M9 L1
XML DTD and HTML5

Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Outline

- XML Type Definition and Validation
- Document Type Definition (DTD)
- HTML5 as a Case Study

Why Do We Need Type and Schema Definition?

What is a well-formed XML document?
- Unique root
- Tags may not overlap
- Other restrictions on characters and words

```
<time>
        <hour>12</hour>
        <minute>72</minute>
        <second>-5</second>
</time>
```

Is this a well-formed XML element?

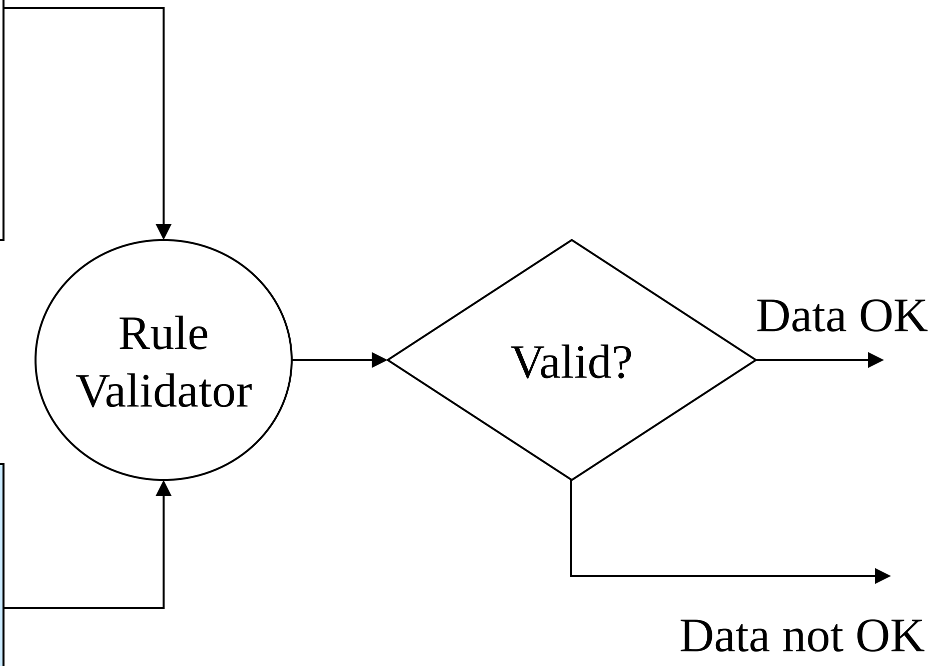
```
<time>
        <hour>18</hour>
        <minute>59</minute>
        <second>2</second>
</time>
```

Is this instance OK?

Apply Business Rules to Validate Data

```
<time>
        <hour>18</hour>
        <minute>59</minute>
        <second>2</second>
</time>
```

XML



hour: integer between 1 and 12
minutes: integer between 0 and 59
second: integer between 0 and 59

Rules

- DTD
- XML Schema

XML Document Type Definition (DTD)

- A piece of information can be encoded in an XML document in different ways.
- When the information is transferred from the source to its destination, the receiver needs to know how the document is structured and need to check if the content is indeed compliant with the structure.
- DTD provides the organization and rules for the XML document instance, or it holds information about the structure or the grammar of an XML document.
- A DTD file can be defined in a separate file (external/global DTD) or within the XML document itself (internal/local DTD).

Syntax Definition of DTD

- <!DOCTYPE *root-element*
  [
      doctype-declarations
  ]>
  It consists of the *name of the root element* and a list of *document type declarations*, *each of which is:*
  - <!ELEMENT *element-name content-model*>
    defines a pattern that associates a *content model* to **an element** of the given name;
    - Content-model: You can define your content model to allow:
      - EMPTY: no content is allowed for the element
      - ANY: any content is allowed for the element (CDATA)
      - (#PCDATA | *element-name* | ...)*: parsed character data

Syntax Definition of DTD (contd.)

- **Expression over element names when defining an element:**
  - choice: (a | b | c)
  - sequence: (a, b, c)
  - optional: a?
  - zero or more: b*
  - one or more: c+

Example of DTD within an XML file

```
<?xml version = ‘1.0’ encoding=‘utf-8’>
<!DOCTYPE instructor [
<!ELEMENT instructor (name, (course+), OfficeHours, (phone | email))>
<!ELEMENT name (first, (middle?), last)>
<!ELEMENT course (#PCDATA)>
<!ELEMENT OfficeHours (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT middle (#PCDATA)>
<!ELEMENT last (#PCDATA)>
]>
<instructor>
      <name>
            <first>Yinong</first>
            <last>Chen</last>
      </name>
      <course>Distributed Software Development</course>
      <course>Introduction to Programming Languages</course>
      <officeHours>4</officeHours>
      <email>yinong@asu.edu</email>
</instructor>
```

DTD File defining the rules

XML file (Instance) complies to the DTD file

, sequence

+ at least one

? optional

| OR

Does not support “integer”

Example of Using an External DTD file

```
<!DOCTYPE instructor [
<!ELEMENT instructor (name, (course+), OfficeHours, (phone | email))>
<!ELEMENT name (first, (middle?), last)>
<!ELEMENT course (#PCDATA)>
<!ELEMENT OfficeHours (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT middle (#PCDATA)>
<!ELEMENT last (#PCDATA)>
]>
```

instructor.dtd

```
<?xml version = ‘1.0’ encoding=‘utf-8’>
<!DOCTYPE instructor SYSTEM “http://venus.sod.asu.edu/WSRepository/xml/instructor.dtd”>
<instructor>
    <name>
        <first>Yinong</first> <last>Chen</last>
    </name>
    <course>Distributed Software Development</course>
    <officeHours>4</officeHours>
    <phone>480-965 2769</phone>
</instructor>
```

Defining Attributes for an Element

instructor.dtd

```
<!EMEMENT course (#PCDATA)>
<!ATTLIST course   level     CDATA    #REQUIRED
                   text      CDATA    #IMPLIED
                   campus    CDATA    “Tempe”
>
```

Default value if not provides

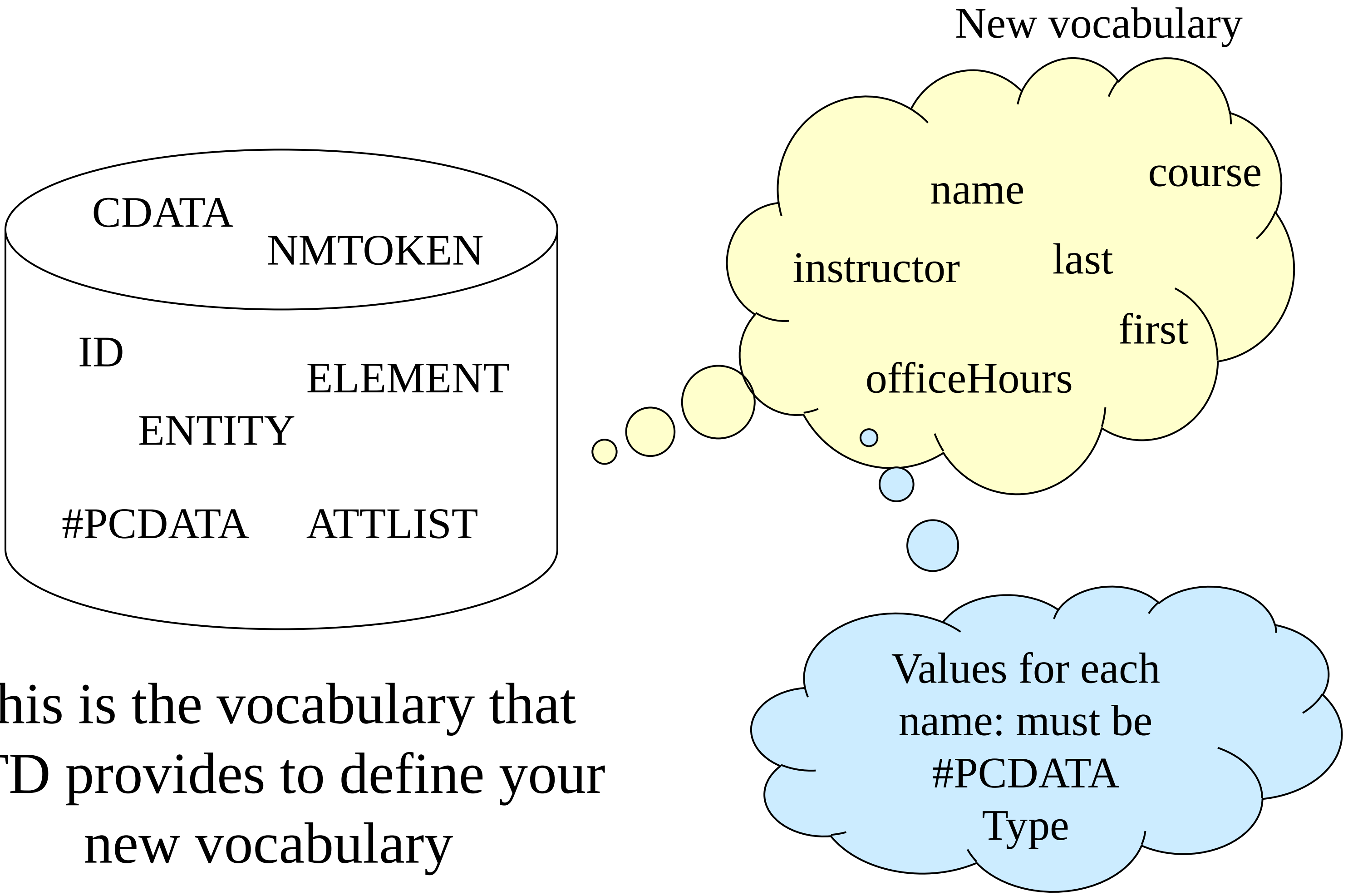
Attribute is optional, no default provided

```
<course
    level = “senior”
    text = “Service-Oriented Architecture & Computing”
    campus = “Downtown” >
    Distributed Software Development
</course>
```

Any data allowed

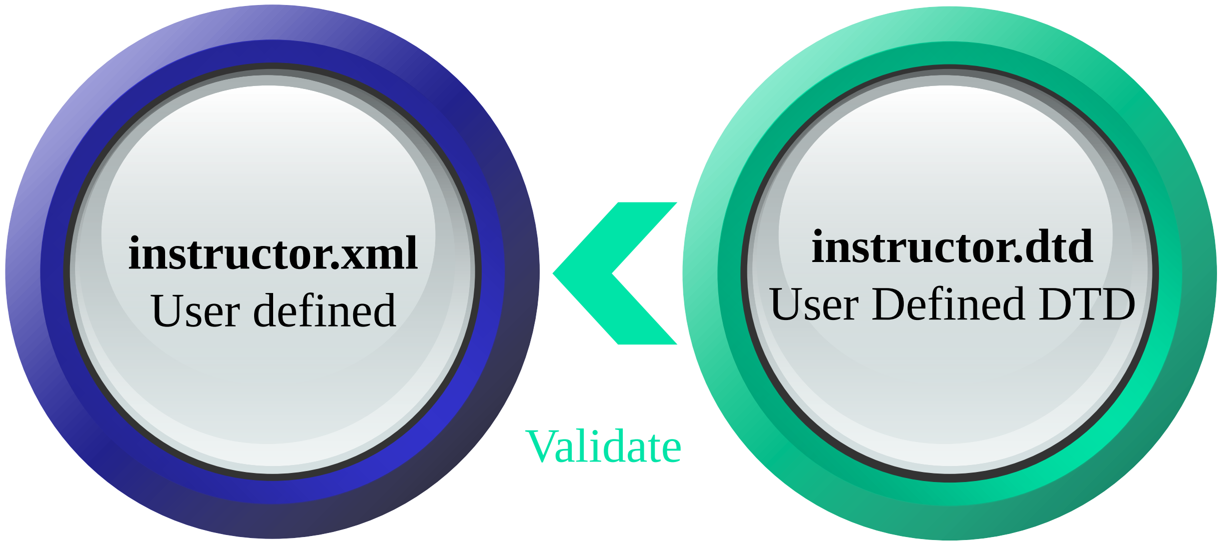
Must be PCDATA, see previous page

DTD Vocabulary (Namespace)



This is the vocabulary that DTD provides to define your new vocabulary

XML Validation Using DTD



The xml document must conform with the rules described in instructor.dtd

Case Study: HTML5 Definition

Reference: http://www.w3schools.com/html/html5_intro.asp

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> Doc  Title </title>
</head>

<body>
Content of the document......
</body>

</html>
```

- New **semantic** elements like <header>, <footer>, <article>, and <section>.
- New form **control attributes** like number, date, time, calendar, and range.
- New **graphic** elements like: <svg> and <canvas>.
- New **multimedia** elements like: <audio> and <video>.

+

JavaScript and JQuery API Calls

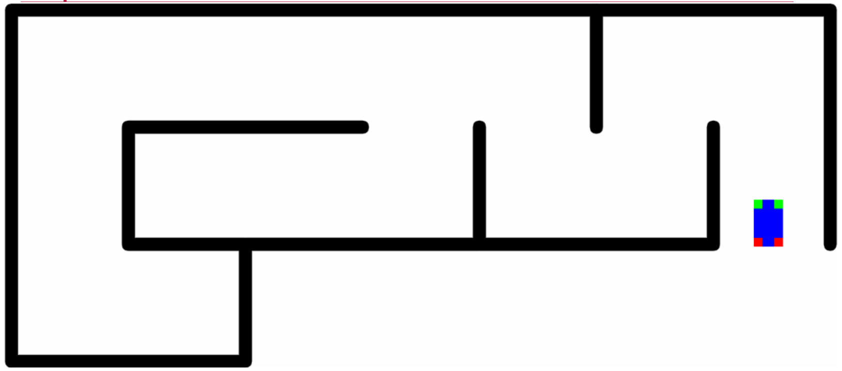
Become a Web Application Development Language

HTML5 JavaScript API Library

Reference: http://html5index.org/

- Web communication protocols
  - Web Sockets
  - Messaging, and
  - WebRTC
    (Web Real-Time Communication)
- Drag and Drop, Fullscreen
- Canvas, SVG, WebGL
- Animation Timing, Media, Pointer Lock, Web Audio
- File API, File System API, Indexed DB, Offline, Web Storage
  Offline and Storage
- Browser, Shadow DOM, Typed Arrays, Web Workers
- DOM and JQuery
- CSS Object Model, Selectors

ASU VIPLE Web Simulator:
https://venus.sod.asu.edu/VIPLE/Web2DSimulator/

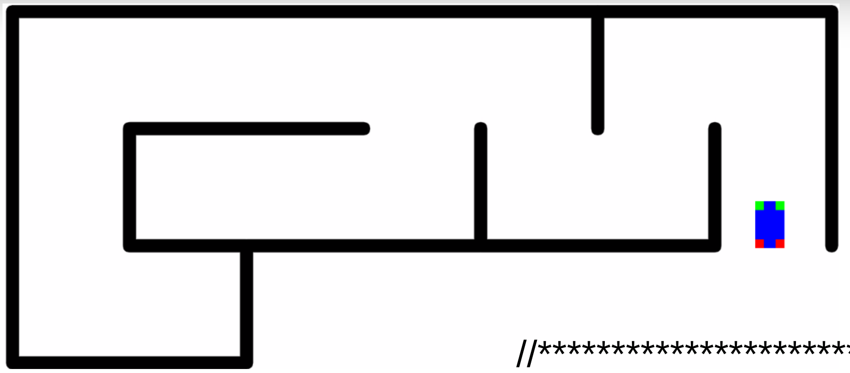


Restricted DOM for Web components

Read text section 4.6.5 for a programming example.

ASU VIPLE Web Simulator:

https://venus.sod.asu.edu/VIPLE/Web2DSimulator/



