INFO/CS 4302
Web Information Systems

FT 2012

Week 4: Structured Data and Document Presentation Formats

(Lecture 7)

Theresa Velden

Today's Program:

- Recap
- **XML Namespaces**
- Expressing constraints on XML documents & document validity:
  - DTD (Document Type Definition)
  - **XML Schema**
  - **RELAX NG** (Introduction)
- **XPath**: addressing entities within XML documents

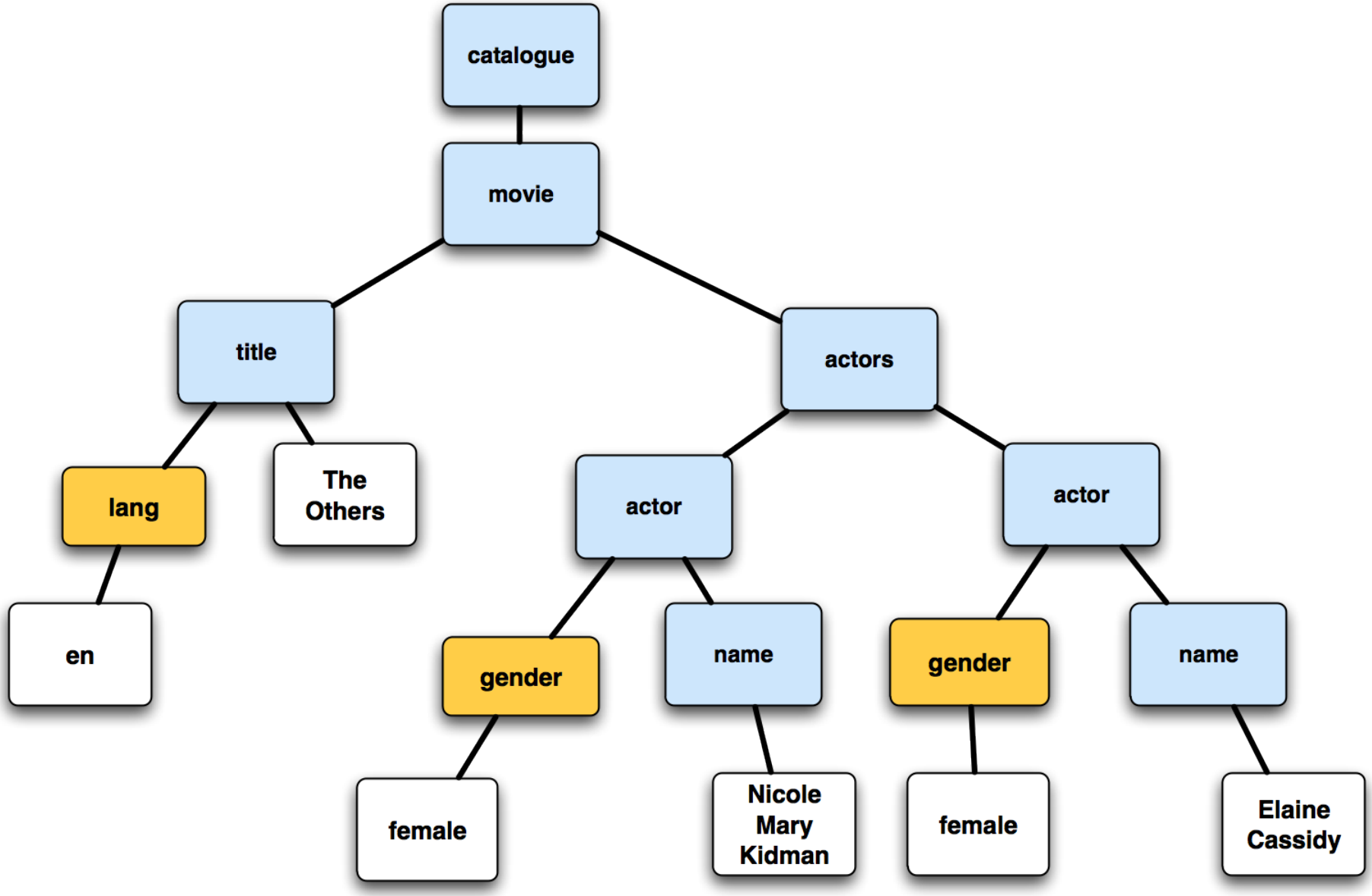
XML Example 2: Element Attributes

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!-- catalogue_snippet.xml  Created: 2012-09-08 17:09 -->

