2023/2024(1)
EF234301 Web Programming

Lecture #5

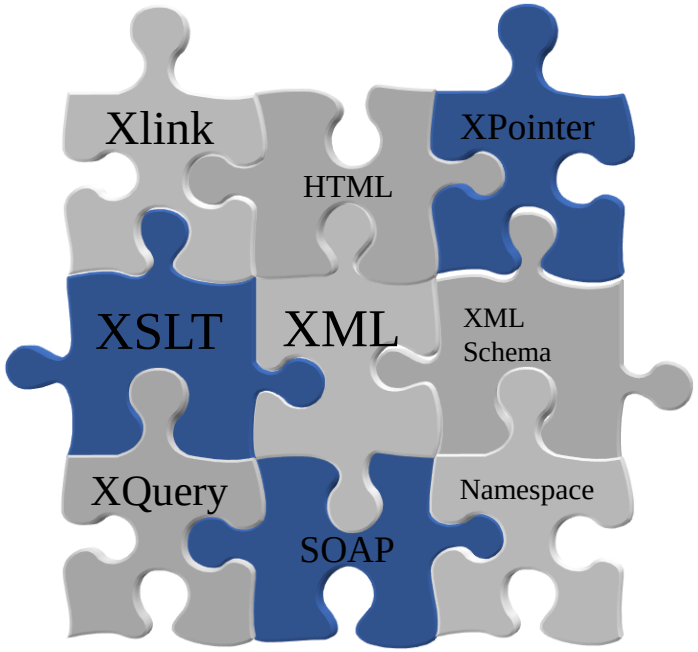
XML

Misbakhul Munir IRFAN SUBAKTI
司馬伊凡
Мисбакхул Мунир Ирфан Субакти

XML

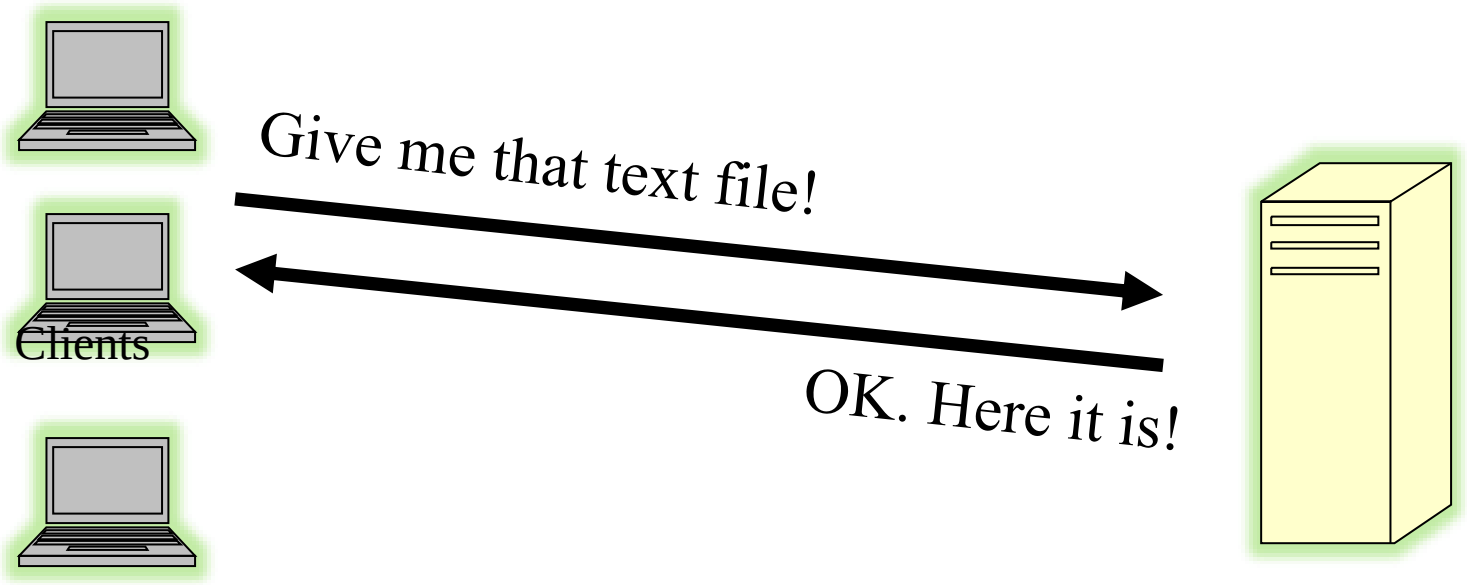
- eXtensible Markup Language

Overview



- What is it?
  - HTML → make up our own tags
- e**X**tensible **M**arkup **L**anguage
  - There're a lot of pieces in the XML puzzle,
  - And they are moving at different speeds
- Very fast growing
  - W3C: governing body
  - XML 1.0 → 1998, XML 1.1 → 2006, XML 2.0 → ? (XML-SW, skunkworks)
  - Valid Unicode characters in XML 1.0 (5th ed) and XML 1.1 (all ed) → Unicode 3.2 forward → contains Balinese (2006, U+1B00–U+1B7F), Javanese (2009, U+A980–U+A9DF), Cham, Phoenician, etc.
  - Binary XML → binary encoding of XML Information Set, XML: text based vs ITU-T & ISO: Fast Infoset

Web: how it works



Browser → web clients

Web server

HTML: hypertext mark-up language

- Formatting language
- Browser → interpreting the tags

```
<H1>This is a heading</H1>
<P>A<B>paragraph</B>started from here</P>
```

XML: what is it?

- It is NOT a mark-up language
- Meta mark-up language
- A set of simple rules
- Provide a uniform method → describing & exchanging structured data
- Describing structure & semantic data → NOT format/layout data