```
// Right-Click, View Page Source
// Code JavaScript behind the HTML5 GUI

//**********************************************************
// Start of Drawing Functions
//**********************************************************
function drawMazeAndRectangle(rectX, rectY) {
    makeWhite(0, 0, canvas.width, canvas.height);
    mazeImg.onload = function() {
        context.drawImage(mazeImg, 0, 0);

        // Draws robot to the canvas at current (x,y) coordinate
        drawRectangleToRobotCanvas();
    };;
}
function drawRobot2() {
    twoRobots = true;
    var robot2 = document.getElementById("robot2");
    var robot2Context = robot2.getContext("2d");
    robot2Context.clearRect(0, 0, canvas.width, canvas.height);
    robot2MiddleX = currRectX2 + rWidth / 2;
    robot2MiddleY = currRectY2 + rHeight / 2;
```

WebGL, Canvas, or SVG? Choose the right API

https://msdn.microsoft.com/en-us/library/dn265058%28v=vs.85%29.aspx?f=255&MSPPError=-2147217396

- Canvas is the typical choice for most HTML5 games. It's simple and speedy, particularly for games with many objects. ASU VIPLE Web Simulator uses Canvas.
- SVG: While Canvas provides simplicity and speed, SVG provides flexibility. For example: Each graphic object is part of the DOM tree for flexible access, and each graphic object in the game can have one or more associated event handlers for process.
- WebGL: If you're already familiar with OpenGL, you *can* use WebGL for a simple 2D game. If you're not, this API is more complex than you need for a simple game.

Ira A. Fulton Schools of Engineering

Arizona State University

M9 L2
XML Schema

Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Outline

- **XML Schema Definition (XSD)**
- **XML Type Definition**
  - Simple Types
  - Complex Types
- **XML Namespace**
- **Case Study: Using XML Schema to Define a New Language**

What is XML Schema and Why?

- Like DTD, XML Schema defines an XML vocabulary for expressing your data's business rules
- XML Schema is a more powerful alternative to DTD that defines the structure (grammar) of an XML document. The advantages of the schema over DTD include:
  - DTD is not XML-based and requires separate tools for processing DTD documents, while XML Schema is XML-based. The same tools can be used for parsing and processing XML documents, as well as XML Schema documents.
  - XML Schema is extendable and reusable. One can easily add new types, restricting existing types, and combine existing schemas.
  - DTD is limited to string type of data. XML schema can define many different types similar to a typical programming language, such as integer, float, string, and structures.

XML Schema Definition (XSD)

- In an XML document, the schema is introduced through an element with an opening tag <xsd:schema>, which is the root element, and
- with a number of optional attributes, e.g., referring to the source and version:

  <schema xmlns ="http://www.w3.org/2000/10/XMLSchema" version = "1.0">

- This schema namespace includes elements and datatypes that can be used for constructing different schemas:
  - schema
  - element
  - complexType
  - sequence
  - boolean
  - integer
  - string

Types Defined in XML Schema W3C 2001

- Forty-four (44) built-in types, in which 19 are simple (primitive) and 25 are derived complex types.
- User can define further types of simple and complex
- Complex types are defined using simple and complex types.
- Type inheritance: Extend one type from an existing type.

Definition of New Simple Types

- Use an element to define a new type with a name and an associated type
- Use a single built-in type to define a new type
- Use a namespace to tell where the element and type used come from.

namespaces

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="Title" type="xsd:string"/>
    <xsd:element name="Author" type="xsd:string"/>
    <xsd:element name="Date" type="xsd:date"/>
    <xsd:element name="ISBN" type="xsd:string"/>
    <xsd:element name="Publisher" type="xsd:string "/>
</xsd:schema>
```

Example: XML Schema of A Bookstore

root

namespaces

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace= "http://venus.sod.asu.edu/WSRepository/xml"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Title" type="xsd:string"
                                minOccurs="1" maxOccurs="1"/>
        <xsd:element name="Author" type="xsd:string"
                         minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="Year" type="xsd:integer"
                                minOccurs="0" maxOccurs="1"/>
        <xsd:element name="ISBN" type="xsd:string"
                                minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Bookstore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xs:attribute name= “Address" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name= "First" type="xsd:string"/>
  <xsd:element name= "Last" type="xsd:string"/>
</xsd:schema>
```

complexType

complexType

simpleType

Definition of Complex Types

- Use an element to define a new type with a name;
- Use sub elements to define the components of the new type;
- Specify the allowed number of occurrences of each component;
- Use namespaces to tell where the used element and types come from;
- Use different kinds of combination options.
  - Sequence
  - Choice
  - all

Definition of complexType

You can define the members of a complexType using three different options:

- *sequence*: All members must appear in the given order
- *all*: All members must appear (unless minOccurs= "0"), but can be in any order
- *choice*: Only one of the list members is allowed

```
<xsd:element name="Book">
    <xsd:complexType>
      <xsd:    all        >
            <xsd:element name="Title" minOccurs="1" maxOccurs="1"/>
            <xsd:element name="Author" minOccurs="1" maxOccurs="unbounded"/>
            <xsd:element name=“Year" minOccurs= "0" maxOccurs="1"/>
            <xsd:element name="ISBN" minOccurs= "0" maxOccurs="1"/>
      </xsd:    all        >
    </xsd:complexType>
```

Choice of complexType

```
<xsd:element name="Book">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Title" />
            <xsd:element name="Author" />
            <xsd:element name=“Year" />
            <xsd:choice>
                <xsd:element name="ISBN-10" />
                <xsd:element name="ISBN-13" />
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element name="Book">
```

Namespaces

- A namespace is declared as an attribute of an element.
- It binds a prefix name (qualifier) to a schema definition, and then uses that prefix wherever required;
- It is not mandatory to declare namespaces at the root element; Namespaces could be declared at any element in the XML document;
- The scope of a declared namespace applies to the entire content of that element;
- Namespace can be overridden: An inner namespace declaration with the same prefix name (qualifier) will override the outer namespace.

Namespaces

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://venus.sod.asu.edu/WSRepository/xml"
      xmlns="http://venus.sod.asu.edu/WSRepository/xml/bookstore.xsd"
         elementFormDefault="qualified"
         attributeFormDefault="unqualified">
```

- schema
- element
- complexType
- sequence
- boolean
- integer
- string

Inherit all names from XMLSchema, plus types being defined in this schema

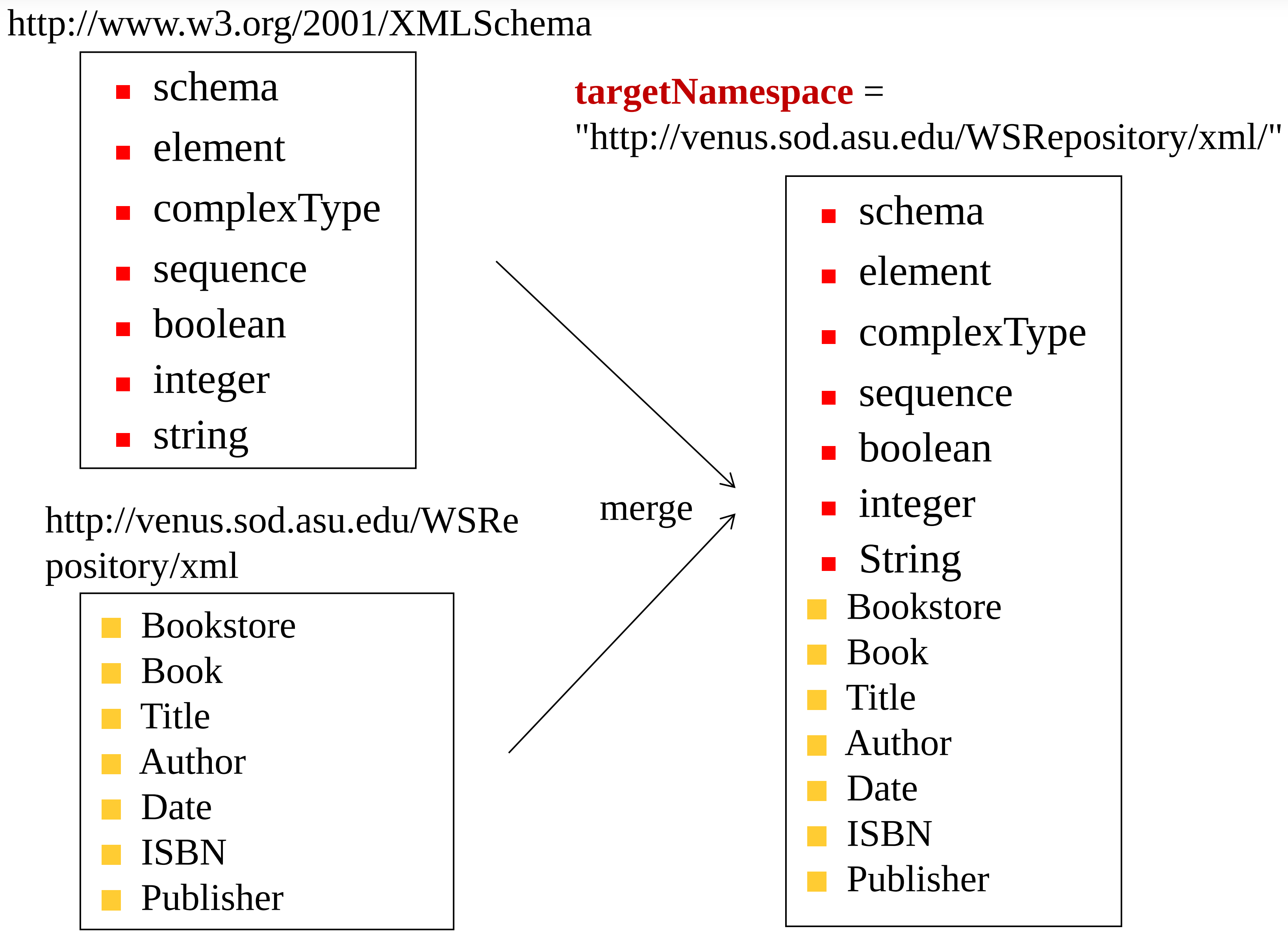
- Bookstore
- Book
- Title
- Author
- Date
- ISBN
- Publisher

4 combinations

This is a directive to any instance documents which conform to this schema: Any **elements** used by the instance document must be namespace qualified. But the **attributes** do not need to be qualified

Define this namespace as the **default** namespace. The names in the default namespace do not need to be qualified

Target Namespace



http://www.w3.org/2001/XMLSchema

- schema
- element
- complexType
- sequence
- boolean
- integer
- string

http://venus.sod.asu.edu/WSRepository/xml

- Bookstore
- Book
- Title
- Author
- Date
- ISBN
- Publisher

merge

targetNamespace = "http://venus.sod.asu.edu/WSRepository/xml/"

- schema
- element
- complexType
- sequence
- boolean
- integer
- String
- Bookstore
- Book
- Title
- Author
- Date
- ISBN
- Publisher

Default Namespace

- It is painful to repeatedly qualify an element or attribute you wish to use from a namespace;
- You can declare one default namespace;
- Important: at any point in time, there can be one default namespace only;
- Declaring a default namespace means that any element within the scope of the namespace will be qualified implicitly, if it is not already qualified explicitly using a prefix.
- The name of the default namespace is simply **xmlns**

```
xmlns = "http://venus.sod.asu.edu/WSRepository/xml/bookstore.xsd"
```

Choice of the default namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
          targetNamespace="http://venus.sod.asu.edu/WSRepository/xml/"
          xmlns:bs="http://venus.sod.asu.edu/WSRepository/xml/bookstore.xsd"
             elementFormDefault="qualified" attributeFormDefault="unqualified">
```

Define the XML Schema as the default namespace. The names in the default namespace do not need to be qualified

The bookstore namespace was the default, and it is no longer default now. The elements defined in it must be qualified by “bs”.

Example: XML Schema of A Bookstore

No default namespaces

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace= "http://venus.sod.asu.edu/WSRepository/xml/"
            elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:element name="Bookstore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
     <xsd:complexType>
       <xsd:sequence>
         <xsd:element name="Title" type="xsd:string" minOccurs="1" maxOccurs="1"/>
         <xsd:element name="Author" type="xsd:string" minOccurs="1" maxOccurs="unbounded"/>
         <xsd:element name="Year" type="xsd:string" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="ISBN" type="xsd:string" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="Publisher" type="xsd:string" minOccurs="0" maxOccurs="1"/>
     </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="First" type="xsd:string"/>
  <xsd:element name="Last" type="xsd:string"/>
</xsd:schema>
```

Better Choice of Default Namespace

Default namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
                targetNamespace= "http://venus.sod.asu.edu/WSRepository/xml/"
               xmlns:bs ="http://venus.sod.asu.edu/WSRepository/xml/bookstore.xsd"
                     elementFormDefault="qualified" attributeFormDefault="unqualified">
    <element name="Bookstore">
        <complexType>
            <sequence>
                <element name="Book" minOccurs="1" maxOccurs="unbounded"/>
            </sequence>
        </complexType>
    </element>
    <element name="Book">
         <complexType>
            <sequence>
                <element name="Title" type="xsd:string" minOccurs="1" maxOccurs="1"/>
                <element name="Author" type="xsd:string" minOccurs="1" maxOccurs="unbounded"/>
                <element name="Year" type="xsd:string" minOccurs="0" maxOccurs="1"/>
                <element name="ISBN" type="xsd:string" minOccurs="0" maxOccurs="1"/>
            </sequence>
        </complexType>
    </element>
    <element name="First" type="string"/>
    <element name="Last" type="string"/>
    <element name="Size" type="bs:DIN"/>
</schema>
```

Complex Type

Complex Type

Simple Type

Using a Schema in an Instance

xmlns:
Default XML
namespace

xsi:
XML
Schema
Instance

