

XML

마크업 언어와 XML

마크업이란? 어떤 내용에 뭔가 특별히 지시하는 문장을 추가할 수 있는데 이러한 추가적인 정보를 **마크업**이라고 합니다.

명함관리 HTML 문서

```
<body>
```

```
<fontsize=3> 명함관리
```

마크업

```
<p> 홍길동 </p>
```

내용

```
<p> 011-123-3451 </p>
```

```
<p> hong@hotmail.com </p>
```

```
</font>
```

```
</body>
```

```
</html>
```

HTML의 문제점과 XML이 등장하게 된 이유]

ch01-01.html	ch01-02.html
<pre><html> <body> <fontsize=3> 제품정보 <p> 배 </p> <p> 나주시</p> <p> 1 </p> <p> 56000 </p> </body> </html></pre>	<pre><html> <body> <fontsize=3> 여행정보 <p> 배 </p> <p> 금강호</p> <p> 16:30</p> <p> 900 </p> </body> </html></pre>

마크업 언어와 XML

HTML의 문제점과 XML이 등장하게 된 이유]

ch01-01.xml

```
<?xml version="1.0" encoding="euc-kr" ?>
<제품정보>
  <과일> 배 </과일>
  <생산지역> 나주시 </생산지역>
  <수량> 1 </수량>
  <가격> 56000 </가격>
</제품정보>
```

ch01-02.xml

```
<?xml version="1.0" encoding="euc-kr" ?>
<여행정보>
  <교통수단> 배 </교통수단>
  <교통편> 금강 호 </교통편>
  <출발시간> 16:30 </출발시간>
  <인원> 900 </인원>
</여행정보>
```

XML의 정의와 특징

XML이란?

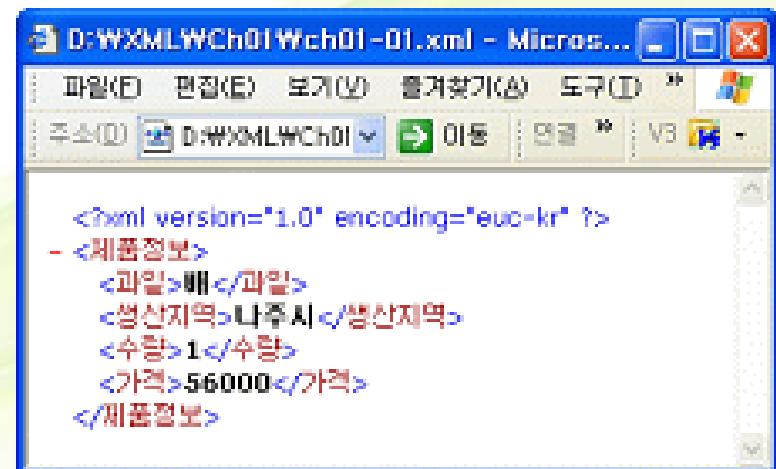
XML = eXtensible(확장 가능한) + Markup Language(마크업 언어)

마크업 언어란?

마크업 언어는 문서의 내용을 조직화하고 구조화시켜 문서의 내용을 정확하게 전달합니다. 마크업 언어의 이러한 특징은 마크업을 사용하여 문서의 내용을 구조적으로 구분함으로써 내용을 쉽게 이해할 수 있기 때문입니다.

XML은 새롭게 만들어 사용할 수 있는 사용자 정의 마크업을 만들기 위한 표준입니다.

XML 문서에서는 태그가 데이터의 구조를 반영하는 용도로 사용됩니다.



XML의 정의와 특징

XML 문서의 특징과 장점

- (1) 태그의 확장성 때문에 새로운 태그를 만들 수 있습니다.
- (2) 문서의 기본 내용이 <이름>과 같은 태그로 구분되므로 검색과 저장이 용이합니다.
- (3) 문서의 구문 오류를 검색해 주는 파서(parser)기능이 지원되어서 쉽게 오류를 찾아 낼 수 있습니다.
- (4) 모든 플랫폼에 독립적입니다. 유니 코드를 적용한 텍스트 형식의 문서이기에 하드디스크는 물론 운영체제나 프로그램 언어에 제약 없이 사용할 수 있습니다. 웹은 물론 애플리케이션에서도 사용가능합니다.
- (5) XML문서를 비XML문서로 바꾸는 등의 다양한 문서 변환이 가능하며, 모든 HTML태그를 수용합니다.

XML의 정의와 특징

항목	SGML	HTML	XML
태그의 제한성	사용자 정의 태그 무제한	태그 제한적 사용자 정의 태그 불가	사용자 정의 태그 무제한 SGML보다는 못함
검색효율	정확한 검색 가능	비효율적 검색	구조와 내용의 분리로 효율적 검색 가능
문서작성의 편리성	복잡하고 어려움	쉽고 간결하나 논리 구조 작성은 어려움	SGML을 간편화시켜 문서 작성이 편리

XML의 정의와 특징

XML 관련기술

[DTD(Document Type Definition)와 XML Schema] 업계별 데이터 공유를 용이하게 하기 위한 목적으로 사용됩니다. XML 문서의 표준을 규정하기 위한 모든 규칙을 말합니다.

[Namespace] 서로 다른 기업이나 기관, 조직 간에 정보를 공유할 때 동일한 이름의 태그가 다른 목적으로 사용되어 발생할 수 있는 충돌을 방지할 목적으로 사용됩니다.

[XSL(eXtensible Stylesheet Language)] 문서를 구조화하는 XML 문서를 HTML과 같이 익숙한 형태로 표현하게 해주는 스타일링 기술입니다.

[XSLT(XSL Transformation)] XML 기반의 문서를 HTML이나 WML 또는 다른 XML 타입을 가진 영역으로 변환하는 기술입니다.

[XLL(eXtensible Linking Language)] HTML의 하이퍼링크와 유사한 연결능력을 제시하며 XLink와 XPointer로 구성됩니다.

[XPath(XML Path Language)] XML 문서를 이루고 있는 구성요소들의 위치를 편리하게 지정하고 요소의 위치를 파악하기 위해 고안되었습니다.

[DOM(Document Object Model)] XML 문서를 객체 형태로 모델링함으로써 애플리케이션에서 XML 문서의 특정 부분의 값을 추출하거나 수정, 삭제, 추가하는 등의 작업을 가능케 합니다.

XML 문서의 구조

XML 문서의 구조

XML 문서 = Prolog + Body

ch01-03.xml

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

```
<!-- 문서를 표준화를 위한 DTD -->
```

```
<!DOCTYPE booklist SYSTEM "ch01-03.dtd">
```

```
<!-- 출력 형태를 표현해주는 XSL -->
```

```
<?xml-stylesheet type="text/xsl" href="ch01-03.xsl"?>
```

Prolog

```
<!-- B O D Y -->
```

```
<제품정보>
```

```
<과일 종류="배">
```

```
<생산지역> 나주시 </생산지역>
```

```
<수량> 1 </수량>
```

```
<가격> 56000 </가격>
```

```
</과일>
```

```
</제품정보>
```

Body

XML 문서의 구조

XML 문서의 구성요소

- PI(Processing Instruction, 처리 지시문)
- 주석(Comment)
- 요소(Element)
- 속성(Attribute)

XML 문서의 구성요소

`<?xml version="1.0" encoding="euc-kr" ?>` PI(Processing Instruction)

`<!-- This is Comment -->` 주석문(Comment)

`<제품정보>` 루트 요소(Root Element)

`<과일 종류="배">` 속성(Attribute)

`<생산지역> 나주시 </생산지역>`

`<수량> 1 </수량>`

`<가격> 56000 </가격>`

`</과일>` 요소(Element)

`</제품정보>`

XML 문서의 구조

PI

PI의 기본 형식

```
<?xml version="버전번호" encoding="인코딩방식" standalone="yes|no"?>
```

(1) version

PI의 version 속성의 사용 예

```
<?xml version="1.0"?>
```

(2) encoding

한글을 사용하기 위한 encoding 속성의 사용 예

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

또는

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

XML 문서의 구조

XML 주석

XML 주석의 사용 예

```
<<!-- XML 문서에 기술된 내용에 대한 설명을 기술하세요. -->
```

3.3 XML 요소

XML 요소를 구성하는 3가지(시작태그, 내용, 끝태그)

요소

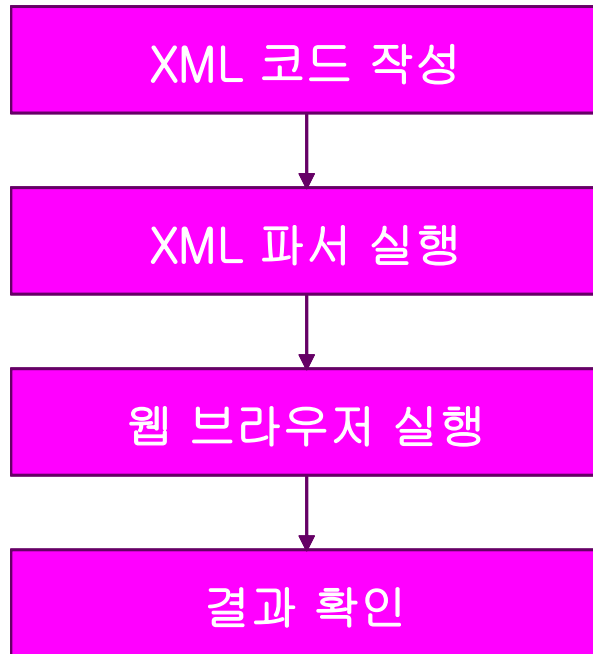
<code><생산지역></code>	<code>나주시</code>	<code></생산지역></code>
시작태그	내용	끝태그

