CH 11

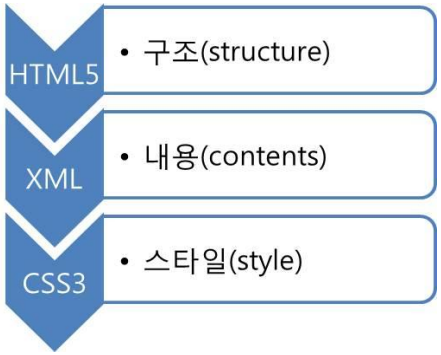
XML, Ajax와 제이쿼리 활용

>> 학습목표 <<

❖XML의 특성과 문서 작성 방법을 알아본다.
❖XML의 네임스페이스 개념을 이해한다.
❖XML 문서에 CSS 적용 방법을 알아본다.
❖Ajax 메소드의 종류를 알아보고 예제에 활용해본다.
❖XML과 제이쿼리를 활용한 예제를 만들어본다.

1.1 XML

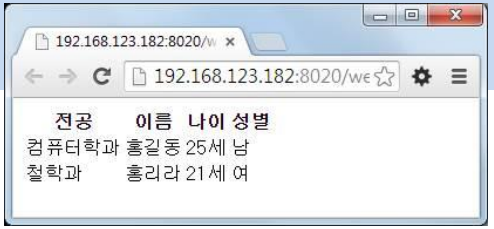
- 웹
  - 단순 정보 제공의 수단 -> 정보 서비스 토대 (중요성 증가)
  - 웹 정보 교환이 빈번, 다양해짐에 따라 웹 정보의 구조와 의미를 보다 명확히 처리할 수 있는 기술이 필요해짐
- XML
  - 정보 표현이 자유롭고 또, 다양한 환경에서도 인식이 가능하기 때문에 웹 정보 교환의 핵심 기술로 인식
  - 웹 문서의 내용 (특히 문서의 의미)을 표현하는데 적합한 마크업 언어
  - SGML로부터는 문서 내용과 구조를 분리한 기능성과 확장성을, HTML로부터는 사용하기 쉽고 편리한 장점을 계승
  - B2B, B2C와 같은 전자상거래의 표준 문서 형식으로 사용
  - 다양한 웹 애플리케이션 간의 데이터 교환 표준으로 폭넓게 사용
- 웹 문서의 구성 요소



1.2 HTML5과 XML

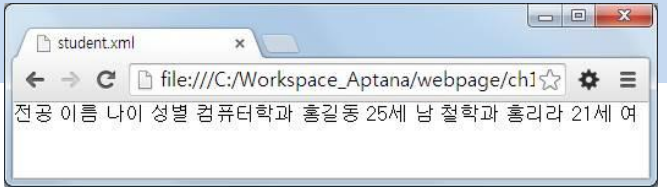
[예제11-1-1] 학생 정보 html5 문서 예 **/ch11/student.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<link rel="stylesheet" href="student1.css"/>
</head>
<body>
        <table>
            <tr>
                <th>전공</th>
                <th>이름</th>
                <th>나이</th>
                <th>성별</th>
            </tr>
            <tr>
                <td>컴퓨터학과</td>
                <td>홍길동</td>
                <td>25세</td>
                <td>남<td>
            </tr>
            <tr>
                <td>철학과</td>
                <td>홍리라</td>
                <td>21세</td>
                <td>여<td>
            </tr>
        </table>
</body>
</html>
```



[예제11-1-2] 학생 정보 xml 문서 예 **/ch11/student.xml**

```
?xml version="1.0" encoding="utf-8" ?>
<?xml-stylesheet type="text/css" href="student2.css"?>
<studentinfo>
        <student_heading>
            <major>전공</major>

            <name>이름</name>
            <age>나이</age>
            <gender>성별</gender>
        </student_heading>
        <student>
            <major>컴퓨터학과</major>
            <name>홍길동</name>
            <age>25세</age>
            <gender>남</gender>
        </student>
        <student>
            <major>철학과</major>
            <name>홍리라</name>
            <age>21세</age>
            <gender>여</gender>
        </student>
</studentinfo>
```



HTML5과 XML 문서의 스타일 적용

- 스타일 적용 결과
  - 두 유형의 문서와 관련된 CSS3 파일 안에 다음 스타일을 적용

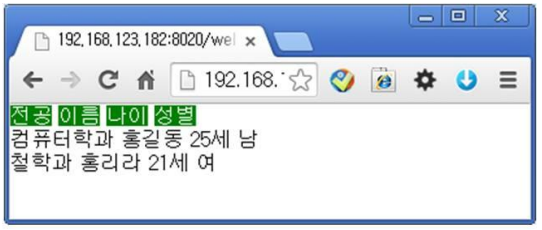
[예제11-1-3] 학생 정보 html5 스타일 예 **/ch11/student1.css**