```
<?xml version="1.0"?>
<Bookstore xmlns ="http://venus.sod.asu.edu/WSRepository/xml/bookstore.xsd"
        xmlns:bs ="http://venus.sod.asu.edu/WSRepository/xml/bookstore.xsd"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://venus.sod.asu.edu/WSRepository/xml/bookstore.xsd">
    <Book>
        <Title>Introduction to Programming Languages</Title>
        <Author>Yinong Chen</Author>
        <Author>Gennaro De Luca</Author>
        <Year>2022</Year>
        <ISBN> 978-1-7924-9416-1</ISBN>
        <Size>10x8</Size>
    </Book>        ...
</Bookstore>
```

xsi
is used to
define the
location

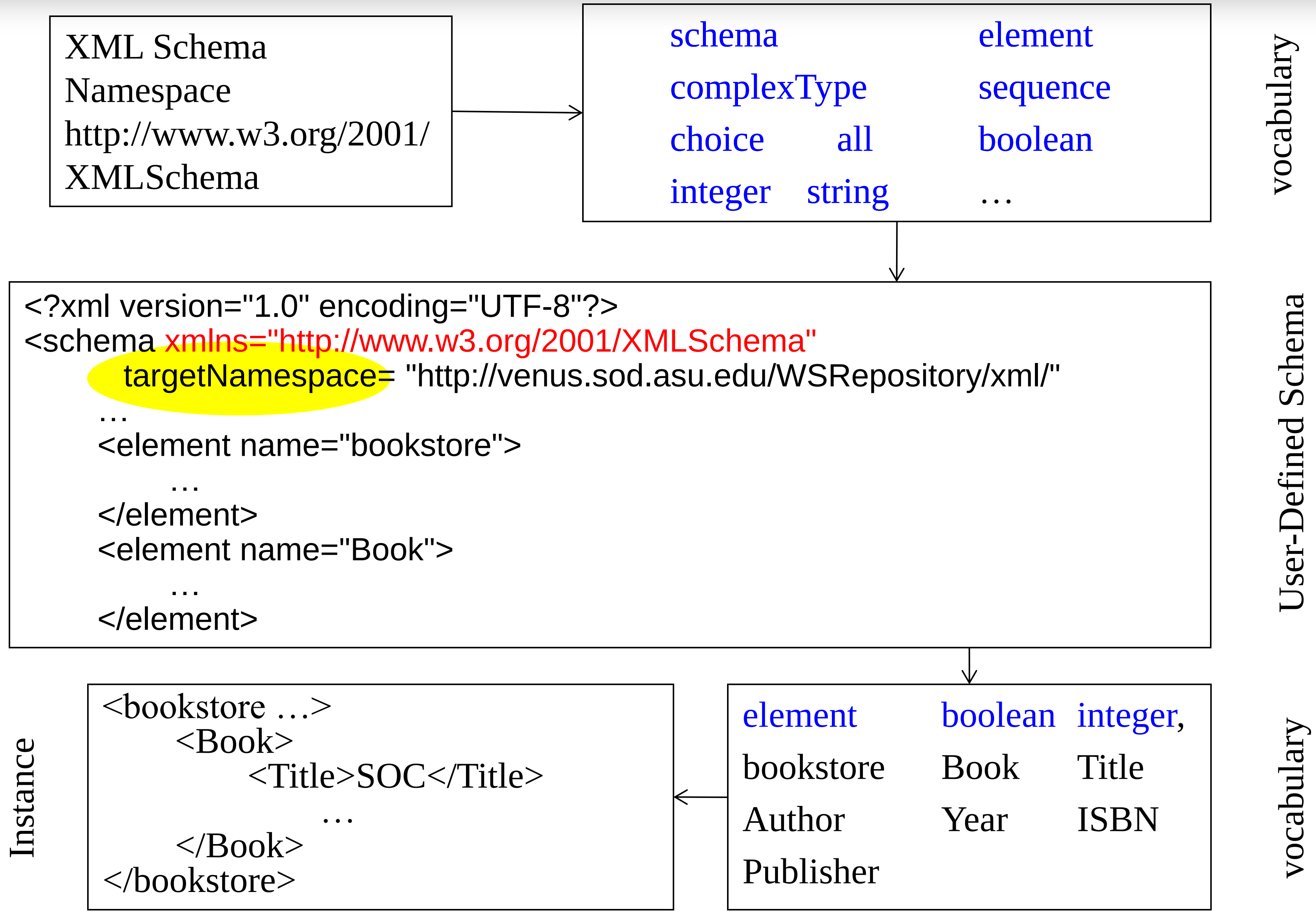
Default Namespace and Attributes

- Prefixed namespace and the default namespace **do not** apply to attributes within the element;
- To apply a namespace to an attribute, the attribute must be explicitly qualified.
- In the example below, the attribute "edition" has no namespace, whereas the attribute "cover" is associated with the namespace bs.

Default namespace

```
<Book  xmlns = "http://venus.sod.asu.edu/WSRepository/xml/bookstore.xsd">
        <Title  bs: cover = "paperback" > Programming Languages </Title>
        <Author>Yinong Chen</Author>
        <Author> Gennaro De Luca </Author>
        <Year edition = "2" >2022</Year>
        <ISBN> 978-1-7924-9416-1</ISBN>
</Book>
```

Summary of XML Schema Example



XML Schema Namespace http://www.w3.org/2001/XMLSchema

schema		element
complexType		sequence
choice	all	boolean
integer	string	…

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace= "http://venus.sod.asu.edu/WSRepository/xml/"
        …
        <element name="bookstore">
                …
        </element>
        <element name="Book">
                …
        </element>
```

```
<bookstore …>
        <Book>
                <Title>SOC</Title>
                        …
        </Book>
</bookstore>
```

element	boolean	integer,
bookstore	Book	Title
Author	Year	ISBN
Publisher		

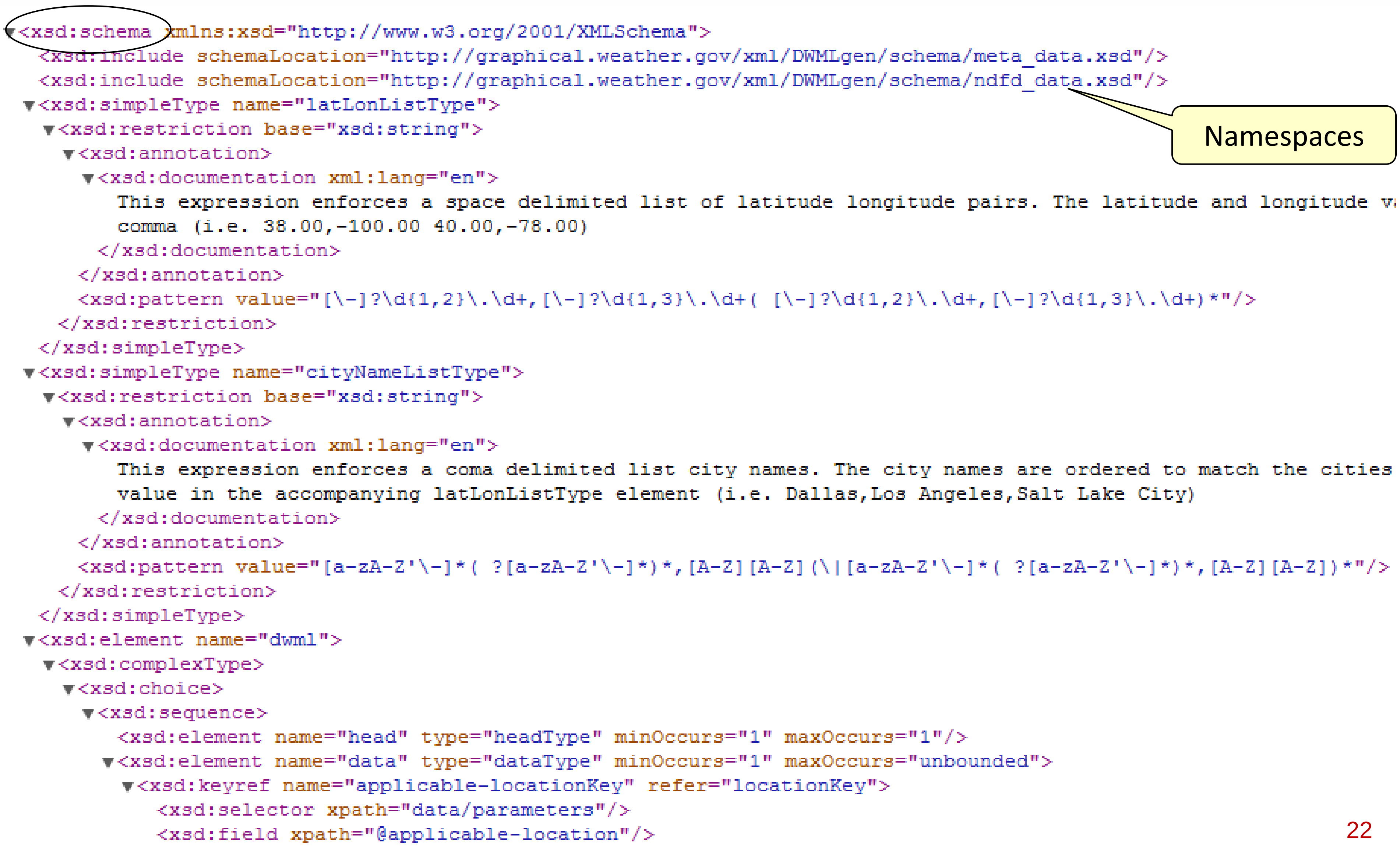
Case Study

https://vlab.noaa.gov/web/mdl/ndfd-web-services
https://graphical.weather.gov/xml/SOAP_server/ndfdXML.htm

- Digital Weather Markup Language (DWML)
- DWML is an XML language for supporting the exchange of the National Weather Service's (NWS) National Digital Forecast Database (NDFD) data;
- Also used in other environmental science applications. DWML site provides a definition of DWML types based on restrictions.
- Weather forecast services: http://www.weather.gov/, Enter City, State names
- 7-day forecast data downloadable in the format of http://forecast.weather.gov/MapClick.php?lat=33.43417&lon=-112.05111&FcstType=dwml

DWML XML Schema

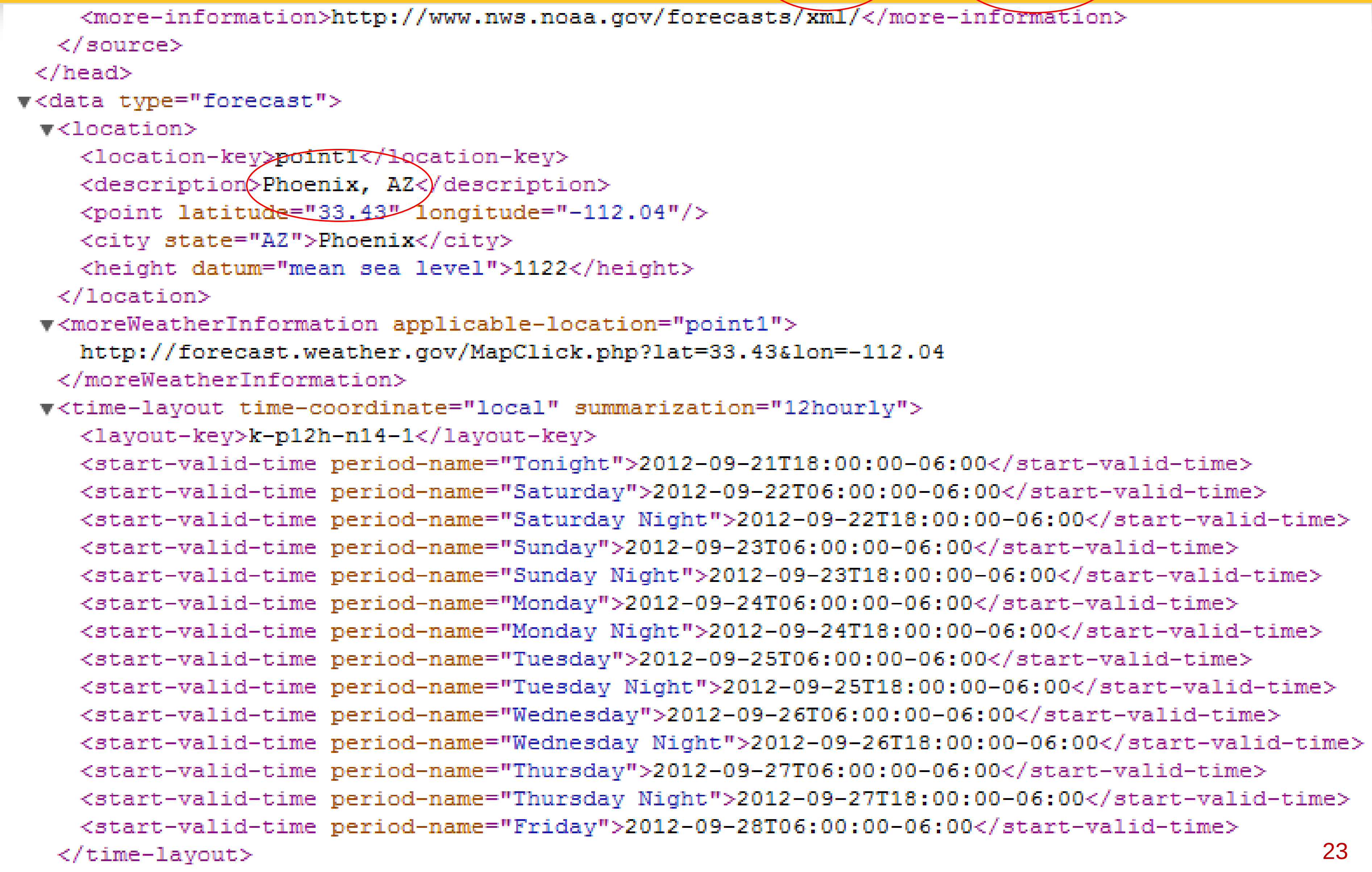
https://graphical.weather.gov/xml



```
▼<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:include schemaLocation="http://graphical.weather.gov/xml/DWMLgen/schema/meta_data.xsd"/>
   <xsd:include schemaLocation="http://graphical.weather.gov/xml/DWMLgen/schema/ndfd_data.xsd"/>
 ▼<xsd:simpleType name="latLonListType">
   ▼<xsd:restriction base="xsd:string">
     ▼<xsd:annotation>
       ▼<xsd:documentation xml:lang="en">
          This expression enforces a space delimited list of latitude longitude pairs. The latitude and longitude v
          comma (i.e. 38.00,-100.00 40.00,-78.00)
        </xsd:documentation>
      </xsd:annotation>
      <xsd:pattern value="[\-]?\d{1,2}\.\d+,[\-]?\d{1,3}\.\d+( [\-]?\d{1,2}\.\d+,[\-]?\d{1,3}\.\d+)*"/>
    </xsd:restriction>
  </xsd:simpleType>
 ▼<xsd:simpleType name="cityNameListType">
   ▼<xsd:restriction base="xsd:string">
     ▼<xsd:annotation>
       ▼<xsd:documentation xml:lang="en">
          This expression enforces a coma delimited list city names. The city names are ordered to match the cities
          value in the accompanying latLonListType element (i.e. Dallas,Los Angeles,Salt Lake City)
        </xsd:documentation>
      </xsd:annotation>
      <xsd:pattern value="[a-zA-Z'\-]*( ?[a-zA-Z'\-]*)*,[A-Z][A-Z](\|[a-zA-Z'\-]*( ?[a-zA-Z'\-]*)*,[A-Z][A-Z])*"/>
    </xsd:restriction>
  </xsd:simpleType>
 ▼<xsd:element name="dwml">
   ▼<xsd:complexType>
     ▼<xsd:choice>
       ▼<xsd:sequence>
          <xsd:element name="head" type="headType" minOccurs="1" maxOccurs="1"/>
         ▼<xsd:element name="data" type="dataType" minOccurs="1" maxOccurs="unbounded">
           ▼<xsd:keyref name="applicable-locationKey" refer="locationKey">
              <xsd:selector xpath="data/parameters"/>
              <xsd:field xpath="@applicable-location"/>
```

