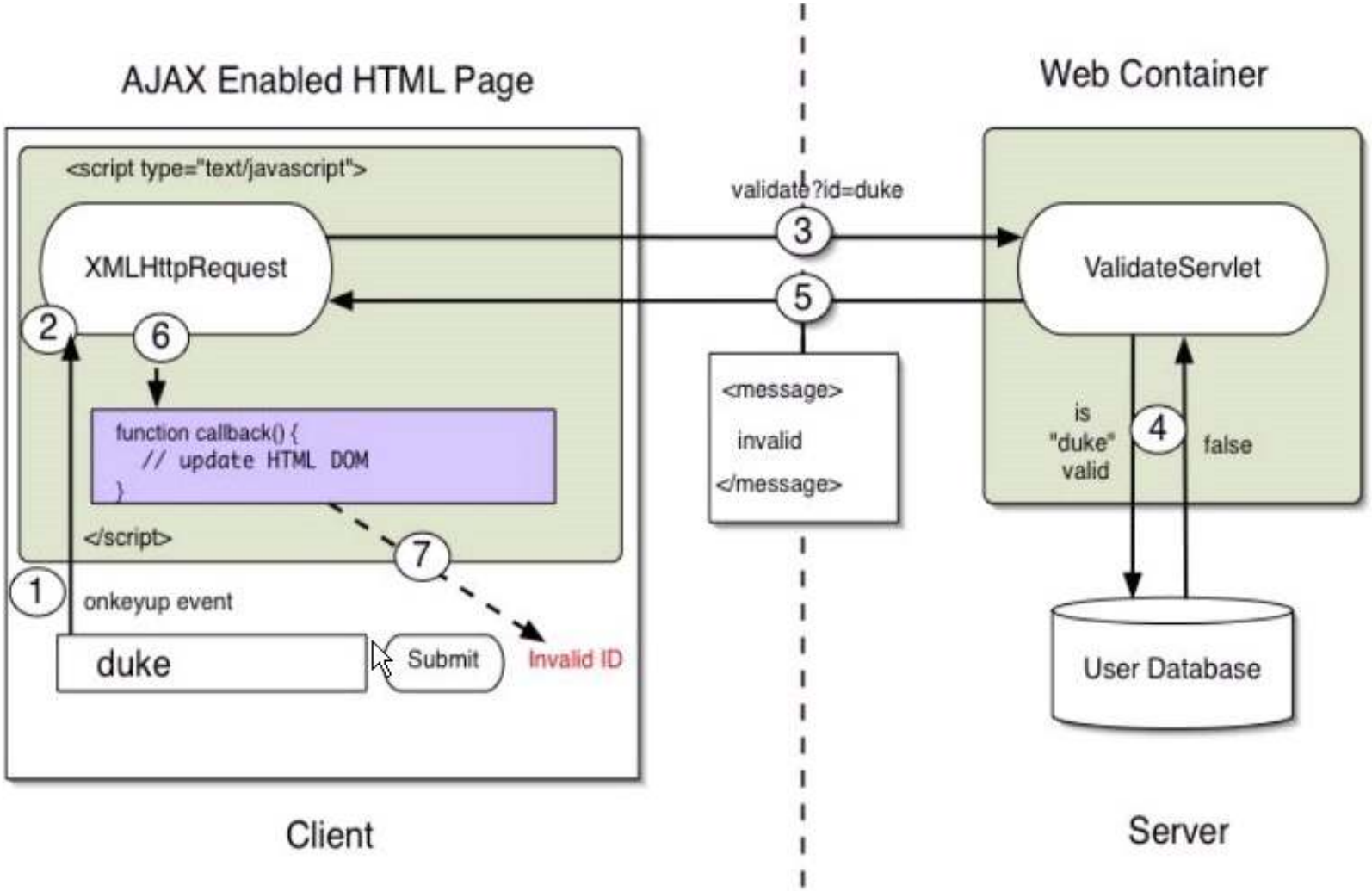
Anatomy of an AJAX Interaction



Ajax(Asynchronous JavaScript)

