 사용한 xmlDocText.jsp문서 그대로 사용하자.
- 문서 구조 파일 작성 : xslDoc.xsl 문서 정의
- 메인 파일 작성 : xslLoad.html 파일 작성
- 기타 : XMLHttpRequest 객체를 생성하기 위한 자바 스크립트 getXML.js는 위에서 사용한 그대로 사용한다.

4.3.3. 문서 구조 파일 작성(xsl)

xsl 문서는 문서의 구조를 나타내는 스타일시트 파일의 일종이다. 아래와 같은 파일을 만들어 저장한다.

파일명 : xslDoc.xsl

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <xsl:stylesheet
3.        xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
4.  <xsl:output method="html" indent="yes" encoding="utf-8"/>
5.  <xsl:template match="members">
6.  <body>
7.  <p>itdocument fighting...</p>
8.    <table border="1" width="200">
9.      <tr>
10.       <th>Name</th>
11.       <th>Address</th>
12.       <th>Phone</th>
13.     </tr>
14.   <xsl:for-each select="member">
15.     <tr>
16.       <td><xsl:value-of select="name" /> </td>
17.       <td><xsl:value-of select="addr" /> </td>
18.       <td><xsl:value-of select="phone" /> </td>
```

```
19.      </tr>
20.    </xsl:for-each>
21.    </table>
22. </body>
23. </xsl:template>
24. </xsl:stylesheet>
```

3~4행 : 문서의 행태가 stylesheet임을 나타냄
5행 : method="html"은 출력형태가 html문서형태이며, indent="yes"의 의미는 화이트스페이스를 유지하겠다는 의미.(화이트 스페이스는 태그 사이의 공백을 의미함)
7행 : 템플레이트룰를 나타내며, 문서의 구조를 변경하겠다는 의미이다. match="members"는 문서 구조를 변경하게될 대상 노드를 지칭함.
16행 : 반복문. member 노드를 모두 순회하면서 반복처리함.
18~20행 : 값을 가져와야할 대상 노드

4.3.4. 메인파일(xslLoad.html)문서 작성

소스에 대한 설명은 소스 중간 중간에 있는 주석으로 대신한다.

파일명 : xslLoad.html

```
1.  <!--
2.  서버로 부터 XML 문서를 읽어 들여 DOM API를
3.  사용하여 표시하는 프로그램
4.  -->
5.  <html>
6.  <head>
7.  <meta http-equiv="Content-Type" content="text/html;charset=utf-8">
8.  <title>XML+XSL </title>
9.  <!-- XMLHttpRequest객체를 생성하기 위한 함수 정의 -->
10. <script src="getXML.js"></script>
11. <script>
12. var xmlDoc = "";
13. var xslDoc = "";
14.
15. // 버튼이 클릭되면 호출되며 XMLHttpRequest객체를
16. // 생성하는 함수를 호출하고
17. // xmlDocText.jsp 문서를 오픈하고 상태를 체크하기
18. // 위한 콜백함수지정
19. function loadFunc(){
20.   xhr = getXMLHttpRequest();
21.   xhr.onreadystatechange=xmlLoad;
22.   xhr.open("get","xmlDocText.jsp",true);
23.   xhr.send("");
```

```
24. }
25.
26. // xmlDostext.jsp 파일을 xml유형으로 읽어들여 xmlDoc
27. // 에 저장하고 xslDoc.xsl 파일을 오픈한뒤 상태를 체크하기 위한
28. // 콜백 함수 지정
29. function xmlLoad(){
30.   if(xhr.readyState == 4){
31.     if(xhr.status==200){
32.       xmlDoc = xhr.responseXML;
33.
34.       xhr = getXMLHttpRequest();
35.       xhr.onreadystatechange=xslLoad;
36.       xhr.open("get", "xslDoc.xsl", true);
37.       xhr.send("");
38.     }
39.   }
40. }
41.
42. // xslDoc.xsl문서를 읽어들여 xslDoc에 저장하고
43. // xml + xsl을 처리하기 위해 doXSLT 함수 호출
44. function xslLoad(){
45.   if(xhr.readyState == 4){
46.     if(xhr.status==200){
47.       xslDoc = xhr.responseXML;
48.       doXSLT();
49.     }
50.   }
51. }
52.
53. // 읽어들인 xml문서를 xsl형태로 변형하기 위해
54. // 브라우저 계열별로 변환작업.
55. // IE계열은 간단히 transformNode()를 사용하여 표현할수 있으나
56. // W3C 계열은 XSLTProcessor()함수를 사용하여 XLSTprocess를 생성한뒤
57. // 관련 메서드들을 사용하여 변환작업.
58. function doXSLT(){
59.   var rs = document.getElementById("result");
60.   if( xmlDoc == null || xslDoc == null){
61.     rs.innerHTML="xmlDoc or xslDoc file fail...";
62.     return;
63.   }
64.
65.   if(window.ActiveXObject){ // IE 계열이면
66.     rs.innerHTML = xmlDoc.transformNode(xslDoc);
67.   }else{
```

```
68.     var xslt = new XSLTProcessor();
69.     xslt.importStylesheet(xslDoc);
70.     var frag = xslt.transformToFragment(xmlDoc, document);
71.     rs.appendChild(frag);
72.   }
73. }
74.
75. // ------------------------------------
76. // 소스를 화면에 보여지게 하기 위한 함수
77. // ------------------------------------
78. function sourceView(file){
79.   xhr = getXMLHttpRequest();
80.   xhr.onreadystatechange = viewer;
81.   xhr.open("get",file, true);
82.   xhr.send(null);
83. }
84.
85. function viewer(){
86.   if(xhr.readyState == 4){
87.     if(xhr.status == 200){
88.       result.innerText = xhr.responseText;
89.     }else{
90.       alert("fail");
91.     }
92.   }
93. }
94.
95. // ------------------------------------
96. </script>
97.
98. </head>
99. <body>
100. <input type="button" value="XML Load" id="btn" onclick="loadFunc()">
101. <input type="button" value="소스보기"  onClick="sourceView('xslLoad.html')">
102. <hr>
103. <div id="result"></div>
104. </body>
105. </html>
```

[실행결과]


XML Load
소스보기
itdocument fighting...
Name
Address
Phone
김씨
서울
1111
이씨
부산
2222
디씨
대구
none

5. JSON 표기법

5.1. JSON이란

JavaScript Object Notation 의 약자로 서로 다른 프로그래밍 언어 간에 데이터를 교환하기 위한 표기법으로 읽고 쓰기 쉽도록 고안해낸 데이터 표기법이다.

5.2. 표현 방법

대부분의 언어에서 친숙하게 사용되고 있는 구조로 이루어져 있으며 크게 2가지의 표현 방법을 사용하고 있다.

- { 이름:값, ... }
- [배열]
- 혼합형

5.2.1. 이름:값 유형

과일을 "이름:값"으로 정의하고 사용해 보도록 하자.