<catalogue>
    <movie>
        <title lang="en">The Others</title>
        <actors>
            <actor>
                <name gender="female">Nicole Mary Kidman</name>
            </actor>
            <actor>
                <name gender="female">Elaine Cassidy</name>
            </actor>
        </actors>
    </movie>
</catalogue>
```

The XML Tree


catalogue
movie
title
actors
lang
The Others
actor
actor
en
gender
name
gender
name
female
Nicole Mary Kidman
female
Elaine Cassidy

Beyond elements and attributes…

- XML 'References':
  - Character References, e.g. < → U+003C
    - See http://www.w3.org/TR/xml-entity-names/
  - Entity References, e.g. & → &
    - Refers to the content of a named entity

- CDATA sections
  - <![CDATA[Some character data]]>
  - Data within is ignored by XML parser

- Processing instructions,
  - E.g. <? xml-stylesheet href="mystyle.xsl" type="text/xsl" >

Well-formed XML Documents

- Follow a list of syntax rules
- A document is not an XML document if it is not well-formed
- An xml document that is well-formed guarantees that the document can be unambiguously parsed and transformed into a unique tree structure

NAMESPACES

Motivation: Interoperability

Namespaces

- Collections of element and attribute names
- Uniquely identified by URIs

The need for namespaces:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- catalogue_snippet.xml-->

<catalogue>
    <movie>
        <title>The Others</title>
    </movie>
</catalogue>
```

[Carl Lagoze, INFO 4302, Fall 2011]

The need for namespaces:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Theresa's Personal Homepage -->
<html>
<head>
    <title>Theresa Velden</title>
</head>
<body>
    <p>My Hobby</p>
    <p>Watching movies</p>
</body>
</html>
```

[Carl Lagoze, INFO 4302, Fall 2011]

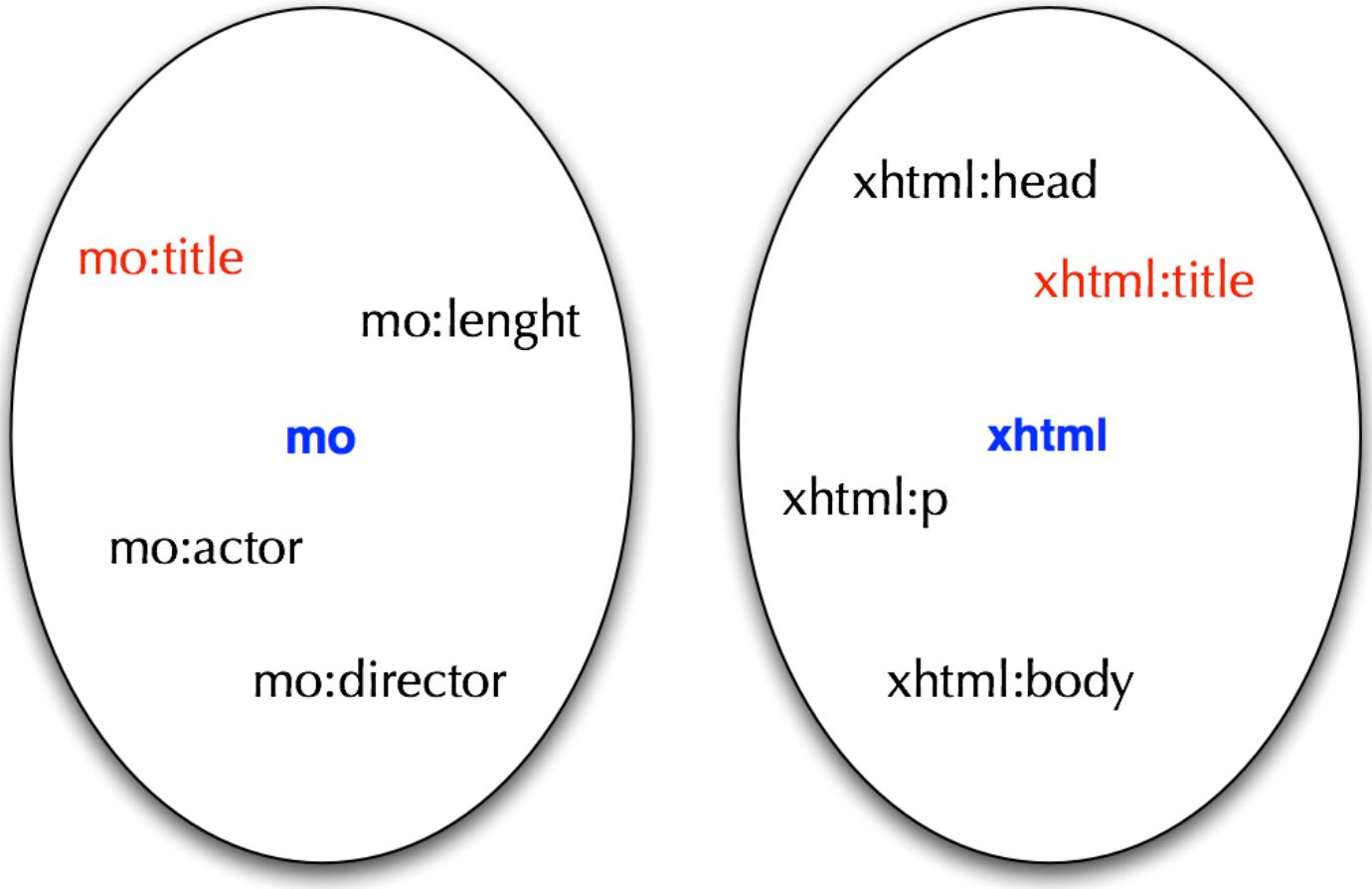
Element Name Conflicts

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Theresa's Personal Homepage -->
<html>
<head>
   <title>Theresa Velden</title>
</head>
<body>
   <p>My hobby</p>
   <p>Watching Movies</p>
   <movies>
      <movie>
         <title>The Others</title>
      </movie>
   </movies>

</body>
</html>
```