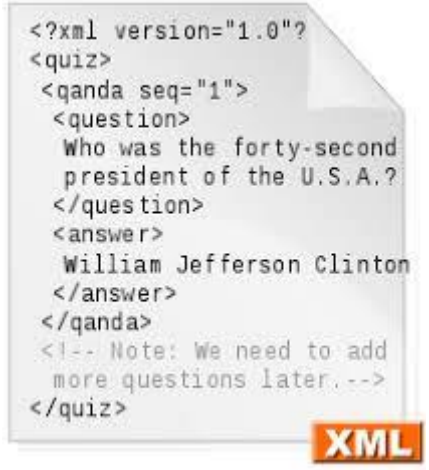
XML: just a normal text? Well, …

- HTML replacement → HTML can be made from XML
- Presentation format → XML can be changed to be presentation format
- Programming language → XML can be used for almost all of programming languages
- Network transfer protocol → XML can be transferred via network
- Database → XML can be stored in a database

Analogy

Portable application

Portable data



XML: meta language

1. Using alphabet
2. Space between words
3. Read from left to right

…

XML: the application

- File configuration
  - J2EE architecture → Java Enterprise Edition → Jakarta EE
- Media for exchanging data
- B2B (Business to Business) transaction
  - Electronic Business Order (ebXML)
  - Interactive Financial eXchange (IFX)
  - Messaging Exchange (SOAP, Simple Object Access Protocol)

XML: specification

- Tag
  - Cannot be overlapped
  - Case-sensitive
  - Need to have tag root
- SVG
  - Scalable Vector Graphics
  - Defined vector-based graphics for the Web
  - Defines the graphics in XML format
- WML
  - Wireless Markup Language
- MathML
  - Mathematical Markup Language
  - Dialect of XML for describing mathematical notation and capturing both its structure and content

XML: the rules

- XML document writing rules

XML: the rules (continued)