```
th { background-color: silver; }
```

[예제11-1-4] 학생 정보 xml 스타일 예 **/ch11/student2.css**

```
student_heading * { color: white; background-color: green; }
student { display: block; }
```

student.html 적용 결과



student.xml 적용 결과

  - 인지 능력이 있는 사람 : HTML5, 기계적인 처리를 하는 프로그램 : XML
  - '콘텐츠로서의 웹 문서' : HTML5, '데이터로서의 웹 문서' : XML

1.3 XML의 특성

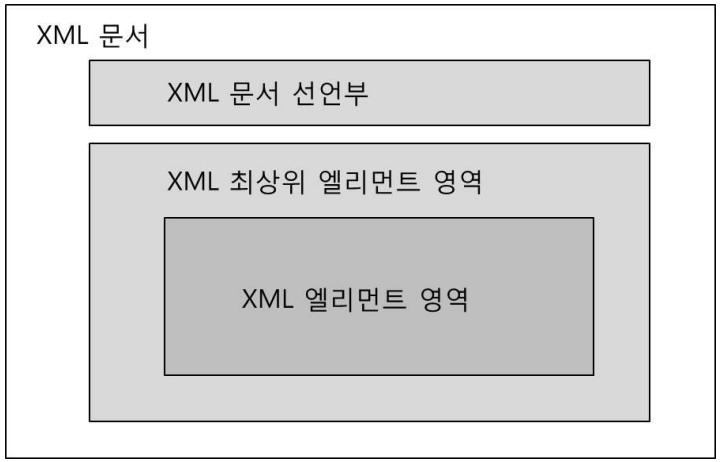
- XML의 장점
  - 범용성 : 어떤 환경에서도 데이터를 인식하고 사용할 수 있는 미니 데이터베이스 기능
  - 유연성 : 자유로운 정보의 결합과 분리 기능
  - 확장성 : 다양한 응용 분야에 적합한 언어를 개발할 수 있는 메타 언어 기능
  - 역동성 : 입력 또는 수정을 통해 내용이 동적으로 변화함

- XML 관련 기술

분류	기술	기능
구조 정의	DTD	XML 문서의 구조 명세(비XML 형식)
	XML 스키마	XML 문서의 구조 명세(XML 형식)
스타일 정의	CSS	브라우저 출력 형식을 명세(비XML 형식)
	XSL	브라우저 출력 형식을 명세(XML 형식, XSLT이용)
문서 변환	XSLT	문서 변환 형식을 명세(XML 형식)
위치 지정	XPATH	XML 문서 내 위치 지정 방식
	XPointer	XML 문서 내 검색 대상을 명세(XPATH 사용)
내용 변경	DOM	DOM 트리 기반의 노드 조작 API
	SAX	XML 문서 조작 API

2.1 XML 문서 구조

- XML 문서
  - XML 선언부를 제외하고는 기존 HTML5의 기본 구조와 사용 방법이 거의 유사
  - XML 문서 선언부
    - 반드시 맨 앞에 명세, XML 문서 유형을 지정
    - XML 문서 구조를 정의한 DTD(또는 XML Schema) 선언, 스타일을 정의한 CSS 연결에 대한 선언도 명세



  - 하나의 최상위 엘리먼트의 <시작태그>로 시작해서 </종료태그>로 끝남
  - 최상위 엘리먼트를 포함하여 XML 문서의 모든 태그들은 자유롭게 정의
  - 엘리먼트의 시작 태그 안의 속성도 자유롭게 정의

XML 문서 작성 예

- XML 문서도 HTML5처럼 모든 종류의 텍스트 편집기를 사용
- 작성한 파일의 확장자는 '.xml'로 지정

[예제11-2] 기본 xml 문서 작성하기1 **/ch11/product.xml**

