25장. Ajax

강사 강태광

Semantic web

1. 미래의 웹은 시맨틱 웹
   1) XML을 활용하여 컴퓨터가 정보의 뜻을 이해하고 자동화된 처리를 할 수 있는 인공지능형 웹
   2) 웹 2.0 서비스로 주목 받는 많은 웹 서비스가 XML을 사용한 정보 교환에 기반
   3) 하이퍼링크로 연결된 단순한 거미줄 -> 의미로 연결된 아주 촘촘한 그물망

2. 시맨틱 웹을 위한 기본
   1) 정보를 활용하기 위해서는 모양과 내용이 분리되어야 한다. 바이너리 형태의 정보는 HTML, XML 형태로 바꾸어야 한다.
   2) URI는 변하지 않아야한다 * URL(Uniform Resource Locator) -> URI(Uniform Resource Identifier
   3) 웹 페이지의 주소는 변하지 않고 항상 동일하게 유지되어 두려움 없이 연결

Ajax: Asynchronous JavaScript and XML

AJAX : Asynchronous JavaScript + XML
- JavaScript를 사용한 비동기 통신, 클라이언트와 서버간에 XML 데이터를 주고받는 기술

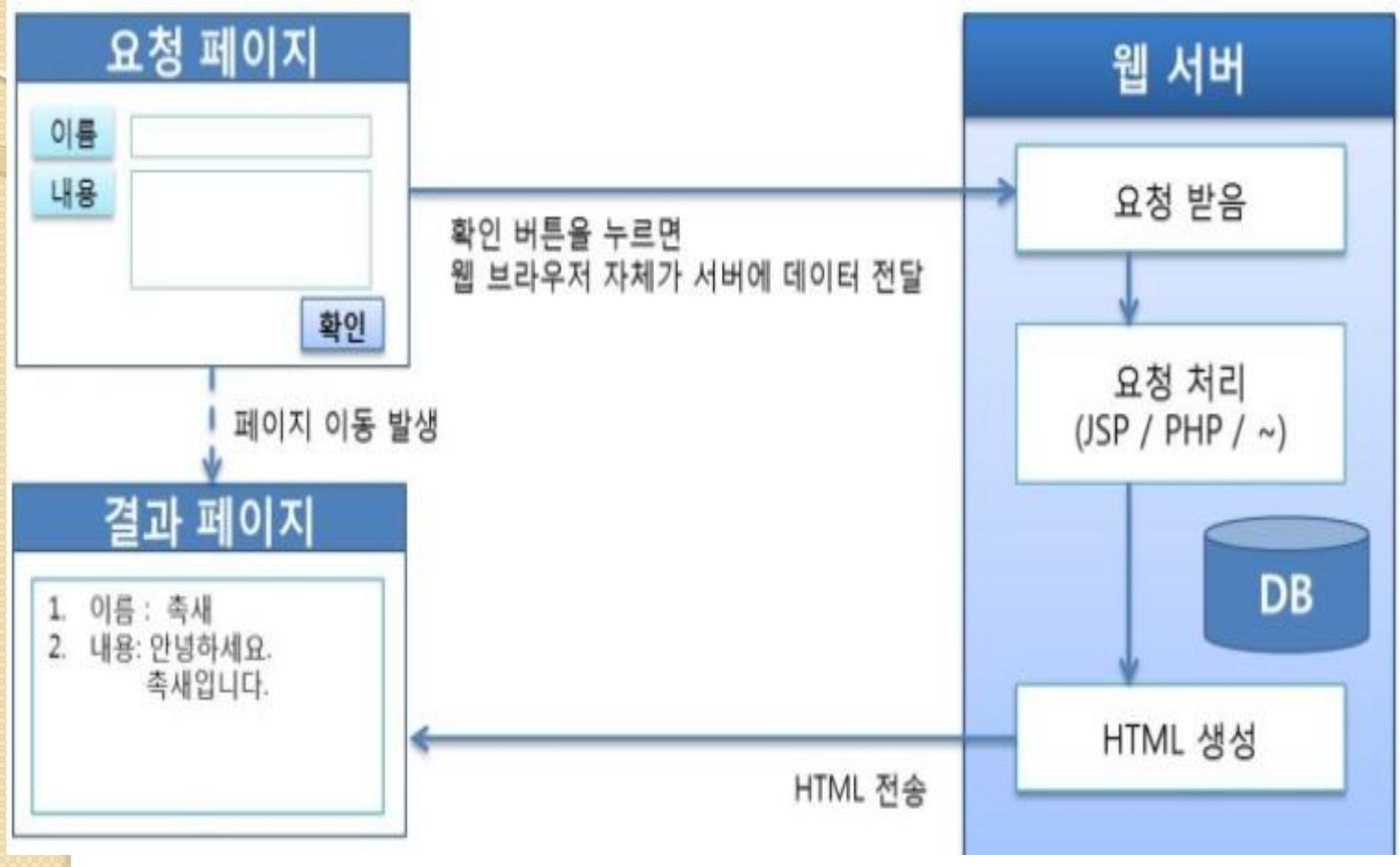
AJAX 장점
- 페이지 이동 없이 고속으로 화면 전환
- 서버 처리를 기다리지 않고 비동기 요청이 가능
- 서버 측 처리를 각 PC에 분산 가능
- 수신하는 데이터의 양을 줄임