3.4 XML 속성

XML 요소의 속성과 속성값

```
<과일 종류 = "배" >  
</과일>
```

XML 문서의 작성과 실행



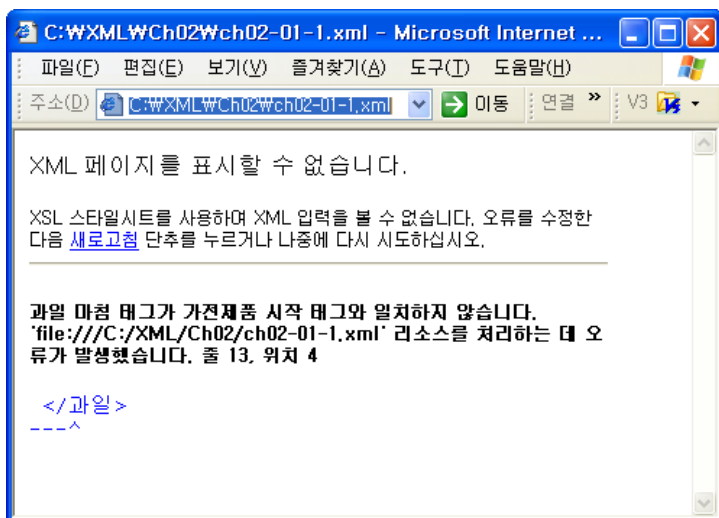
XML 사용하기 위해서 필요한 S/W

XML 문서를 작성할 수 있는 에디터
XML 문서를 검사할 수 있는 S/W
디스플레이 할 브라우저

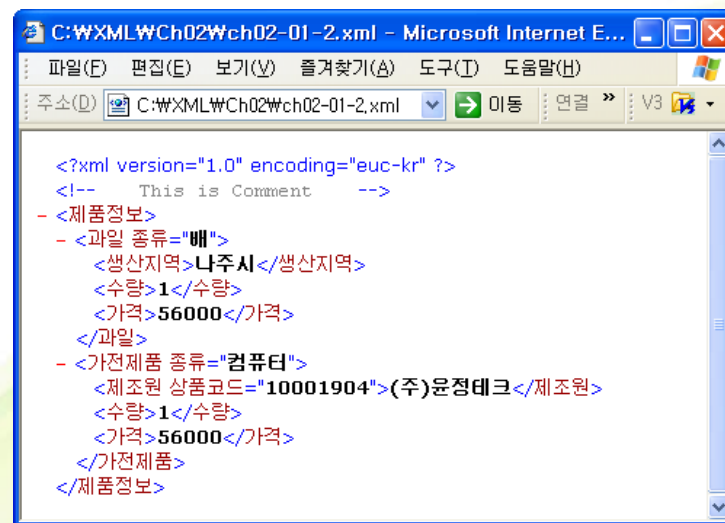
Well-Formed XML의 개념

Well-Formed XML 문서는 XML 1.0 스펙에 정의된 문법규칙을 잘 지켜서 작성된 XML 문서를 의미합니다.

Well-Formed 하지 않은 XML 문서]



[에러를 수정한 Well-Formed XML 문서]



Well-Formed(잘 구성된) 요소

요소와 나만의 태그

① 태그의 이름은 영문자와 숫자 특수 문자 중에서는 “_” , “.” 의 조합으로 구성합니다.

잘된 예	이 유
<author_id>	영문자와 “_”로 구성되었습니다.
<number3>	영문자와 숫자로 구성되었습니다.

② 태그이름의 시작은 영문자로 합니다. **encoding**에 “**euc-kr**”을 지정한 경우에는 한글도 태그이름의 시작으로 사용할 수 있습니다. 숫자나 구두점으로 태그이름을 시작할 수 없습니다. 하지만 특수문자 중에서는 언더바(“_”)로 태그 이름을 시작 할 수 있습니다.

잘된 예	이 유
<author>	첫 글자를 영문자로 시작하였습니다.
<_1000>	숫자 앞에 “_”를 덧붙였으므로 태그이름으로 적당합니다.

Well-Formed(잘 구성된) 요소

요소와 나만의 태그

③ 태그이름에 공백을 포함시킬 수 없습니다 .

잘못된 예	이 유
<author id>	두 단어로 태그의 이름을 지정할 경우에는 공백 대신 “-”를 사용합니다. 예를 들어 다음과 같이 지정합니다. <author-id>

④ 특수문자 중에 “:”문자는 네임스페이스를 지정할 때 사용하는 기호이므로 일반 태그이름에는 포함하지 않도록 합니다.

잘못된 예	이 유
<author:id>	XML 파서에 의해 에러를 발생하지는 않지만 태그의 이름으로 권장하지 않습니다.

Well-Formed(잘 구성된) 요소

요소와 나만의 태그

⑤ 태그이름은 대소문자를 구별하므로 철자(스펠링)만 같다고 같은 이름으로 혼동해서는 안 됩니다.

잘못된 예	이 유
<author> </Author>	시작 태그와 끝 태그에 기술한 태그의 이름이 일치하지 않으므로 에러를 발생합니다. XML 문서는 대소문자를 구분하므로 조심해야 합니다.

⑥ 태그의 시작을 알리는 “<” 기호 다음에 공백 없이 붙여서 태그의 이름을 기술해야 합니다. 하지만 태그의 끝을 알리는 “>” 기호는 공백이 있어도 상관없습니다.

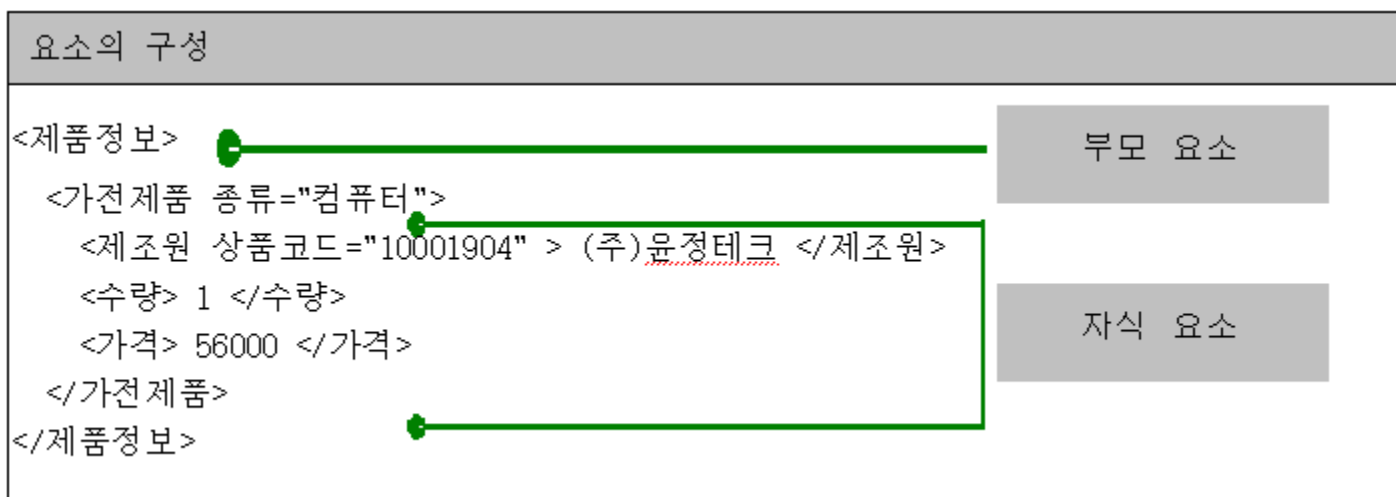
잘못된 예	이 유
< author >	시작을 알리는 < 기호 다음에 공백을 두었기에 에러를 발생합니다.

⑦ 태그이름은 **xml**로 시작할 수 없습니다.

잘못된 예	이 유
<xmlTest>	태그의 이름으로 xml을 포함하였기에 에러를 발생합니다.

Well-Formed(잘 구성된) 요소

XML 문서의 루트 요소



루트 요소란?

계층구조로 존재하는 XML 문서에서 최상위에 존재하는 요소를 루트 요소라고 하며 루트 요소는 반드시 하나만 유일하게 존재해야 합니다.