Instance of the Schema: Phoenix Seven-Day Weather Data Download

http://forecast.weather.gov/MapClick.php?lat=33.43417&lon=-112.05111&FcstType=dwml



```
    <more-information>http://www.nws.noaa.gov/forecasts/xml/</more-information>
  </source>
</head>
▼<data type="forecast">
  ▼<location>
    <location-key>point1</location-key>
    <description>Phoenix, AZ</description>
    <point latitude="33.43" longitude="-112.04"/>
    <city state="AZ">Phoenix</city>
    <height datum="mean sea level">1122</height>
  </location>
  ▼<moreWeatherInformation applicable-location="point1">
    http://forecast.weather.gov/MapClick.php?lat=33.43&lon=-112.04
  </moreWeatherInformation>
  ▼<time-layout time-coordinate="local" summarization="12hourly">
    <layout-key>k-p12h-n14-1</layout-key>
    <start-valid-time period-name="Tonight">2012-09-21T18:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Saturday">2012-09-22T06:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Saturday Night">2012-09-22T18:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Sunday">2012-09-23T06:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Sunday Night">2012-09-23T18:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Monday">2012-09-24T06:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Monday Night">2012-09-24T18:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Tuesday">2012-09-25T06:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Tuesday Night">2012-09-25T18:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Wednesday">2012-09-26T06:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Wednesday Night">2012-09-26T18:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Thursday">2012-09-27T06:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Thursday Night">2012-09-27T18:00:00-06:00</start-valid-time>
    <start-valid-time period-name="Friday">2012-09-28T06:00:00-06:00</start-valid-time>
  </time-layout>
```

To Get Forecast from the URL

http://forecast.weather.gov/MapClick.php?lat=33.43417&lon=-112.05111&FcstType=dwml

```
// The operation takes latitude and longitude as inputs, and obtain 7-day forecast
public void GetForecast(string latitude, string longitude) {
     // form the URI
    UriBuilder fullUri = new UriBuilder("http://forecast.weather.gov/MapClick.php");
    fullUri.Query = "lat=" + latitude + "&lon=" + longitude + "&FcstType=dwml";
    // initialize a new WebRequest: RESTful style service access
    HttpWebRequest forecastRequest =
                                    (HttpWebRequest)WebRequest.Create(fullUri.Uri);
    // set up the state object for the async request
    ForecastUpdateState forecastState = new ForecastUpdateState();
    forecastState.AsyncRequest = forecastRequest;
    // start the asynchronous request
    forecastRequest.BeginGetResponse(
            new AsyncCallback(HandleForecastResponse), forecastState);
}
```

Process the Forecast Response (XML File)

Asynchronous invocation

```
private void HandleForecastResponse(IAsyncResult asyncResult) {
    // get the state information
    ForecastUpdateState forecastState = (ForecastUpdateState)asyncResult.AsyncState;
    HttpWebRequest forecastRequest = (HttpWebRequest)forecastState.AsyncRequest;
    // end the async request
    forecastState.AsyncResponse = (HttpWebResponse)forecastRequest.EndGetResponse(asyncResult);
    Stream streamResult;
    string newCredit = ""; string newCityName = ""; int newHeight = 0;
    // create a temp collection for the new forecast information for each time period
    ObservableCollection<ForecastPeriod> newForecastList =
        new ObservableCollection<ForecastPeriod>();
    try {
        // get the stream containing the response from the async call
        streamResult = forecastState.AsyncResponse.GetResponseStream();
        XElement xmlWeather = XElement.Load(streamResult);  // load the XML
        // start parsing the XML.  You can see what the XML looks like by browsing to:
        // http://forecast.weather.gov/MapClick.php?lat=44.52160&lon=-87.98980&FcstType=dwml
        // find the source element and retrieve the credit information
        XElement xmlCurrent = xmlWeather.Descendants("source").First();
        newCredit = (string)(xmlCurrent.Element("credit"));
        // find the source element and retrieve the city and elevation information
        xmlCurrent = xmlWeather.Descendants("location").First();
        newCityName = (string)(xmlCurrent.Element("city"));
        newHeight = (int)(xmlCurrent.Element("height"));
        // find the first time layout element
        xmlCurrent = xmlWeather.Descendants("time-layout").First();
```

Read Text 10.4.5: LINQ to XML Processing

Ira A. Fulton Schools of Engineering

Arizona State University

M9 L3
XML Validation

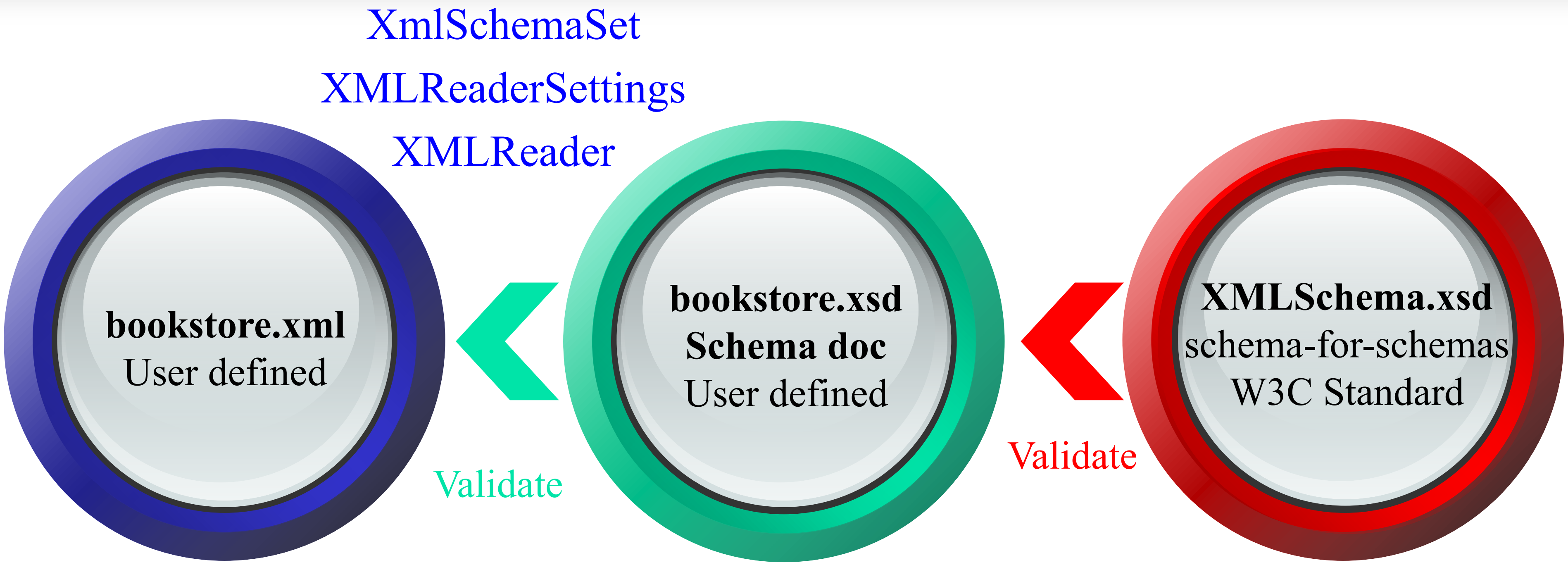
Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Outline

- XML Validation Classes
- XML Reader with Validation

Multiple Levels of Validation

http://validator.w3.org/



The xml instance document conforms with the rules described in bookstore.xsd and in W3C standard

bookstore.xsd is a valid schema document, i.e., it conforms with the rules described in the schema-for-schemas

XML Validation Using DTD and XSD

http://validator.w3.org/

bookstore.xml
User defined

XmlSchemaSet
XMLReaderSettings
XMLReader

Validate

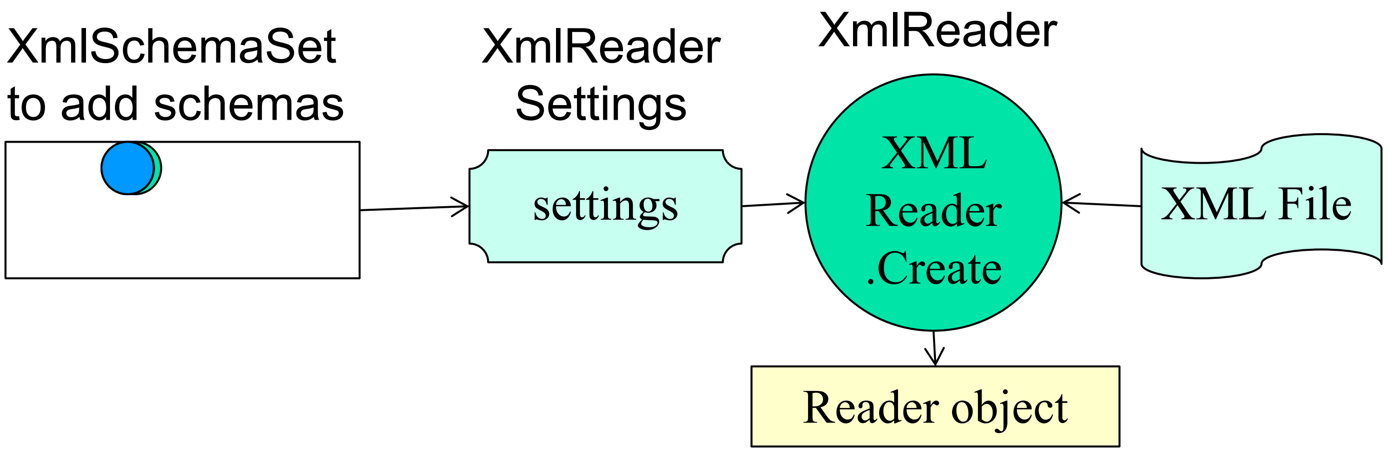
User defined
bookstore.xsd
bookstore.dtd

The xml instance document conforms with the rules described in bookstore.xsd or bookstore.dtd

Validation Classes in .Net Framework

- Three classes are used, and they support both DTD and XSD: XmlSchemaSet, XMLReaderSettings, XMLReader for Validation

1. Use XmlSchemaSet object to hold the schemas
2. Link the schema set to XMLReaderSettings object
3. Use XmlReader to create a reader object that associates the XML file to be validated with the XMLReaderSettings object



Using XmlSchemaSet Class to add schemas

```
using System; using System.Xml;
using System.Xml.Schema; using System.IO;

public class Sample {
    public static void Main() {
        // Create the XmlSchemaSet class.
        XmlSchemaSet sc = new XmlSchemaSet();
●       // Add the schema to the collection before performing validation
       sc.Add(null, "http://venus.sod.asu.edu/WSRepository/xml/Courses.xsd");
        // Define the validation settings.
        XmlReaderSettings settings = new XmlReaderSettings();
        settings.ValidationType = ValidationType.Schema;
        settings.Schemas = sc; // Association
```

Do not list .xsd file as a namespace of the .xml file.

Schema definition file

You could choose .DTD

settings

Using XmlReader.Create Method

```
    settings.ValidationEventHandler += new
      ValidationEventHandler(ValidationCallBack);
    // Create the XmlReader object.
    XmlReader reader =
     XmlReader.Create("http://venus.sod.asu.edu/WSRepository/xml/Courses.xml", settings);
    // Parse the file.
    while (reader.Read()) ;        // will call event handler if invalid
    Console.WriteLine("The XML file validation has completed");
  }
  // Display any validation errors.
  private static void ValidationCallBack(object sender, ValidationEventArgs e)
  {
    Console.WriteLine("Validation Error: {0}", e.Message);
  }
}
```