```
<?xml version="1.0" encoding="utf-8"?>                    <!-- XML 문서 선언 -->
<!DOCTYPE productinfo SYSTEM "product.dtd">               <!-- XML 구조 선언 -->
<?xml-stylesheet type="text/css" href="product.css"?>     <!-- 스타일 문서 선언 -->
<productinfo>                                             <!-- 최상위 엘리먼트 시작 -->
    <product id="001">                                    <!-- 속성 -->
        <name>갤럭시S5</name>
        <company>삼성</company>
        <price>980,000</price>
    </product>
    <product id="002">                                    <!-- 상위 엘리먼트-->
        <name>G3</name>                                   <!-- 하위 엘리먼트 -->
        <company>LG</company>
        <price>880,000</price>
    </product>
</productinfo>                                            <!-- 최상위 엘리먼트 종료 -->
```

2.2 XML 코딩 규칙

- HTML5와 비슷한 코딩 규칙
  - XML 문서는 반드시 최상위 엘리먼트를 하나만 갖는다. 이를 문서 엘리먼트(document element) 또는 루트 엘리먼트(root element)라고 한다.
  - 하나의 엘리먼트는 시작 태그로 시작하며 종료 태그가 마지막에 위치한다. 태그는 엘리먼트의 내용을 감싸는 역할을 한다.
  - 시작 태그와 종료 태그는 반드시 쌍으로 존재해야 한다.
  - 태그는 열린 순서대로 차례로 닫혀야 한다.
  - 대소문자는 구별한다.
  - 엘리먼트(태그) 안에 또 다른 엘리먼트(태그)가 포함될 수 있다. 상위 엘리먼트와 하위 엘리먼트 관계가 문서 내에 존재한다.
  - 태그들은 중첩되지 않아야 한다. 모든 하위 태그들이 모두 열렸다 닫혀야만 상위 태그는 닫힐 수 있다.
  - 문서 안에 설명을 추가할 경우에는 '<!--'와 '-->'를 사용하여 주석 처리한다. 주석은 XML 선언 이후 어디든지 명세할 수 있다. 다만, 주석 안에는 불가능하고 '--' 문자열을 포함할 수 없다.
  - 속성은 시작 태그 안에 위치하며 속성이름과 속성값을 갖는다.

2.3 XML 기본 요소(1)

- XML 문서 선언부
  - XML 문서임을 선언하는 부분으로 반드시 입력
  - 단일 태그로 공백 없이 '<?xml'로 시작하여 '?>'로 끝남
  - 버전(필수 요소, 생략불가, 제일 먼저 지정)과 인코딩 형식을 속성으로 지정

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

- XML 스타일 선언
  - xml-stylesheet 태그를 통해 스타일시트를 적용하도록 지시
  - 지정한 CSS 형식의 파일을 연결해서 명세된 스타일을 처리

```
<?xml-stylesheet type="text/css" href="student2.css"?>
```

XML 기본 요소(2)

- XML 엘리먼트 (element)
  - HTML5 문서처럼 XML 문서도 다양한 엘리먼트들의 집합
  - XML 문서 안에서 데이터를 표현하는 기본 단위
  - XML 문서에서 명세 가능한 3가지 엘리먼트 유형
- 기본 엘리먼트 유형 : 태그 내용으로 문자열 데이터만을 갖음

```
<address>성남시 분당구</address>                    <!-- 기본 엘리먼트 -->
```

- 복합 엘리먼트 유형 : 태그 내용으로 하위 태그들을 포함

```
<address>                                          <!-- 복합 엘리먼트 -->
      <city>성남시</city>
      <gu>분당구<gu>
</address>
```

- 혼합 엘리먼트 유형 : 태그 내용으로 문자열 데이터와 하위 태그들을 포함

```
<address>경기도                                    <!-- 혼합 엘리먼트 -->
      <city>성남시</city>
      <gu>분당구<gu>
</address>
```

- 빈 엘리먼트 유형 : 내용이 없음

XML 기본 요소(3)

- XML 속성(attribute)
  - 이름 자유롭게 직접 정의 가능, XML 명명 규칙을 따름
  - 속성값은 큰 따옴표(" ")와 작은 따옴표(' ') 모두 가능하지만 가급적 큰 따옴표를 사용하고 반드시 속성값(공백 문자 포함)을 가져야 한다.
  - 엘리먼트의 시작 태그 안에 같은 이름의 속성을 여러 번 명세 불가

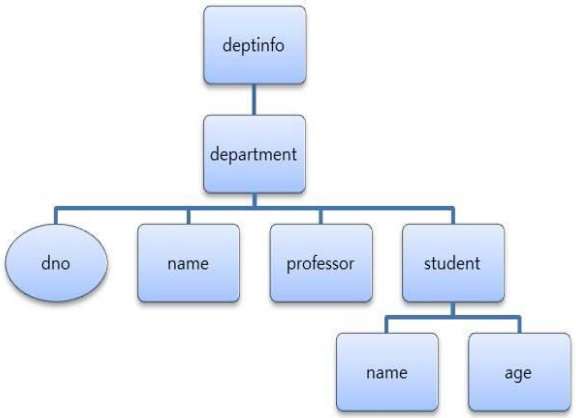
```
<address type="집주소">성남시 분당구 서판교로 123</address>