- Root element → single, unique
- Tag → opening & closing have to match
- Consistent capitalisation
- Nested element → cannot be overlapped
- Attribute's value surrounded by quotation
- No attribute repetition in an element

```
1  <?xml version = "1.0"?>
2  <institution id = "4859">
3      <name>ITS</name>
4      <type>University</type>
5      <address>
6          <street>Jl Raya ITS</street>
7          <city>Surabaya</city>
8          <country>Indonesia</country>
9      </address>
10 </institution>
```

Root element → single, unique

- Each XML document need to have a root element

```
<root>
    <child>
        <subchild> ... </subchild>
    </child>
</root>
```

Tag → opening & closing have to match

- XML element → nested, as in HTML
- Nested element needs to be matched → cannot be overlapped
- HTML, e.g.,

```
<b><i>It’s bold and italic</b></i>
```

- XML, e.g.,

```
<b><i>It’s bold and italic</i></b>
```

Attribute’s value surrounded by quotation

- Wrong example

```
<note date=16 October 2020>
    <to>My dear daughter</to>
    <from>Your beloved one</from>
</note>
```

- Correct example

```
<note date="16 October 2020">
    <to>My dear daughter</to>
    <from>Your beloved one</from>
</note>
```

Entity references

- Some characters have a special meaning in XML
  - E.g., “<“ in XML element → error, because it deemed as a new element
- Wrong example

```
<kidage>if age < 18 then</kidage>
<adultage>if age > 18 then</adultage>
<pair>boy & girl</pair>
```

- Correct example

```
<kidage>if age &lt; 18 then</kidage>
<adultage>if age &gt; 18 then</adultage>
<pair>boy &amp; girl</pair>
```

Entity references (continued)

- 5 entity references in XML need to be written correctly

Entity name	Character	Decimal reference	Hexadecimal reference
quot	"	"	"
amp	&	&	&
apos	'	'	'
lt	<	<	<
gt	>	>	>

XML: comment/remark

- As in HTML

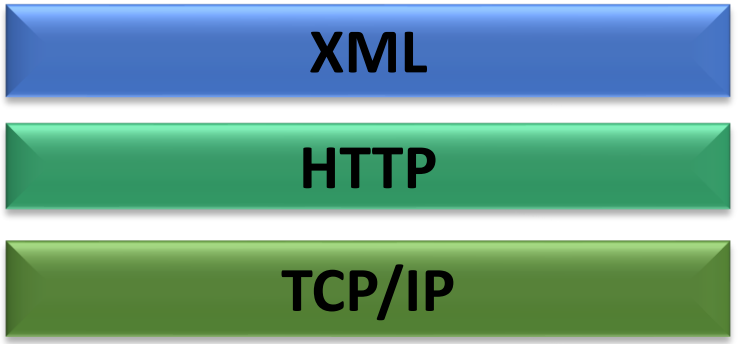
```
<!– As in HTML, this the way comment in XML -->
```

XML: the history

- SGML → Standard Generalised Markup Language
- 10 February 1998 → XML 1.0
- 29 September 2006 → XML 1.1
- XML → subset of SGML (SGML-lite)
- XML → lesser and simpler syntax
- SGML development → base for XML

XML: the power

- Internationally independent standard
- Simple structure
- Universality → software & hardware
- Extensible
- Scalable → data & design separation
- Integrity