XML Namespaces

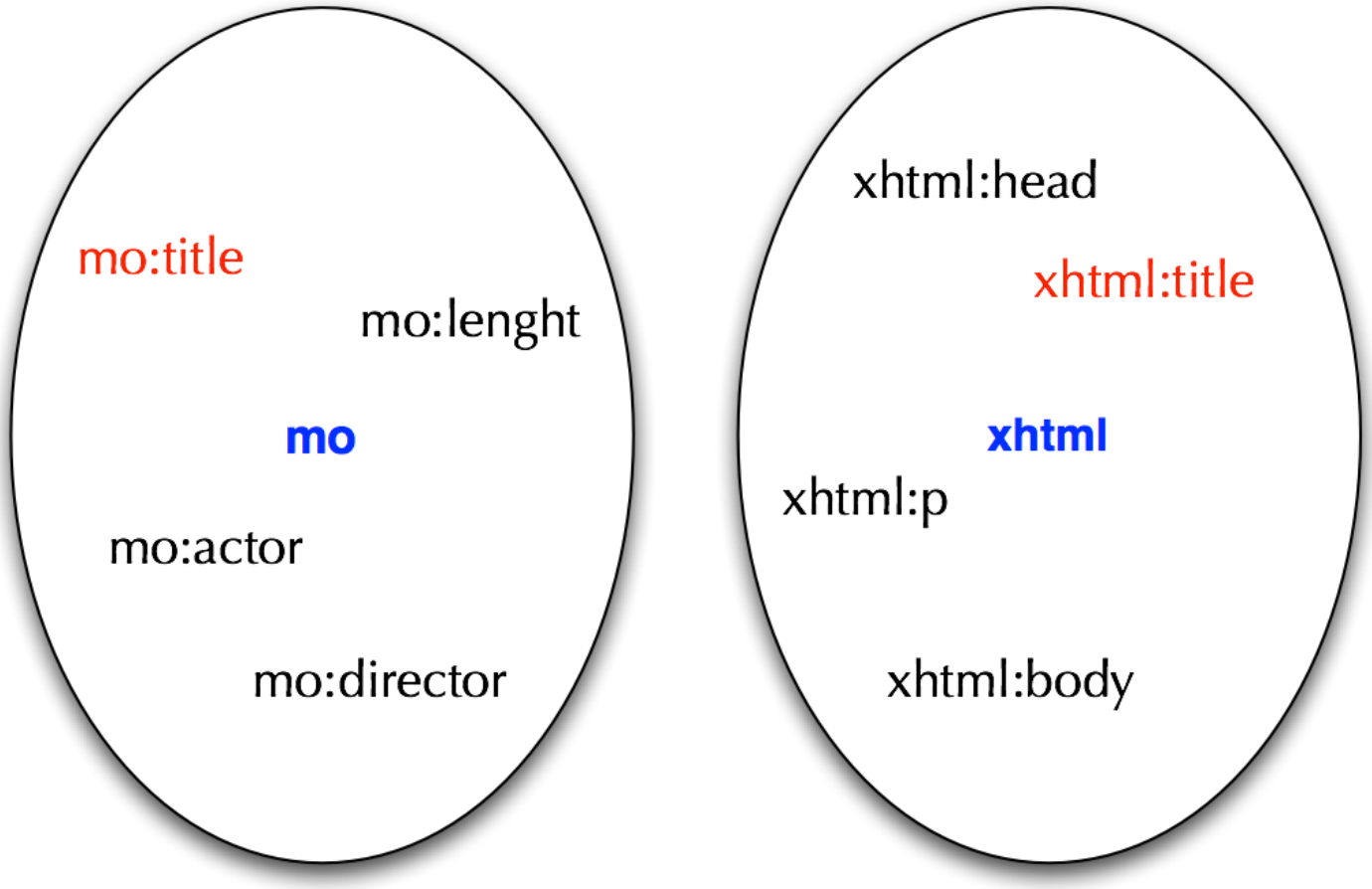


vocabulary mo

vocabulary xhtml

[Carl Lagoze, INFO 4302, Fall 2011]

XML Namespaces



vocabulary mo

vocabulary xhtml

But who guarantees uniqueness of pre-fixes?

[Carl Lagoze, INFO 4302, Fall 2011]

Resolving Collision

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Theresa's Personal Homepage -->
<xhtml:html>
<xhtml:head>
    <xhtml:title>Theresa Velden</xhtml:title>
</xhtml:head>
<xhtml:body>
    <xhtml:p>My hobby</xhtml:p>
    <xhtml:p>Watching Movies</xhtml:p>
    <mo:movies>
        <mo:movie>
            <mo:title>The Others</mo:title>
        </mo:movie>
    </mo:movies>
</xhtml:body>
</xhtml:html>
```

Use of **'Qualified Names'** or Qnames

prefix part : local part

Prefix must be associated with an URI:

xhtml: http://www.w3.org/1999/xhtml

mo: http://nogood.com/movie

Prefix itself is arbitrary, decisive is identification by name space URI

XML Namespaces

- Are collection of names for elements and attributes
- Associate local prefixes with a global namespace name
  - A unique name for a namespace: use URI in domain of party creating the namespace
  - Doesn't have any meaning, i.e. does not have to resolve into anything
  - Are compared as strings (URI equivalence rules do not hold http://authority.org/~namespaceX

    ≠ http://authority.org/%7EnamespaceX

Rationale for Namespaces

- How the web works:
  - Individually created documents linked by ambiguous references
- Towards a global database of knowledge?