- JavaScript에 의한 비동기적인 통신으로 XML 기반의 데이터를 클라이언트인 웹 브라우저와 서버 사이에서 교환
- 페이지의 이동 없이 웹 브라우저 화면을 동적 변경
- Ajax 장점
  - 서버측의 부담 중 일부를 웹 클라이언트에게 넘겨주게 되어 서버 어플리케이션 성능 향상
  - “웹 브라우저는 요청을 송신하면 응답을 기다리지 않는다”
  - “서버는 필요한 데이터만을 응답한다”

Ajax(Asynchronous JavaScript)

- Ajax 구성 기술 요소
  - Javascript
    - 사용자 이벤트가 발생하면 XMLHttpRequest 객체를 사용해서 웹 서버에 요청을 전달
    - XMLHttpRequest 객체로부터 응답이 오면 DOM, CSS 등을 사용해서 화면을 조작
  - DOM
    - 문서의 구조
    - 폼 등의 정보나 화면 구성을 조작할 때 사용
  - CSS
    - 글자색, 배경색, 위치, 투명도 등 UI와 관련된 부분을 담당
  - XMLHttpRequest
    - 웹 서버와 통신
    - 사용자의 요청을 웹 서버에 전송하고, 웹 서버로부터 받은 결과를 웹 브라우저에 전달

XMLHttpRequest 지원 브라우저

- MS IE 4.0 이후
- Firefox 1.0 이후
- Opera 7.6 이후
- Safari 1.2 이후
- Netscape 7 이후
- Konqueror 3 이후

Ajax – 사용자 요청 처리 과정

- 사용자가 이벤트(마우스 클릭 등)를 발생
- 자바스크립트는 DOM을 사용해서 필요한 정보를 구함
- XMLHttpRequest 객체를 통해서 웹 서버에 요청을 전달
- 웹 서버는 XMLHttpRequest로부터의 요청을 알맞게 처리
- 웹 서버는 결과를 XML이나 단순 텍스트로 생성해서 XMLHttpRequest에 전송
- 클라이언트 브라우저로 서버로부터의 응답이 도착하면 XMLHttpRequest 객체는 자바스크립트에 도착 사실을 알림
- 자바 스크립트는 응답 데이터와 DOM을 조작해서 사용자 화면에 반영

XMLHttpRequest(XHR)객체

- Ajax에서 서버와 클라이언트 간에 통신을 담당
- W3C의 표준이 아님(모든 웹 브라우저가 XHR 객체를 지원하는 것은 아님)
- 서버와 클라이언트간에 요청과 응답을 처리하기 위해서 필요한 객체
- 웹 브라우저가 IE5.0/6.0인 경우에는 ActiveX 객체를 사용해서 생성
  - var httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
- 웹 브라우저가 IE7.0, 파이어폭스, 사파리, 오페라등의 경우에는 자바스크립트의 내장 객체를 사용해서 생성
  - var httpRequest = new XMLHttpRequest();

XMLHttpRequest(XHR)객체

```
 //getXMLHttpRequest() 함수로 생성한 객체를 저장하기 위해 선언한 전역 변수
var  httpRequest = null;
function  getXMLHttpRequest(){
            if(window.ActiveXObject){  //IE에서 XMLHttpRequest 객체 구하기
                        try{
                              return new ActiveXObject("Msxml2.XMLHTTP");
                        }catch(e){
                                    try{
                                        return new ActiveXObject("Microsoft.XMLHTTP");
                                    }catch(e1){
                                        return null;
                                    }
                        }
            }else if (window.XMLHttpRequest)           {
       //IE이외의 Firefox, Opera와 같은 브라우저에서 XMLHttpRequest 객체 구하기
                        return new XMLHttpRequest();
    }else {
                        return null;
            }
}
```