XML: tree structure

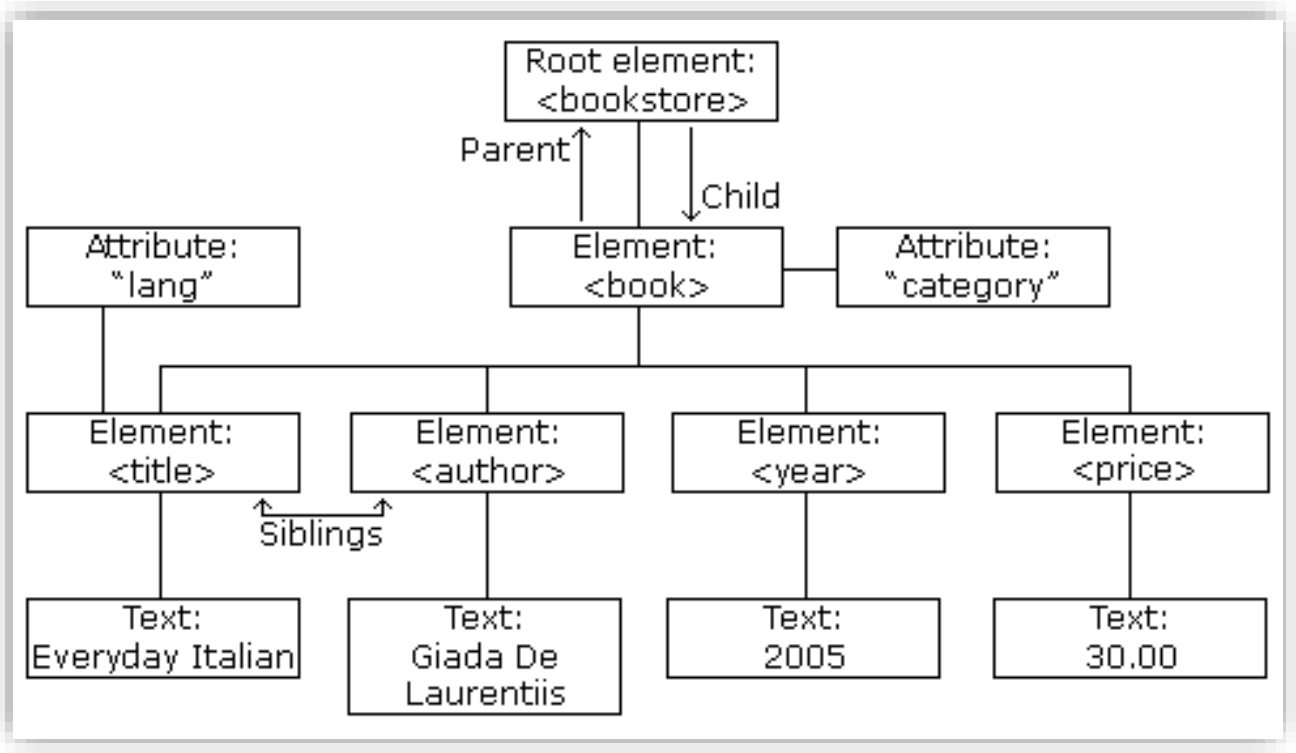
- XML document has to have root element → the parent of all elements
- XML elements similar to tree structure → starting from the root goes to the leaves as the lowest level
- Each XML element can have sub element (child)

XML: tree structure → example

```
<root>
    <child>
        <subchild> ... </subchild>
    </child>
</root>
```

- Parent, child & sibling → relation between elements
- Parent has child(ren) element
- Children elements within same level → sibling (brothers & sisters)
- XML elements → content & attribute (as in HTML)

XML: visualisation example



XML: self description

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<note>
    <to>Yarik</to>
    <from>Naya</from>
    <heading>Reminder</heading>
    <body>Don't forget to help mommy!</body>
</note>
```

- 1st line is XML declaration → defined XML version 1.0 and using the encoding ISO-8859-1 = Latin-1/West European character set
- Next line → defined the root element

```
<note>
```

XML: self description (continued)

- Next 4 lines → defined the children elements (to, from, heading & body)

```
<to>Yarik</to>
<from>Naya</from>
<heading>Reminder</heading>
<body>Don't forget to help mommy!</body>
```

- Last line → root element closing

```
</note>
```

XML: document example

```
<bookstore>
<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>
```

XML markup languages/vocabularies

- XML → meta language
- Everyone can create her/his own language
- What do you want to create?
- A language created → for some purpose

MathML: Mathematical Markup Language

- $x^2 + 4x + 4 = 0$