AJAX 단점
- 크로스 브라우저화의 노하우 필요
- AJAX를 사용하지 못하는 브라우저
- 오픈 소스이므로 차별화가 어려움
- 보안에 더욱 신경을 써야 함

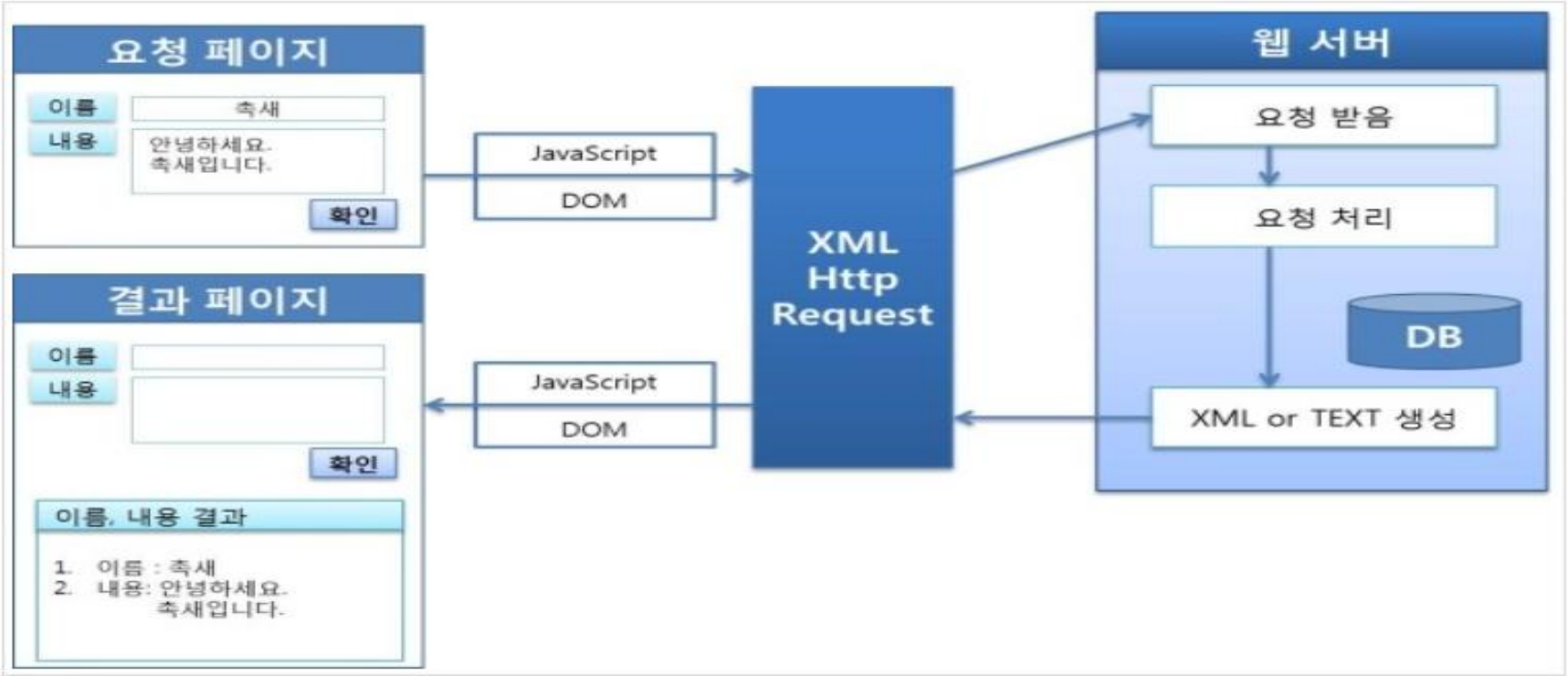
기존방식과 AJAX의 차이

1) 기존방식
- 웹 브라우저가 웹 서버에 요청 전송
- 웹 서버는 JSP 등의 서버 어플리케이션을 사용해 사용자의 요청 처리 후 결과를 HTML로 생성해서 웹 브라우저에 전송
- 웹 브라우저는 응답으로 받은 HTML을 분석한 뒤 그 내용을 화면에 그림
- 결과적으로 웹 브라우저가 웹 서버와 통신을 하고 요청 결과는 HTML로 생성되고 사용자 입장에서는 **페이지 이동이 발생함**

Ajax: Asynchronous JavaScript and XML



Ajax 사용 이유



AJAX의 주요 구성 요소

-XMLHttprequest : 웹서버와 통신을 담당함. 사용자의 요청을 웹서버에 전송 및 웹서버로부터 받은 결과를 웹브라우저에 전달함

- DOM : 문서의 구조를 나타냄, 폼 등의 정보나 화면 구성을 조작할때 사용함
- CSS : 글자색,배경색,위치,투명도 등 UI관련 부분을 담당
- 자바스크립트 : 사용자가 마우스를 드래그하거나 버튼을 클릭하면 XMLHttpRequest객체를 사용해 웹서버에 요청을 전송함.

XMLHttpRequest 객체로부터 응답이 오면 DOM, CSS등을 사용해 화면을 조작함

.

Ajax 적용 해결 방법

Ajax의 활용방향

- 페이지의 일부분에만 새로운 콘텐츠를 로드하는 기능은 사용자의 전체적인 사용 경험을 향상시킬 수 있다. 이는 페이지의 일부만 수정하게 되면 사용자가 전체 페이지가 로드될 때까지 기다릴 필요가 없기때문

Ajax 활용예시

① 라이브검색

라이브 검색(혹은 자동완성)이라고 불리는 기능은 주로 Ajax를 이용한다.
오늘날 검색사이트의 대부분이 사용하는 기술이다. 이 방법은 검색막대나 홈페이지에서 검색어를 입력하는 동시에 검색 결과가 나타나도록 한다.

② 사용자 정보 표시