XMLHttpRequest(XHR) 메소드

메소드명	설명
abort()	현재의 요청을 취소(중단)한다.
getAllResponseHeaders()	HTTP 요청에 대한 모든 응답 헤더드을 키와 값의 쌍으로 리턴한다.
getResponseHeader(headerrName)	매개변수로 주어진 HTTP 헤더의 값을 리턴한다.
open(method, url, async)	사용자의 요청을 설정하는 메소드
send(content)	요청을 서버로 보낸다.
setRequestHeader(header, value)	헤더의 값을 설정하는 것으로, 헤더명이 header인 값을 value로 설정한다.
http://en.wikipedia.org/wiki/XMLHttpRequest참고	

XMLHttpRequest(XHR) 프로퍼티

프로퍼티명	설명
onreadystatechange	상태의 변경이 발생했을 때,이벤트를 처리하기 위한 이벤트 핸들러를 기술한다.
readyState	요청 객체의 상태를 리턴한다.
responseText	문자열로 이루어진 서버의 응답을 받는다.
responseXML	XML로 이루어진 서버의 응답을 받는다.
responseBody	이진코드 문자열로 서버의 응답을 받는다.ActiveX컴포넌트를 사용해서 생성한 XMLHttpRequest 객체에서만 사용된다.
satus	서버로부터 응답받는 HTTP 상태코드로, 숫자로 리턴된다.
statusText	서버로부터 HTTP 상태를 문자열로 리턴받는다.
http://en.wikipedia.org/wiki/XMLHttpRequest참고	

웹 서버에 요청 전송하기

- open(method, url, async)
  - 요청의 초기화
  - GET/POST 선택
  - 접속할 URL입력
  - 동기/비동기 방식 지정
- send()
  - 웹 서버에 요청 전송
- 크로스 브라우저를 지원하기 위해서는 항상 비동기 방식 사용권장 (세번째 인자 true)
- XMLHttpRequest 객체가 웹 서버에 전송하는 요청 파라미터의 경우에는 반드시 UTF-8로 인코딩해서 전송

웹 서버에 요청 전송하기

- GET 방식 전달

```
httpRequest = getXMLHttpRequest();
httpRequest.open( “GET”, “/test.jsp?id=guest&pwd=1234” , true);
httpRequest.send(null);
```

```
httpRequest = getXMLHttpRequest();
httpRequest.open( “GET” , “/test.jsp”, true);
httpRequest.send(null);
```

- POST 방식 전달

```
httpRequest = getXMLHttpRequest();
httpRequest.open( “POST” , “/test.jsp”, true);
httpRequest.send( “id=guest&pwd=1234” );
```

서버 응답 처리하기

- onreadystatechange
  - 웹 서버로부터 응답이 도착할 때 호출될 함수를 지정
  - 콜백(callback)함수
  - 화면을 변경하거나 경고 창을 띄우는 등 응답 결과에 따라 알맞은 작업을 수행
  - 콜백함수는 readyState라는 프로퍼티의 값이 변경될때마다 호출

XMLHttpRequest 객체 상태

- readyState

값	의미	설 명
0	Uninitialized	객체만 생성되고 아직 초기화되지 않은 상태(open)메서드가 호출되지 않음)
1	Loading	open메서드가 호출되고 아직 send메서드가 불리지 않은 상태
2	Loaded	send 메서드가 불렸지만 status와 헤더는 도착하지 않은 상태
3	Interactive	데이터의 일부를 받은 상태
4	Completed	데이터를 전부 받은 상태, 완전한 데이터의 이용 가능

- readyState의 값이 2와 3인 경우는 웹 브라우저에 따라서 다르게 처리되므로 1과 4를 사용하는 것이 크로스 브라우저를 지원할 수 있는 가장 알맞은 방법이다.