  - Key: allow for distributed knowledge creation and lazy integration
- Problems:
  - Collisions (of how things are named)
  - Joins (how to link related content)
- Namespaces:
  - Build on URI notion
  - Uniquely qualify intra-document name collisions
  - Provide technology for cooperation

[Carl Lagoze, INFO 4302, Fall 2011]

How to use namespaces:

Namespace (binding) declarations

- Declared using a family of reserved attributes: such an attribute's name must either be "xmlns" or begin with "xmlns:"

  e.g. <mo:movies **xmlns:**mo="http://www.nogood.com/movie">
- Scope of name space declaration:
  - Begins at element for which declared
  - Applies to entire content of that element (except when overwritten)
- Variants:
  - For the whole document (root element)
  - For a child node & its content (allowing to use several name spaces in one document)
  - Default namespace (no prefix applied)
    - All descendent elements assumed to be from this namespace unless specified otherwise locally for a child element

XML Namespaces

```
<?xml version="1.0" encoding="UTF-8"?>

<xhtml:html
   xmlns:xhtml="http://www.w3.org/1999/xhtml"
   xmlns:mo="http://www.nogood.com/movie">
<xhtml:head>
   <xhtml:title>Theresa Velden</xhtml:title>
</xhtml:head>
<xhtml:body>
   <xhtml:p>My hobby</xhtml:p>
   <xhtml:p>Watching Movies</xhtml:p>
   <mo:movies>
      <mo:movie>
         <mo:title>The Others</mo:title>
      </mo:movie>
   </mo:movies>
</xhtml:body>
</xhtml:html>
```

Root element declaration: holds for whole document

XML Namespaces