```
<script>
var fruits={apple:'사과', banana:'바나나', melon:'메론', watermelon:'수박'}

// 이름으로 가져오기
document.write("apple -->" + fruits.apple + "<br>");
document.write("banana -->" + fruits.banana + "<br>");
document.write("melon -->" + fruits.melon + "<br>");
document.write("watermelon -->" + fruits.watermelon + "<br>");
</script>
```

위와 같이 객체.이름 형태로 그 값을 불러올 수 있으며 이름을 배열의 요소값으로 사용할 수 도 있다.

```
<script>
var fruits={apple:'사과', banana:'바나나', melon:'메론', watermelon:'수박'}

// 배열첨자로 가져오기
document.write("apple -->" + fruits['apple'] + "<br>");
document.write("banana -->" + fruits['banana'] + "<br>");
document.write("melon -->" + fruits['melon'] + "<br>");
document.write("watermelon -->" + fruits['watermelon'] + "<br>");
</script>
```

5.2.2. 배열 유형

자바스크립트에서 배열의 초기값을 지정할때는 {}를 사용하지만 JSON 표기법에서는 []를 사용하여 배열을 정의한다.

```
<SCRIPT LANGUAGE="JavaScript">
  var mountine=['백두산','지리산','한라산','설악산'];
  for(i=0 ; i<mountine.length ; i++){
    document.write(mountine[i] + "<br>");
  }
</SCRIPT>
```

5.2.3. 혼합형

이름:값 유형과 배열 유형을 혼합하여 사용할 수도 있다. 위 예제에서 사용되었던 fruits와 mountine의 객체를 하나로 합하는 예를 사용해 보자.

전체적인 구조는 이름:값 형태를 취하고 배열형 또한 배열명:[값] 형태로 변경하면 된다.

```
var mountineFruits = {
    apple:'사과', banana:'바나나', melon:'메론', watermelon:'수박',
    mountine:['백두산','지리산','한라산','설악산']
};
```

접근 방법을 살펴 보도록 하자.

```
str = "apple = " + mountineFruits.apple + "<br>";

for(i=0 ; i<mountineFruits.mountine.length ; i++){
    str += (mountineFruits.mountine[i] + "<br>");
}

rs.innerHTML =str;
```

5.2.4. JSON으로 2차원 배열 만들기

Map 구조

```
var data =[
       {apple:'사과', banana:'바나나', melon:'메론', watermelon:'수박'},
       {apple:'사과1', banana:'바나나1', melon:'메론1', watermelon:'수박1'},
];
```

Map 구조 형태로 2차원 배열을 선언한 경우 아래와 같이 제어할 수 있다.

data[0].apple --> 사과
data[1].apple --> 사과1

2차원 배열 구조

```
var data = [
  [ 1,2,3,4], [5,6,7,8], [9,10,11,12]
]
```

2차원 배열 구조 형태로 배열을 선언하여 사용할 경우 여타 언어와 동일한 방법으로 제어할 수 있다.

data[0][0] --> 1
data[1][0] --> 5

5.3. JSON을 사용한 클래스 정의

JSON 표현식을 사용하지 않고 클래스를 정의하려면 아래와 같이 prototype을 따로 정의해야 했다.

MyClass = function(){ ... }
MyClass.prototype.setName = function(){ ... }
MyClass.prototype.setAge = function(){ ... }
MyClass.prototype.setAddress = function(){ ... }

위의 소스를 JSON 표현법을 사용하여 정의하면 아래와 같다.

```
1.  // ---------------------------------------
2.  // JSON을 이용한 객체 정의
3.  // ---------------------------------------
4.  // 클래스 정의
5.  MyClass = function(){
6.
7.  }
8.
9.  // 메서드를 JSON 표현법으로 정의
10. MyClass.prototype ={
11.   setName:function(n){
12.     this.name = n;
13.   },
14.
15.   setAge:function(a){
16.     this.age = a;
17.   },
```

```
18.
19.    setAddress:function(a){
20.      this.address = a;
21.    }
22. }
23.
24. // MyClass를 사용하여 객체를 생성하고 메서드를
25. // 사용하여 값을 저장한후 출력하는 함수 정의
26. function first(){
27.    var str = "";
28.    var rs = document.getElementById("result"); // result는 DIV태그의 ID
29.
30.    mc = new MyClass();
31.    mc.setName("itdocument");
32.    mc.setAge(10);
33.    mc.setAddress("대한민국");
34.
35.
36.    str = "Name : " + mc.name + "<br>";
37.    str += "Age : " + mc.age + "<br>";
38.    str += "Address : " + mc.address;
39.
40.    rs.innerHTML =str;
41. }
```

* 기타 자세한 사항은 JSON 관련 사이트를 방문하여 살펴 보기 바란다.
(http://www.json.org/json-ko.html)

5.4. JSON 유형 응답 처리

JSON 유형의 응답처리를 하려면 데이터 발생도 JSON 타입으로 발생시켜야 하며 수신 데이터는 JSON.parse()를 사용하여 데이터를 JSON유형의 객체로 변환하여 사용해야 편리하다.

[예1] 텍스트 파일로 이루어진 JSON 타입 데이터 읽기