<address type="직장주소" zip="123-456">서울시 충정로 1가 25</address>
```

- XML 명명 규칙(naming rule)
  - 첫 글자는 문자나 '_' 기호로 시작한다.
  - 첫 글자와 중간에는 공백 문자를 사용할 수 없다.
  - 소문자 'xml'로 시작할 수 없다.
  - 예약어는 사용할 수 없다.

  - 엘리먼트의 태그 이름은 내용의 의미를 잘 표현할 수 있게, 너무 단축된 이름 은 피함
  - HTML5와는 다르게 대소문자를 구분(가급적 소문자, 영문명으로 작성)

2.4 XML 문서 작성

- [그림 11-4] department.xml 문서의 DOM 트리 구조(예제11-3)



[예제11-3] 기본 xml 문서 작성하기2 **/ch11/department.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<deptinfo>
	<department dno="001">
		<name>컴퓨터학과</name>
		<professor>박인우</professor>
		<student sno="001">
			<name>홍길동</name>
			<age>25세</age>
		</student>
	</department>
	<department dno="002">
		<name>철학과</name>
		<professor>박현우</professor>
		<student sno="021">
			<name>홍리라</name>
			<age>21세</age>
		</student>
		<student sno="022">
			<name>김미선</name>
			<age>22세</age>
		</student>
	</department>
</deptinfo>
```

XML 문서 작성

- 속성 & 엘리먼트 중심의 XML 문서 작성

[예제11-4] 속성 중심 xml 문서 작성하기 **/ch11/attribute-dept.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<deptinfo>
	<department dno="001" name="컴퓨터학과" professor="박인우">
		<student sno="001" name="홍길동" age="25세"></student>
	</department>
	<department dno="002"  name="철학과" professor="박현우">
		<student sno="021" name="홍리라" age="21세"></student>
		<student sno="022" name="김미선" age="22세">  </student>
	</department>
</deptinfo>
```

[예제11-5] 엘리먼트 중심 xml 문서 작성하기 **/ch11/element-dept.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<deptinfo>
	<department>
		<dno>001</dno>
		<name>컴퓨터학과</name>
		<professor>박인우</professor>
		<student>
			<sno>001</sno>
			<name>홍길동</name>
			<age>25세</age>
		</student>
	</department>
	<department>
		<dno>002</dno>
		<name>철학과</name>
		<professor>박현우</professor>
		<student>
			<sno>021</sno>
			<name>홍리라</name>
			<age>21세</age>
		</student>
		<student>
			<sno>022</sno>
			<name>김미선</name>
			<age>22세</age>
		</student>
	</department>
</deptinfo>
```

3.1 네임스페이스

- 이름 충돌(name conflict)
  - 서로 다른 의미를 갖는 태그의 이름 중복성 때문에 발생하는 현상
    - '<name>' 태그가 사람 이름, 건물 이름, 학과 이름 등으로 사용되는 경우
  - XML 문서 안에서 똑같은 이름의 엘리먼트 태그가 다른 의미로 사용될 수 있음
  - 서로 XML 문서를 교환하거나 합병할 경우, 다른 의미의 중복된 태그와 속성 존재 가능
  - 충돌을 해결하기 위한 개념 -> 네임스페이스

- 네임스페이스(namespace)
  - '이름 공간'을 의미하는 추상적인 개념
    - 네임스페이스가 다르면 명명 방식이 다르고 사람(또는 그룹) 수만큼 각자의 독립된 네임스페이스를 갖는으며 같은 네임스페이스를 갖는 XML 문서간에는 서로 이름 충돌이 발생하지 않는다고 가정
  - 이름 앞에 네임스페이스를 식별할 수 있는 코드를 포함하면 모든 이름 충돌을 해결

3.2 네임스페이스 선언 방법(1)

- 기본 네임스페이스 선언
  - 네임스페이스는 XML 문서 안에 'xmlns' 속성(속성값은 유일한 네임스페이스 이름으로 URI(Unique Resource Identifier) 값 지정)값을 이용하여 선언
  - 기본 네임스페이스는 보통 xmlns 속성은 최상위 엘리먼트의 시작 태그 안에 선언
  - 적용 받는 태그 이름 앞에 접두어 생략 가능

```
<studentinfo xmlns="http://www.good.ac.kr/2014/xmlns/student">
```

  - xmlns 속성은 모든 엘리먼트의 시작 태그 안에 선언 가능
  - 선언된 네임스페이스는 하위 엘리먼트들에게 상속되어 적용

[예제11-6] 기본 네임스페이스 선언하기 **/ch11/namespace1.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<studentinfo xmlns="http://www.good.ac.kr/2014/xmlns/student">

