Rendering Data to HTML

- Direct HTML generation
- Cascading style sheets
- XSLT
Direct HTML Generation

- Output HTML content directly from code
- E.g.:

```cpp
cout << "<HTML>" << endl;
cout << "<BODY> <H1>" << endl;
cout << "<FONT COLOR=""#FF0000">" << endl;
cout << getTitle();
cout << "</FONT>" << endl;
cout << "</H1> </BODY>" << endl;
cout << "</HTML>" << endl;
```
Direct HTML Generation

- This is yucky – the model and view are tightly coupled
- What if I want to change the color of all instances of the title?
  - Global search and replace
  - Recompile
CSS: Extracting commonality

- Standard way of referencing an (external) library of styles: stylesheet
- Make style changes once, in stylesheet, and changes are reflected across many documents
- Change styles of HTML tags or user defined classes
Cascading Style Sheets

- `<HEAD> <LINK rel="stylesheet" href="style.css" type="text/css"/> </HEAD>`

- `style.css:`
  ```
  h1 {
    color: blue;
    font-variant: small-caps;
  }
  ```
Cascading Style Sheets

- Now we’ve extracted some of the low-level view contents, but the view structure is still hard-coded in the app.
- Why should we have to print HTML from the code? It would be cleaner to keep HTML out of the code and use fancy HTML editors on it.
XSLT

- XML Stylesheet Language for Transformations
- Given XML content, apply pattern matching rules to transform the content
- Output format can vary
  - HTML
  - XML
  - Foo 17.5
XSLT: XML Input

- **XML**: eXtensible Markup Language
- Hierarchical, user defined structure

```xml
<pets>
  <pet name=“Spike” type=“dog”>
    <trick> Play dead </trick>
    <trick> Roll over </trick>
  </pet>
  <pet name=“Bob” type=“camel”>
    <trick> Drool </trick>
  </pet>
</pets>
```
XSLT is special XML

- List pets:

```xml
<xsl:template match="//pet">
  <p> <b>
    <xsl:value-of select="@name"/>
  </b> is a
  <xsl:value-of select="@type"/>
  </p>
</xsl:template>
```

- Output: <p> <b> Spike </b> is a dog. </p>  
  <p> <b> Bob </b> is a camel. </p>
Many Stylesheets, many views

- XSLT is Turing complete – this means you can do anything
  - This doesn’t mean you want to – XSLT is not a procedural language
- Changing the stylesheet changes the view without recompiling application
  - Model is completely decoupled from view
  - Not limited to topical changes like CSS
Basic XSLT procedure

1. Generate/Load model data
2. Convert model to Document Object Model (DOM – XML in memory)
3. Load XSLT transformation
4. Feed the DOM to the transformer, which outputs the result

Xerces/Xalan handle all of this, in C++ or Java
http://xml.apache.org/