Well-Formed(잘 구성된) 요소

XML 문서의 루트 요소

```
<?xml version="1.0" encoding="euc-kr" ?>  
<!-- This is Comment -->
```

```
<과일 종류="배">
```

```
    <생산지역> 나주시 </생산지역>
```

```
    <수량> 1 </수량>
```

```
    <가격> 56000 </가격>
```

```
</과일>
```

```
<가전제품 종류="컴퓨터">
```

```
    <제조원 상품코드="10001904" > (주)윤정테크 </제조원>
```

```
    <수량> 1 </수량>
```

```
    <가격> 56000 </가격>
```

```
</가전제품>
```

```
// roor가 두개이므로 에러
```

Well-Formed(잘 구성된) 요소

XML 문서의 루트 요소

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

```
<!-- This is Comment -->
```

```
<제품정보>
```

```
  <과일 종류="배">
```

```
    <생산지역> 나주시 </생산지역>
```

```
    <수량> 1 </수량>
```

```
    <가격> 56000 </가격>
```

```
  </과일>
```

```
  <가전제품 종류="컴퓨터">
```

```
    <제조원 상품코드="10001904" > (주)윤정테크 </제조원>
```

```
    <수량> 1 </수량>
```

```
    <가격> 56000 </가격>
```

```
  </가전제품>
```

```
</제품정보>
```

Well-Formed(잘 구성된) 요소

시작 태그와 끝 태그 일치

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

```
<!-- This is Comment -->
```

```
<제품정보>
```

```
  <과일 종류="배">
```

```
    <생산지역> 나주시 </생산지역>
```

```
    <수량> 1 </수량>
```

```
    <가격> 56000
```

```
  </과일>
```

```
  <가전제품 종류="컴퓨터">
```

```
    <제조원 상품코드="10001904" > (주)윤정테크 </제조
```

```
원>
```

```
    <수량> 1 </수량>
```

```
    <가격> 56000 </가격>
```

```
  </가전제품>
```

```
</제품정보>
```

Well-Formed(잘 구성된) 요소

Empty 요소

HTML의 Empty 요소

```
<IMG SRC="photo.jpg">
```

XML에서의 Empty 요소 표현

```
<IMG SRC="photo.jpg"></IMG>
```

XML에서의 Empty 요소 표현

```
<IMG SRC="pretty.jpg"/>
```

Well-Formed(잘 구성된) 요소

Empty 요소

```
<?xml version="1.0" encoding="euc-kr"?>
<!-- Well Formed XML 문서입니다. -->
<제품정보>
  <과일 종류="배">
    <생산지역> 나주시 </생산지역>
    <주문 수량="1" 가격="56000"></주문>
  </과일>
  <가전제품 종류="컴퓨터">
    <제조원 상품코드="10001904" > (주)윤정테크 </제조원>
    <주문 수량="1" 가격="56000"/>
  </가전제품>
</제품정보>
```

Well-Formed(잘 구성된) 요소

태그를 지정할 때 대소문자 구분

태그명의 대소문자를 맞추어주지 않아서 Well-Formed 하지 않은 요소

```
<price> 546000 </PRICE>
```


Well-Formed(잘 구성된) 요소

요소 내포

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

<!-- 잘못된 XML 문서입니다. -->

<제품정보>

<과일 종류="배">

<생산지역> 나주시

</과일>

</생산지역>

<주문 수량="1" 가격="56000"></주문>

<가전제품 종류="컴퓨터">

<제조원 상품코드="10001904" > (주)윤정테크 </제조원>

<주문 수량="1" 가격="56000"/>

</가전제품>

</제품정보>

004:~008:를 다음과 같이 수정하면 됩니다.

004: <과일 종류="배">

005: <생산지역> 나주시

006: </생산지역>

007: <주문 수량="1" 가격="56000"></주문>

008: </과일>

Well-Formed 속성

속성을 포함한 요소

```
<주문 수량="1" 가격="56000"></주문>
```

속성값을 따옴표로 묶기

속성값을 단일 따옴표로 묶은 예

```
<주문 수량='1' 가격='56000' ></주문>
```

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

```
<!-- 잘못된 XML 문서입니다. -->
```

```
<제품정보>
```

```
  <과일 종류="배">
```

```
    <생산지역> 나주시
```

```
  </생산지역>
```

```
    <주문 수량 가격=56000></주문>
```

```
  </과일>
```

```
<가전제품 종류="컴퓨터">
```

```
  <제조원 상품코드="10001904" > (주)윤정테크 </제조원>
```

```
  <주문 수량='1' 가격="56000"/>
```

```
</가전제품>
```

```
</제품정보>
```

// 수량에 값, 5600에 따옴표

Well-Formed 속성

하나의 요소에 동일한 속성은 반복 사용 불가

```
<?xml version="1.0" encoding="euc-kr"?>
<!-- 잘못된 XML 문서입니다. -->
<제품정보>
  <과일 종류="배">
    <생산지역> 나주시 </생산지역>
    <주문 수량="1" 가격="56000" 가격="60000"></주문>
  </과일>
  <가전제품 종류="컴퓨터">
    <제조원 상품코드="10001904" > (주)윤정테크 </제조원>
    <주문 수량="1" 가격="56000"/>
  </가전제품>
</제품정보>
```

// 가격 두번

개체 참조와 문자 참조

PCDATA로 사용하면 안 되는 문자

'&' , '<' , '>' , '"' , "'"

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

```
<!-- 잘못된 XML 문서입니다. -->
```

```
<test>
```

```
    <and> YOU & ME </and>
```

```
</test>
```

문자	개체 참조	문자 참조	
		10진수	16진수
&	&	&	&
<	<	<	<
>	>	>	>
"	"	"	"
'	'	'	'

개체 참조와 문자 참조

PCDATA로 사용하면 안 되는 문자를 사용한 Well-Formed 하지 않은 예

```
<and> YOU & ME </and>
```

개체 참조와 문자 참조를 사용한 예

```
<and> YOU &amp; ME </and>
```

```
<and> YOU &#38; ME </and>
```

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

```
<!-- 잘된 XML 문서입니다. -->
```

```
<test>
```

```
  <!-- 개체 참조 -->
```

```
  <and> YOU &amp; ME </and>
```

```
  <!-- 십진문자참조 -->
```

```
  <and> YOU &#38; ME </and>
```

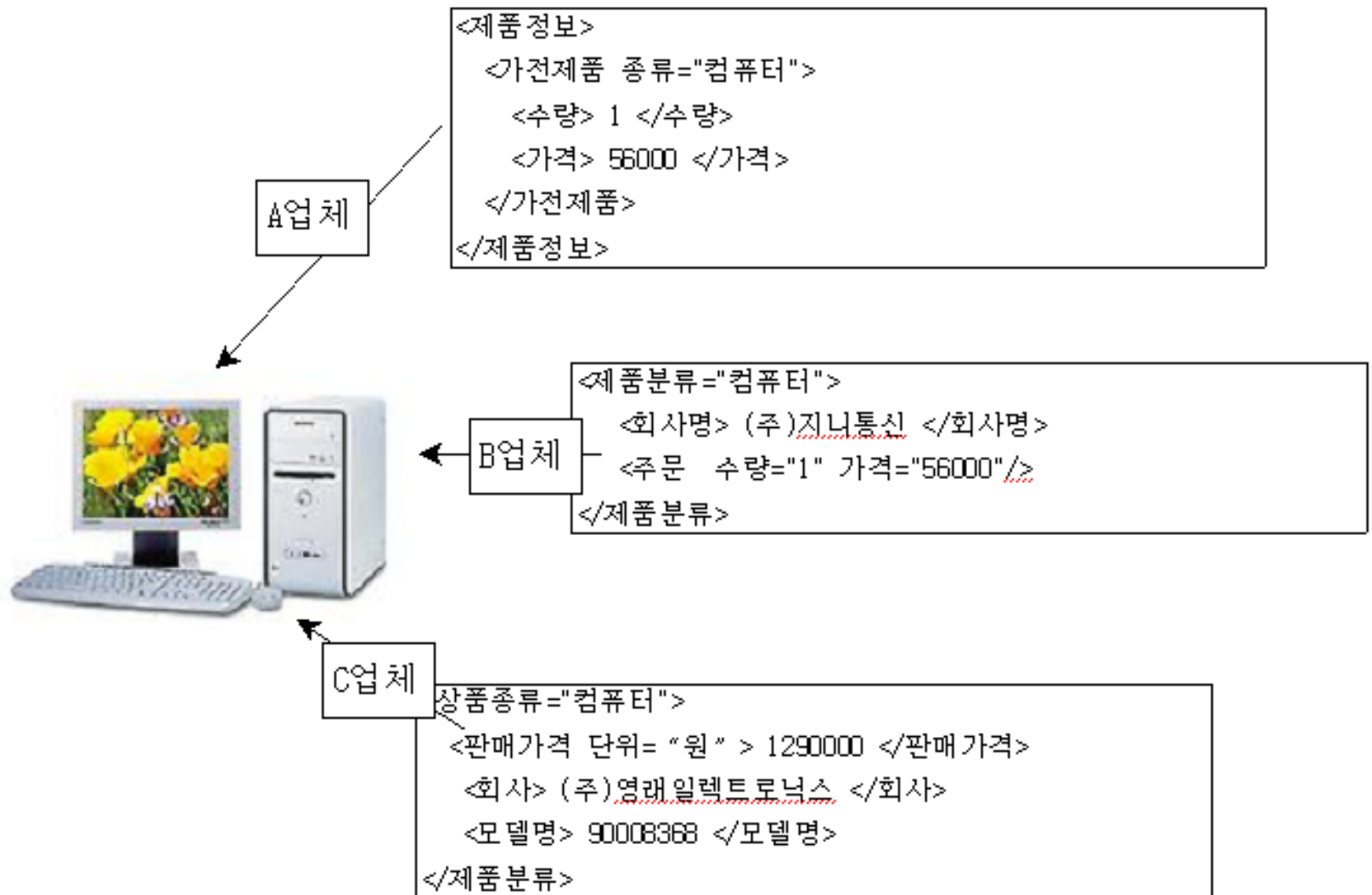
```
</test>
```

```
<?xml version="1.0" encoding="euc-kr"?>  
<!-- 잘된 XML 문서입니다. -->  
<test>  
    <and> <![CDATA[YOU & ME]]> </and>  
</test>
```