XMLHttpRequest 객체 상태

- readyState

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

```
unction callbackFunction(){
   if ( httpRequest.readyState == 1 || httpRequest.readyState == 2 ||
httpRequest.readyState == 3 )          {
                        //화면에  서버에서 작업중이라는 메시지 출력
    } else   if ( httpRequest.readyState == 4 )     {
                        //서버에서 완전한 응답이 도착한 경우
              //응답 결과에 따라 알맞은 작업 처리
       }
  }
```

서버 응답 상태

- status/statusText

값	텍스트	설 명
200	OK	요청 성공
403	forbidden	접근 거부
404	Not Found	페이지 없음
500	Internal Server Error	서버 오류 발생

```
function callbackFunction(){
   if ( httpRequest.readyState == 4 )    {
      if( httpRequest.status == 200 ) {
        //서버에서 완전한 응답이 도착한 경우
      }else{
            alert("문제 발생 :"+httpRequest.status);
      }
   }
}
```

서버로부터의 응답 상태

- statusText
  - Status 프로퍼티의 값을 설명하는 텍스트 문장을 저장한다.
- 웹서버는 단순 텍스트 파일의 경우 응답 character set을 변경할 수 없다
- 단순 텍스트 파일의 응답 character set은 XMLHttpRequest는 기본 캐릭터셋인 UTF-8을 사용해서 데이터를 읽어온다.
- JSP는 응답 character set을 UTF-8로 지정하고, pageEncoding속성을 사용해서 소스 코드의 character set은 EUC-KR로 지정함으로써,작업은 EUC-KR로 하면서 결과는 UTF-8로 생성할 수 있게 된다.
- Opera8.5 버전까지는 statusText 프로퍼티를 제공하고 있지 않으므로, 크로스 브라우저를 실현하기 위해서는 statusText는 가급적 사용하지 않는 것이 좋다

파라미터 한글 처리 방법

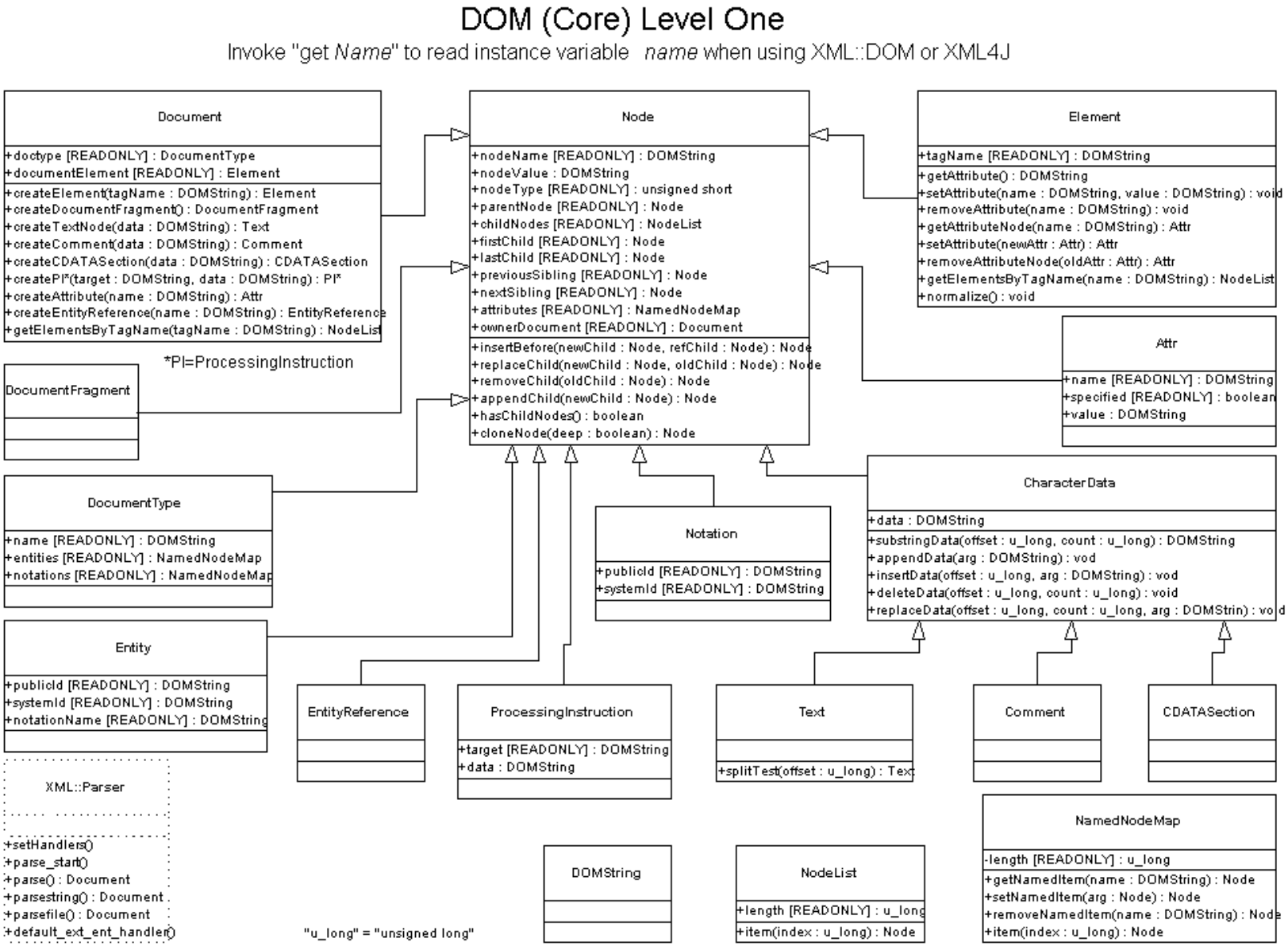
- XMLHttpRequest 객체가 웹 서버에 전송하는 요청 파라미터의 경우에는 반드시 UTF-8로 인코딩해서 전송해야 한다.
- 자바 스크립트는 문자열 UTF-8로 인코딩해 주는 함수인 codeURIComponent() 함수를제공하고 있다

```
Var params = "name="+encodeURIComponent("대한민국");
XMLHttpRequest.open("GET", "/hello.jsp"+params, true);
```