    <student sno="001">
        <major>컴퓨터학과</major>
        <name>홍길동</name>
        <age>25세</age>
        <gender>남</gender>
    </student>
</studentinfo>
```

네임스페이스 선언 방법(2)

- 접두어를 사용한 네임스페이스 선언
  - XML 문서에 네임스페이스를 여러 개 사용하거나 기본 네임스페이스가 아닌 네임스페이스를 선언할 때 접두어를 함께 선언
  - 접두어는 문자나 '_' 문자로 시작하는 문자열이면 가능

```
<st:student xmlns:st="http://www.good.ac.kr/2014/xmlns/student">
```

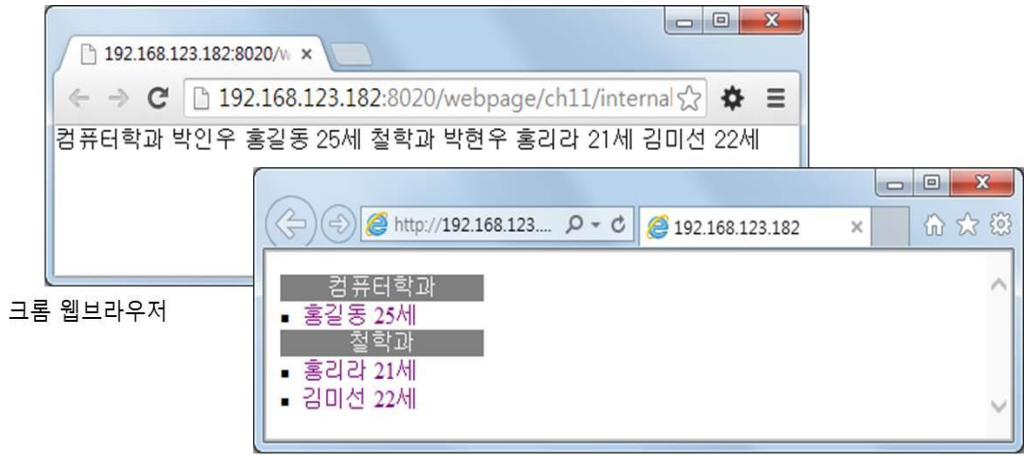
  - 기본 네임 스페이스가 아닌 다른 특정 네임스페이스에 속함을 정의하려면 태그 이름이나 속성 이름 앞에 접두어와 ':' 기호를 붙임

[예제11-7] 네임스페이스 접두어 적용하기 **/ch11/namespace2.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<deptinfo xmlns="http://www.good.ac.kr/2014/xmlns/department ">
    <department>
        <dno>001</dno>
        <name>컴퓨터학과</name>
        <professor>박인우</professor>
        <st:student xmlns:st="http://www.good.ac.kr/2014/xmlns/student">
            <st:sno>001</st:sno>
            <st:name>홍길동</st:name>
            <st:age>25세</st:age>
        </st:student>
    </department>
</deptinfo>
```

4. XML 문서의 CSS 적용(1)

- 내부 CSS3 스타일시트 적용
  - XML 문서 내부에 스타일시트를 선언한 예
  - [그림 11-5] internal-css.xml의 실행 결과(예제11-8)

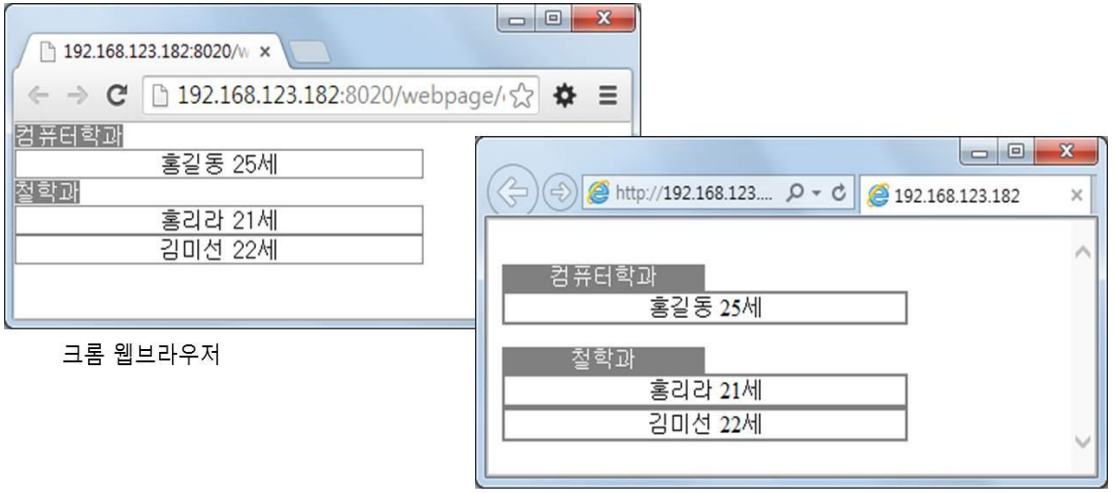


크롬 웹브라우저

IE 웹브라우저

XML 문서의 CSS 적용(2)