<Well-Formed XML 문서의 규칙>

- ① **XML** 선언문으로 문서를 시작해야 한다.
- ② **XML** 문서는 한 개의 유일한 루트 요소를 가져야 한다.
- ③ 시작과 끝 태그가 일치해야 한다.
- ④ 태그를 지정할 때 대소문자를 구분한다.
- ⑤ 내용을 포함하고 있지 않고 하나의 태그만을 사용하는 요소는 **/>**로 끝나야 한다.
- ⑥ 요소들은 내포되어야 한다.
- ⑦ 속성의 값은 반드시 인용부호를 사용해야 한다.
- ⑧ 하나의 요소에서 동일한 이름의 속성을 반복하여 사용할 수 없다.
- ⑨ **<**, **&**는 태그에만 사용할 수 있고, 데이터에는 사용할 수 없다. 사용하려면 개체 참조를 사용한다.
- ⑩ 내장된 개체 참조에는 **&(&)**, **<(<)**, **>(>)**, **'(')**, **"(")** 등이 있다. 필요시 **DTD**에서 사용자가 정의해서 개체 참조를 사용할 수 있다.
- ⑪ 특수문자를 제약 없이 사용하려면 **CDATA Section**을 사용하면 된다. **CDATA Section**은 개체 치환 없이 특수문자들을 자유롭게 사용할 수 있다.

DTD의 필요성



DTD의 필요성

표준이 정의되어 있지 않은 XML 문서의 문제점

보내온 XML 문서를 이해하는데 시간이 걸리며,
애플리케이션에서 XML 문서를 사용할 경우 서로 다른 규칙을 적용해야 하며,
XML 문서를 제작하는데도 효율적이지 못하므로
XML 문서 교환이 결국은 불편하고 복잡하며 혼란스러울 것입니다.

DTD의 필요성

Valid XML 문서의 개념

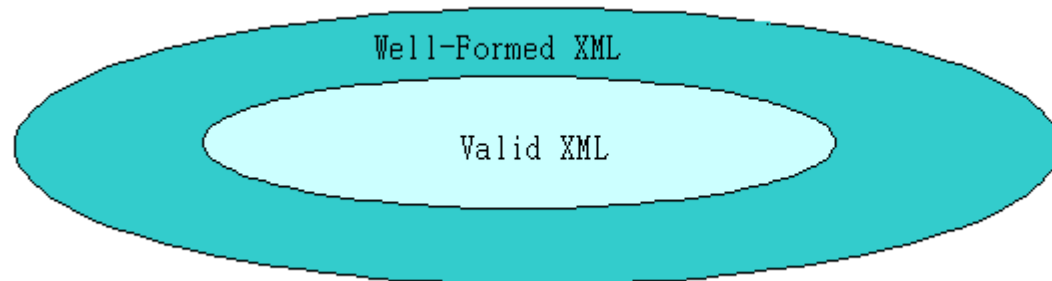
DTD란?

DTD는 Document Type Definition의 약어로서 직역해보면 문서의 형태를 정의하는 것이라고 해석됩니다. DTD는 업계별로 표준안을 정의하는 용도로 사용됩니다.

Well-Formed XML문서와 Valid XML문서란?

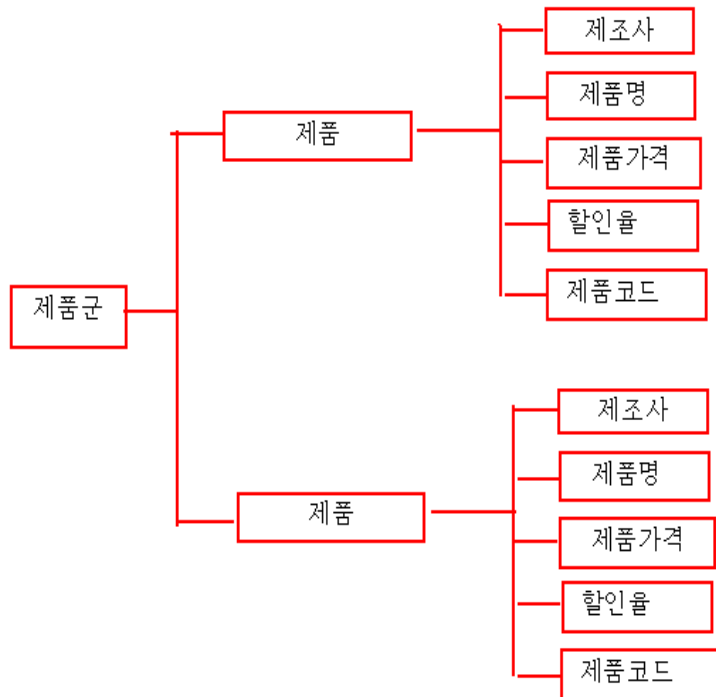
Well-Formed XML문서란 XML 문법에 맞추어서 작성된 문서를 말합니다.

Valid XML문서란 DTD에 정의된 규칙을 따르는 Well-Formed XML 문서입니다.



Valid XML문서 = Well-Formed XML문서 + DTD

DTD의 필요성



A업체
ch03-A.xml

```
<?xml version="1.0" encoding="euc-kr" ?>
<!-- ch03-A.xml -->
- <제품군>
- <제품>
  <제조사>(주) 한지 정보통신</제조사>
  <제품명>컴퓨터</제품명>
  <제품가격>1270000</제품가격>
  <할인율>10</할인율>
  <제품코드>10001904</제품코드>
</제품>
</제품군>
```

B업체
ch03-B.xml

```
<?xml version="1.0" encoding="euc-kr" ?>
<!-- ch03-B.xml -->
- <제품군>
- <제품>
  <제조사>(주) 지니 통신</제조사>
  <제품명>컴퓨터</제품명>
  <제품가격>1070000</제품가격>
  <할인율>5</할인율>
  <제품코드>10001936</제품코드>
</제품>
</제품군>
```

C업체
ch03-C.xml

```
<?xml version="1.0" encoding="euc-kr" ?>
<!-- ch03-C.xml -->
- <제품군>
- <제품>
  <제조사>(주) 영래 일렉트로닉스</제조사>
  <제품명>컴퓨터</제품명>
  <제품가격>1290000</제품가격>
  <할인율>15</할인율>
  <제품코드>90008368</제품코드>
</제품>
</제품군>
```

DTD 문서 작성

내부 DTD

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

```
<!DOCTYPE 제품군[  
  <!ELEMENT 제품군 (제품*)>  
  <!ELEMENT 제품 (제조사, 제품명, 제품  
가격, 할인율, 제품코드)>  
  <!ELEMENT 제조사 (#PCDATA)>  
  <!ELEMENT 제품명 (#PCDATA)>  
  <!ELEMENT 제품가격 (#PCDATA)>  
  <!ELEMENT 할인율 (#PCDATA)>  
  <!ELEMENT 제품코드 (#PCDATA)>  

```

```
<제품군>
```

```
  <제품>
```

```
    <제조사>
```

```
      (주) 한지 정보통신
```

```
    </제조사>
```

```
    <제품명>
```

```
      컴퓨터
```

```
    </제품명>
```

```
    <제품가격>
```

```
      1270000
```

```
    </제품가격>
```

```
    <할인율>
```

```
      10
```

```
    </할인율>
```

```
    <제품코드>
```

```
      10001904
```

```
    </제품코드>
```

```
  </제품>
```

```
</제품군>
```

네임스페이스의 필요성

요소 이름 충돌

User 사용자	
ID	사용자의 아이디
Password	사용자의 암호
Name	사용자의 이름
Email	사용자의 이메일
Phone	사용자의 전화번호

```
<User>
  <ID>pinkgirl</ID>
  <Password>123-456</Password>
  <Name>전원지</Name>
  <Email>pink@hello.com</Email>
  <Phone>02-777-7777</Phone>
</User>
```

Goods 상품	
ID	상품에 대한 아이디
CategoryID	상품의 종류
Name	상품의 이름
Price	상품의 가격

```
<Goods>
  <ID>19-908-098</ID>
  <CategoryID>7</CategoryID>
  <Name>NoteBook</Name>
  <Price>2250000</Price>
</Goods>
```

네임스페이스의 필요성

요소 이름 충돌

```
<?xml version="1.0" encoding="euc-kr"?>
<User>
  <ID>pinkgirl</ID>
  <Password>123-456</Password>
  <Name>전원지</Name>
  <Email>pink@hello.com</Email>
  <Phone>02-777-7777</Phone>
  <Goods>
    <Good>
      <ID>19-908-098</ID>
      <CategoryID>7</CategoryID>
      <Name>NoteBook</Name>
      <Price>2250000</Price>
    </Good>
  </Goods>
</User>
```