```
1.  <%@ page language="java" contentType="text/html; charset=UTF-8"
2.     pageEncoding="UTF-8"%>
3.  <!DOCTYPE html>
4.  <html>
5.  <head>
6.  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7.  <title>json_loader</title>
```

```
<script>
var xhr;
function first(){
  if(xhr == null) xhr = new XMLHttpRequest();
  xhr.onreadystatechange = loadDoc; // callback 지정
  xhr.open("get","json_data1.txt", true);
  xhr.send(null);
}
//eval() 함수 사용
function loadDoc(){
  if(xhr.status==200 && xhr.readyState==4){
      var doc = xhr.responseText;
      var rst = document.getElementById("result");

      var list = JSON.parse(doc);
      var str = '<ul>';
      for(i=0; i<list.length; i++){
          str += '<li>' + list[i].id;
          str += '<li>' + list[i].pwd;
          str += '<li>' + list[i].address;
          str += '<hr/>';
      }
      str += '</ul>';
      rst.innerHTML  = str;
  }
}

</script>
</head>
<body>
<h2>json 타입으로 입력된 데이터를 읽어들이는 예제</h2>

<input type="button" value="실행"  onClick="first()">
<hr>
<div id="result"></div>

</body>
</html>
```

json 형태로 만들어진 데이터 파일

json_data1.txt

```
[
 { "id": "a001" , "pwd":"1111","address":"종로1"},
 { "id": "a002" , "pwd":"1111","address":"종로2"},
 { "id": "a003" , "pwd":"1111","address":"종로3"}
 ]
```

[실행결과]

[예2] JSP로 만들어진 JSON 타입 데이터 읽기

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html >
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>json_loader</title>
<script>
var xhr;
function first(){
  if(xhr == null) xhr = new XMLHttpRequest();
  xhr.onreadystatechange = loadDoc; // callback 지정
  xhr.open("get","data.jsp", true);
  xhr.send(null);
}

// JSON.parse사용
function loadDoc(){
  if(xhr.status==200 && xhr.readyState==4){
      var doc = xhr.responseText;

      var newDoc = document.implementation.createHTMLDocument("title");
      newDoc.write(doc);
      var id = newDoc.getElementById("json_data");
      var data = JSON.parse(id.innerHTML);
      var rst = document.getElementById("result");

      var str = "<h2>result</h2>";
      for(i=0 ; i<data.length; i++){
          str += "<li><b>ID : </b>" + data[i].mid;
          str += "<li><b>Address : </b>" + data[i].address;
          str += "<li><b>Phone : </b>" + data[i].phone;
          str += "<hr/>";
      }

      rst.innerHTML = str;
  }
}
```

```
</script>
</head>
<body>
<h1>json 타입 배열 데이터를 읽어들이는 예제</h1>
<input type="button" value="실행" onClick="first()">
<hr>
<div id="result"></div>
</body>

</html>
```

데이터 파일

data.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>json_data</title>
</head>
<body>
<div id='json_data'>
<%
// json 배열 타입의 일반 데이터
// 작은 따옴표를 큰 따옴표로 바꾸어 전달.
String data ="[{'mid':'aaa', 'address':'서울', 'phone':'123-1234'},";
data += "{'mid':'bbb', 'address':'부산', 'phone':'3333-4444'}]";
String newData = data.replaceAll("'", "\"");
out.print(newData);

%>
</div>

</body>
</html>
```

6. 패키지 정의하기

6.1. 패키지의 개념

패키지는 클래스와 유사한 개념이다. 클래스도 하나의 바구니의 개념이 있는것 처럼, 패키지도 하나의 바구니이다. 그러나 클래스 보다는 더 커다란 바구니라 생각하면 된다.

날짜를 제어하는 클래스를 구성했다 하자. 아마도 대다수의 프로그래머들은 Date란 클래스명을 생각하거나 Nalja(? ㅋㅋㅋ) 일수도 있을 것이다. 그러나 같은 이름의 클래스명을 생각했다 하더라도 그 안에 들어가 있는 프로그램의 기능까지도 동일하게 만들 수 있는 확률을 얼마나 될까 ???

또는 같은 이름의 클래스명이더라도 그 기능이 서로 달라 모두 필요하다라고 가정해 보자. 상황은 더욱 복잡해 질뿐이다.

이럴때 간편하게 사용할 수 있는 개념이 패키지이다. 같은 이름의 Date 클래스 이더라도 서로 다른 바구니에 담아두었다면 필요할 때 꺼내 쓸 수 있지 않겠는가.

6.2. 정의하기

자바스크립트를 사용하여 패키지를 정의하는 방법은 의외로 간단하면서 단순하다. itdocument란 패키지를 정의하고 싶다면 아래와 같이 단 한줄로 정의가 가능하다.

var itdocument = new Object();

너무도 간단하지 않는가. 이렇게 패키지를 정의한후 해당 패키지 안에 클래스를 구성하려고 할 때면 클래스명 앞에 패키지명과 "."을 붙여 서로 다른 바구니임을 표현하면 된다.

var itdocument.Date = function(){...}

위와 같이 정의했다면 itdocument 패키지안에 있는 Date 클래스임을 나타낸 것이다. itdocument 패키지 안에 Date클래스를 새롭게 정의하여 사용하는 예는 다음과 같다.

파일명 : package_1.html(일부분)

```
1. // ---------------------------------------
2. // package 정의
3. // ---------------------------------------
4. var itdocument = new Object();
5.
6. // package 의 함수 정의하기
7. itdocument.Date =
8. function(){
9.   this.d = new
```