- 외부 CSS3 스타일시트 적용
  - 외부에 CSS3 스타일 파일을 정의하여 여러 XML 문서에 같은 스타일을 적용 가능
  - XML 문서 외부에 스타일시트를 선언한 예
  - [그림 11-6] external-css.xml의 실행 결과(예제11-9-1,2)



크롬 웹브라우저

IE 웹브라우저

5.1 제이쿼리 Ajax

- Ajax(Asynchronous JavaScript and XML)
  - 서버와 비동기 HTTP 통신을 하기 위한 기술
  - 서버에 HTTP 요청을 보낸 뒤 XML, JSON 형식 등으로 응답을 받아 페이지의 일부만을 변경
  - 매번 HTML5 페이지 전체를 새로 고침 하지 않고도 웹 페이지 내용을 새롭게 갱신 가능

  -> 페이지 이동 없이 웹 서버와 데이터를 주고 받는 사용자 상호 작용의 새 패러다임 제공
  - 빠르고 동적인 웹 페이지 생성을 위한 핵심 기술로 브라우저나 플랫폼에 독립적임

- 제이쿼리는 웹 브라우저 종류에 상관없이 같은 방식으로 Ajax 기능을 구현하도록 다양한 메소드를 제공
- 제이쿼리 모바일은 페이지 이동을 위해 Ajax 기술을 사용

주요 제이쿼리 Ajax 관련 메소드

Ajax 메소드	기능/예

$.ajax( )	모든 Ajax 메소드의 기본이 되는 메소드
	예) $.ajax({
	url: 'service.php',
	success: function(data) {
	$('#area').html(data);
	}
	});

$.get( )	GET 방식의 ajax( ) 메소드 예) $.get('sample.html', function(data) { $('#area').html(data); });
$.post( )	POST 방식의 ajax( ) 메소드 예) $.post('sample.html', function(data) { $('#area').html(data); });
$.getJSON( )	JSON 형식으로 응답 받는 ajax( ) 메소드 예) $.getJSON('sample.json', function(data) { $('#area').html(' ' + data.age + ' '); });
load( )	서버로부터 데이터를 받아서 일치하는 요소 안에 HTML을 추가 예) $('#area').load('sample.html', function( ) { ; });
$.getScript( )	자바스크립트 형식으로 응답 받는 ajax( ) 메소드 예) $.getScript('sample.js', function( ) { ; });
$.ajaxSetup( )	ajax( ) 메소드의 선택 사항들에 대한 기본값 설정 예) $.ajaxSetup({ url: 'service.php' });

제이쿼리 Ajax 이벤트 메소드

Ajax 메소드	기능
ajaxStart( )	첫 번째 Ajax 요청이 시작될 때 호출되는 이벤트 메소드 예) $('#img1').ajaxStart(function( ){ $(this).show(); });
ajaxStop( )	모든 Ajax 요청이 끝날 때 호출되는 이벤트 메소드 예) $('#img1').ajaxStop(function( ){ $(this).fadeOut(2000); });
ajaxSend( )	특정 Ajax 요청을 보내기 전에 호출되는 이벤트 메소드 예) $("#msg").ajaxSend(function(event, request, settings){ $(this).append(" " + settings.url + "페이지 요청 시작 "); });
ajaxSuccess( )	특정 Ajax 요청이 성공적으로 완료될 때마다 호출되는 이벤트 메소드 예) $("#msg").ajaxSuccess(function(event, request, settings){ $(this).append(" 요청 성공 "); });
ajaxError( )	Ajax 요청들에 대한 오류 발생시 호출되는 이벤트 메소드 예) $("#msg").ajaxError(function(event, request, settings){ $(this).append(" " + settings.url + "페이지 요청 실패 "); });
ajaxComplete( )	Ajax 요청들이 완료되면(성공/실패 관련 없이) 호출되는 이벤트 메소드 예) $("#msg").ajaxComplete(function(event,request, settings){ $(this).append(" 요청 완료 "); });

5.2 $.ajax( ) 메소드의 XML 문서 적용 예

- $.ajax( ) 메소드
  - 모든 Ajax 메소드가 내부적으로는 사용하는 기본 메소드
  - Ajax 요청을 기본적인 부분부터 직접 설정하고 제어할 수 있어 다른 Ajax 메소드로 할 수 없는 요청도 수행 가능
  - $.ajax() 메소드의 기본 형식

```
$.ajax( options ) ;
```

```
$.ajax({ url: URL주소 [,type: 요청방식] [,data: 요청내용] [,timeout: 응답제한시간] [,dataType: 응답데이터유형] [,async: 비동기여부] [,success: 성공콜백함수] [,error: 실패콜백함수] });
```

