Theme 4-1

XSLT

XSL

- You can now create, read and validate XML docs.
- **The next step:** To learn how to format XML docs.
- Details for formatting XML were originally in the XSL spec.
- **XSL** = e**X**tensible **S**tylesheet **L**anguage.
- The spec took too long to finish, so the W3C split it in three.

XSL

- **These three parts:**
  - **XSLT** (for **T**ransformations)
  - **XPath** (to access or refer to parts of an xml document)
  - **XSL-FO** (for **F**ormatting **O**bjects)

- First, we'll look at how to use **XSLT** to transform XML docs.

- XML is commonly transformed to HTML or to another XML.

- You can transform XML into practically any format you like.

XSLT

- **Transforming an XML doc means…**
  - ...to use XSLT to analyse its contents...
  - ...and then take actions to generate output...
  - ...depending on what elements are found.

- You can, e.g., output the data in a different order...
- ...or display only certain pieces of info, and so on.

XPath

- After XSLT, we'll look at **XPath**.

- XPath was created to address **different parts** of an XML document.

- In addition:
  - Manipulation of strings, numbers and Booleans;
  - Contains matching functionalities.

XSL-FO

- After XPath, we'll look at **XSL-FO**.

- XSL-FO is typically used to format XML for **print** output.
  - E.g. an output in **PDF** format.

- XSL-FO has little to no browser support.

- XSL-FO requires specific parser software to use.

XSLT

Extensible Stylesheet Language Transformations

Transforming XML with XSLT

- **For XSLT transformation, you need:**
  1. The **XML** doc that contains the source data to transform;
  2. The **XSLT** stylesheet that describes the transformation rules.

- You also need an **XSLT processor**.
  - It is a piece of software that does the transformation work;
  - It can be a standalone app or part of a browser.

XSLT stylesheet

- XSLT style sheets are xml **files**.
  - All XML rules apply

- Save them using the **.xsl** extension.
  - Not **.xslt**

- XSLT uses **XPath** to identify nodes.
  - We will cover XPath in detail later.

Analysing the Source XML

- You must **link** the XML source to the XSLT style sheet.

- Do so by adding a **processing instruction** to the **source**.

- Then, when you open the source XML doc...
- ...in an XSLT processor or a browser with XSLT support...
- ...the instruction tells the processor to transform the XML.

Analysing the Source XML

- **The linking processing instruction looks like this:**

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

- Add it **after** the XML declaration but **before** the root.

- Replace **style.xsl** with the name of your style sheet.

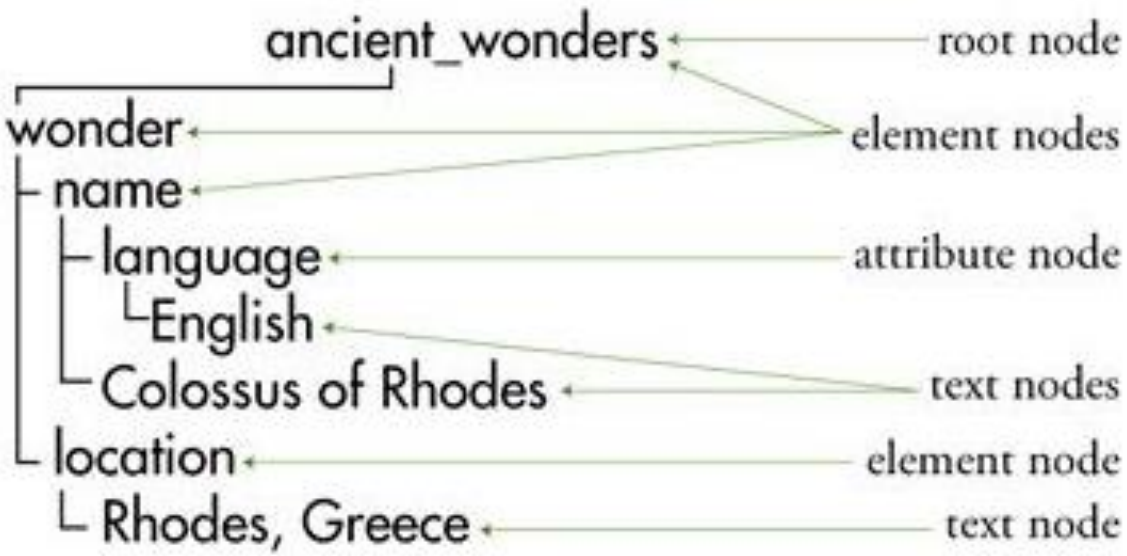
Analysing the Source XML

- Once the XSLT processor has your source and style sheet...
- ...it analyses the source and converts it into a **node tree**.