네임스페이스의 필요성


네임스페이스로 이름 충돌 방지

네임스페이스의 역할

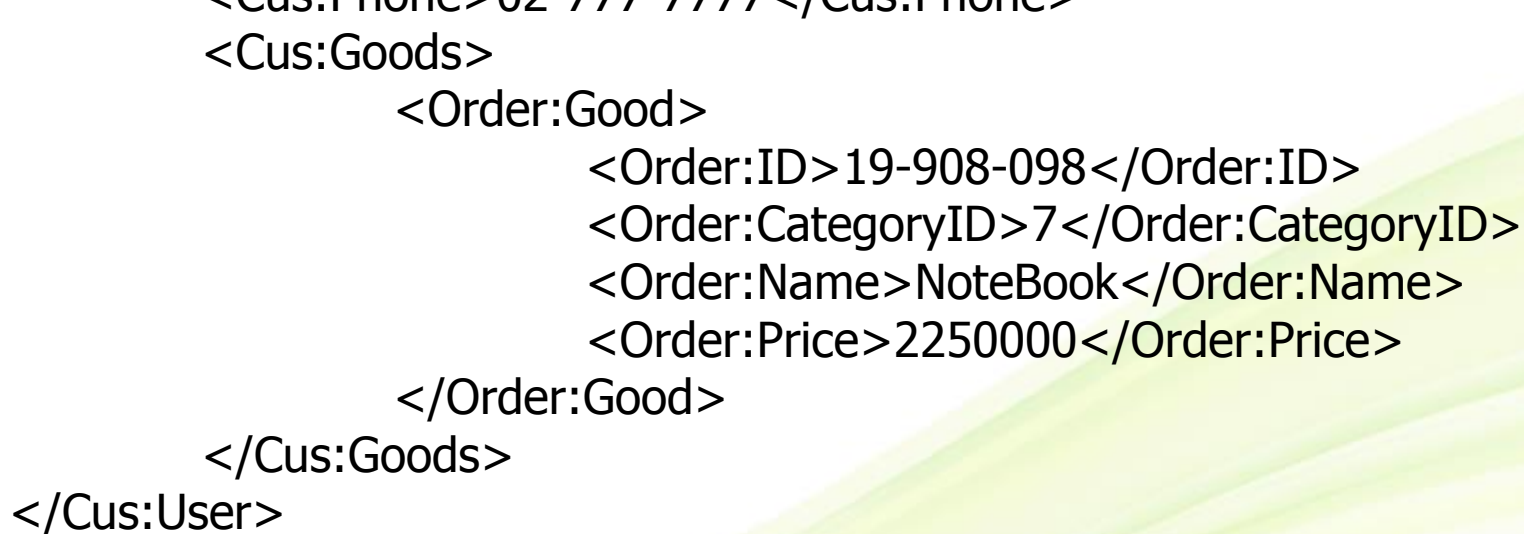
네임스페이스는 요소를 구별하기 위한 용도로 사용되며 이름 공간으로 번역할 수 있듯이 요소와 속성의 이름이 사용되는 이름 공간의 집합을 설정해 줍니다.

네임스페이스의 접두사

네임스페이스의 기본개념은 접두사로 이름의 충돌을 피한다는 데에서 출발합니다.



```
<?xml version="1.0" encoding="euc-kr"?>
<Cus:User>
  <Cus:ID>pinkgirl</Cus:ID>
  <Cus:Password>123-456</Cus:Password>
  <Cus:Name>전원지</Cus:Name>
  <Cus:Email>pink@hello.com</Cus:Email>
  <Cus:Phone>02-777-7777</Cus:Phone>
  <Cus:Goods>
    <Order:Good>
      <Order:ID>19-908-098</Order:ID>
      <Order:CategoryID>7</Order:CategoryID>
      <Order:Name>NoteBook</Order:Name>
      <Order:Price>2250000</Order:Price>
    </Order:Good>
  </Cus:Goods>
</Cus:User>
```

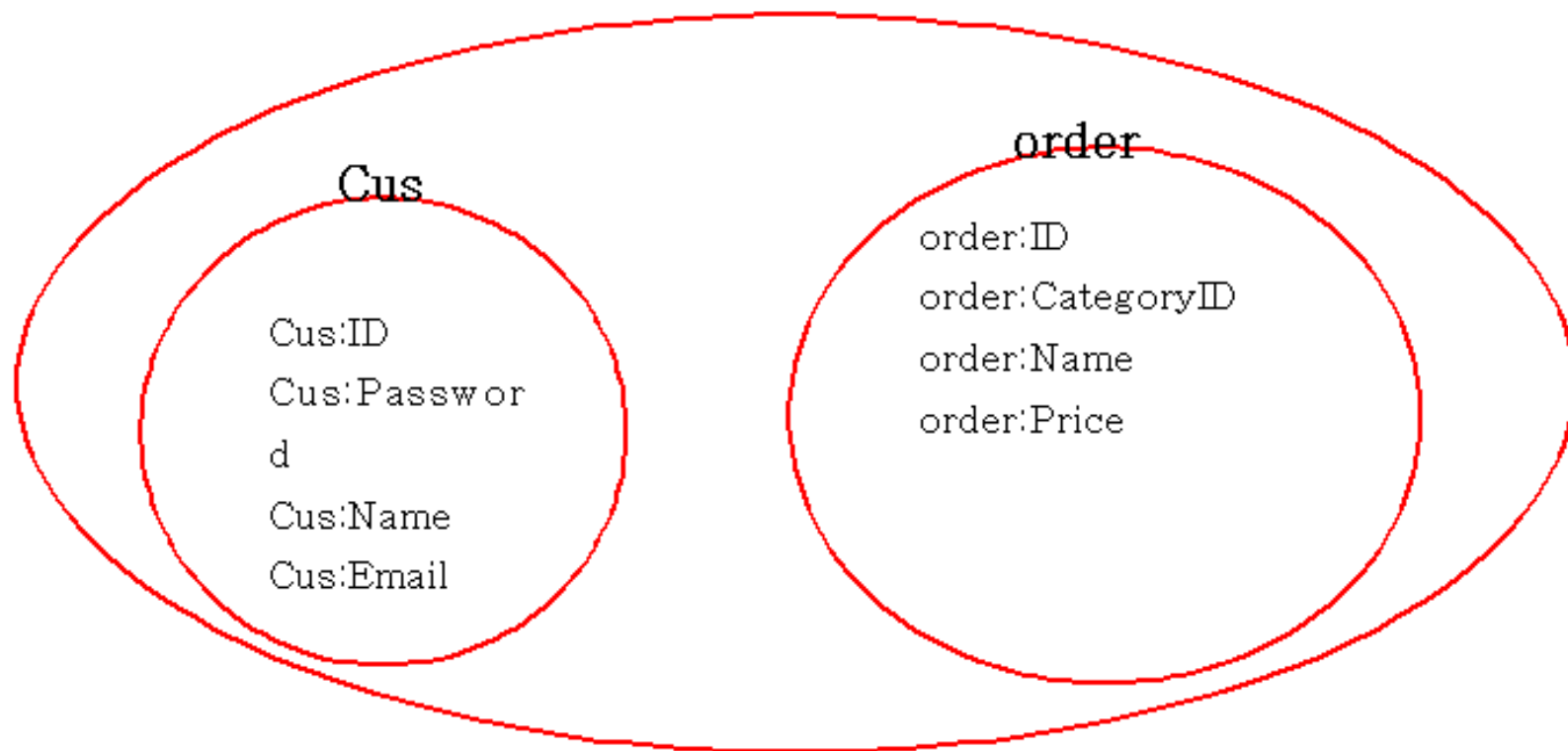


네임스페이스의 필요성

네임스페이스로 이름 충돌 방지

Cus = { ID, Password, Name, Email, Phone }

Order = { ID, CategoryID, Name, Price }



네임스페이스의 선언

네임스페이스 선언의 기본 형식

```
<prefix:ElementName xmlns:prefix=URL>
```

네임스페이스 선언의 예

```
<Cus:User xmlns:Cus="http://www.pop.com/buy/Cus">
```

네임스페이스의 선언

```
<?xml version="1.0" encoding="euc-kr"?>
<Cus:User xmlns:Cus="http://www.pop.com/buy/cus"
  xmlns:Order="http://www.techcom.net/2002/Order">
  <Cus:ID>pinkgirl</Cus:ID>
  <Cus:Password>123-456</Cus:Password>
  <Cus:Name>전원지</Cus:Name>
  <Cus:Email>pink@hello.com</Cus:Email>
  <Cus:Phone>02-777-7777</Cus:Phone>
  <Cus:Goods>
    <Order:Good>
      <Order:ID>19-908-098</Order:ID>
      <Order:CategoryID>7</Order:CategoryID>
      <Order:Name>NoteBook</Order:Name>
      <Order:Price>2250000</Order:Price>
    </Order:Good>
  </Cus:Goods>
</Cus:User>
```

기본 네임스페이스

기본 네임스페이스를 선언하는 기본 형식

```
<ElementName xmlns=URL>
```

기본 네임스페이스를 사용한 예

```
<User xmlns="http://www.pop.com/buy/cus">  
  <ID>pinkgirl</ID>  
  <Password>123-456</Password>  
  :  
</User>
```

기본 네임스페이스

```
<?xml version="1.0" encoding="euc-kr"?>
<User xmlns="http://www.pop.com/buy/cus"
      xmlns:Order="http://www.techcom.net/2002/Order">
  <ID>pinkgirl</ID>
  <Password>123-456</Password>
  <Name>전원지</Name>
  <Email>pink@hello.com</Email>
  <Phone>02-777-7777</Phone>
  <Goods>
    <Order:Good>
      <Order:ID>19-908-098</Order:ID>
      <Order:CategoryID>7</Order:CategoryID>
      <Order:Name>NoteBook</Order:Name>
      <Order:Price>2250000</Order:Price>
    </Order:Good>
  </Goods>
</User>
```

DTD의 한계와 대안으로서의 스키마

① **DTD**는 구현이 어렵다.