웹 서버 응답 결과 처리

- 웹 서버의 응답 텍스트의 값이 HTML인 경우
  - 응답 텍스트를 innerHTML을 사용하여 그대로 화면에 반영
- 웹 서버의 응답 텍스트가 임의의 포맷인 경우
  - 자바스크립트는 응답 텍스트를 분석해서 HTML 코드를 생성한 뒤 innerHTML을 사용해서 화면에 반영
- Document.documentElement는 DOM API로서 문서의 루트 노드를 나타낸다.
- Document.documentElement.innerHTML은 전체 HTML 문서의 코드를 나타낸다.

Logical View



DOM(Document Object Model)

- 문서를 객체로 표현하기 위한 표준
- HTML이나 XML 등의 문서를 객체로 표현할 때 사용되는 API
- 플랫폼/언어 독립적
- 구조화된 문서를 표현하는 W3C 표준 모델
- 동적으로 문서의 내용, 구조, 스타일에 접근하고 변경하는 수단

DOM 표준안

- Level 0 :
  - DOM이 만들어지기 이전의 모든 벤더 종속적인 DOM을 포함, 표준화 과정 이전에 있었던 단계에 대한 표현
- Level 1 :
  - DOM 문서에 대한 탐색과 조정에 대한 최초의 표준 명세
- Level 2 :
  - XML 네임스페이스(Namespace) 지원, 필터링된 뷰(view)와 이벤트
- Level 3 : 6가지 명세로 구성
  - Core
  - Load and Save
  - Xpath
  - Views and Formatting
  - Requirements
  - Validation

DOM API 인터페이스

- Document
  - 전체 문서를 나타낸다
- Element
  - 각 태그를 나타낸다
- Text
  - 문자열 데이터가 Text 노드로 표현된다.
- CDataSection
  - XML 문서의 CDATA 영역의 문자열 값을 저장한다.
- Node 인터페이스
  - Element, Document, Text등 주요 인터페이스가 상속받는 부모 인터페이스