```
<apply><plus/>
    <apply><power/>
        <ci>x</ci>
        <cn>2</cn>
    </apply>
    <apply><times/>
        <cn>4</cn>
        <ci>x</ci>
    </apply>
    <cn>4</cn>
</apply>
```

XML: new language

- Any new internet language can be created by using XML
- E.g.,
  - XHTML → a new version of HTML
  - WSDL (Web Service Definition Language) → describe web service
  - WAP (Wireless Application Protocol) & WML (Wireless Markup Language) → handheld devices markup languages
  - RSS → news feeds

Synchronised Multimedia Integration Language (SMIL)

```
<DIV CLASS="time" t:timeline="seq">>
    <P class="time" t:dur="1">
        This will be appeared for 1 second then will be gone.
    </P>
    <P class="time" t:dur="1">>
        This will be appeared for 1 second then will be appeared for next 1 second.
    </P>
    <P class="time" t:dur="1">>
        This will be appeared after 2 seconds then will be appeared for next 1 second.
    </P>
</DIV>
```

Vector Markup Language

```
<v:shape style='top: 0; left: 0; width: 250; height: 250'
    stroke="true" strokecolor="red" strokeweight="2" fill="true"
    fillcolor="green" coordorigin="0 0" coordsize="175 175">
<v:path v="m 8,65 |
    72,65,92,11,112,65,174,65,122,100,142,155,92,121,42,155,60,100
  x e"/>
</v:shape>
```

Wireless Markup Language

```
<wml>
<!-- root element -->
<card id="card1" title="Example 1">
    <p>
        <!-- a card can only contain P blocks or DO blocks -->
        <do type="accept" label="go to card 2">
            <go href="#card2"/>
        </do>
        This is the first card.
    </p>
</card>
<card id="card2" title="Example 1">
    <p> This is the second card. </p>
</card>
</wml>
```

Hypertext Markup Language (HTML)

```
<h1>Introduction/h1>
<p>
    Welcome to our <b>website</b>
</p>
```

- Is it XML?
- Next version of HTML → XHTML

XML schemas & DTDs

- XML → communication
- Talk with the same language
- Schema/DTDs describes language's vocabulary
- DTD → Document Type Definition
- E.g.,
  - Which tags are used
  - How they can be arranged
- Schemas will replace DTDs

Schema: the example

```
<?xml version="1.0" encoding="UTF-8"?>
<PressRelease>
   <Title>Studend Loan Problems</Title>
   <Date>20/7/01</Date>
   <Content>Bla Bla Bla</Content>
</PressRelease>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xsd:element name="PressRelease">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Date" type="xsd:date"/>
      <xsd:element name="Content" type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Document Type Definition (DTD)

- Describes specification & rules for the elements & attributes an XML document needs to have
- A set of these rules → schema
- Schema is not mandatory → but it needs to ensure the document consistency

Document Type Definition (continued)

- Defines the elements in DTD:

```
<!ELEMENT animal (lion)>
```

  - Element animal only has one element lion.

```
<!ELEMENT picture EMPTY>
```

  - Element picture doesn't have any other element

```
<!ELEMENT animal ANY>
```

  - Element animal can have any other element

- Defines an element which has the text

```
<!ELEMENT name(#PCDATA)>
```

Document Type Definition (continued)

- Defines element which has some elements

```
<!ELEMENT animal(name, weight)>
```

- Based on the rule above, then an element has to have element name and weight

```
<animal>
<name>lion</name>
<weight>350 pounds</weight>
</animal>
```

Document Type Definition (continued)

- Defines element which has some choices/options

```
<!ELEMENT animal ((name, weight) | (picture)>
```

- Based on the rule above then an element animal has to have element name & weight or only has element picture
- Defines Unit in an element:

```
<!ELEMENT animal (name+, weight?, picture, subspecies*)
```

- Explanation
  - name need to be appeared at least once
  - weight can be appeared once or none
  - picture only need to be appeared once
  - subspecies can be appeared many times or none