```
10.  Date();
11. }
12.
13. // 패키지내의 메서드를 JSON 표현법으로 정의
14. itdocument.Date.prototype ={
15.   setYear:function(y){
16.     this.dsetYear(y);
17.   },
18.
19.   setMonth:function(m){
20.     this.d.setMonth(m);
21.   },
22.
23.   setDate:function(d){
24.     this.d.setDate(d);
25.   }
26. }
27.
28. // 패키지내의 객체를 생성하고 메서드를
29. // 사용하여 값을 저장한후 출력하는 함수 정의
30. function first(){
31.   var str = "";
32.   var rs = document.getElementById("result"); // result는 DIV태그의 ID
33.
34.   mc = new itdocument.Date();
35.   mc.setDate(2009);
36.   mc.setMonth(10);
37.   mc.setDate(25);
38.
39.
40.   str = "Year : " + mc.d.getYear() + "<br>";
41.   str += "Month : " + mc.d.getMonth() + "<br>";
42.   str += "Date : " + mc.d.getDate();
43.
44.   rs.innerHTML =str;
45.
46. }
```

6.3. 패키지 중첩

패키지안에 또 다른 패키지를 정의하는 것을 패키지 중첩이라 한다. 물론 거의 모든 객체 지향언어에는 중첩된 패키지들이 수없이 많이 존재한다. 사용 목적과 용도에 따라 그룹화하여 바구니에 바구니, 또 그 안에 다른 바구니를 만들어 정의하고…

이 역시 아주 간단하에 패키지를 중첩하여 선언할 수 있다.

```
var itdocument = new Object();
var itdocument.itjava = new Object();
```

위에서도 알 수 있듯이 패키지와 패키지의 구분도 "."를 사용하여 구분하고 그 포함관계도 나타낸다.
함수의 정의도 크게 다르지 않다.

```
itdocument.itjava.MyClass = function(){...}
...
mc = new itdocument.itjava.MyClass(); // 패키지안에 있는 클래스 생성
```

6.4. JSON의 패키지 정의

Object를 사용하지 않고 JSON 표현방법을 사용하여 패키지를 정의할 수 있다.

```
var itdocument = {};
itdocument.itjava = {};
itdocument.itjava.MyClass = function(){ ... }
```

클래스를 생성하여 사용하는 방법은 동일함으로 생략하도록 한다.

7. 응용편

7.1. 응용편-제시어 기능

최근들어 검색 사이트 뿐만 아니라 많은 사이트들이 검색 기능들을 제공하고 있으며, 오히려 검색 기능을 제공하지 않는 사이트들이 뭔가 2% 부족하다는 느낌 마져 들게 하고 있다. 이 검색 기능의 기능을 살펴보면, 사용자가 특정 단어를 입력 하였을 때 이와 유사하거나 관련된 단어들을 보여주고 사용자로 하여금 보다 빠르게 검색할 수 있도록 도와주고 있다.

이런 기능을 "제시어기능"이라 표현하고 이런 기능을 Ajax를 사용하여 구현해 보고자 한다.

7.1.1. 개발범위

제시어 목록은 서버가 10여개중에서 적당한 것을 제공하고 사용자가 검색한 단어를 근거로 조합하여 제시하도록 한다.

7.1.2. 프로세스 흐름

제시어 기능은 서버의 기능과 클라이언트의 기능이 서로 연합되어 만들어져야 한다.

7.1.3. 서버 기능

사용자가 입력한 검색어를 기준으로 관련 검색어에 해당하는 제시어를 자신이 갖고 있는 데이터를 사용하여 만들어야 한다. 이때 자신이 갖고 있는 데이터라 함은 소비자들의 성향들이 적용되어 있어야 함은 물론, 자주 검색되어지는 문장, 사업적 요구사항에 근거한 문장 등등을 기준으로 제시어 목록을 만들어야 한다.

7.1.4. 클라이언트 기능

사용자로부터 검색어를 입력받아 서버로 전달하는 기능과 서버로부터 전달 받은 제시어를 적당한 표현 형태로 표시해 주는 기능이 있어야 한다.

표시해 주는 방법들이야 여러 가지가 있지만 이곳에서는 태그와 style속성중 layer 기능을 사용하여 표시해 주고자 한다.

7.1.5. 프로그램 초기화

다음과 같은 내용의 작업을 초기화 단계에서 처리한다.

1) 입출력 화면을 설계.
2) 프로그램에 필요한 내용을 초기화.
3) XMLHttpRequest 객체를 생성.
4) 각 이벤트에 해당하는 함수를 정의.

7.1.6. 사용자 인터페이스 구성

입력창과 버튼은 화면상에 나타나나,
6행의 select 박스는 초기 속성이 감춤으로 되어 있어 보이지 않게 된다.

```
1.  <body>
2.  <p> Ajax를 활용한 제시어 입력 테스트</p>
3.  <div id='div1'>
4.    <input type="text" size=30 name="findStr" id="findStr" value='it'>
5.    <input type="button" value="검색" id="btn"><br>
6.    <select size=10 id="suggest" style="width:225px;"></select>
7.  </div>
8.  <div id="result"></div>
9.  </body>
```

스타일 속성

```
<style>
#div1{
   position: relative;
}
#suggest{
   position: absolute;
   display: none;
}

#result{
   border:1px #FF9900 solid;
   width:273px;
   height:150px;
   padding:10px;
   box-sizing: border-box;
   margin-top:10px;
}
</style>
```

7.1.7. 프로그램 초기화

페이지가 로딩 되면 실행되도록 window.onload 이벤트 핸들러를 적용하였다.

```
1.  var userSuggest = ""; // 사용자 검색어
2.  var suggest; //제시어 결과
3.  var xhr;
4.  var rs;
5.
6.  window.onload=function(){
7.    xhr = new XMLHttpRequest();
8.    rs = document.getElementById("result");
9.    suggest = document.getElementById("suggest");
10.
11.   document.getElementById("findStr").onkeyup = keyupFunc;
12.   document.getElementById("btn").onclick = keyupFunc;
13.   suggest.onclick = resultFunc;
14.   suggest.addEventListener("onmouseover",mover);
15. }
16.
```

13~14행은 서로 다른 방법으로 이벤트를 추가하였으나 14행은 기존 방법대로

suggest.onmouseover = mover;

와 같이 작성해도 된다.

7.1.8. 이벤트에 해당하는 함수 정의

각 용도별로 함수명과 그 기능들을 간략히 요약해 놓는다.