Node 인터페이스의 주요 프로퍼티

프로퍼티 타입	프로퍼티 이름	의미
String	nodeName	노드의 이름
String	nodeValue	노드의 값
unsigned short	nodeType	노드타입
Node	parentNode	부모노드
NodeList	childNodes	자식노드목록
Node	firstChild	첫번째 자식 노드
Node	lastChild	마지막 자식 노드
Node	previousSibling	현재 노드와 같은 부모노드를 갖는 자식 노드 중 현재 노드 이전의 자식 노드
Node	nextSibling	현재 노드와 같은 부모노드를 갖는 자식 노드 중 현재 노드의 다음 자식 노드
Document	ownerDocument	이 노드가 포함된 Document 객체

Node 인터페이스의 주요 프로퍼티

Property/Method	Description	비고
childNodes	요소내 모든 노드의 배열을 반환	
firstChild	요소내 첫 번째 노드를 반환	
getAttribute(aAttributeName)	특정 속성 값을 반환	오페라 버그
hasAttribute(aAttributeName)	특정 속성 값이 있는지 여부를 판별.	IE지원 안함
hasChildNodes()	자식 노드가 있는지 여부를 판별	
lastChild	요소내 마지막 노드를 반환	
nextSibling	상위 노드의 다음 자식 노드	
nodeName	현재 노드의 이름을 반환	
nodeType	현재 노드의 형식을 반환한다. 예를 들어 1번은 요소 노드, 2번은 속성 노드 3번은 텍스트, 4번은 CDATA, 5번은 참조 엔티티 등.	
nodeValue	현재 노드의 값을 반환한다. 노드 값이 텍스트이면 텍스트를, 속성이면 속성값을 기타는 null을 반환한다.	
ownerDocument	현재 노드를 포함하고 있는 문서 객체를 반환	
parentNode	현재 노드의 상위 노드를 반환	
previousSibling	상위 노드의 이전 자식 노드	
removeAttribute(aName)	노드의 특정 속성을 지운다.	IE 버그 있음

NodeList

- 자식 노드 목록이 저장
- length – NodeList에 저장된 노드의 개수
- item(i) – 인덱스 i에 저장된 노드를 구함(i는 0부터 시작)

Node에 정의된 nodeType 관련 상수값

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

Document 인터페이스의 메서드

함 수	설 명
NodeList getElementsByTagName(String tagname)	지정한 이름의 태그에 해당하는 모든 Element의 목록을 구한다.
Element getElementById(String elementId)	Id 속성의 값이 elementId인 태그를 구한다.

Element 인터페이스의 속성 관련 함수

함 수	설 명
String getAttribute(String name)	Name에 해당하는 속성의 값을 구한다.
setAttribute(String name, String value)	이름이 name인 속성의 값을 value로 지정한다.
removeAttribute(String name)	이름이 name인 속성을 제거한다.

Document의 Element 생성 메서드

함 수	설 명
Element createElement(String tagName)	지정한 태그 이름을 갖는 Element를 생성한다.
Text createTextNode(String text)	Text를 값으로 갖는 Text노드를 생성한다.

Node의 DOM 트리 조작 관련 함수

함 수	설 명
Node insertBefore (Node newChild, Node refChild)	현재 노드의 자식 노드인 refChild 노드의 previousSibling 자리에 newChild 노드를 삽입한다.
Node replaceChild (Node newChild, Node oldChild)	현재 노드의 자식 노드인 oldChild 노드를 새로운 newChild 노드로 교체한다.
Node removeChild (Node oldChild)	현재 노드의 자식 노드인 oldfChild 를 현재 노드에서 제거한다.
Node appendChild (Node newChild)	newChild 노드를 현재 노드의 마지막 자식 노드로 추가한다.