Attribute

- Attribute can be more useful than decomposing an element into sub-sub element
- These 2 elements are the same

```
<population animal="lion">80</population>

<population>
    <animal>lion</animal>
    <quantity>80</quantity>
</population>
```

- Attribute needs to be declared in DTD so that it can be used

```
<!ELEMENT population (#PCDATA)>
<!ATTLIST population year CDATA #IMPLIED>
```

Attribute (continued)

- Required attribute

```
<!ELEMENT population (#PCDATA)>
<!ATTLIST population year (2000 | 2001) #REQUIRED>
```

  - Element population has to have attribute year whose value 2000 or 2001

```
<population year="2000">80</population>
```

- Default attribute

```
<!ELEMENT population (#PCDATA)>
<!ATTLIST population year CDATA "2000">
```

Attribute (continued)

- Default attribute

```
<!ELEMENT population (#PCDATA)>
<!ATTLIST population year CDATA #FIXED "2000">
```

  - Invalid `<population year="2001">80</population>`
  - Valid `<population year="2000">80</population>`
  - Valid `<population>80</population>`

- Unique attribute

```
<!ELEMENT animal (name)>
<!ATTLIST animal code ID #REQUIRED>
```

DTD declaration

- Internal

```
<?xml version="1.0"?>
<!DOCTYPE animal [ ]>
<animal> </animal>
```

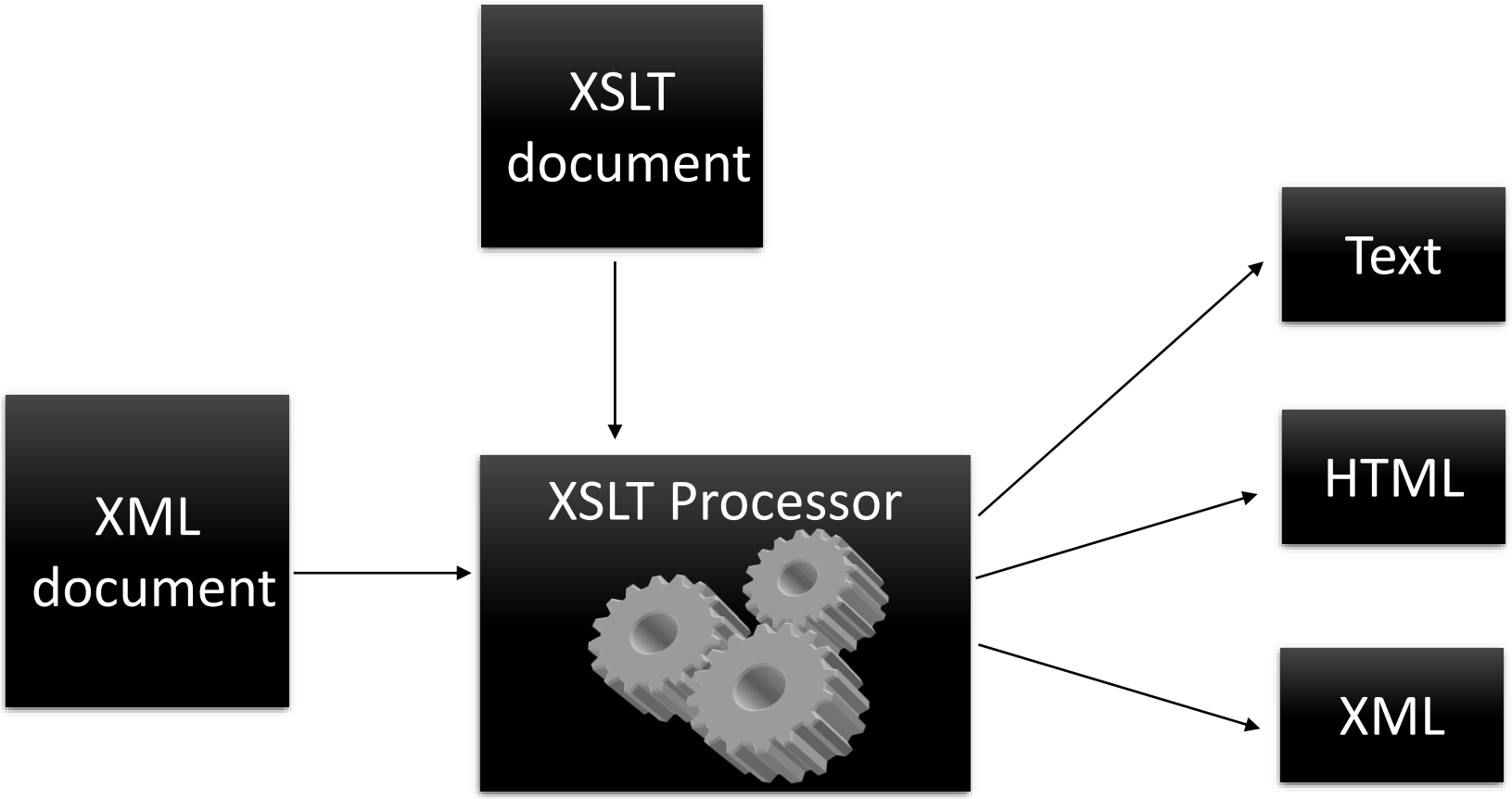
  - animal is the root element
  - The DTD lies inside “[ ]” brackets at “DOCTYPE”
- External

```
<?xml version="1.0"?>
<!DOCTYPE animal SYSTEM "http://www.animals.com/xml/animal.dtd">
```