```
1.  <script>
2.  window.onload=function(){}
3.
4.  //제시어 항목에 마우스가 올라가면
5.  function mover(){}
6.
7.  // 키가 눌려지거나 검색버튼이 클릭되면
8.  function keyupFunc(event){}
9.
10. // 서버로 부터 데이터가 전송되었을때
11. function callBackFunc(){}
12.
```

```
13. //제시어 항목에서 클릭하면
14. function resultFunc(){}
15. </script>
```

7.1.9. 콜백함수 정의

사용자로부터 검색할 데이터가 입력되면 서버에 내용을 전달하고 그 결과를 받아 사용자 화면에 내용을 표시하는 기능을 담는다.

```
1.  // 서버로 부터 데이터가 전송되었을때
2.  function callBackFunc(){
3.    if(xhr.readyState == 4){
4.      if(xhr.status == 200){
5.        temp = userSuggest + xhr.responseText ;
6.        temp = temp.split(",");
7.        resultStr="";
8.        suggest.length = 0; // options 모두 지움
9.        for(i=0 ; i<temp.length-1 ; i++){
10.         suggest.options[i] = new Option(temp[i],i);
11.       }
12.     }
13.   }
14. }
```

3~4행 : 서버로 값이 정상적으로 전달되고 그 응답이 모두 도착했음을 의미함.
5행 : 서버로부터 전달된 값을 텍스트 형태로 받는다.
6~11행 : 도착한 메시지를 ","로 분리하고 <select>태그의 <option>객체를 생성하여 표시한다.

7.1.10. 검색어 입력 기능

검색어가 입력되거나 변경되면 해당 내용을 서버로 전달한다.

```
1.  // 키가 눌려지면
2.  function keyupFunc(event){
3.    ev = window.event || event;
4.    str = document.getElementById("findStr");
5.
6.    // 검색할 단어가 있을때만 제시어창 보여주기
```

```
7.    if(str.value.length >0){
8.       suggest.style.display="block";
9.    }else{
10.      suggest.style.display="none";
11.   }
12.
13.   // 서버에 전달(한글 깨짐 방지)
14.   str = escape(encodeURIComponent(str.value));
15.
16.   url = "./suggest_process.jsp?findStr=" + str;
17.   xhr.onreadystatechange = callBackFunc;
18.   xhr.open("get", url, true);
19.   xhr.send(null);
20.
21. }
```

- 2행 : 키보드가 눌려지면 해당 이벤트의 종류와 이벤트가 발생한 객체를 매개변수로 하여 호출된다.
- 3행 : 크로스 브라우징 기능으로 IE계열과 W3C 표준 계열의 브라우저를 동시에 지원하도록 한다.(IE7이상에서는 W3C의 표준안을 따르므로 불필요하다.)
- 4행 : 입력된 문자열을 저장한다.
- 7~11행 : 입력된 문자열이 0보다 크면 제시문이 있는 문단을 보여주고 그렇지 않으면 감춘다.
- 14행 : 한글이 포함되어 있는 검색어가 정상적으로 전달되도록 인코딩햐였다.
- 16~19행 : 검색정보를 서버에 전달.

7.1.11. 검새결과 표시

사용자 편의성을 제공하려면 더 많은 세세한 기능들을 부여해야 하겠지만 이 예에서는 제시된 용어 선택에 대한 편의성만을 제공토록 하겠다.

```
1.  //제시어 항목에 마우스가 올라가면
2.  function mover(){
3.    suggest.selectedIndex = 0;
4.  }
5.
6.  //제시어 항목에서 클릭하면
7.  function resultFunc(){
8.    if(suggest.selectedIndex < 0 ) return;
9.       findStr.value = suggest[suggest.selectedIndex].text;
10.      suggest.style.display="none";
11.      var result = document.getElementById("result");
12.      result.innerHTML = findStr.value + " 로(으로) 검색됨.";
13. }
```

7.1.12. 서버

사용자로 하여금 검색어가 전달된 경우 그 결과를 사용자에게 되돌려주는 서버파일이다.

suggest_process.jsp

```
<%@page import="java.net.URLDecoder"%>
<%@page contentType="text/html;charset=UTF-8"%>
<%
request.setCharacterEncoding("UTF-8");
String str = request.getParameter("findStr");
//한글 깨짐 방지
str = URLDecoder.decode(str, "utf-8");

String resultStr = "";
String list[] = {
 "itjava","itdocument",
 "박원기","원기왕성","박원기짱",
 "abc","ab","aa","abc","def", "aabbcc",
 "123","234","452","c1331","22"
 };

int pos=-1;
for(int i=0 ; i<list.length ; i++){
 if(list[i].contains(str)) {
       resultStr += (list[i] + ",");
 }
}