사용자가 생성한 콘텐츠를 사용하는 웹사이트(예를들면, 트위터나 플리커 등)는 독자들의 웹사이트에 본인의 정보(최근 트윗이나 사진 등)를 노출하는 기능을 제공
이를 위해 자신들의서버에 데이터를 수집한다.
이 밖에도 회원가입시 중복아이디일 경우, "이미 사용중인 아이디입니다."를 표시하게 하는 기능에서도 활용되고있으며, 온라인쇼핑몰에서 장바구니에 원하는 물품을 추가했을 때 페이지 이동이나 전체 페이지에 대한 새로고침 없이도 물품정보만 추가되는 기능을 구현하고자 할 때, 이 Ajax를 활용

Ajax의 활용 분야

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

2. 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) 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) 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.

XMLHttpRequest 객체를 사용한 데이터 송수신

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

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

3. 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

1. 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)

1.XMLHttpRequest 객체는 웹 서버로부터 응답이 도착하면 특정한 자바 스크립트 함수를 호출하는 기능이 있는데 이 프로퍼티가 onreadystatechange

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

2.코드가 실행되면 /test.jsp에 접속하여 응답이 도착하면 **callbackFunction()** 함수 호출
- **onreadystatechange**에 지정한 함수를 콜백 함수라고 부르며 콜백함수에서 화면을 변경하거나 경고창을 뛰우는 등 응답결과에 따라 수행

