Transforming XML

- **Benefit of XML**
  - Structured, standard
  - Readable, understandable (see iTunes example)

- **We don't always know the format we need for our application**
  - Adapter pattern, fit with XML?
  - If the data is cumbersome, make it agile
  - [http://www.w3.org/Style/XSL](http://www.w3.org/Style/XSL)

What is XSL (*)?

- **Extensible Stylesheet Language Family**
- **XLST adds Transformations**
  - Transform XML into ... (HTML, RDF, XML,...)
  - Complete programming language

- **XPATH**
  - Language for expressing/addressing parts of an XML document, see also XML Linking ([http://www.w3.org/TR/xlink](http://www.w3.org/TR/xlink))

- **XSL-FO**
  - Vocabulary for specification of formatting semantics (don't ask me)

What is XSLT?

- **Rule-based language**
  - Match elements
  - Select elements
  - Find value of elements

- **Need minimal understanding of XPATH to understand rules/templates in XLST**
  - Rule/query in XLST has a context, the node in the XML source being transformed
  - Specify tree-like path from some root to a node

Simple View of XPATH

- **The root is /**
  - This is global, beginning of transformed XML

- **foo/bar**
  - Path from foo to bar (direct parent-child)
  - To be global use /foo/bar

- **//foo and foo/bar**
  - Like foo->*bar, e.g., any number of ancestors/descendants between labelled nodes