Add an event handler

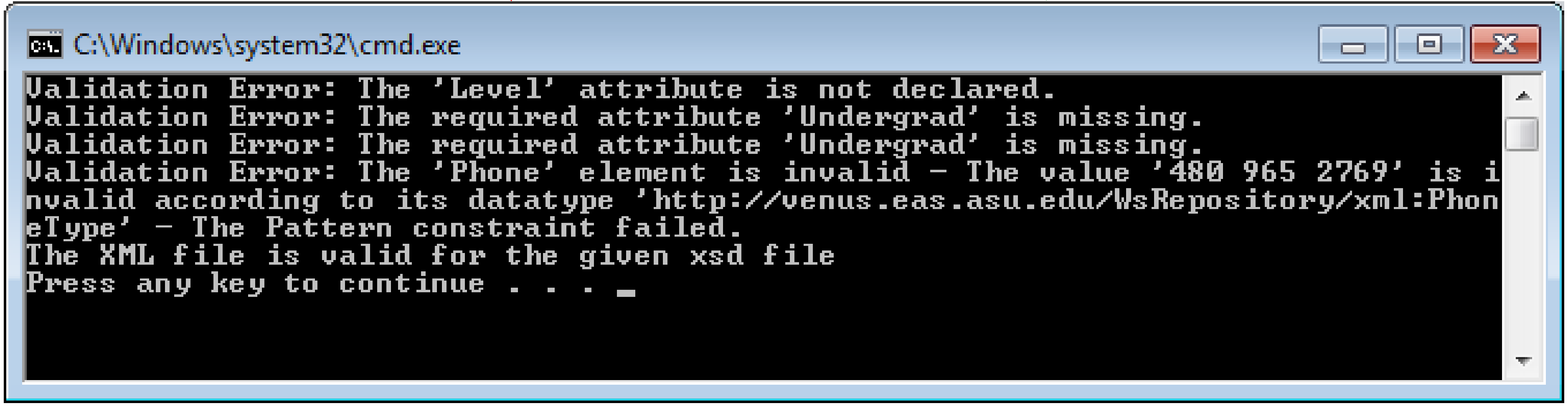
xmlCoursesWithErrors.xml

① XML instance file

② namespaces

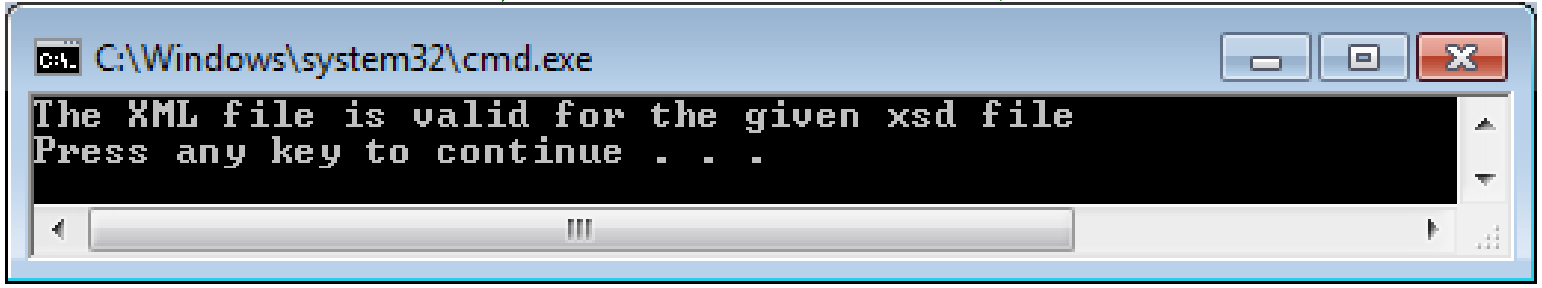
Output with Errors and Without Errors

CoursesWithErrors.xml Courses.xsd



```
Validation Error: The 'Level' attribute is not declared.
Validation Error: The required attribute 'Undergrad' is missing.
Validation Error: The required attribute 'Undergrad' is missing.
Validation Error: The 'Phone' element is invalid - The value '480 965 2769' is i
nvalid according to its datatype 'http://venus.eas.asu.edu/WsRepository/xml:Phon
eType' - The Pattern constraint failed.
The XML file is valid for the given xsd file
Press any key to continue . . . _
```

Courses.xml Courses.xsd



```
The XML file is valid for the given xsd file
Press any key to continue . . .
```

Ira A. Fulton Schools of Engineering

Arizona State University

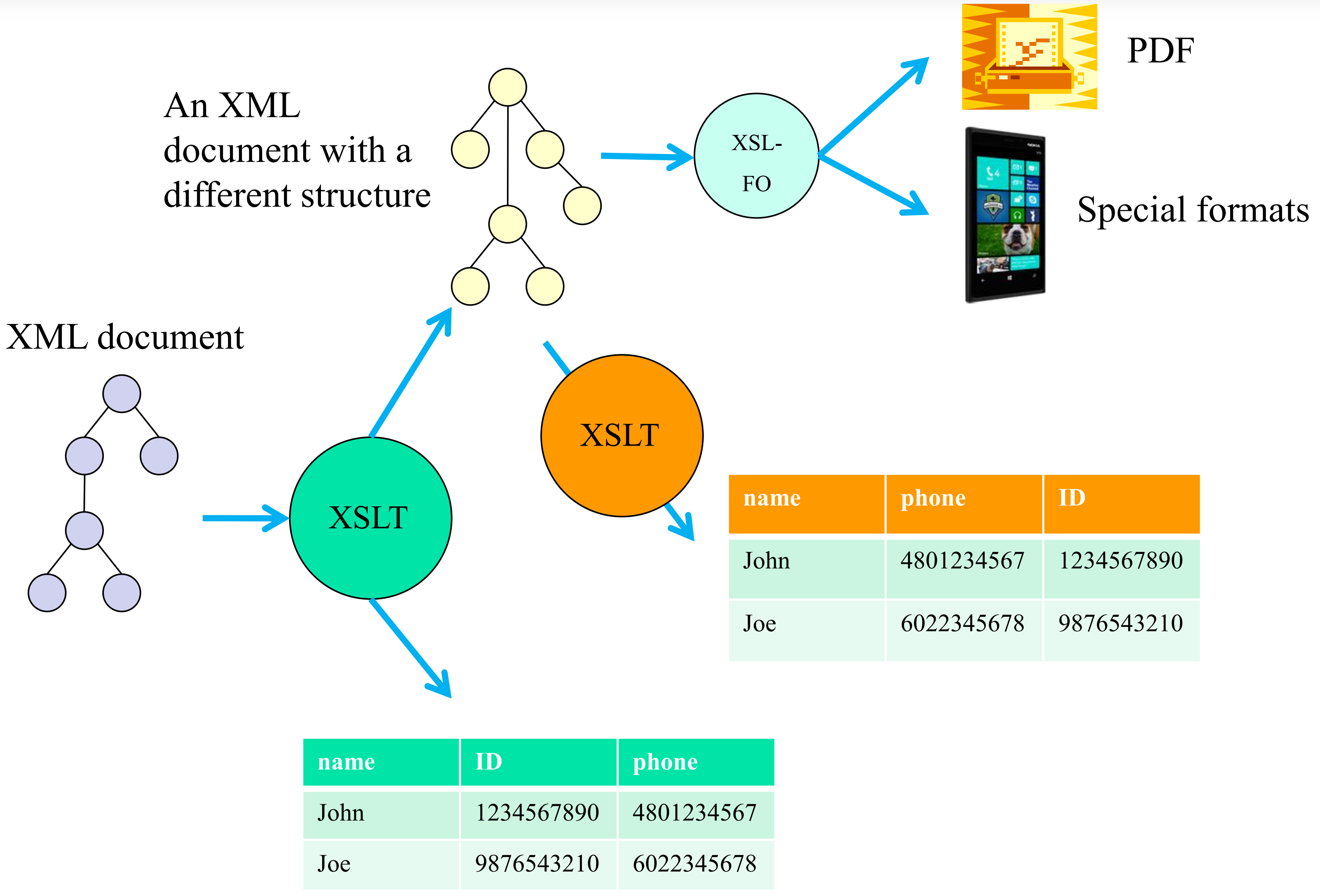
M9 L4
XSL:
Extensible Stylesheet Language and
XSLT:
XSL Transformations

Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Outline

- The Needs of Transformations
- XSL: Extensible Stylesheet Language
- Client-Side Transformation
- Server-Side Transformation
- CSS: Cascading Style Sheets

The Needs of Transformation: XSLT and XSL-FO



name	phone	ID
John	4801234567	1234567890
Joe	6022345678	9876543210

name	ID	phone
John	1234567890	4801234567
Joe	9876543210	6022345678

XSL: Extensible Stylesheet Language

http://www.w3.org/TR/xsl/

- HTML: "Format without Structure"
  - Typesetting language
  - Not extensible
- XML: "Structure Without Format"
  - Defines "elements" using "tags"
  - Creates hierarchical structure of information set
- XSL is a set of language technologies for defining XML document transformation and presentation
  - XSLT: XSL Transformations
  - XSL Formatting Objects or XSL-FO, is a markup language for XML document formatting which is most often used for generating PDFs
  - XSLT is a built-in component in ESB Enterprise Service Bus

XSLT: XSL Transformations

- XSL is a declarative (functional) programming language with constructs such as
  - if
  - choose-when-otherwise (switch) →
  - for-each

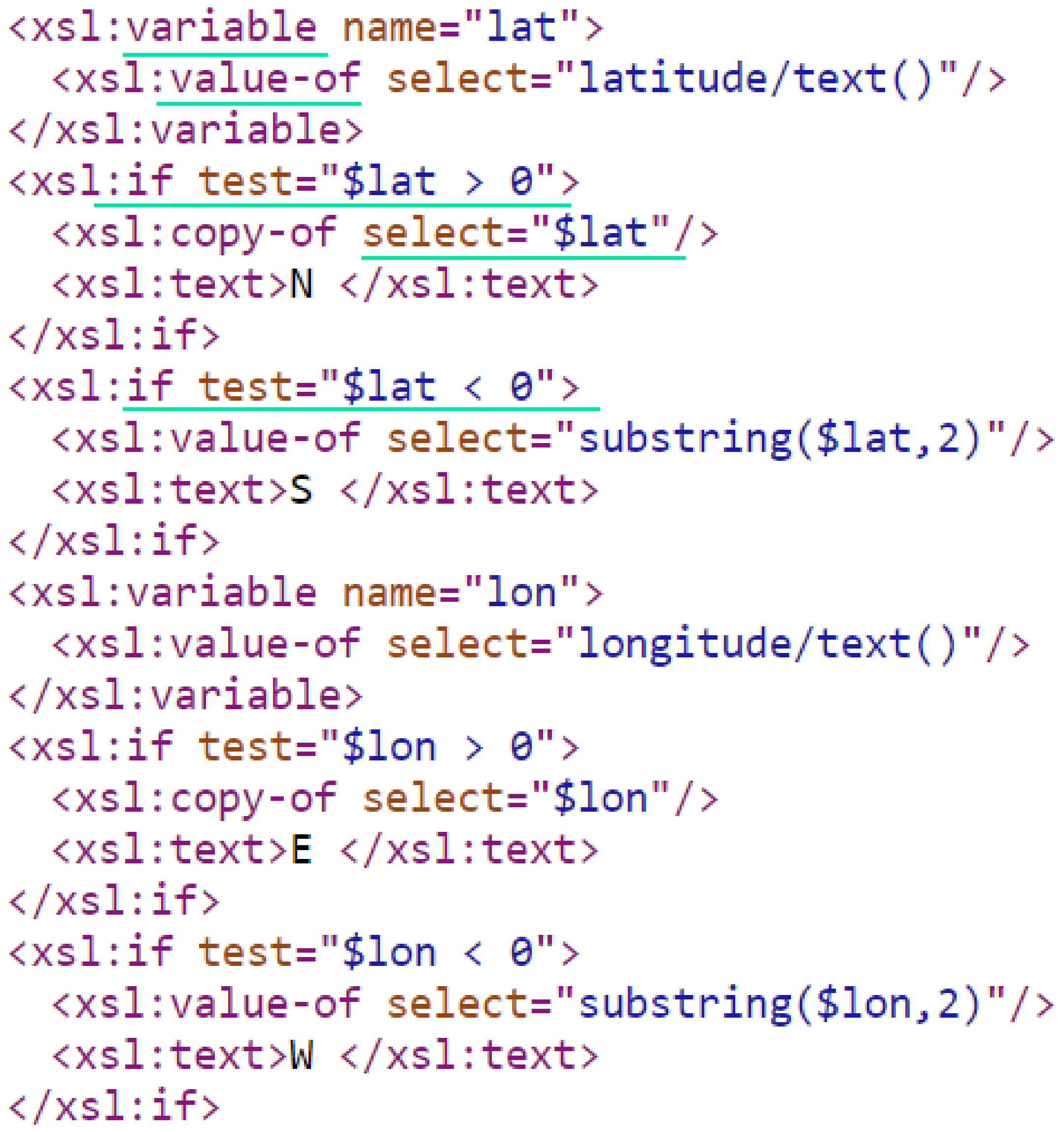
```
<xsl:choose>
          <xsl:when condition">
                    html statements
          </xsl:when>
          <xsl:when test="condition">
                    html statements
          </xsl:when>
          <xsl:otherwise>
                    html statements
          </xsl:otherwise>
</xsl:choose>
```

- XSL is Turing complete in theory, and thus can be used for programming any task, but limited in practice
  - Creates formatted output

- Transforms one XML structure to another
- Transforms XML to HTML: Adding HTML Format to XML Structure

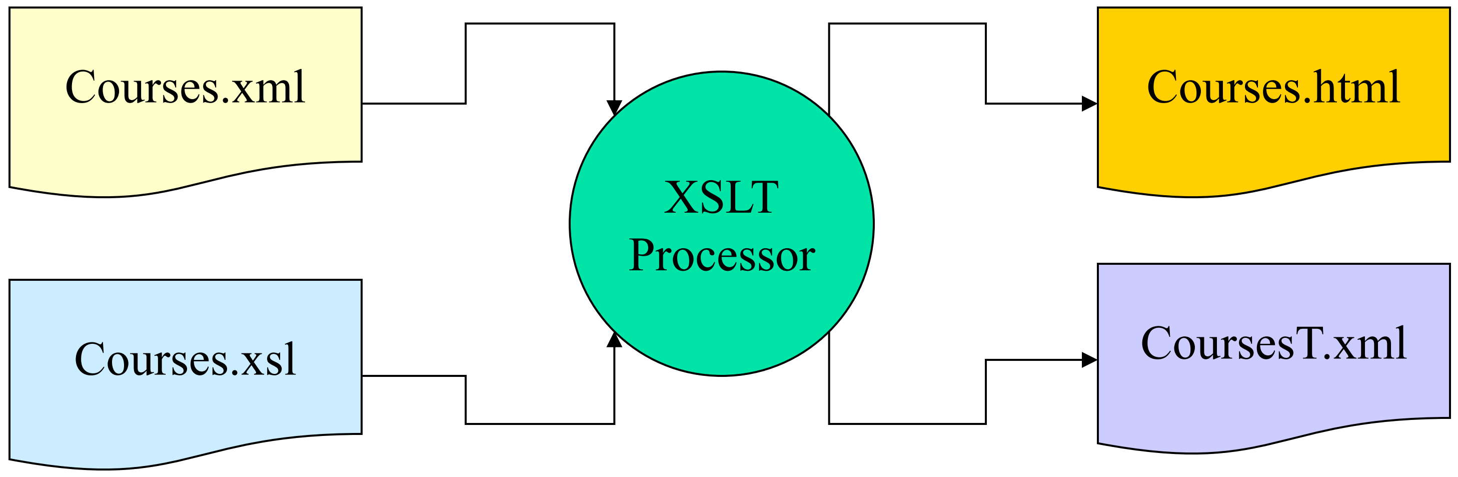
Weather.xsl that defines: variable and if

https://w1.weather.gov/xml/current_obs/latest_ob.xsl



```
<xsl:variable name="lat">
  <xsl:value-of select="latitude/text()"/>
</xsl:variable>
<xsl:if test="$lat > 0">
  <xsl:copy-of select="$lat"/>
  <xsl:text>N </xsl:text>
</xsl:if>
<xsl:if test="$lat < 0">
  <xsl:value-of select="substring($lat,2)"/>
  <xsl:text>S </xsl:text>
</xsl:if>
<xsl:variable name="lon">
  <xsl:value-of select="longitude/text()"/>
</xsl:variable>
<xsl:if test="$lon > 0">
  <xsl:copy-of select="$lon"/>
  <xsl:text>E </xsl:text>
</xsl:if>
<xsl:if test="$lon < 0">
  <xsl:value-of select="substring($lon,2)"/>
  <xsl:text>W </xsl:text>
</xsl:if>
```