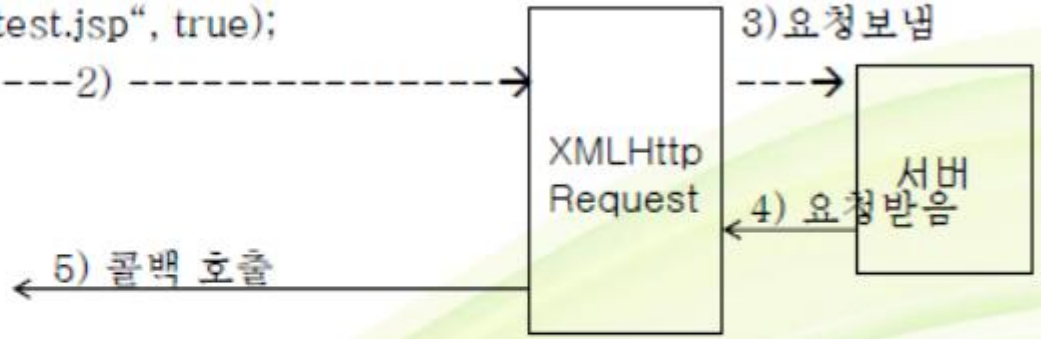
3. 코드가 실행하는 순서
. 사용자의 이벤트가 발생하면 이벤트 처리함수 호출
. 이벤트 처리함수에서는 XMLHttpRequest 객체의 send()함수 호출
. XMLHttpRequest객체의 send함수가 호출되면 웹 서버에 요청이 전송 된다
. XMLHttpRequest 객체에 응답이 도착하면 onreadystatechange 프로퍼티에 지정한 콜백함수 호출

서버 응답처리하기 (onreadystatechange 프로퍼티와 콜백함수)

```
<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 때마다 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 출력 하는 것이 좋음

서버로부터의 응답상태 : status/statusTest

- 서버로부터 응답이 도착하면 웹 서버에서의 처리가 올바르게 수행되었는지 확인
- **주요 상태코드**

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

- 서버로부터 응답이 오면 status 프로퍼티를 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

- 서버로부터 응답이 오면 status 프로퍼티를 사용해서 요청이 성공적으로 수행되었는지 확인

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

innerHTML 속성을 사용한 화면 변경

1. XMLHttpRequest
   - 사용자가 이벤트를 발생시키면 웹 서버에 데이터 전송
   - 웹 서버가 생성한 응답결과를 바탕으로 화면을 조작

2. 사용결과 화면의 내용이 변경되거나 새로운 내용 추가
   - HTML 요소의 innerHTML 속성에 HTML 코드 지정하기
   - DOM(Document Object Model) API 사용하기

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

innerHTML 속성을 사용한 화면 변경

---------- 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>
```

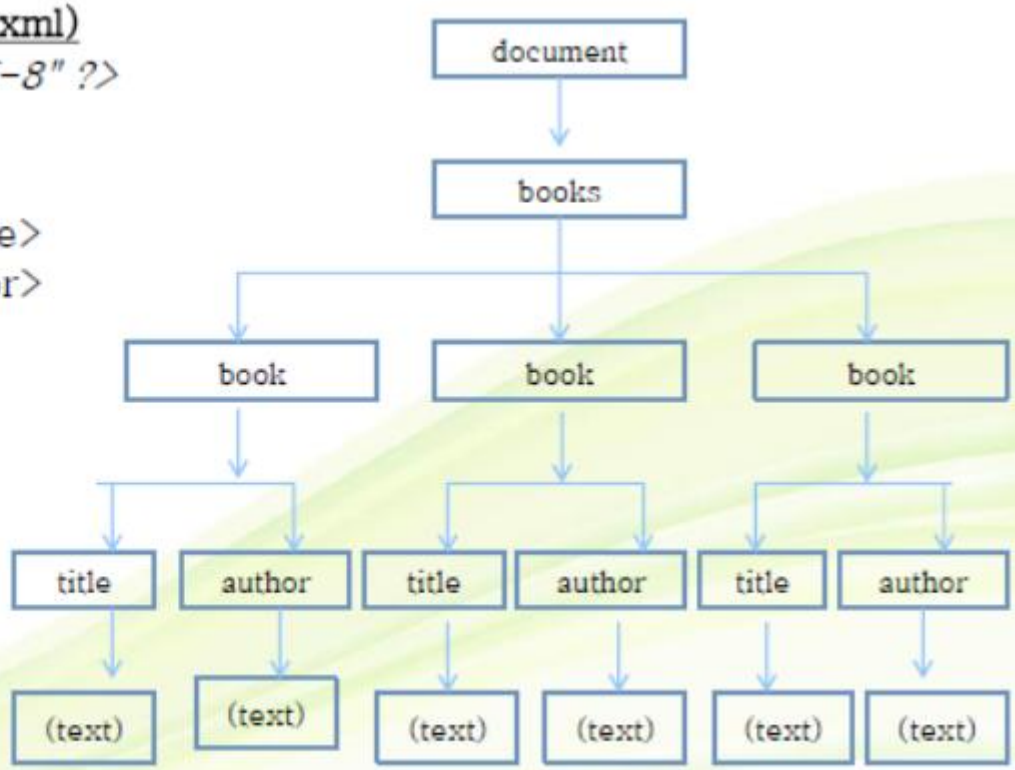
- Document.getElementById()함수는 id 속성의 값을 사용하여 원하는 html태그를 사용할 수 있도록 해주므로 p1변수는 id 속성값이 p1인 <P>태그와 일치하게 된다
- innerHTML 속성을 사용하면 화면의 내용을 변경할 수 있다.
  즉, innerHTML 속성에 원하는 HTML 코드만 값으로 주면 해당 내용 변경