- ajax( ) 메소드 선택 항목들(options)을 맵 형식으로 명세

선택항목 : 항목값	의미
url : URL 주소	요청이 보내질(주로 서버)의 URL 주소(필수 항목, 기본값: 현재페이지) 예) "sample.php", "sample.html", "sample.xml"
type : 요청방식	요청을 위해 사용할 HTTP 메소드 예) "get"(기본값), "post"
data : 요청내용	서버로 전달되는 요청 내용(제이쿼리 객체맵이나 문자열)
timeout : 응답제한시간	요청 응답 제한 시간(밀리초) 예) 20000
dataType : 응답데이터유형	(서버로부터의) 반환될 응답 데이터의 형식 예) "xml", "html", "json", "jsonp", "script", "text"
Async : 논리값	요청이 비동기식으로 처리되는지 여부(기본값: true)
success : function(data)	요청 성공 콜백함수(data: 서버 반환 값)
error : function( )	요청 실패 콜백함수

Ajax 데이터 로드 화면 작성하기

[예제11-10-1] Ajax 데이터 로드 화면 작성하기 /ch11/ajax-data.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<meta name="viewport" content="width=device-width, initial-scale=1"/>
<title>jQuery</title>
	<!-- 제이쿼리 모바일, 제이쿼리 라이브러리 파일 선언 -->
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css"/
>
<script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
<script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js"></script>
	<!-- 사용자 정의 자바스크립트 파일 선언 -->
<script src="ajax-xml.js"></script>
<script src="ajax-json.js"></script>
<script src="ajax-html.js"></script>
</head>
<body>
	<div data-role="page">
		<div data-role="header" data-position="fixed">
			<h1>Ajax 활용</h1>
		</div>
		<div data-role="content">
			<button id="btnLoad1" data-inline="true">XML</button>
			<button id="btnLoad2" data-inline="true">JSON</button>
			<button id="btnLoad3" data-inline="true">HTML</button>

			<ul data-role="listview" id="listArea" data-inset="true">
				<li id="item">데이터 로드 하기 전 . . .</li>
			</ul>
		</div>
		<div data-role="footer" data-position="fixed">
			<h4>꼬리말</h4>
		</div>
	</div>
</body>
</html>
```

[예제11-10-2] Ajax 메소드 적용 /ch11/ajax-stuinfo.xml
xml 문서

```
<?xml version="1.0" encoding="utf-8"?>
<stuinfo>
    <student>
        <gradepoint>1.0 학점</gradepoint>
    </student>
    <student>
        <gradepoint>2.0 학점</gradepoint>
    </student>
    <student>
        <gradepoint>3.0 학점</gradepoint>
    </student>
    <student>
        <gradepoint>4.0 학점</gradepoint>
    </student>
</stuinfo>
```

Ajax 데이터 로드 화면 작성하기

- 리스트 항목의 동적 추가
  - 이미 생성된 리스트뷰에 새로운 항목을 동적으로 추가할 경우, 갱신 내용을 제이쿼리 모바일에 알리기 위해 리스트의 새로 고침을 명시적으로 호출해야 함

```
$('ul').empty();              // 리스트뷰를 비움
$('ul').append(tagList);      // 리스트뷰에 tagList에 저장된 부분 리스트를 추가
$('ul').listview('refresh');  // 리스트뷰를 새로 고침
```

  - [그림 11-7] ajax-data.html의 실행 결과(예제11-10-1~3)

5.3 $.getJSON( ) 메소드의 JSON 형식 데이터 적용 예

- $.getJSON( ) 메소드
  - GET 요청 방식으로 서버로부터 JSON 형식의 데이터를 요청
  - HTTP 요청을 이용하여 원격지의 페이지를 읽어오고 XMLHttpRequest 객체를 생성하여 반환