XSL-T

- eXtensible Stylesheet Language, T = Transformation
- A standard by W3C
- A procedure for transforming an XML document format to other document format

XSL Transformation



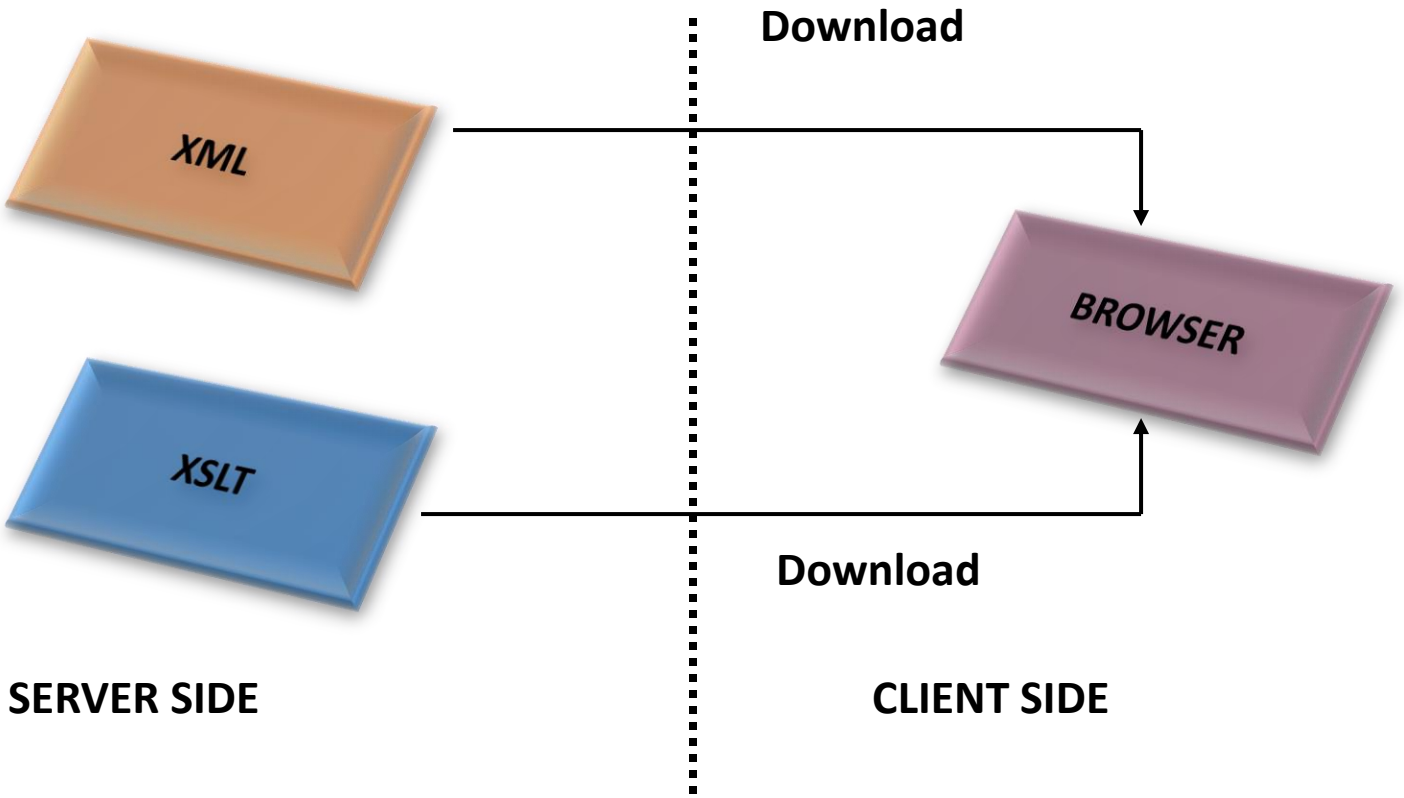
XML query data with XSLT

- As the data storage → data extraction mechanism needed
- Many way for data extraction → XSLT, the simplest one
- XSLT at first for presenting XML document → has been developed so that it can be used to query XML data

XML query with XSLT: the advantages

- Query can be done in client-side → alleviate server's works
- In addition being able for querying data, XSLT has also been able for presenting the data
- XSLT is XML document → has universal property

XML query data with XSLT: the principle



PHP: XML accessing

clients.xml

```xml
<clients>
    <client>
        <name>John Doe</name>