객체지향 자바 스크립트

- 자바 스크립트에서 클래스의 선언은 함수를 정의하는 방식으로 한다.

```
클래스 이름 = function(파라미터) {
  ....
 }
```

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

- 클래스의 객체를 생성시에는 “new” 키워드를 사용한다.

```
var  mb = new  Member(‘korea’, ‘대한민국);
```

객체지향 자바 스크립트

- 새로운 함수를 클래스에 추가할 때에는 protoype을 사용

```
클래스 이름 .prototype.함수이름 = function(파라미터){
  ....
 }
```

```
Member.prototype.setValue  =  function(newId,  newName) {
  this.id = newId;
  this.name = newName;
}
```

```
Member.prototype.toString  =  function() {
   return this.id +"["+ this.name + "]";
}
```

객체지향 자바 스크립트

- 자바스크립트에서는 객체를 클래스없이 직접 정의하는 것이 가능하다
  - 클래스 정의 없이 무명 객체로 정의해서 프로퍼티나 메소드를 정의할 수 있다

```
var 객체명 = {
    프로퍼티명 1 : 프로퍼티값1;
    ....
    프로퍼티명 n : 프로퍼티값n;
    메소드명 1 : function(){
      내용...
    }
    ....
    메소드명 n : function(){
      내용...
    }
}
```

prototype.js

- prototype.js를 사용하면 클래스를 정의할 때 명시적인 서술이 가능하다.
- http://www.prototypejs.org
- 클래스의 생성은 prototype.js에서 제공하는 Class 객체의 create() 메소드로 선언한다.
- 클래스의 메소드 선언은 클래스 변수의 prototype속성에 “메소드명:function()”형태로 정의한다.
- 클래스의 생성자는 클래스 변수의 prototype속성에 initialize()메소드로 정의한다.

```
intialize : function(id){
    this.id = id;
}
```

- Ajax에서는 “prototype.js”라는 표준 라이브러리가 자주 사용되고 있다

prototype.js

- prototype.js에 의한 클래스의 상속은 Object 객체의 extend() 메소드를 사용한다.
  - extend() 메소드의 리턴값을 상속하는 클래스의 prototype 속성에 설정한다.
  - extend() 메소드는 첫 번째 매개변수로 상속해 주는 클래스의 객체를 지정한다.
  - extend() 메소드는 두번째 매개변수에는 상속받는 클래스의 정의를 기술한다.

    Child.prototype = Object.extend(new Parent,{..
  - 슈퍼 클래스의 메소드를 호출 – apply()

    슈퍼클래스명.prototype.슈퍼클래스의 메소드명.apply(this);

prototype.js

- 추상 클래스란 메소드의 내용을 갖지 않고, 단순히 메소드의 선언만을 가진다.
  - test:function(name){}
- 추상 클래스의 목적은 해당 클래스를 사용하는 사용자들이, 특정 메소드의 메소드명이나 메소드의 인수 등을 정형화 시켜 구조를 통일하는 것이 목적이다.
- 추상 클래스를 상속받는 클래스에서 해당 추상 클래스를 재정의한다.
- 이벤트는 Event객체의 observe() 메소드의 매개변수로 지정해서 사용한다
  - Event.observe(태그객체, 이벤트명, 이벤트핸들러메소드);