- A **node tree** is...
  - ...a hierarchical representation of the source XML doc...
  - ...in the form of a tree data structure...
  - ...that is stored in memory.

Analysing the Source XML

- A **node** is one individual piece of the XML doc.
  - E.g. an element, an attribute, or some text content.



Assessing the XSLT Style Sheet

- After creating the node tree...
- ...the XSLT processor looks to the XSLT style sheet...
- ...for **instructions** on what to do with the source nodes.

- The instructions are contained in **templates**.
  - Templates in XSLT are like **functions** in programming.

Performing the Transformation

- Transformation starts by processing the **root template**.
- Every style sheet **must** have a root template.
- The root template applies to the source’s root node.
- You can always find an example in the sample files under the Editix installation folder.

Performing the Transformation

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
            <xsl:template match="/">
                  <!-- basic output here -->
            </xsl:template>
</xsl:stylesheet>
```

Assessing the XSLT Style Sheet

- **Each XSLT template has two parts:**
  1. A **label** saying which source nodes the template applies to;
  2. Instructions (a.k.a. **rules**) on how to transform the nodes.

- **The instructions will…**
  - ...output source nodes...
  - ...output literal (“hard-coded”) result nodes, or...
  - ...further process the source nodes.

Performing the Transformation

- The root template may call **sub-templates**...
- ...which can then apply to other nodes in the source XML.

- Once the last instruction in the root template is processed...
- ...transformation ends.

- Then the output (in memory) is saved to a file, or...
- ...displayed in a browser, or both.

Writing XSLT Stylesheets

Beginning an XSLT Style Sheet

- An XSLT style sheet is also an XML doc.

- XSLT is a standardised **custom XML language**.

- An XSLT doc must follow all XML grammar rules...
- ...as well as use the predefined XSLT tags and structure.

- Start an XSLT style sheet with the **XML declaration**.

Beginning an XSLT Style Sheet

- After the declaration, create the **XSLT root** element.
- The root element name must be **xsl:stylesheet**.
- It must have two attributes, **xmlns:xsl** and **version**.
  - The **xmlns:xsl** attribute defines the XSLT namespace.
  - Its value must be **exactly** the following:

```
http://www.w3.org/1999/XSL/Transform
```

Beginning an XSLT Style Sheet

Close the root element with `</xsl:stylesheet>`.

- All further code goes inside the root element.

- The XML declaration and XSLT root element tags...
- ...will look the same in all your style sheets.

Creating the Root Template

- **Reminder:**
  - The XSLT processor starts by processing the root template.
  - The root template must tell the processor...
  - ...how to transform the content from the source’s root node...
  - ...into some new output.

Creating the Root Template

- **The root template will look like this:**

```
<xsl:template match="/">
          <!-- Template rules go here -->
</xsl:template>
```

- The template element is a child of the XSLT root element.

- The forward slash represents the root of the XML document.

Outputting HTML

- If you're **not** transforming XML to XML (the default)...
- ...you **must** tell the processor what the output format is.

- E.g., if we're transforming XML to HTML...
- ...add the following as the **first child** of the **root element**:

```
<xsl:output method="html" />
```

Outputting HTML

- Now, define template instructions for the root template.

- One type of template instruction is a **literal element**.
  - A literal element is any element in the style sheet…
  - ...that is **NOT** an XSLT instruction (doesn't start with **xsl:**).
  - They are output **exactly as they appear** in the style sheet.

Outputting HTML

- Create the outer structure of the output as literal elements.
- **Example:**

```
<xsl:template match="/">
      <html><head><title>Title!</title></head>
            <body>
            <!- Do something here-->
            </body>
      </html>
</xsl:template>
```

Outputting HTML

- The **HTML** code is sent to the **output** exactly as it appears.

- In the next lecture we will put XSLT instructions...
- ...inside the HTML outer structure...
- ...that retrieves data from the XML source...
- ...which then forms part of the HTML output.

Theme 4: XSL

TO BE CONTINUED...