out.print(resultStr);
%>
```

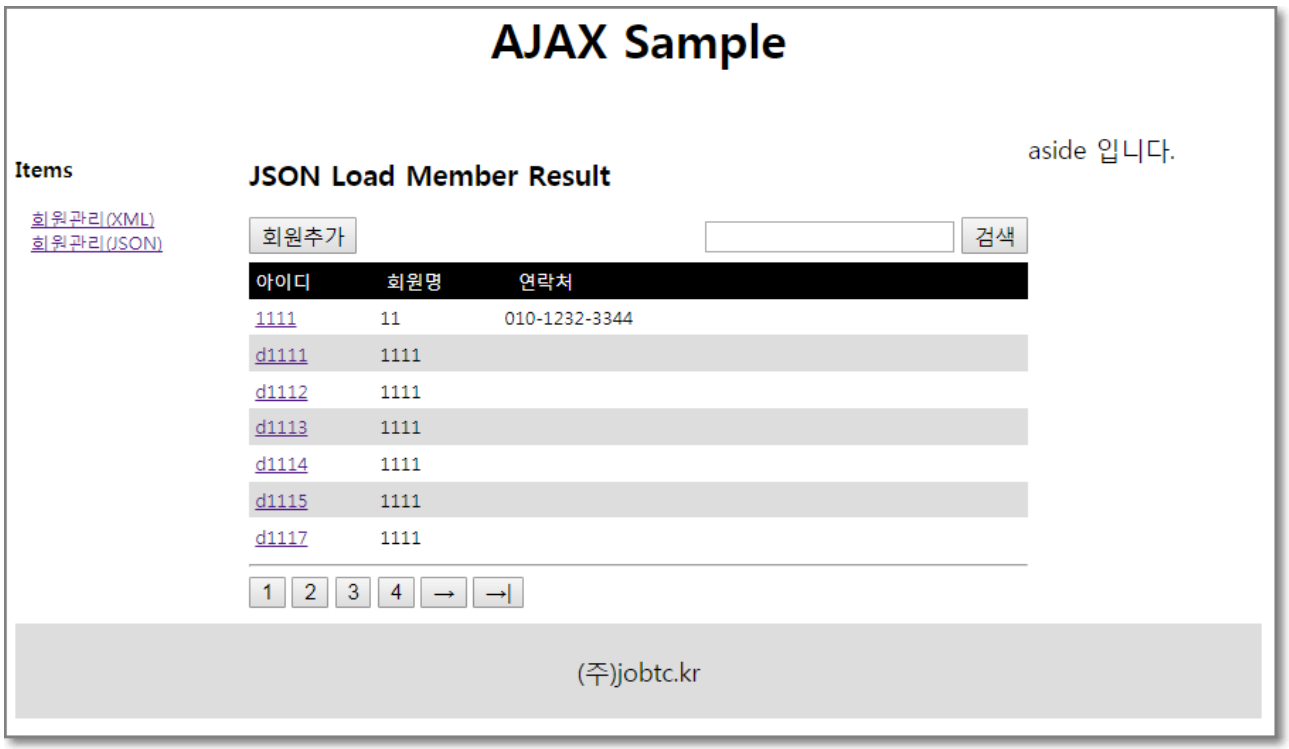
실무에서 쓰이는것과 같이 완벽한 소스를 구현하지는 않았지만 제시어 기능들이 어떤 프로세스를 갖고 실행되는지에 대한 이해는 되었을 것으로 보고 제시어 기능에 대한 예를 마치도록 하겠다.

7.2. XML / JSON 처리를 위한 공통부분

7.2.1. 테이블 구조

	PK	Column Name	Data Type	Not Null
1	PK	MID	VARCHAR2(20 BYTE)	☑
2		IRUM	VARCHAR2(30 BYTE)	☐
3		RDATE	DATE	☐
4		POINT	NUMBER(10,1)	☐
5		GENDER	VARCHAR2(1 BYTE)	☐
6		EMAIL	VARCHAR2(50 BYTE)	☐
7		NICNAME	VARCHAR2(30 BYTE)	☐
8		PWD	VARCHAR2(20 BYTE)	☐
9		ADDRESS1	VARCHAR2(50 BYTE)	☐
10		ADDRESS2	VARCHAR2(50 BYTE)	☐
11		POST	VARCHAR2(10 BYTE)	☐
12		PHONE	VARCHAR2(20 BYTE)	☐

7.2.2. 전체 구조



7.2.3. index.jsp

```
<body>

<div id='ajax'>
 <div id='header'>AJAX Sample</div>

 <div id='body'>
   <div id='menu'>menu</div>
   <div id='content'>content</div>
   <div id='aside'>aside</div>
 </div>

 <div id='footer'>(주)jobtc.kr</div>

</div>
<script>init()</script>
</body>
</html>
```

7.2.4. javascript

```
<script>
var xhr;

function init(){
 xhr = new XMLHttpRequest();
 xhr.open('get', 'menu.jsp');
 xhr.send();
 xhr.onreadystatechange=loadMenu;

}


function loadMenu(){
 if(xhr.status == 200 && xhr.readyState == 4){
   var doc = xhr.responseText;
   var menu = document.getElementById("menu");
     menu.innerHTML = doc;
     loadIntro();
   }
}
```

```
21.
22. function loadIntro(){
23.  xhr.open('get', 'intro.jsp');
24.    xhr.send();
25.  xhr.onreadystatechange = function(){
26.    if(xhr.status == 200 && xhr.readyState==4){
27.      var doc = xhr.responseText;
28.      var content = document.getElementById('content');
29.        content.innerHTML = doc;
30.        loadAside();
31.      }
32.    }
33. }
34.
35. function loadAside(){
36.  xhr.open('get', 'aside.jsp');
37.    xhr.send();
38.  xhr.onreadystatechange = function(){
39.    if(xhr.status==200 && xhr.readyState == 4){
40.      var doc = xhr.responseText;
41.      var aside = document.getElementById('aside');
42.        aside.innerHTML = doc;
43.      }
44.    }
45. }
46.
47. function loadPage(page){
48.  xhr.open('get', page);
49.    xhr.send();
50.  xhr.onreadystatechange = function(){
51.    if(xhr.status == 200 && xhr.readyState ==4){
52.      var doc = xhr.responseText;
53.      var content = document.getElementById("content");
54.        content.innerHTML = doc;
55.
56.      if (match = doc.match(/<script[^>]*>(([^<]|\n|\r|<[^\/])+)<\/s
   cript>/)){
57.          eval.call(window, match[1]);
58.        }
59.
60.        start();
61.      }
62.    }
63. }
64.
```

```
65. </script>
```

7.3. 회원 정보를 XML로 처리하기

MyOraclePool.java 를 통하여 데이터는 ArrayList<MemberVo> 타입으로 리턴된다고 가정합니다.

Step 1.

JSP 페이지를 사용하여 간단한 UI와 Ajax 객체를 선언합니다.

memberXMLAjax.jsp