XML Transformations Using a Stylesheet



Converting XML to HTML on Client (Browser)

- Put Courses.xml and Courses.xsl into a Website, e.g., in

  C:\Inetpub\wwwroot (IIS virtual directory); or

  http://venus.sod.asu.edu/WSRepository/xml/Courses.xml

- Using the browser to convert the xml file into html by:
  - Test Courses.xml, with and without this line:

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

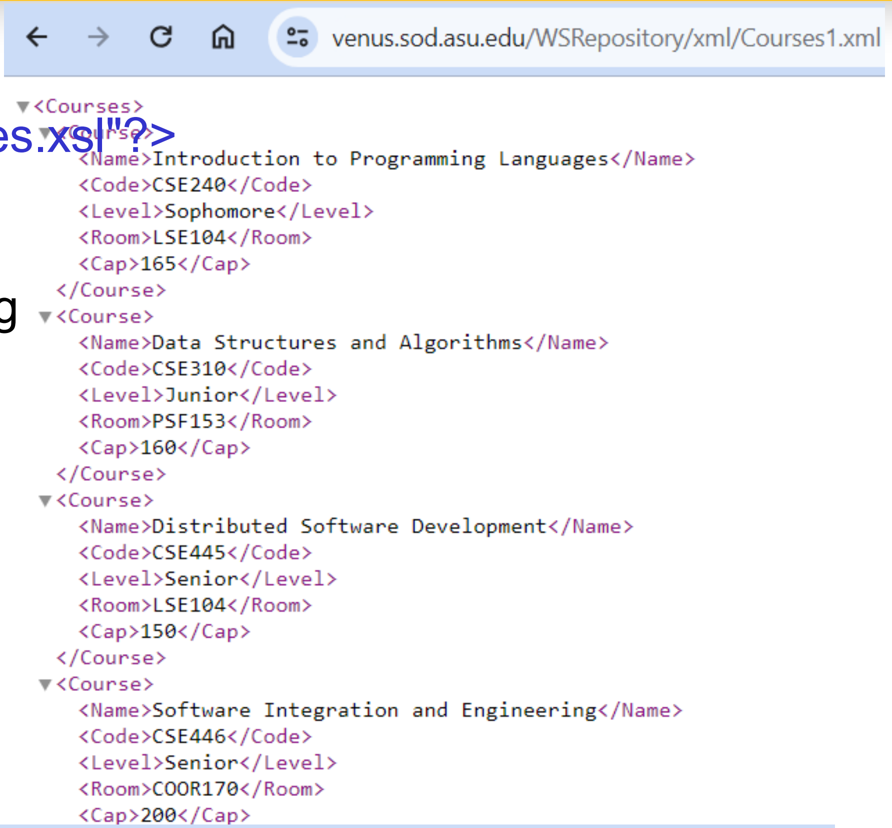
    and open the file by typing the address:

    C:\Inetpub\wwwroot\Courses.xml
  - Repeat by adding the line

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

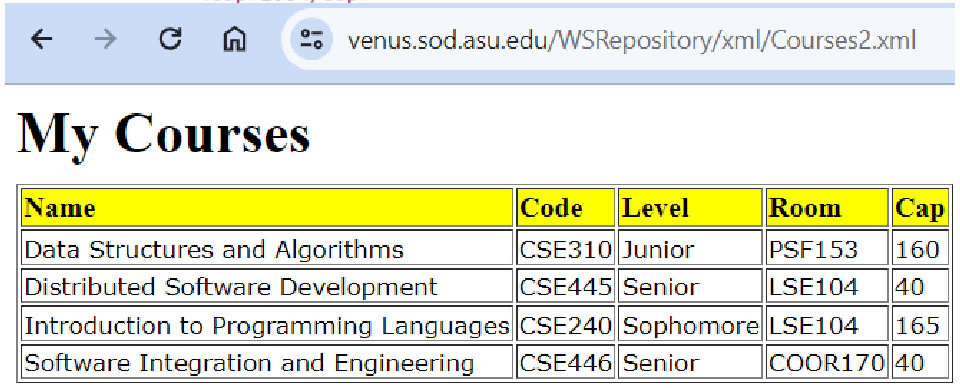
Converting XML to HTML on Client (Browser)
Example: Courses.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="Courses.xsl"?>
<Courses>
    <Course>
        <Name>Introduction to Programming
    Languages</Name>
        <Code>CSE240</Code>
        <Level>Sophomore</Level>
        <Room>BYAC110</Room>
        <Cap>82</Cap>
    </Course>
    <Course>
        <Name>Distributed Software
    Development</Name>
        <Code>CSE445</Code>
        <Level>Senior</Level>
        <Room>BYAC210</Room>
        <Cap>40</Cap>
    </Course>
</Courses>
```



venus.sod.asu.edu/WSRepository/xml/Courses1.xml

```
▼<Courses>
  ▼<Course>
     <Name>Introduction to Programming Languages</Name>
     <Code>CSE240</Code>
     <Level>Sophomore</Level>
     <Room>LSE104</Room>
     <Cap>165</Cap>
   </Course>
  ▼<Course>
     <Name>Data Structures and Algorithms</Name>
     <Code>CSE310</Code>
     <Level>Junior</Level>
     <Room>PSF153</Room>
     <Cap>160</Cap>
   </Course>
  ▼<Course>
     <Name>Distributed Software Development</Name>
     <Code>CSE445</Code>
     <Level>Senior</Level>
     <Room>LSE104</Room>
     <Cap>150</Cap>
   </Course>
  ▼<Course>
     <Name>Software Integration and Engineering</Name>
     <Code>CSE446</Code>
     <Level>Senior</Level>
     <Room>COOR170</Room>
     <Cap>200</Cap>
```



venus.sod.asu.edu/WSRepository/xml/Courses2.xml

My Courses

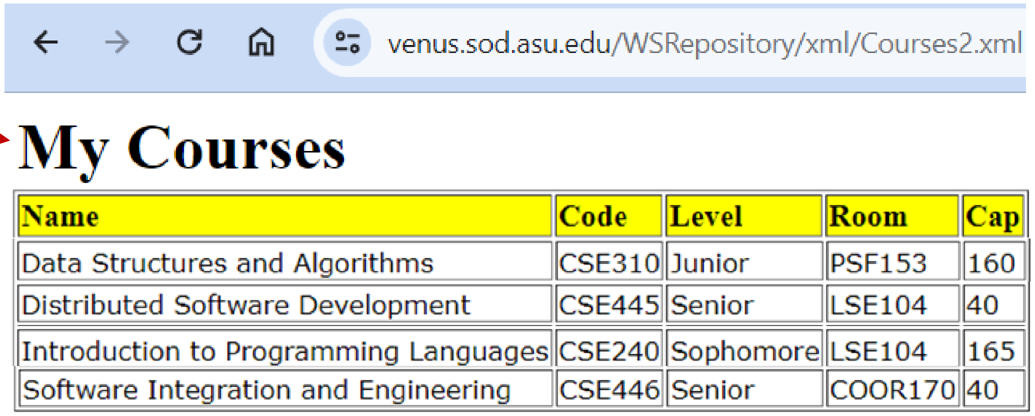
Name	Code	Level	Room	Cap
Data Structures and Algorithms	CSE310	Junior	PSF153	160
Distributed Software Development	CSE445	Senior	LSE104	40
Introduction to Programming Languages	CSE240	Sophomore	LSE104	165
Software Integration and Engineering	CSE446	Senior	COOR170	40

Courses.xsl that defines: for-each

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="/">
        <html>  <body>
            <h1>My Courses</h1>
            <table border="1">
                <tr bgcolor="yellow">
                    <td><b>Name </b></td>
                    <td><b>Code</b></td>
                    <td><b>Level</b></td>
                    <td><b>Room</b></td>
                    <td><b>Cap</b></td>
                </tr>
                <xsl:for-each select="Courses/Course">
                    <xsl:sort select="Name" />
                    <tr style="font-size: 10pt; font-family: verdana">
                        <td><xsl:value-of select="Name"/></td>
                        <td><xsl:value-of select="Code"/></td>
                        <td><xsl:value-of select="Level"/></td>
                        <td><xsl:value-of select="Room"/></td>
                        <td><xsl:value-of select="Cap"/></td>
                    </tr>
                </xsl:for-each>
            </table>
        </body> </html>
    </xsl:template>
</xsl:stylesheet>
```

Print header

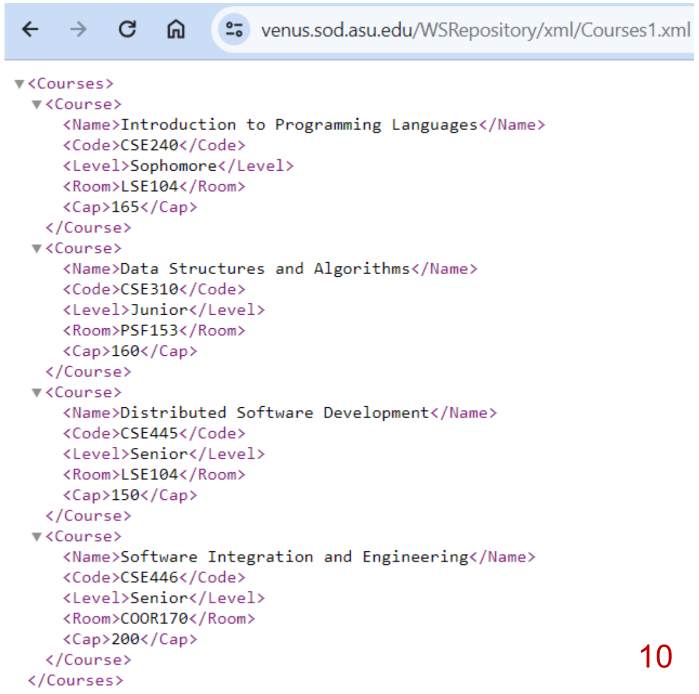
Print contents



venus.sod.asu.edu/WSRepository/xml/Courses2.xml

My Courses

Name	Code	Level	Room	Cap
Data Structures and Algorithms	CSE310	Junior	PSF153	160
Distributed Software Development	CSE445	Senior	LSE104	40
Introduction to Programming Languages	CSE240	Sophomore	LSE104	165
Software Integration and Engineering	CSE446	Senior	COOR170	40



venus.sod.asu.edu/WSRepository/xml/Courses1.xml

```
▼<Courses>
  ▼<Course>
     <Name>Introduction to Programming Languages</Name>
     <Code>CSE240</Code>
     <Level>Sophomore</Level>
     <Room>LSE104</Room>
     <Cap>165</Cap>
   </Course>
  ▼<Course>
     <Name>Data Structures and Algorithms</Name>
     <Code>CSE310</Code>
     <Level>Junior</Level>
     <Room>PSF153</Room>
     <Cap>160</Cap>
   </Course>
  ▼<Course>
     <Name>Distributed Software Development</Name>
     <Code>CSE445</Code>
     <Level>Senior</Level>
     <Room>LSE104</Room>
     <Cap>150</Cap>
   </Course>
  ▼<Course>
     <Name>Software Integration and Engineering</Name>
     <Code>CSE446</Code>
     <Level>Senior</Level>
     <Room>COOR170</Room>
     <Cap>200</Cap>
   </Course>
</Courses>
```

Another Example:
Fallacies.xml: The file to be displayed

```
<?xml version="1.0"?>
<Fallacies>
  <Fallacy>
    <Number>1</Number>
    <Text>The network is reliable.</Text>
    <Author>Bill Joy and Tom Lyon</Author>
  </Fallacy>
  <Fallacy>
    <Number>2</Number>
    <Text>Latency is zero.</Text>
    <Author>Bill Joy and Tom Lyon</Author>
  </Fallacy>
  <Fallacy>
    <Number>3</Number>
    <Text>Bandwidth is infinite.</Text>
    <Author>Bill Joy and Tom Lyon</Author>
  </Fallacy>
  <Fallacy>
    <Number>4</Number>
    <Text>The network is secure.</Text>
    <Author>Bill Joy and Tom Lyon</Author>
  </Fallacy>
  <Fallacy>
<Number>5</Number>
    <Text>Topology does not change.</Text>
    <Author>Peter Deutsch</Author>
  </Fallacy>
  <Fallacy>
    <Number>6</Number>
    <Text>There is one administrator.</Text>
    <Author>Peter Deutsch</Author>
  </Fallacy>
  <Fallacy>
    <Number>7</Number>
    <Text>Transport cost is zero.</Text>
    <Author>Peter Deutsch</Author>
  </Fallacy>
  <Fallacy>
    <Number>8</Number>
    <Text>The network is homogeneous.</Text>
    <Author>James Gosling</Author>
  </Fallacy>
</Fallacies>
```