DTD는 XML 문법을 따르지 않기 때문에 작성방법을 따로 익혀야 한다는 번거로움이 있습니다.

② **DTD**는 제한된 데이터 형만 제공된다.

DTD는 문자열 이외의 데이터 형(숫자나 날짜)을 표현할 수 없으며, 값의 범위 제한이나 데이터 유형에 대한 제한을 할 수 없습니다. DTD를 사용하는 문서 내에서는 숫자도 문자열 형태로 기술할 수밖에 없습니다.

③ **DTD**는 재사용성과 확장성이 불가능하다.

DTD는 클래스의 상속 개념도 없고, 자식 클래스를 파생시키는 개념도 존재하지 않습니다. 이것은 코드의 재사용이라는 측면에서 비효율성을 초래합니다. 또한 DTD 동작은 XML 1.0에 구속되어 있기 때문에 스펙을 개정하지 않는 한 새로운 선언이나 새로운 특징을 추가할 방법이 없습니다.

XML 스키마

스키마 작성하기

스키마 문서의 최상위 요소인 <schema>의 기본 형식

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
</xs:schema>
```

스키마의 <element/>로 제품군 요소를 데이터 형으로 정의

```
<xs:element name="제품군" type="xs:string"/>
```

ch06-01.xsd

```
<?xml version="1.0" encoding="euc-kr"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >  
  <xs:element name="제품군" type="xs:string">  
  
    </xs:element>  
</xs:schema>
```

스키마 문서의 물리적 구성

스키마 문서의 루트 요소인 **schema**

스키마 문서의 최상위 요소인 `<schema>`의 기본 형식

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
</xs:schema>
```

요소 선언

스키마의 `<element/>`로 제조사 요소를 데이터 형으로 정의

```
<xs:element name="제조사" type="xs:string"/>
```

속성 선언

스키마의 `<attribute/>`로 제품코드 속성을 정의

```
<xs:element name="제품" >  
  <attribute name='제품코드' />  
</xs:element>
```

XML 문서에서 적용한 예

```
<제품 제품코드="A1002"/>
```



```
<?xml version="1.0" encoding="euc-kr"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="제품군">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="제품">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="제조사" type="xs:string"/>
              <xs:element name="제품명" type="xs:string"/>
              <xs:element name="제품가격" type="xs:integer"/>
              <xs:element name="할인율" type="xs:integer"/>
              <xs:element name="제품코드" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

CSS 설정과 적용

스타일시트는 스타일 설정과 스타일 적용으로 나뉩니다.

스타일 설정

스타일 설정의 기본 형식

```
selector { 속성 : 값 }
```

H1 태그에 글자 색상을 지정하는 예

```
H1{color : red}
```

H2 태그의 글꼴체와 글자 색상을 지정한 예

```
H2{font-family : 굴서체; color : red}
```

CSS 설정과 적용

XML 문서에 외부 CSS를 적용하기 위한 기본 형식

```
<?xml:stylesheet type="text/css" href="경로명 + 파일이름.css"?>
```

ch07-01.css

good {font-size: 20pt; color: red}

company {font-size: 15pt; color : navy}

name {font-weight : bold; color : blue;}

price {font-style : italic; color : green;}

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

```
<?xml:stylesheet type="text/css" href="ch07-01.css"?>
```

```
<goods>
```

```
  <good>
```

```
    <company>(주)한지 정보통신</company>
```

```
    <name>컴퓨터</name>
```

```
    <price> 1270000</price>
```

```
    <discount>10</discount>
```

```
  </good>
```

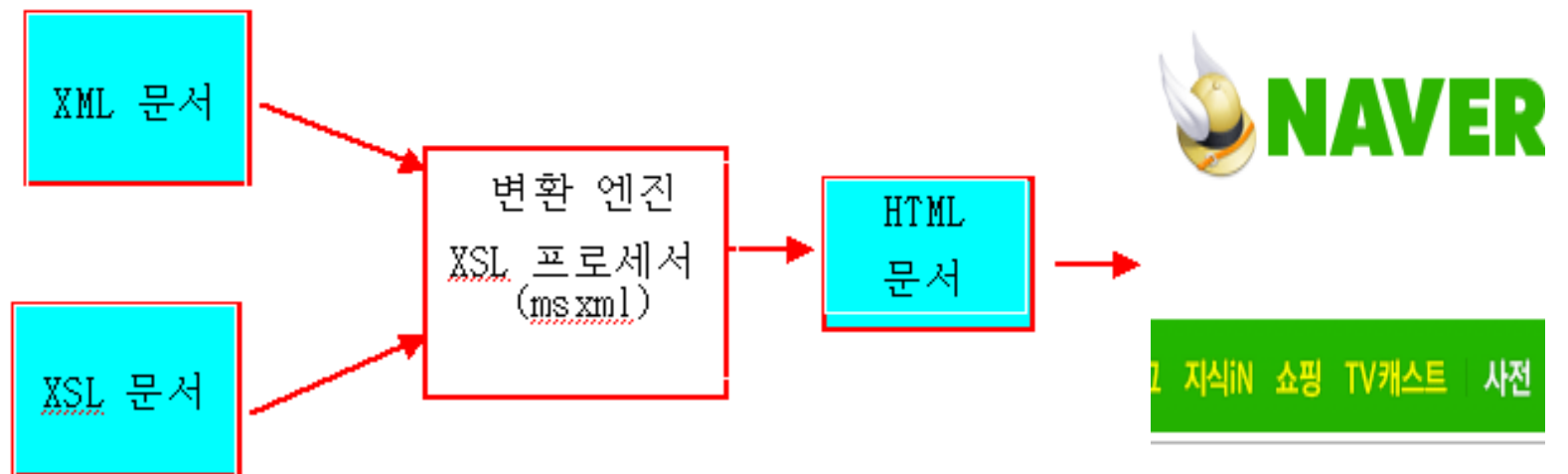
```
</goods>
```

XML 문서에 XSL를 적용하는 이유

XML의 가장 큰 특징 중의 하나가 문서의 구조 및 의미를 나타내는 부분과 문서의 표현 (Presentation)을 나타내는 부분으로 분리한다는 점입니다.

XML의 특성상 문서를 어떻게 표현할 것인가를 정의할 수 있는 별도의 언어가 필요하게 되었는데, 그 언어가 바로 XSL입니다.

XSL문서의 문법



XML 문서에 XSL를 적용하는 이유

제품 내역입니다.

주소(D) C:\WXML\WCh08\Wch08-04.xml

핸드폰 리스트

모델명	통신사	구입형태	제조사	색상	수량	가격	이미지
SCH-X147	017	기기변경	삼성전자	블루	30	253000	
CX-300L	019	신규가입	LG전자	실버	30	270000	

완료 내 컴퓨터

[동일한 XML 문서에 다른 XSL을 적용하였을 때의 결과 화면]

제품 내역입니다.

주소(D) C:\WXML\WCh08\Wch08-04.xml

핸드폰 리스트

모델명	통신사	구입형태	제조사	색상	수량	가격
SCH-X147	017	기기변경	삼성전자	블루	30	253000
CX-300L	019	신규가입	LG전자	실버	30	270000

완료 내 컴퓨터

[XML 문서에 XSL을 적용해서 테이블 형태로 표현한 화면]

XPath의 개념

XPath란?

XPath는 XML 계층 구조에서 특정 요소까지 도달하기 위한 경로를 표현할 수 있도록 하는 새로운 개념의 스펙입니다.

XPath의 개념

XPath의 노드