```
1.  <%@ page language="java" contentType="text/html; charset=UTF-8"
2.   pageEncoding="UTF-8"%>
3.  <!DOCTYPE html>
4.  <html>
5.  <head>
6.  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7.  <title>memberXMLAjax</title>
8.  <script>
9.   var xhr;
10.  function init() {
11.     document.getElementById('btnFind').onclick = function() {
12.        if (xhr == null) xhr = getXMLHTTP();
13.        var findStr = document.getElementById('findStr').value;
14.        xhr.open('get', './memberXMLAjax_result.jsp?findStr=' + findStr, tru
    e);
15.        xhr.onreadystatechange = resultFunc;
16.        xhr.send(null);
17.     }
18. }
19.
20.  function getXMLHTTP() {
21.     if (window.ActiveXObject) {
22.        xhr = new ActiveXObject("Msxml2.XMLHTTP");
23.     } else {
24.        xhr = new XMLHttpRequest();
25.     }
26.     return xhr;
27. }
28.
29.  function resultFunc() {
30.     var rs = document.getElementById('result');
31.     if (xhr.readyState == 4 && xhr.status == 200) {
32.
33.        var doc = xhr.responseXML;
34.        var list = doc.getElementsByTagName("member");
35.        rs.innerHTML = "length:" + list.length;
36.        for (i = 0; i < list.length; i++) {
```

```
            var data = list.item(i);
            var mid = data.getElementsByTagName("mid").item(0).firstChild.nodeV
alue;
            var irum = data.getElementsByTagName("irum").item(0).firstChild.nod
eValue;
            rs.innerHTML += "<hr/><li>mid:" + mid + "<li>irum:"
                + irum;
        }


    }

 }
</script>
</head>
<body>
  <h3>회원 정보를 XML로 처리하여 Ajax로 처리</h3>
  <input type='text' name='findStr' id='findStr'>
  <input type='button' id='btnFind' value='검색' />
  <hr />
  <div id='result'></div>
  <script>
     init();
  </script>
</body>
</html>
```

Step 2.

MyOralcePool로 부터 데이터를 읽어 들여 XML 유형으로 리턴합니다. 물론 테그라이브러리는 필수 사항은 아닙니다.

memberXMLAjax_result.jsp

```
<?xml version="1.0" encoding="utf-8" ?>

<%@page import="bean.MemberVo"%>
<%@page import="java.util.ArrayList"%>
<%@page import="bean.MyOraclePool"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<myroot>
 <%
    //회원 정보 데이터를 xml로 변환하여 처리
    String findStr = request.getParameter("findStr");
    MyOraclePool pool = new MyOraclePool();
```

```
14.    ArrayList<MemberVo> al = pool.select(findStr);
15.    request.setAttribute("al",al);
16.    response.setContentType("text/xml");
17.  %>
18.  <c:forEach items="${al }" var="data">
19.    <member>
20.      <mid>${data.mid }</mid>
21.      <irum>${data.irum }</irum>
22.    </member>
23.  </c:forEach>
24.
25. </myroot>
```

7.4. 회원정보를 JSON으로 처리하기

MyOraclePool.java 를 통하여 데이터는 ArrayList<MemberVo> 타입으로 리턴된다고 가정합니다.

Step 1.

기본적인 UI를 만들고 Ajax를 통하여 서버로 부터 데이터를 수신받을 수 있도록 프로그래밍 합니다.

memberJSONAjax.jsp

```
1.  <%@ page language="java" contentType="text/html; charset=UTF-8"
2.    pageEncoding="UTF-8"%>
3.  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w
    3.org/TR/html4/loose.dtd">
4.  <html>
5.  <head>
6.  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7.  <title>Insert title here</title>
8.  <script>
9.   var xhr;
10.  function init() {
11.    document.getElementById('btnFind').onclick = function() {
12.      if (xhr == null) xhr = getXMLHTTP();
13.      var findStr = document.getElementById('findStr').value;
14.      xhr.open('get', './memberJSONAjax_result.jsp?findStr=' + findStr, tru
    e);
15.      xhr.onreadystatechange = resultFunc;
16.      xhr.send(null);
17.    }
18. }
19.
20.  function getXMLHTTP() {
21.    if (window.ActiveXObject) {
22.      xhr = new ActiveXObject("Msxml2.XMLHTTP");
23.    } else {
24.      xhr = new XMLHttpRequest();
25.    }
26.    return xhr;
27. }
28.
29.  function resultFunc() {
30.    var rs = document.getElementById('result');
31.    if (xhr.readyState == 4 && xhr.status == 200) {
32.
33.      var temp = xhr.responseText;
34.
35.      var doc = JSON.parse(temp);
36.      rs.innerHTML = "";
```

```
      for(i=0; i<doc.length; i++){
         rs.innerHTML += "<li>" + doc[i].mid + "---" + doc[i].irum;
      }

   }

}
</script>
</head>
<body>
 <h3>회원 정보를JSON으로 처리하여 Ajax로 처리</h3>
 <input type='text' name='findStr' id='findStr'>
 <input type='button' id='btnFind' value='검색' />
 <hr />
 <div id='result'></div>
 <script>
   init();
 </script>
</body>
</html>
```

Step 2.

MyOraclePool 로 부터 데이터를 전달받아 JSON형식으로 데이터를 구조화 시킵니다. 태그 라이브러리 사용은 선택적 사항입니다.

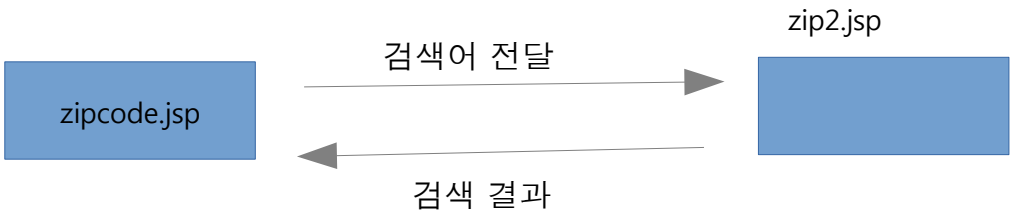
memberJSONAjax_result.jsp

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

<%@page import="org.apache.catalina.startup.SetContextPropertiesRule"%>
<%@page import="bean.MemberVo"%>
<%@page import="java.util.ArrayList"%>
<%@page import="bean.MyOraclePool"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%
 //회원 정보 데이터를 xml로 변환하여 처리
 String findStr = request.getParameter("findStr");
 MyOraclePool pool = new MyOraclePool();
 ArrayList<MemberVo> al = pool.select(findStr);
 request.setAttribute("al", al);
%>
[
 {"mid": "${al[0].mid }", "irum": "${al[0].irum }"}
```

```
<c:forEach items="${al }" var="data" begin='1'>
 ,{"mid": "${data.mid }", "irum": "${data.irum }"}
</c:forEach>

]
```