DOM(Document Object Model)과 XML

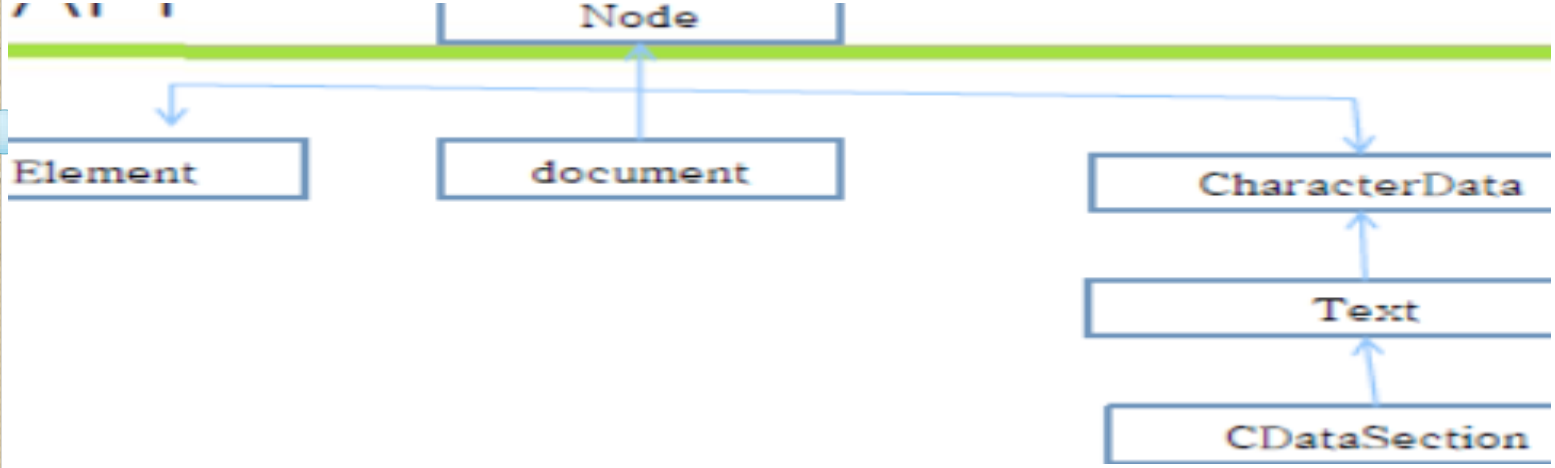
1. DOM문서를 객체로 표현하기 위한 표준으로서 HTML이나 XML 등의 문서를 객체로 표현할 때
   - 사용되는 API
2. DOM API는 문서를 Tree구조로 표현하여 이해하기 쉬움
3. XMLHttpRequest객체는 응답텍스트 대신 XML응답 결과를 사용할 수 있음
   이때 DOM API를 사용해서 서버가 생성한 XML로부터 데이터를 추출

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>
```



주요 DOM API



1. DOM API에서 모든 것은 Node로 표현하며 문서의 구성요소들은 모두 Node 또는 하위 인터페이스로 매핑
2. Document ; 문서 전체
3. Element : 각 태그
4. Text ; 문자열 데이터가 Text 노드로 표현
5. 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

주요 DOM API

2. 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

1) 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

2) % 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

4) 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 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

[1] <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 oldChild 노드를 새로운 newNode로 교체한다
- Node removeChild(Node oldChild) 현재 노드의 자식노드인 oldChild 노드를 현재노드에서 제거
- Node appendChild(Node newChild) newChild를 현재 노드의 마지막 자식 노드로 추가

JSON 표기법

1. JSON(JavaScript Object Notation) 표기법은 서로 다른 프로그래밍 언어간에
[1] 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'];
```

[2] 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

[3] JSON표기법을 사용한 클래스 정의
- JSON표기법의 이름/값 부분에서 함수 이름이 '이름'에 들어가도 함수의 정의가 '값' 들어감