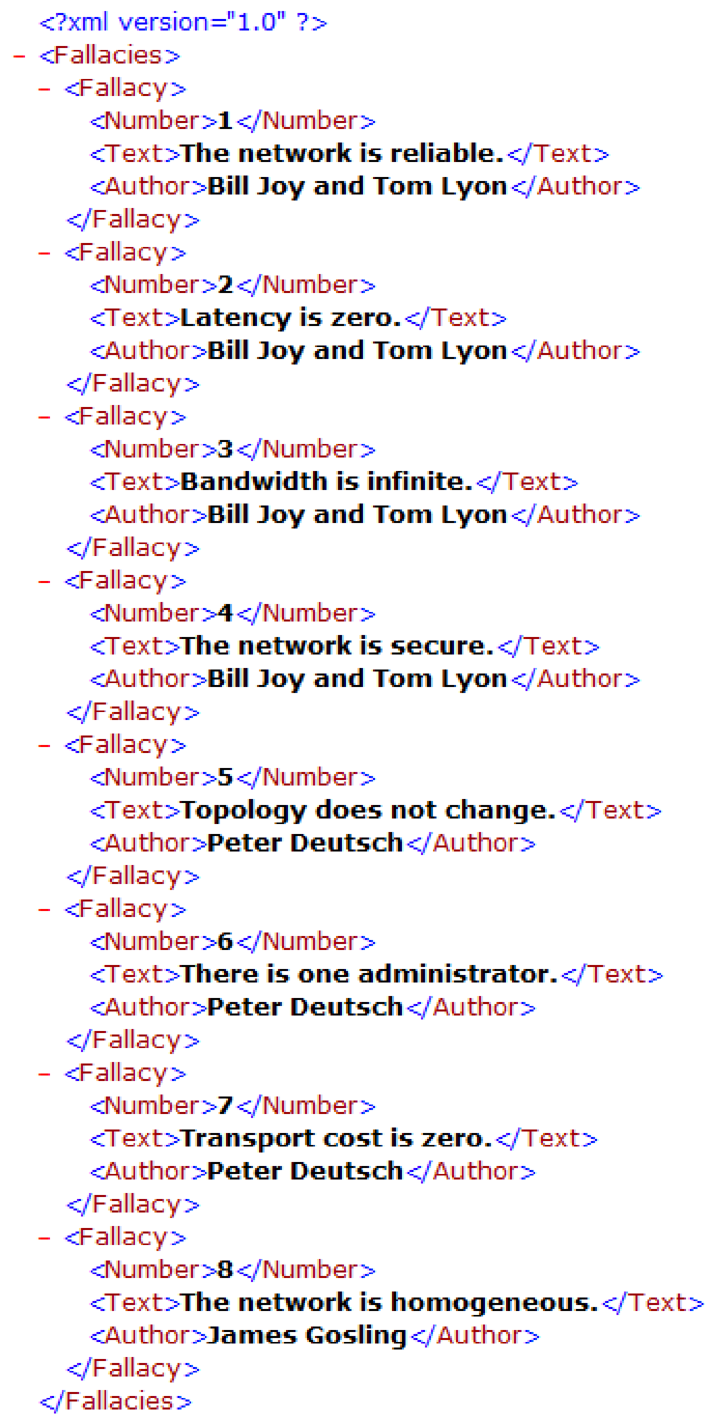
Fallacies.xsl: The stylesheet file

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <html> <body>
        <h1 style="background-color: blue; color: white; font-size: 18pt; text-align: center">
          Eight Fallacies in Distributed Computing
        </h1>
        <table border="1">
          <tr style="font-size: 12pt; font-family: verdana; font-weight: bold">
            <td style="text-align: center">Number</td>
            <td style="text-align: center">Fallacy</td>
            <td style="text-align: center">Author</td>
          </tr>
          <xsl:for-each select="Fallacies/Fallacy">
            <xsl:sort select="Number" />
            <tr style="font-size: 12pt; font-family: verdana">
              <td><i><xsl:value-of select="Number"/></i></td>
              <td><xsl:value-of select="Text"/></td>
              <td><xsl:value-of select="Author"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body> </html>
  </xsl:template>
</xsl:stylesheet>
```

Print header

Print contents

Without using the XSL File



```
  <?xml version="1.0" ?>
- <Fallacies>
  - <Fallacy>
      <Number>1</Number>
      <Text>The network is reliable.</Text>
      <Author>Bill Joy and Tom Lyon</Author>
    </Fallacy>
  - <Fallacy>
      <Number>2</Number>
      <Text>Latency is zero.</Text>
      <Author>Bill Joy and Tom Lyon</Author>
    </Fallacy>
  - <Fallacy>
      <Number>3</Number>
      <Text>Bandwidth is infinite.</Text>
      <Author>Bill Joy and Tom Lyon</Author>
    </Fallacy>
  - <Fallacy>
      <Number>4</Number>
      <Text>The network is secure.</Text>
      <Author>Bill Joy and Tom Lyon</Author>
    </Fallacy>
  - <Fallacy>
      <Number>5</Number>
      <Text>Topology does not change.</Text>
      <Author>Peter Deutsch</Author>
    </Fallacy>
  - <Fallacy>
      <Number>6</Number>
      <Text>There is one administrator.</Text>
      <Author>Peter Deutsch</Author>
    </Fallacy>
  - <Fallacy>
      <Number>7</Number>
      <Text>Transport cost is zero.</Text>
      <Author>Peter Deutsch</Author>
    </Fallacy>
  - <Fallacy>
      <Number>8</Number>
      <Text>The network is homogeneous.</Text>
      <Author>James Gosling</Author>
    </Fallacy>
  </Fallacies>
```

Demonstration using Client Conversion

Put the two Files into an IIS virtual Directory on localhost or on server

- Using client-side conversion: add a line of directive:

  <?xml-stylesheet type="text/xsl" href="Fallacies.xsl"?>

  after the first of prolog.

- Type the address in your browser:

  C:\Inetpub\wwwroot\Fallacies.xml

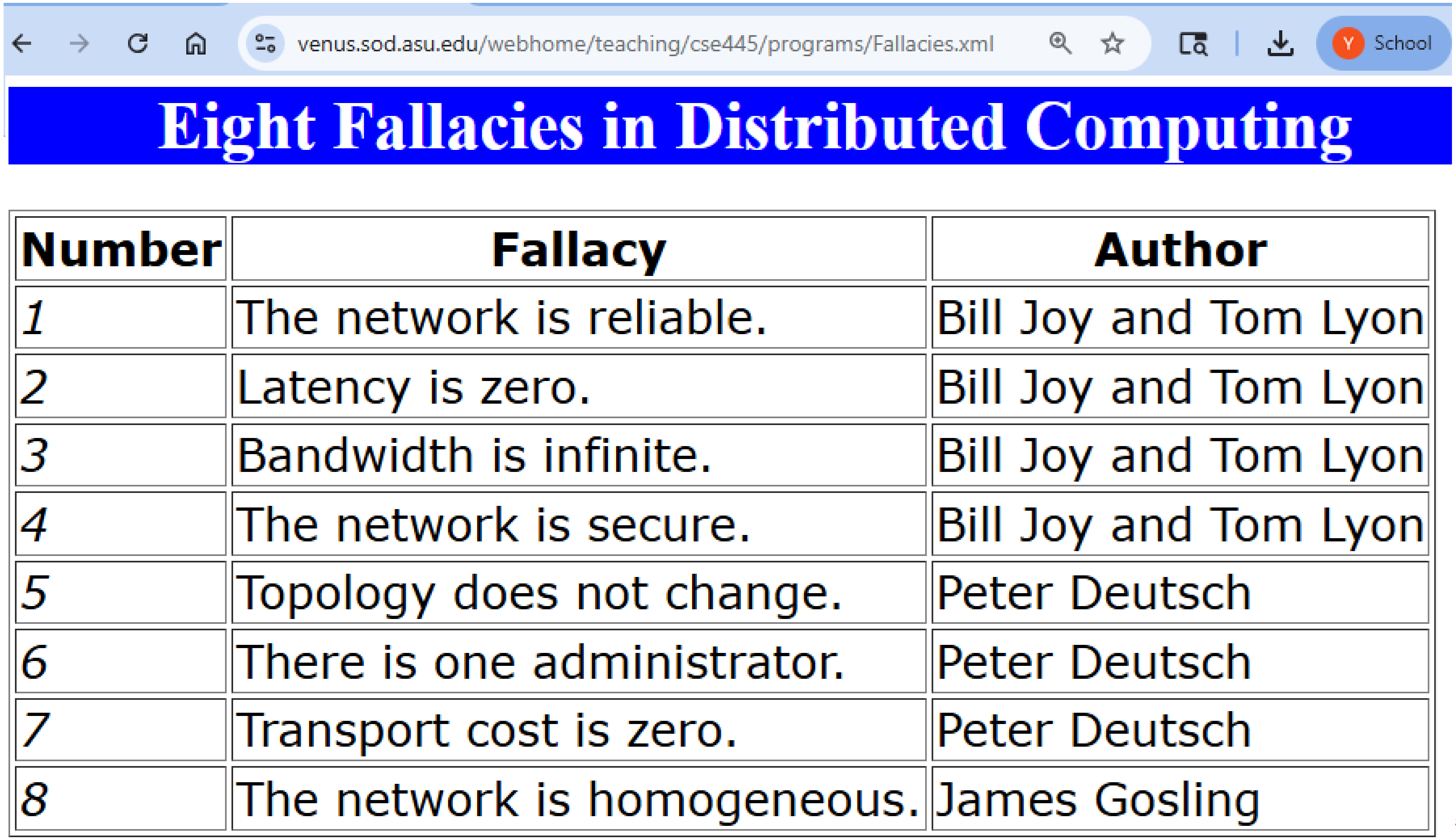
You can test it on your Windows IIS

OR, put them in any remote Web server, e.g.: http://venus.sod.asu.edu/WSRepository/xml/Fallacies.xml

The Display of the Page

https://venus.sod.asu.edu/webhome/teaching/cse445/programs/Fallacies.xml

https://venus.sod.asu.edu/webhome/teaching/cse445/programs/Fallacies.xsl


venus.sod.asu.edu/webhome/teaching/cse445/programs/Fallacies.xml
School

Eight Fallacies in Distributed Computing

Number	Fallacy	Author
1	The network is reliable.	Bill Joy and Tom Lyon
2	Latency is zero.	Bill Joy and Tom Lyon
3	Bandwidth is infinite.	Bill Joy and Tom Lyon
4	The network is secure.	Bill Joy and Tom Lyon
5	Topology does not change.	Peter Deutsch
6	There is one administrator.	Peter Deutsch
7	Transport cost is zero.	Peter Deutsch
8	The network is homogeneous.	James Gosling

Converting XML to HTML on Server Side

Two potential problems with the client-side conversion:

- Need to modify the xml file (to reference an xsl file)
- Browser dependent
  - Not all browsers have an embedded XSLT processor;
  - Not all versions of the browser has an embedded XSLT processor.
- To ensure that your page can be displayed properly, you need to write a program the conversion on the server side: Writing a .aspx file using a program to do the conversion.

Members of XslCompiledTransform Class

Method Name	Description
Equals	Overloaded. Determines whether two Object instances are equal. (inherited from Object)
GetHashCode	Serves as a hash function for a particular type. (inherited from Object)
GetType	Gets the Type of the current instance. (inherited from Object)
Load	Overloaded. Loads the XSLT style sheet, including style sheets referenced in xsl:include and xsl:import elements.
ReferenceEquals	Determines whether the specified Object instances are the same instance. (inherited from Object)
ToString	Returns a String that represents the current Object. (inherited from Object)
Transform	Overloaded. Transforms the XML data using the loaded XSLT style sheet.

Do it in C# in Console: Translate XML into html

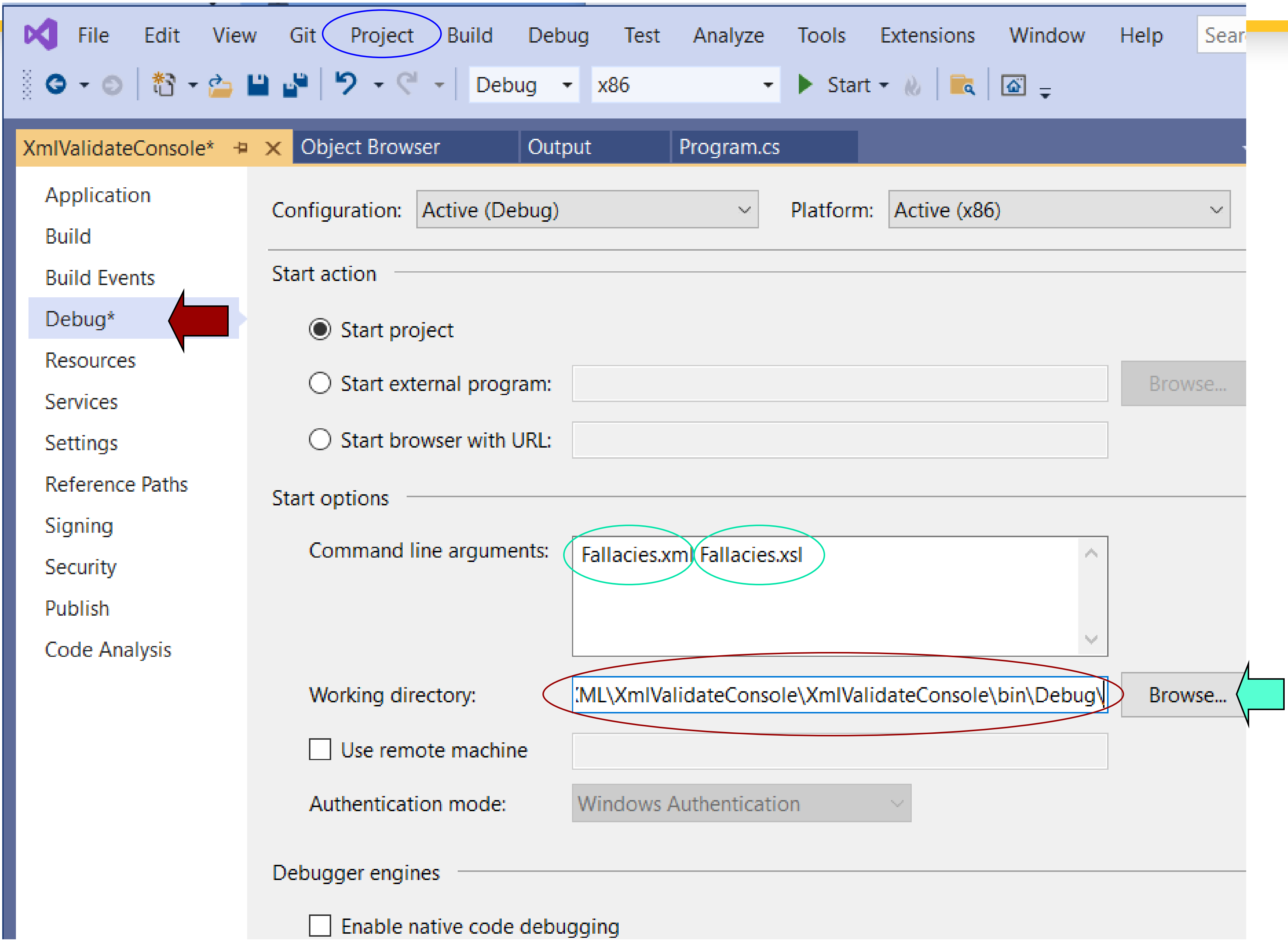
Command line input:
Fallacies.xml   Fallacies.xsl

