Displaying XML

- Style sheets
  - CSS
  - XSL
- Parse the XML document (using Java, Perl, or any other language) and map the parsed structure to a HTML page(s).

The Book Order Displayed using XSL
What are style sheets?

- Style sheets describe how documents are presented (on screens, in print, or perhaps how they are pronounced).

- W3C has actively promoted the use of style sheets on the Web since the Consortium was founded in 1994.

- The Style Sheets Activity has produced two W3C Recommendations (CSS1 and CSS2) which are widely, although not consistently, implemented in browsers.

- The W3C Style Sheets Activity is also developing XSL.

CSS in Action: Computer Science Department Web Page

```html
<HTML>
<HEAD>
<TITLE>WPI Computer Science Department</TITLE>
<Link REL=stylesheet TYPE=text/css HREF="/cs-style.css"/>
<META NAME="Description" CONTENT="This is the web page of the Computer Science Department."/>

<META NAME="Keywords" CONTENT="computer science … significant bits">

<IMG SRC="/images/banner.gif"
    ALT="WPI Worcester Polytechnic Institute"
    BORDER=0 WIDTH=418 HEIGHT=40/>
</HEAD>
```
XSL: how it works

- Generate an XSL formatting specification as a **hierarchical sequence of formatting objects** and their properties. Examples of such formatting objects are images, text lines, paragraph blocks and so on.
- XSL additionally offers the programmer a number of **formatting functions**: borders, padding, rectangular areas, font, font-size, background image, hyphenation, etc.
- The formatting objects used in XSL are **based on prior work on CSS and DSSSL** - the Document Style Semantics & Specification Language. XSL is designed to be easier to use than DSSSL, which was only for use by expert programmers. Nonetheless, in practice it is expected that people will use tools to simplify the task of creating XSL style sheets.
A simple XML document.

```xml
<?xml version='1.0'?>
<doc><title>My Document</title>
<para>This is a <em>short</em> document.</para>
<para>It only exists to demonstrate a <em>simple</em> XML document.</para>
<figure><title>My Figure</title>
<graphic fileref="myfig.gif"/>
</figure>
</doc>
```

A simple XSL style sheet that generates HTML from XML

```xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template pattern="doc">
    <HTML>
      <HEAD>
        <TITLE>A Document</TITLE>
      </HEAD>
      <BODY>
        <xsl:process-children/>
      </BODY>
    </HTML>
  </xsl:template>

  <xsl:template pattern="title">
    <H1>
      <xsl:process-children/>
    </H1>
  </xsl:template>

  <!-- this stylesheet handles only a subset of the sample document -->
</xsl:stylesheet>
```
Formatting paragraphs is easy:

```xml
<xsl:template pattern="para">
  <P>
    <xsl:process-children/>
  </P>
</xsl:template>
```

Emphasis

- Emphasis. Designating emphasis is a little more interesting because it can be nested. The following template handles the simple, unnested case:

```xml
<xsl:template pattern="em">
  <I>
    <xsl:process-children/>
  </I>
</xsl:template>
```

If this is the only template for em, the result will be nested <I> tags in the output. We could rely on the browser to handle this case, but let's not. The following rule applies boldface to text that is nested within an already emphasized text segment:

```xml
<xsl:template pattern="em/em">
  <B>
    <xsl:process-children/>
  </B>
</xsl:template>
```

If necessary, additional rules could be added for triply nested emphasis and beyond.
Drawbacks of CSS and XSL

- So far very much platform dependent:
  - CSS:
    - Set of CSS features that works well in Netscape Navigator and IE5 is very small
  - XSL:
    - Mostly supported by IE5

DOM: Document Object Model

- Overview
- Features of Interest
- The Object Model
- Locating Objects
- Examples
- XML for Java
- Populating a DOM tree in java
Overview

• The Specification
  – Document Object Model Level 1 Specification
    Version 1.0 issued by the World Wide Web
    Consortium October 1998

• What is DOM Level 1 ?
  – A “platform and language neutral” application
    programming interface (API) for documents.
  – Defines the logical structure of documents and
    the way the document is accessed and
    manipulated.

Overview

• What is DOM Level 1 ?
  – Enables programs and scripts to dynamically
    access and update the content structure and
    style of documents.
  – Standard interface and logical model
  – Provides a set of objects for supporting
    representations of HTML and XML
    documents.
Overview

- Two Parts of DOM Level 1
  - Core
  - HTML
- Core Level 1
  - Defines a set of low level interfaces that can represent any structured document.
  - Extensions for representing XML specific items.
- HTML Level 1
  - Extends core to describe objects and methods specific to HTML

Features of Interest

- Build document
- Navigate Structure
- Add, Modify, Delete elements and content
- Work with whole or fragments of documents
- Direct access to nodes in the structure by name or class
The Model

- The Object Model is the backbone to support document manipulation.
- DOM’s *logical* structure is tree based.
- Although the model is tree-like from the programmer’s view, the internal structure implementation is not defined.
- In order to be implementation neutral, the term “Structure Model” is used to describe the representation.