- ASP:
  - JSON for ASP.
  - JSON ASP utility class.
- ActionScript:
  - ActionScript1.
  - ActionScript2.
  - ActionScript3.
  - JSONConnector.
- C.
  - JSON_checker.
  - JSON parser.
  - json-c.
  - M's JSON parser.
  - YAJL.
- C++:
  - TinyJSON.
  - jsoncpp.
  - zoolib.
  - Jaula.
  - JOST.
  - JSON Spirit.
  - CAJUN.
- C#:
  - JSON_checker.
  - Jayrock.
  - Json.NET - LINQ to JSON.
  - JSONSharp.
  - LitJSON.
  - JSON for .NET.
- Erlang.
- Fan.
- Flex.
- Haskell:
  - RJson package.
  - json package.
  - JSON.hs.
  - HaskellNet.
- haXe.
- Java:
  - org.json.
  - org.json.me.
  - Json-lib.
  - JSON Tools.
  - json-simple.
  - Stringtree.
  - SOJO.
  - VRaptor.
  - Restlet.
  - Jettison.
  - json-taglib.
  - FLEXJSON.
  - XStream.
  - JsonMarshaller.
  - Flexjson.
  - Jackson JSON Processor.
  - JON tools.
  - google-gson.
- JavaScript:
- LotusScript:
  - JSON LS.
  - JSON Reader/Writer.
- Lua:
  - Json4Lua.
  - Json.lua.
- Objective C:
  - BSJSONAdditions.
  - Cocoa JSON Framework.
- Objective CAML.
- OpenLaszlo.
- Perl.
- PHP:
  - PHP 5.2.
  - json.
  - Services_JSON.
  - Zend_JSON.
  - JSONRPC.
  - Solar_Json.
  - SCA_SDO.
  - Comparison of php json libraries.
- Pike:
  - Public.Parser.JSON.
  - Public.Parser.JSON2
- pl/sql.
- PowerShell.
- Prolog:
  - SWI-I
  - json.pr

JSON

JavaScript Object Notation

JSON(JavaScript Object Notation)

- http://www.json.org/json-ko.html
- 경량의 DATA-교환 형식
- 사람이 읽고 쓰기에 용이
- 기계가 분석하고 생성함에도 용이
- JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999의 일부에 토대를 두고 있다.
- 인터넷에서 자료를 주고받을 때 그 자료를 표현하는 방법
- 자료의 종류에 큰 제한은 없다
- 특히 컴퓨터 프로그램의 변수값을 표현하는 데 적합
- 형식은 자바스크립트의 구문 형식을 따르지만, 프로그래밍 언어나 플랫폼에 독립적이므로 C, C++, 자바, 자바스크립트, 펄, 파이썬 등 많은 언어에서 이용할 수 있다.

JSON(JavaScript Object Notation)

- JSON의 DATA 구조
  1. name/value 형태의 쌍으로 collection 타입
     다양한 언어들에서, 이는 object, record, struct(구조체), dictionary, hash table, 키가 있는 list, 또는 연상배열로서 구현
  2. 값들의 순서화된 리스트
     대부분의 언어들에서, 이는 array, vector, list, 또는 sequence로서 구현

JSON(JavaScript Object Notation)

- object는 name/value 쌍들의 비순서화된 SET이다.
- object는 { (좌 중괄호)로 시작하고 } (우 중괄호)로 끝내어 표현한다.
- 각 name 뒤에 : (colon)을 붙이고 , (comma)로 name/value 쌍들 간을 구분한다.

  {"name2": 50, "name3": "값3", "name1": true}

- array은 값들의 순서화된 collection 이다.
- array는 [(left bracket)로 시작해서] (right bracket)로 끝내어 표현한다.
- , (comma)로 array의 값들을 구분한다.

  [10, {"v": 20}, [30, "마흔"]]

JSON(JavaScript Object Notation)

- JSON 표기법으로 배열에 접근할 때에는 '객체[인덱스]'형식으로 접근할 수 있으며, 인덱스 값은 0부터 시작한다.

```
var countries = {ko, fr, uk, 'us'}
var koName = countries.ko;
var frName = countries[0];
```

- value는 큰따옴표안에 string, number ,true ,false , null, object ,array이 올수 있다. 이러한 구조들을 포함한다.

- string은 큰따옴표안에 둘러 싸인 zero 이상 Unicode 문자들의 조합이며, 쌍다옴표안에 감싸지며,backslash escape가 적용된다. 하나의 문자(character)도 하나의 문자열(character string)로서 표현된다.

- number는 8진수와 16진수 형식을 사용하지 않는 것을 제외하면 C와 Java number 처럼 매우 많이 비슷하다.