```
class MyXSLTApp {
    static void Main(string[ ] args) {
        if (args.Length < 2) {
            Console.WriteLine(“Error: Files required not found");
            return;
        }
        try {
            XPathDocument doc = new XPathDocument(args[0]);
            XslCompiledTransform xt = new XslCompiledTransform();
            xt.Load(args[1]);
            xt.Transform(doc, null, Console.Out);
        }
        catch (Exception ex) {
            Console.WriteLine(ex.Message);
        }
        string x = Console.ReadLine();
    }
}
```

Do it in C#: How to do it step by step?

- Create a C# Console Application
- Copy and Paste the C# code into the application
- Build/Compile your program
- Choose Visual Studio menu “Project” -> “Properties…”
- Click on Debug -> Command Arguments
- Enter Fallacies.xml Fallacies.xsl files as the command line arguments (See next page)
- Browse to the Working directory where your Fallacies.xml and Fallacies.xsl files are located
- Run the program
- The html file will be generated (see following page).

Project → Properties …
Setting Up Command Line Input



Configuration: Active (Debug)   Platform: Active (x86)

Start action

- [✓ Start project]
- Start external program:
- Start browser with URL:

Start options

Command line arguments: Fallacies.xml Fallacies.xsl

Working directory: :ML\XmlValidateConsole\XmlValidateConsole\bin\Debug\

- [ ] Use remote machine

Authentication mode: Windows Authentication

Debugger engines

- [ ] Enable native code debugging

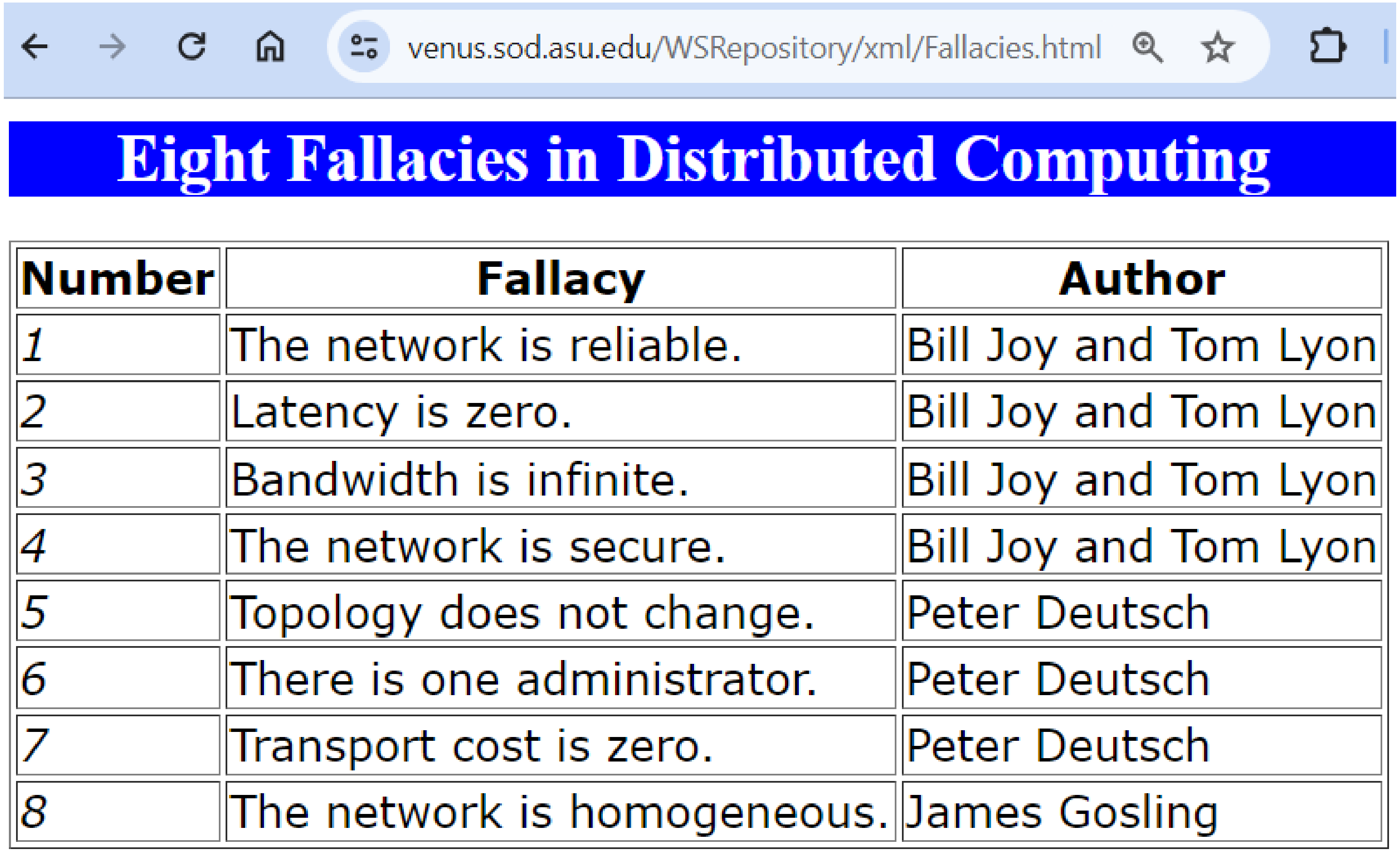
The output of the program: html file

```
<html>
 <body>
  <h1 style="background-color: blue; color: white; font-size: 18pt; text-align: center">
     Eight Fallacies in Distributed Computing
    </h1>
  <table border="1">
   <tr style="font-size: 12pt; font-family: verdana; font-weight: bold">
    <td style="text-align: center">Number</td>
    <td style="text-align: center">Fallacy</td>
    <td style="text-align: center">Author</td>
   </tr>
   <tr style="font-size: 12pt; font-family: verdana">
    <td>
     <i>1</i>
    </td>
    <td>The network is reliable.</td>
    <td>Bill Joy and Tom Lyon</td>
   </tr>
   <tr style="font-size: 12pt; font-family: verdana">
    <td>
     <i>2</i>
    </td>
    <td>Latency is zero.</td>
    <td>Bill Joy and Tom Lyon</td>
   </tr>
   <tr style="font-size: 12pt; font-family: verdana">
    <td>
     <i>3</i>
    </td>
    <td>Bandwidth is infinite.</td>
    <td>Bill Joy and Tom Lyon</td>
   </tr>
   <tr style="font-size: 12pt; font-family: verdana">
    <td>
     <i>4</i>
    </td>
```

```
<td>The network is secure.</td>
    <td>Bill Joy and Tom Lyon</td>
   </tr>
   <tr style="font-size: 12pt; font-family: verdana">
    <td>
     <i>5</i>
    </td>
    <td>Topology does not change.</td>
    <td>Peter Deutsch</td>
   </tr>
   <tr style="font-size: 12pt; font-family: verdana">
    <td>
     <i>6</i>
    </td>
    <td>There is one administrator.</td>
    <td>Peter Deutsch</td>
   </tr>
   <tr style="font-size: 12pt; font-family: verdana">
    <td>
     <i>7</i>
    </td>
    <td>Transport cost is zero.</td>
    <td>Peter Deutsch</td>
   </tr>
   <tr style="font-size: 12pt; font-family: verdana">
    <td>
     <i>8</i>
    </td>
    <td>The network is homogeneous.</td>
    <td>James Gosling</td>
   </tr>
  </table>
 </body>
</html>
```

Put this html file into a Website

http://venus.sod.asu.edu/WSRepository/xml/Fallacies.html



venus.sod.asu.edu/WSRepository/xml/Fallacies.html

Eight Fallacies in Distributed Computing

Number	Fallacy	Author
1	The network is reliable.	Bill Joy and Tom Lyon
2	Latency is zero.	Bill Joy and Tom Lyon
3	Bandwidth is infinite.	Bill Joy and Tom Lyon
4	The network is secure.	Bill Joy and Tom Lyon
5	Topology does not change.	Peter Deutsch
6	There is one administrator.	Peter Deutsch
7	Transport cost is zero.	Peter Deutsch
8	The network is homogeneous.	James Gosling

Web Application: Embed C# Code in Fallacies.aspx

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Xml.XPath" %>
<%@ Import Namespace="System.Xml.Xsl" %>

<%
  XPathDocument doc =
      new XPathDocument (Server.MapPath ("Fallacies.xml"));
  XslCompiledTransform xt = new XslCompiledTransform();
  xt.Load (Server.MapPath ("Fallacies.xsl"));
  xt.Transform(doc, null, Response.OutputStream);
%>
```

Send the return value as an html stream to Web browser.

Ira A. Fulton Schools of Engineering

Arizona State University

M9 L5
CSS:
Cascading Style Sheets

Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Outline

- Cascading Style Sheets in HTML
- Cascading Style Sheets in XML
- XHTML (Extensible HTML)?
- Development of Web Presentation Languages

CSS: Cascading Style Sheets

- In HTML, within an element tag:
  <link REL="STYLESHEET" TYPE="text/css"
  HREF="*UrlOfCssStyleSheet*">
- In XML: As a Processing Information (PI) the prologue of an XML document:
  <?xml-stylesheet href="*UrlOfCssStyleSheet*"
  type="text/css"?>

- Example (http://www.w3.org/Style/styling-XML.en.html):

```
<?xml-stylesheet href="common.css"?>
<?xml-stylesheet href="modern.css" title="Modern" media="screen"
                                                   type="text/css"?>
<?xml-stylesheet href="classic.css" alternate="yes" title="Classic"
                      media="screen, print" type="text/css"?>
```

CSS Language Rules

Element selected for applying the style

Style for the element

```
body
{
     font-family:      Arial;
     color:            blue;
     background:       yellow;
     margin:           8px;
}
```

Attribute

Value

HTML Example

common.css

```
body
{
   font-family: Tahoma, Arial, sans-serif;
   font-size: 14px;
   color: blue;
   background: yellow;
   margin: 8px;
}
h1 {
   font-size: 18px;
   margin-top: 14px;
   margin-bottom: 6px;
   border-bottom: 2px solid black
}
```

HTML:

```
<link href="common.css"
   rel="stylesheet"
   type="text/css" />

<body>
   <h1>First Level Heading</h1>
      One evening, just as he was getting
       his flute ready and his musicians
       were assembled, an officer brought
       him a list of the strangers who had
      arrived.
   <h2>Second Level Heading</h2>
         A new paragraph here.
</body>
```

CSS Application Example in XML Files

http://www.w3.org/Style/styling-XML.en.html

Define multiple CSS files to switch the view of the page

```
<?xml-stylesheet href="common.css"?>
<?xml-stylesheet href="modern.css" title="Modern" media="screen"
                                   type="text/css"?>
<?xml-stylesheet href="classic.css" alternate="yes" title="Classic"
               media="screen, print" type="text/css"?>
<ARTICLE>
   <HEADLINE> Fredrick the Great meets Bach </HEADLINE>
   <AUTHOR>Johann Nikolaus Forkel</AUTHOR>
   <PARA> One evening, just as he was getting his
      <INSTRUMENT>flute</INSTRUMENT>
      ready and his musicians were assembled, an officer brought him
      a list of the strangers who had arrived.
   </PARA>
</ARTICLE>
```

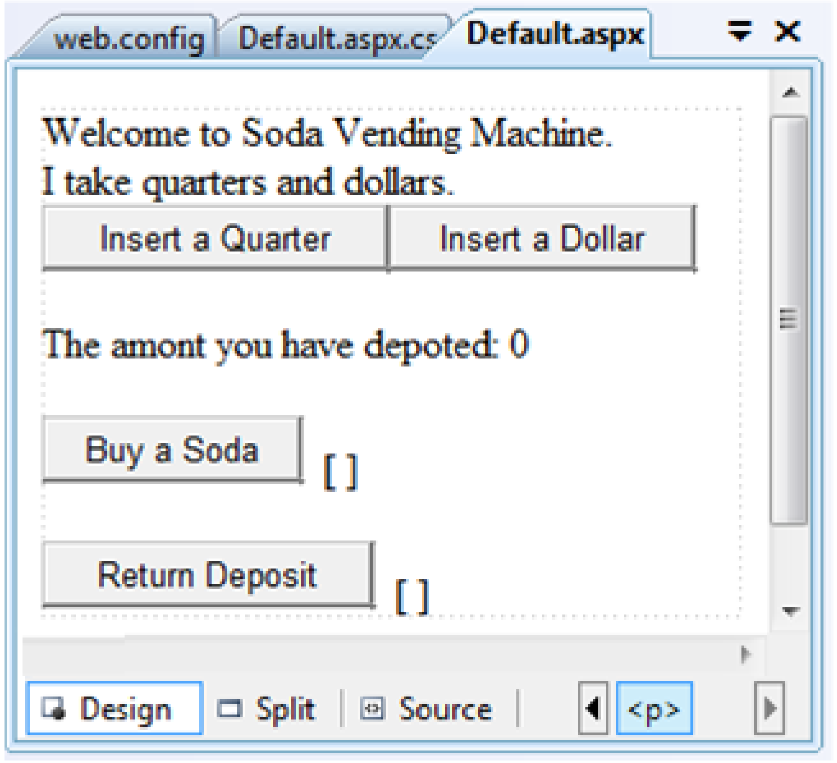
Define styles for the elements in .css file

What is XHTML (Extensible HTML)?

- Problems with HTML:
  - It has a fixed set of tags, and thus is not extensible;
  - Many browser-specific tags are added.
- How can these problems be addressed?
  - Add a document type definition file (DTD) or XSD schema
- XHTML allows three different ways to add a DTD file
  - Strict DTD: Enforce the structure by restricting to those tags defined in the DTD;
  - Transitional DTD: Allow all the depreciated features/tags to be migrated into the new page;
  - Frameset DTD: Same as Transitional DTD, but replaced the <body> with the <frame>, which allows developers to share the frames in multiple pages.

Application of XHTML in ASP .Net

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!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">
<head runat="server">
   <title>Vending Machine</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      Welcome to Soda Vending Machine. <br />
      I take quarters and dollars.<br />
   </div>
   <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
      Text="Insert a Quarter" />
   <asp:Button ID="Button2" runat="server" onclick="Button2_Click"
      Text="Insert a Dollar" />
   <p>
      The amont you have depoted:
      <asp:Label ID="lblAmount" runat="server" Text="0"></asp:Label>
   </p>
   <asp:Button ID="btnSoda" runat="server" onclick="Button3_Click"
      Text="Buy a Soda" />
   <asp:Label ID="lblSoda" runat="server" Text="[  ]"></asp:Label> <br />
   <br />
   <asp:Button ID="btnRtn" runat="server" onclick="Button4_Click"
      Text="Return Deposit" />
   <asp:Label ID="lblRtn" runat="server" Text="[        ]"></asp:Label>
   <br />
   </form>
</body>
</html>
```



Welcome to Soda Vending Machine.
I take quarters and dollars.

The amont you have depoted: 0

[ ]

[ ]

Development of Web Presentation Languages

- 1991 HTML
- 1994 HTML 2
- 1996 CSS 1 + JavaScript
- 1997 HTML 4
- 1998 CSS 2
- 2000 XHTML
- 2005 AJAX
- 2008 XAML = XHTML + CSS + AJAX + ~~Silverlight~~
- 2009 HTML5 = HTML + CSS + JavaScript / JQuery

JavaScript / JQuery

Animation can be programmed

Animation can be programmed

Ira A. Fulton Schools of Engineering

Arizona State University