```
$.getJSON( url [, data] [,function(data)] ) ;
```

  - $.getJSON 메소드 입력인자

인자	의미
url	요청이 보내질(주로 서버)의 URL 주소(필수 항목, 기본값: 현재 페이지) 예) "sample.json"
data	서버로 전달되는 요청 내용(제이쿼리 객체 맵이나 문자열)
function(data)	요청 성공 콜백 함수 (data: 서버 반환 값)

JSON

- JSON(JavaScript Object Notation)
  - 데이터 교환을 위한 형식으로 '미니 XML'이라 불리움
  - 사람이 직관적으로 이해하기 쉽고 파싱하고 생성하기도 쉬움
  - 대부분의 언어에서 JSON을 사용할 수 있고 XML보다 가볍고 빨라 효율적
- JSON 명세 방법
  - 자바스크립트에서 객체를 표현하는 방법과 비슷
  - JSON 형식과 XML 형식 비교

JSON 형식	XML 형식
{ "students" : { "student" : [{"name":"홍길동", "gender":"남"}, {"name":"홍길순", "gender":"여"},] } }	<students> <student> <name>홍길동</name> <gender>남</gender> </student> <student> <name>홍길순</name> <gender>여</gender> </student> </students>

json 문서 로드 getJSON() 메소드 적용하기

- [그림 11-8] ajax-data.html의 실행 결과(예제11-10-1, 예제11-11-1~2)

[예제11-11-1] Ajax 메소드 적용 json 문서 /ch11/ajax-stuinfo.json

```
{
    "stuinfo" : [
        {"schoolyear":"1학년"},
        {"schoolyear":"2학년"},
        {"schoolyear":"3학년"},
            {"schoolyear":"4학년"}
    ]
}
```

[예제11-11-2] json 문서 로드 getJSON() 메소드 적용하기 /ch11/ajax-json.js

```
$(document).ready( function() {
    $('#btnLoad2').click( function() {
        $.getJSON('ajax-stuinfo.json', function(jsonData) {
            var tagList = "";
            $.each(jsonData.stuinfo, function() {
                tagList += "<li>" + this.schoolyear + "</li>";
            });
            $('#listArea').empty();
            $('#listArea').append(tagList);
            $('#listArea').listview('refresh');
        });
    });
});
```

5.4 $.load( ) 메소드의 HTML 문서 적용 예

- $.load( ) 메소드
  - 서버로부터 데이터를 받아오는 가장 간단한 메소드로 많이 이용
  - 서버로부터 데이터를 받아 메소드를 실행하는 대상 엘리먼트에 직접 추가 -> 복잡한 선택 사항을 설정하지 않고도 빠르고 간단하게 웹 페이지의 동적 갱신이 가능
  - 요청이 성공하면 메소드가 실행되는 대상 엘리먼트 내용이 서버에서 응답 받은 HTML5 마크업 데이터로 대체

```
$.load( url [, data] [,function(data)] ) ;
```

  - $.load( ) 메소드 선택 항목

인자	의미
url	요청이 보내질(주로 서버)의 URL 주소(필수 항목, 기본값: 현재페이지) 예) "sample.html"
data	서버로 전달되는 요청 내용(제이쿼리 객체맵이나 문자열)
function(data)	요청 성공 콜백 함수(data: 서버 반환 값)

html load( ) 메소드 적용하기

- [그림 11-9] ajax-data.html의 실행 결과(예제11-10-1, 예제11-12-1~2)

[예제11-12-1] Ajax 메소드 적용 html 문서 **/ch11/ajax-listinfo.html**

```
<!DOCTYPE html>
<html>
<body>
	<ul>
		<li>봄</li>
		<li>여름</li>
		<li>가을</li>
		<li>겨울</li>
	</ul>
</body>
</html>
```

[예제11-12-2] html load() 메소드 적용하기 **/ch11/ajax-html.js**

```
$(document).ready( function() {
	$('#btnLoad3').click( function() {
		$('#listArea').empty();
		$('#listArea').load('ajax-listinfo.html li', function(htmlData){
			$('#listArea').listview('refresh');
		});
	});
});
```

6.1 관광지 안내 앱 : tourApp

- [그림 11-10] tourApp 앱 정보 저장 XML 문서

tour1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<attractionlist  xmlns="http://www.hk.ac.kr/
2014/xmlns/tour">
  <attraction country="미국">
    <name>자유의여신상</name>
    <location>뉴욕</location>
    <picture>pic1.png</picture>
  </attraction>
</attractionlist>
```

tour2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<attractionlist  xmlns="http://www.hk.ac.kr/
2014/xmlns/tour">
  <attraction country="미국">
    <name>자유의여신상</name>
    <location>뉴욕</location>
    <picture>pic1.png</picture>
  </attraction>
  <attraction country="이탈리아">
    <name>콜롯세움</name>
    <location>로마</location>
    <picture>pic2.png</picture>
  </attraction>
  <attraction country="영국">
    <name>런던아이</name>
    <location>런던</location>
    <picture>pic3.png</picture>
  </attraction>
  <attraction country="프랑스">
    <name>에펠탑</name>
    <location>파리</location>
    <picture>pic4.png</picture>
  </attraction>
</attractionlist>
```

tour1.xml 정보를 로드한 경우

tour2.xml 정보를 로드한 경우

- [그림 11-11] tourapp-ajax-xml.html의 실행 결과(실습11-1)