```
<?xml version="1.0" encoding="UTF-8"?>

<html
   xmlns="http://www.w3.org/1999/xhtml"
   xmlns:mo="http://www.nogood.com/movie">
<head>
   <title>Theresa Velden</title>
</head>
<body>
   <p>My hobby</p>
   <p>Watching Movies</p>
   <mo:movies>
      <mo:movie>
         <mo:title>The Others</mo:title>
      </mo:movie>
   </mo:movies>
</body>
</html>
```

Default name space element plus root element name space declaration.

XML Namespaces

```
<?xml version="1.0" encoding="UTF-8"?>

<html
   xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>Theresa Velden</title>
</head>
<body>
   <p>My hobby</p>
   <p>Watching Movies</p>
   <mo:movies xmlns:mo="http://www.nogood.com/movie">
      <mo:movie>
         <mo:title>The Others</mo:title>
      </mo:movie>
   </mo:movies>
</body>
</html>
```

Root element declaration: holds for whole document; and child note (incl. its content)

XML SCHEMA ET AL

XML Meta Documents

- Express constraints on an xml document:
  - What element names and attributes to use
  - How often an element may occur
  - How elements are nested (complex elements)
  - What values attributes may have
  - What content elements may have… etc.
- Examples:
  - DTD (Document Type Definition developed for SGML)
  - XML Schema
  - RELAX NG

Document Type Definition (DTD)

- Reflection of XML roots in SGML
- Problems:
  - Not extensible: can import declarations but not refine or inherit declarations
  - Document must be valid according to 1 DTD: cannot build on elements from different DTDs
  - Limited support for name spaces
  - Poor data typing; mainly intended for text
  - Not defined using XML syntax hence cannot use XML tools!

```
<!ELEMENT catalogue (movie)>

<!ELEMENT movie (title,actors)>

<!ELEMENT title (#PCDATA)>
<!ATTLIST title  lang NMTOKEN #REQUIRED>

<!ELEMENT actors (actor)+>
<!ELEMENT actor (name)>

<!ELEMENT name (#PCDATA)>
<!ATTLIST name gender NMTOKEN #REQUIRED>
```

XML Schema & RELAX NG

- XML Schema
  - W3C recommendation
  - Powerful & complex (3-part recommendation)
- RELAX NG
  - Tree constraint language written in XML (with additional compact notation)
  - Integrate well with data type libraries (such as from XML Schema)
  - Supports namespaces
  - For many purposes equivalent to XML Schema

XML Document Example

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!-- catalogue_snippet.xml  Created: 2012-09-08 17:09 -->

<catalogue>
    <movie>
        <title lang="en">The Others</title>
        <actors>
            <actor>
                <name gender="female">Nicole Mary Kidman</name>
            </actor>
            <actor>
                <name gender="female">Elaine Cassidy</name>
            </actor>
        </actors>
    </movie>
</catalogue>
```

Called in the following the "XML document instance" to distinguish it from the XML schema document (which is also an XML document)

XML Schema Basics

root element := <schema>

XML Schema (catalogue-example.xsd):

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
:
:
:
</xs:schema>
```

Name space declaration: since the schema itself is an xml document and the elements and data types used come form the W3C XML Schema namespace.

XML Schema Basics

XML Schema (catalogue-example.xsd):

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
:
:
:
</xs:schema>
```

....and in the XML Document instance (catalogue-example.xml):

```
<?xml version="1.0" encoding="UTF-8"?>

<catalogue xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xsi:noNamespaceSchemaDeclaration=”catalogue-example.xsd”>
:
:
:
</catalogue>
```

attribute (from XMLSchema-instance namespace): specifies schema to validate document against

XML Schema: defining elements

A SIMPLE element (cannot contain attributes or child elements):

```
<xs:element name="name" type="xs:string"/>
```

The 'name' attribute defines the name of an element in the XML instance document.

A COMPLEX element:

```
<xs:element name="name" type="xs:string">
  <xs:complexType >
    <xs:attribute name="gender" type="xs:string"/>
  </xs:complexType>
```

This 'name' attribute specifies the attribute name in the XML document instance.