        <desc>Private banking</desc>
    </client>
    <client>
        <name>Barbara Smith</name>
        <desc>Business</desc>
    </client>
</clients>
```

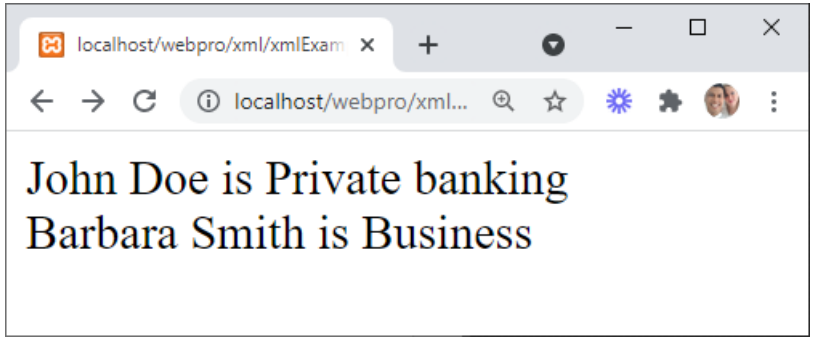
xmlExample1.php

```php
<?php
    $dom = new DomDocument();
    $dom -> load("clients.xml");

    $clients = $dom -> getElementsByTagName("client");
    foreach ($clients as $client) {
        $name_node = $client -> getElementsByTagName("name");
        $name = $name_node -> item(0) -> textContent;

        $desc_node = $client -> getElementsByTagName("desc");
        $desc = $desc_node -> item(0) -> textContent;

        echo "$name is $desc<BR>";
    }
?>
```



xmlExample2.php

```php
<?php
    $clients = simplexml_load_file("clients.xml");
    foreach ($clients -> client as $client) {
        echo $client -> name . " is " . $client -> desc . "<BR>";
    }
?>
```