```
<제품>
  <핸드폰 모델명="SCH-X147">
    <통신사>017</통신사>
    <구입형태>기기변경</구입형태>
    <제조사>삼성전자</제조사>
  </핸드폰>
  <핸드폰 모델명="CX-300L">
    <통신사>019</통신사>
    <구입형태>신규가입</구입형태>
    <제조사>LG전자</제조사>
  </핸드폰>
</제품>
```

문서 루트
(가상노드)

루트 노드

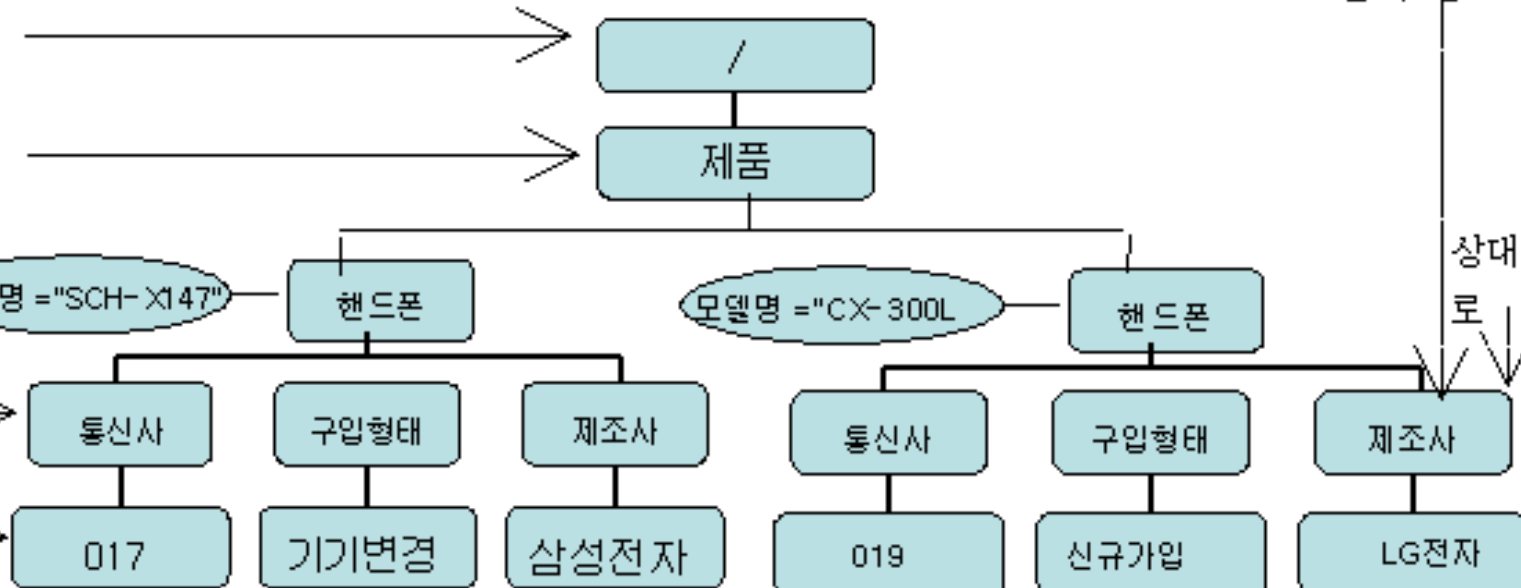
속성노드 → 모델명="SCH-X147"

요소 노드 → 통신사

텍스트 노드 → 017

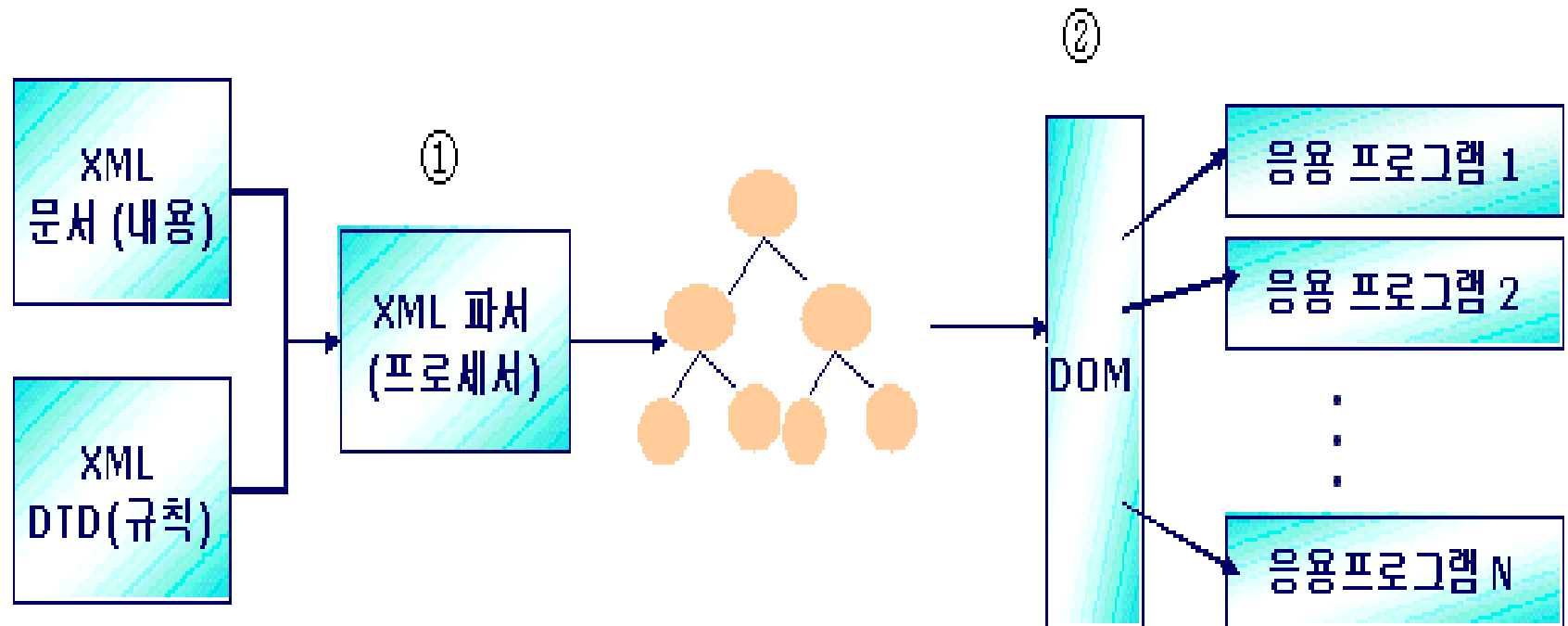
절대 경로

상대 경
로

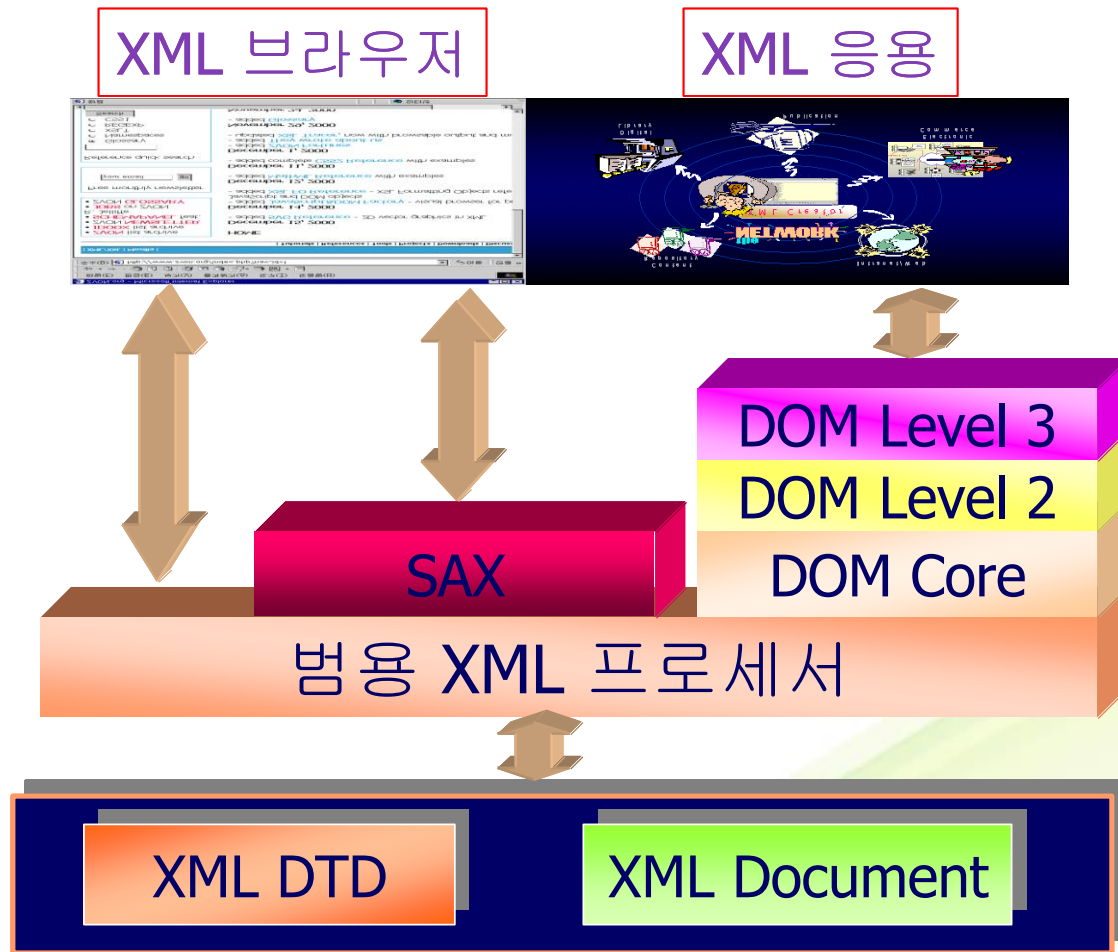


XML DOM

HTML의 하이퍼링크



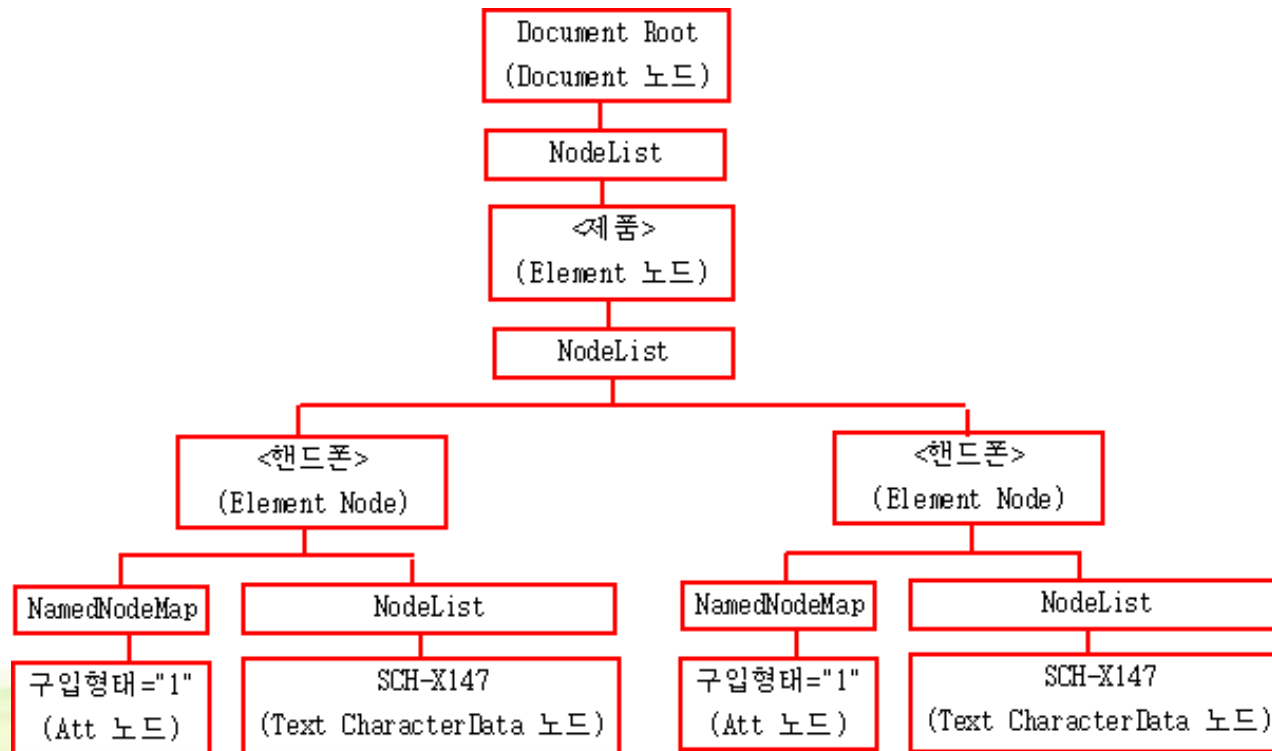