XML Schema: defining elements

A SIMPLE element (cannot contain attributes or child elements):

```
<xs:element name="name" type="xs:string"/>
```

The 'type' attribute defines the data type of the element's content

A COMPLEX element:

```
<xs:element name="name" type="xs:string">
  <xs:complexType >
    <xs:attribute name="gender" type="xs:string"/>
  </xs:complexType>
```

This 'type' attribute specifies the data type of the attribute's value in the XML instance document.

XML Schema: Built-in Data Types

E.g.

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date, e.g. <dateborn>1970-03-27</dateborn>
- xs:time

XML Schema Example

Snippet of an XML Document instance

```
<actors>
      <actor>
            <name gender="female">Nicole Mary Kidman</name>
      </actor>
      <actor>
            <name gender="female">Elaine Cassidy</name>
      </actor>
</actors>
```

Corresponding XML Schema snippet [part 1]

```
<xs:element name="actors">
   <xs:complexType>
    <xs:sequence>
     <xs:element maxOccurs="unbounded" ref="actor"/>
    </xs:sequence>
   </xs:complexType>
 </xs:element>
```

'Order indicator': child elements have to occur in the specified order

'Occurrence indicator': specifies how often an element can be repeated

'element reference': allows to modularize the content of the schema document.

XML Schema Example

Snippet of an XML Document instance

```
<actor>
    <name gender="female">Nicole Mary Kidman</name>
</actor>
```

Corresponding XML Schema snippet [part 2]

```
<xs:element name="actor">
  <xs:complexType>
    <xs:all>
      <xs:element ref="name"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

'Order indicator:' child elements can appear in any order but may only appear once

'element reference' again

XML Schema Example

Snippet of an XML Document instance

```
<name gender="female">Nicole Mary Kidman</name>
```

Corresponding XML Schema snippet [part 3]

```
<xs:element name="name">
  <xs:complexType mixed="true">
    <xs:attribute name="gender" type="xs:string"/>
  </xs:complexType>
 </xs:element>
```

Attribute of complexType element: child element may contain elements text, and attributes

XML Schema: Indicators

Order indicators:

- All := child elements can appear in any order, and each child element must occur only once
- Choice := either one child element or another can occur
- Sequence := the child elements must appear in a specific order

Occurrence indicators: how often an element may appear or must appear

- maxOccurs
- minOccurs

Group indicators: allows to define groups of elements or attributes and reference them elsewhere in the schema

- Group name
- attributeGroup name

XML Schema

- View in XML Editor (Oxygen)

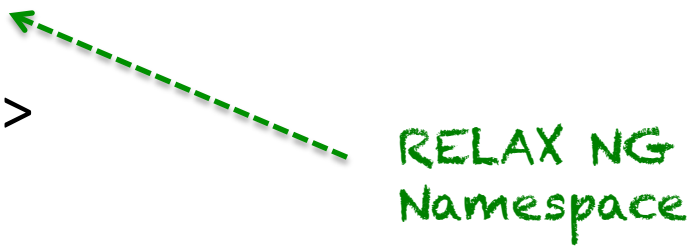
INTRODUCTION TO RELAX NG

RELAX NG Introduction

- An alternative to XML Schema
- There is an XML syntax and a compact non-XML syntax
- A REALX NG document is itself an XML document
- Integrates well with data typing libraries (such as from XML schema)
- Allows for easy referencing of chunks that can be re-used (define, ref)
- Supports namespaces
- Concept of ‘pattern’, not ‘simple’ and ‘complex types’

RELAX NG Introduction

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <start>
        <element name=”catalogue">
          .../...
        </element>
    </start>
</grammar>
```



RELAX NG Example

```
<element name="actors”>
    <oneOrMore>
      <element name="actor">
        <element name="name">
          <attribute name="gender">
            <data type=”string"/>
          </attribute>
          <text/>
        </element>
      </element>
    </oneOrMore>
</element>
```

The element ‘actors’ may have one or child elements ‘actor’

Data type of attribute value

Text node (arbitrary text content of the element node – could be empty)

XML Notation

RELAX NG Example

```
element actors {
    element actor {
        element name {
            attribute gender { xsd:string },
                text
        }
    }+
}
```

Data type of attribute value

Text node (arbitrary text content of the element node – could be empty)

The element 'actors' may have one or child elements 'actor'