The Model Structure

- Tree of Nodes (Object Hierarchy)
  - Documents are modeled using objects
  - Node object is a single tree node
  - Document object is the root node of the tree
  - Each node can have child nodes
  - All nodes will have parent nodes (except root)
  - Each node is numbered sequentially and can be named
  - Nodes can be inserted or removed from the structure
The Model Notes

- The Object Model captures the document structure and the behavior of the document and object of which it is composed
  - Nodes in the object diagram represent objects that have functions and identity.
  - Nodes/Trees do not represent data structures.
- Any two implementations of the DOM will produce the same structural model, with the same object representations and relationships on a given document.

Objects

- The structure models the structure of the document
  - A document tree is an ordered collection of elements
  - An element is an object that contains all the content between the start and end tags of the element as object, and any set of attributes that are defined for that element.
- Elements can be nested
  - E.g. Paragraphs, sections of a document, or cells of a table
Example

```html
<TABLE ID="myTable">
  <TR><TD> C1R1 </TD> <TD> C2R1 </TD></TR>
  <TR><TD> C1R2 </TD> <TD> C2R2 </TD></TR>
</TABLE>
```

Locating Objects

- **Object Identity**
  - Each element has a name, its tag name.

  ```html
  <MyTag ID="001">This is a paragraph </MyTag>
  ```

  - The element “MyTag” has an attribute, ID=“001”, that uniquely identifies it.
  - You can get a list of all the MyTag elements in the document or list the elements with by attribute, e.g. ID=“001”.

Department DTD

<!ELEMENT department (employee)*>
<!ELEMENT employee (name, (email | url))>
<!ATTLIST employee id CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT url EMPTY>
<!ATTLIST url href CDATA #REQUIRED>

<?xml version="1.0" ?>
<!DOCTYPE department (doctype...)>
<department>
  <employee id="J.D">
    <name>John Doe</name>
    <email>John.Doe@foo.ibm.com</email>
  </employee>
  <employee id="B.S">
    <name>Bob Smith</name>
    <email>Bob.Smith@foo.com</email>
  </employee>
  <employee id="A.M">
    <name>Alice Miller</name>
    <url href="http://www.trl.jp.ibm.com/~amiller"/>
  </employee>
</department>
Department

employee
    name
    "John Doe"
    email
    "John.Doe@foo.ibm.com"

employee
    name
    "Bob Smith"
    email
    "Bob.Smith@foo.com"

employee
    name
    "Alice Miller"
    url
    "http://www.trl.jp.ibm.com"

From "XML and Java"

Relationships

• Two kinds of relationships
  – Ability to traverse the object hierarchy
    • From a given object you can locate it in the tree and refer to parent, child, sibling.
    • A NodeList can be generated to produce and ordered list of nodes.
  – Ability to access a collection of objects by name
    • NamedNodeList object which represents nodes that can be accessed by name.
XML for Java

- Existing DOM binding
- Provides resources to build a well-formed DOM tree from a database.
- XML document generation ability
  - Generates valid document based on given DTD and constructed document structure.

Populating a DOM tree in java

```java
Element root = new createElement("department");
doc.appendChild(root);

Element emp = doc.createElement("Employee");
root.appendChild(emp);

Element item = doc.createElement("Name");
item.appendChild(doc.createTextNode("John Doe");
emp.appendChild(item);

item = doc.createElement("email");
item.appendChild(doc.createTextNode("JohnDoe@wpi.edu");
emp.appendChild(item);
```
DOM use in the Digital Telemetry Library

• Construct DOM structure using the telemetry data and DTD/metadata generated by the description wizard.
• Generate an XML telemetry document from populated DOM structure.
• The generator will output well-formed and valid XML message stream.
• Future use in manipulating documents for based on queries from client

Summary

• Platform/Language neutral API for docs
• Logical structure for XML and HTML
• Access to manipulate structure and contents using script/language bindings
• Static documents are transformed into dynamic “living” documents/applications by DOM services
• “XML for Java” provides services for generation of valid and well-formed XML documents via DOM and DTDs
References

• Hiroshi Maruyama, Kent Tamura, and Naohiko Uramoto, "XML and Java", Addison-Wesley, May 1999
• SAX 1.0, www.megginson.com/SAX
• XML for Java, IBM, alpaworks.ibm.com

References