객체 모델로서의 XML



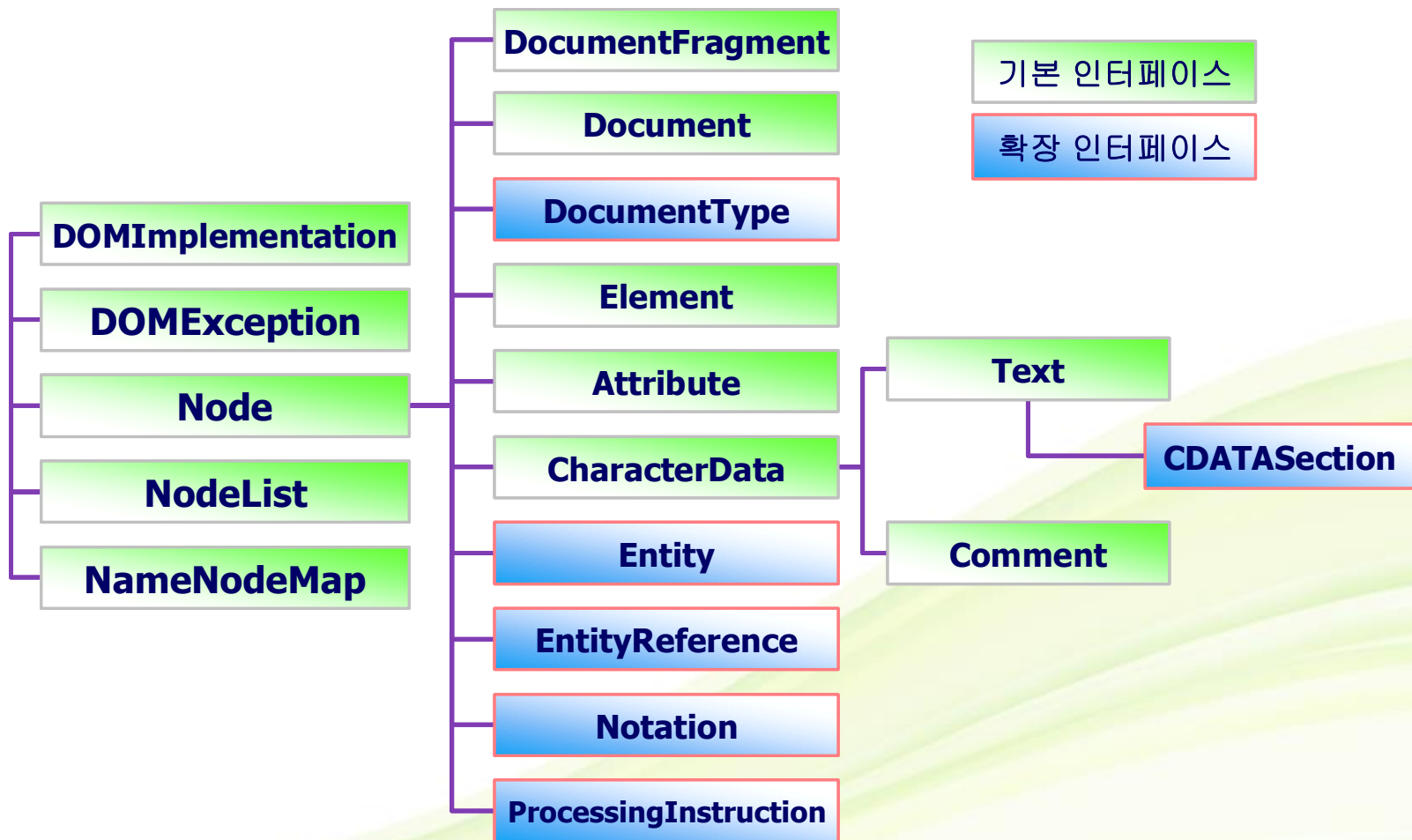
객체 모델로서의 XML

XML 문서(Ch12-01.xml)

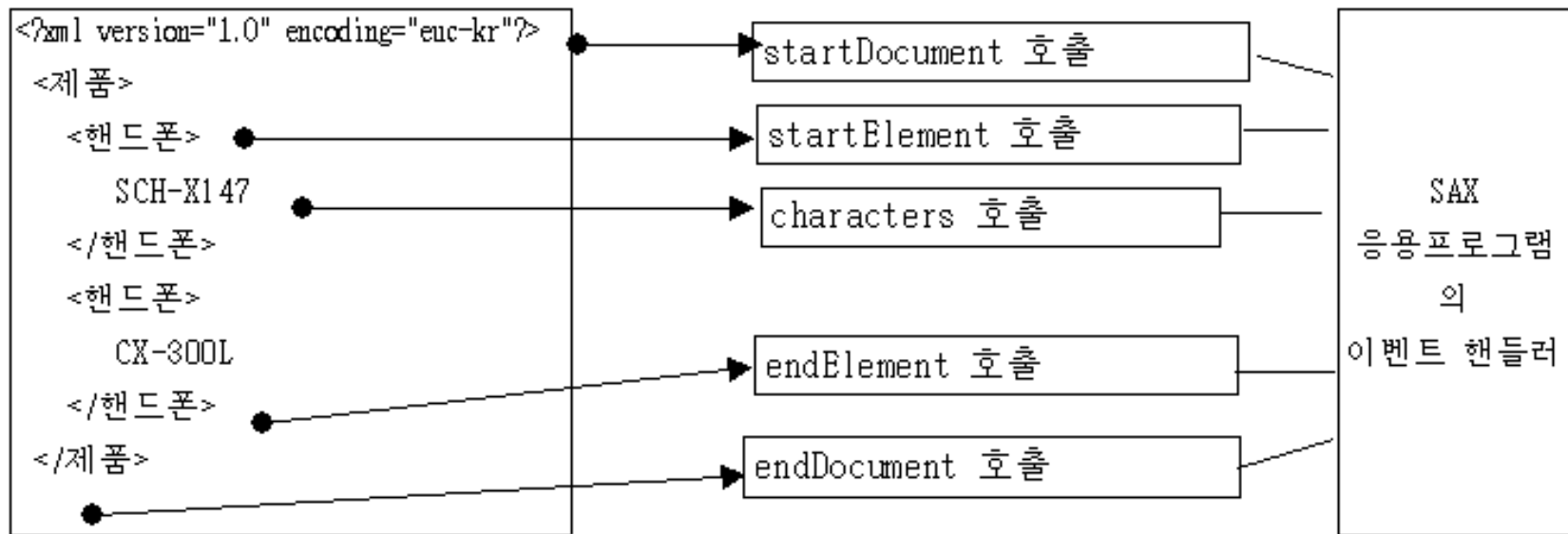
```
<제품>
  <핸드폰>
    <모델명 구입형태="1"> SCH-X147 </모델명>
  </핸드폰>
  <핸드폰>
    <모델명 구입형태="2"> CX-300L </모델명>
  </핸드폰>
</제품>
```



객체 모델로서의 XML



이벤트 기반 인터페이스 SAX



`startDocument/endDocument` : XML문서가 시작/끝날 때 호출됩니다.

`startElement/endElement` : XML문서의 각각의 요소가 시작/끝날 때 호출됩니다.

`characters` : XML의 태그가 아닌 데이터, 즉 `textNode`를 만났을 때 호출됩니다.