7.5. 공공DB를 사용한 우편 번호 검색

data.go.kr 에서 재공해 주는 DB를 사용하여 우편번호를 검색해 보자.



zipcode.jsp에서 ajax를 사용하여 공공 DB에 접근하여 데이터를 직접가져올 수 도 있지만 크로스 도메인 문제를 해결하기 위해 여러가지 설정들이 필요하다. 때문에 직접 처리하지 않고 zip2.jsp를 호출하여 URLConnection으로 데이터를 가져오도록 하였다.

[사전준비]
data.go.kr 사이트를 방문하여 계정을 생성한뒤 우편 번호 검색에 필요한 키를 생성해야 한다. 또한 우편 번호 검색과 관련한 문서를 참조하여 데이터 요청 구조와 응답 구조를 분석하자.

[zipcode.jsp]

```
<%@page import="java.net.URLEncoder"%>
<%@page import="javax.print.URIException"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" autoFlush
="true"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%
request.setCharacterEncoding("utf-8");
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>zipCode.jsp</title>
<style>
#result{
 width:350px;
 margin-top:3px;
 font-size:12px;
 overflow-x: scroll;
}
</style>
<script>
var xhr;
```

```
function zipFind(){
  var frm = document.frm_zip;
  var road = frm.road.value;

  var ip = './zip2.jsp?srchwrd=' + encodeURIComponent(road);

  xhr = new XMLHttpRequest();
  xhr.open('get', ip);
    xhr.send();
  xhr.onreadystatechange = function(){
    if(this.status==200 && this.readyState == 4){
      var rs = document.getElementById('result');
        rs.length=0;
      var doc = this.responseText;
      //메모리공간의 document에 저장
      var newDoc = document.implementation.createHTMLDocument("");
        newDoc.open();
        newDoc.write(doc);

      var nodeList = newDoc.getElementsByTagName("newAddressListAreaCd");

      if(nodeList.length==0){
        var op = new Option("찾는 자료가 없습니다.");
          rs.options[rs.length] = op;
        }

      for(i=0 ; i<nodeList.length ; i++){
        var node = nodeList.item(i);
        var zipNo = node.getElementsByTagName('zipNo')
                    .item(0).firstChild.nodeValue;
        var rnAdres = node.getElementsByTagName('rnAdres')
                    .item(0).firstChild.nodeValue;
        var lnmAdres = node.getElementsByTagName('lnmAdres')
                    .item(0).firstChild.nodeValue;

        var op = new Option('[ ' + zipNo + ' ] '+rnAdres, zipNo);
          rs.options[rs.length] = op;
        }
      }
    }

  frm.result.ondblclick = function(){
    var idx = frm.result.selectedIndex; //선택된 항목
    var zipCode = frm.result[idx].value;
    var tempAddr = frm.result[idx].text.split('] ');

    var pFrm = opener.document.frm_insert;
```

```
70.     if(pFrm == null){
71.         pFrm = opener.document.frm_modify;
72.       }
73.
74.       pFrm.post.value = zipCode;
75.       pFrm.address1.value = tempAddr[1];
76.       self.close();
77.     }
78.
79. }
80.
81.
82. </script>
83.
84. </head>
85. <body>
86.
87. <div id='zipCode'>
88.  <h3>우편번호 검색</h3>
89.  <form name='frm_zip' method='post'>
90.     <input type='text' name='road' value='세종로 17'/>
91.     <input type='button' id='btnZipFind' onclick='zipFind()'
92.            value='검색' /><br/>
93.     <select size='20' name='result' id='result'></select>
94.  </form>
95. </div>
96. </body>
97. </html>
```

69~76행 까지는 우편 번호를 필요로 하는 메인 창에 값을 넘기는 부분이다.

[zip2.jsp]

```
1.  <?xml version='1.0' encoding='utf-8' ?>
2.  <%@page import="org.w3c.dom.Document"%>
3.  <%@page import="org.w3c.dom.NodeList"%>
4.  <%@page import="org.w3c.dom.Element"%>
5.  <%@page import="javax.xml.parsers.DocumentBuilderFactory"%>
6.  <%@page import="java.io.InputStreamReader"%>
7.  <%@page import="java.io.BufferedReader"%>
8.  <%@page import="java.net.URLConnection"%>
9.  <%@page import="java.net.URL"%>
10. <%@page import="java.net.URLEncoder"%>
11. <%@ page language="java" contentType="text/xml; charset=UTF-8"
```

```
12.  pageEncoding="UTF-8"%>
13.  <%
14.  String inputLine = "";
15.  String buffer = "";
16.  request.setCharacterEncoding("utf-8");
17.
18.  String srchwrd = "&srchwrd="
19.         + URLEncoder.encode(request.getParameter("srchwrd"), "utf-8");
20.
21.  String ip = "http://openapi.epost.go.kr/postal/retrieveNewAdressAreaCdSe
    rvice/retrieveNewAdressAreaCdService/getNewAddressListAreaCd";
22.  ip += "?ServiceKey=사용자키값";
23.  ip += "&searchSe=road" + srchwrd;
24.
25.  URL url = new URL(ip);
26.
27.    URLConnection connection = url.openConnection();
28.  connection.setRequestProperty("CONTENT-TYPE", "text/xml");
29.  BufferedReader in = new BufferedReader(new InputStreamReader(url.openStr
    eam(), "utf-8"));
30.  while ((inputLine = in.readLine()) != null) {
31.       buffer += inputLine.trim();
32.    }
33.    in.close();
34.
35.    out.print(buffer);
36.
37.  %>
```