Compact Non-XML Notation

RELAX NG - Modularity

```
<?xml version="1.0" encoding="UTF-8"?>

<grammar xmlns=http://relaxng.org/ns/structure/1.0
datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

<define name=“name”>
      <element name=“name”>
            <data type=“string”>
      </element>
</define>

<start>
      <element name=“cast”>
            <oneOrMore>
                  <element name=”actor">
                        <ref name=“name”>
                  </element>
            </oneOrMore>
      </element>
 </start>

</grammar>
```

Definition of a pattern

Reference to a pattern

RELAX NG

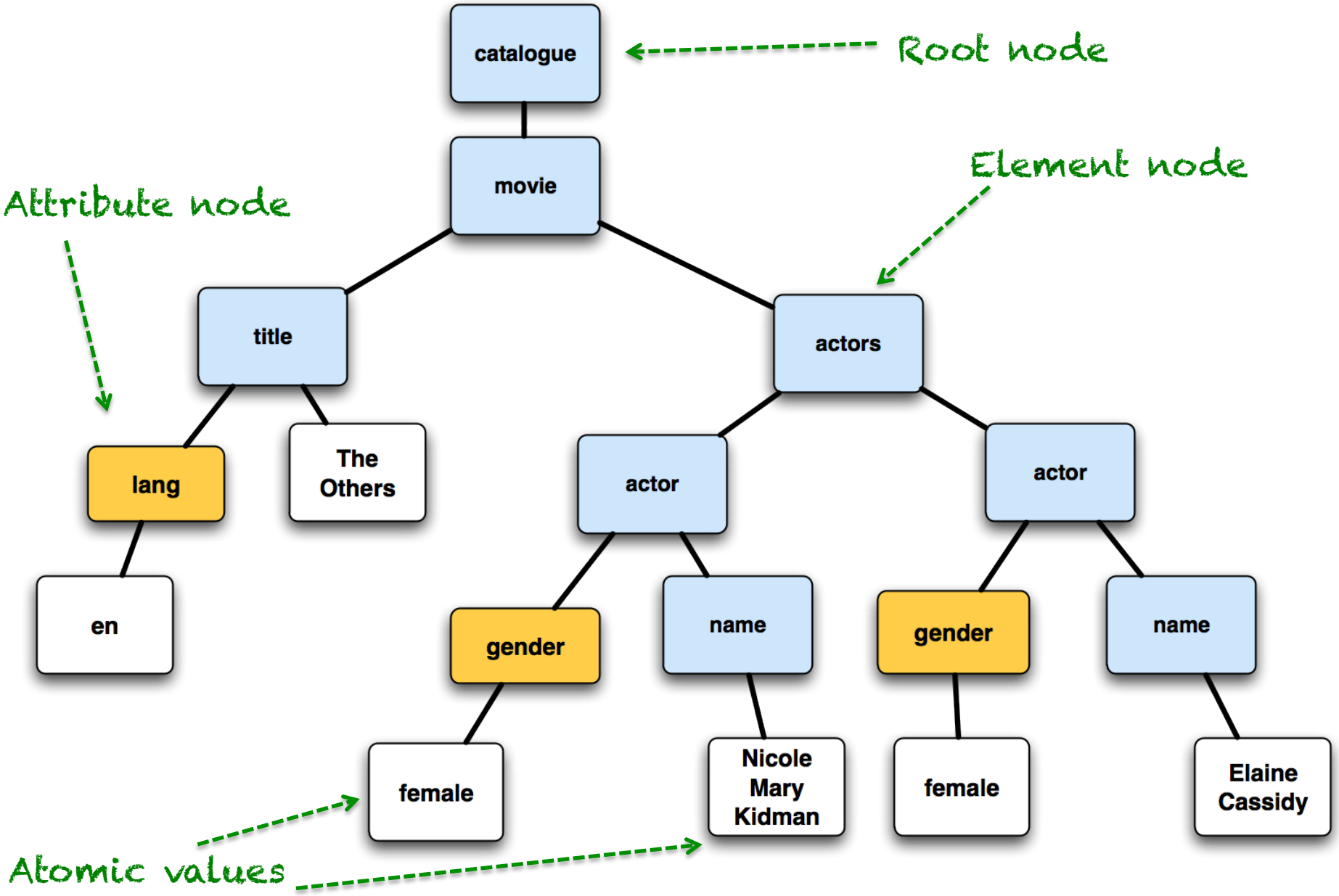
- View in XML Editor (Oxygen)

XPATH

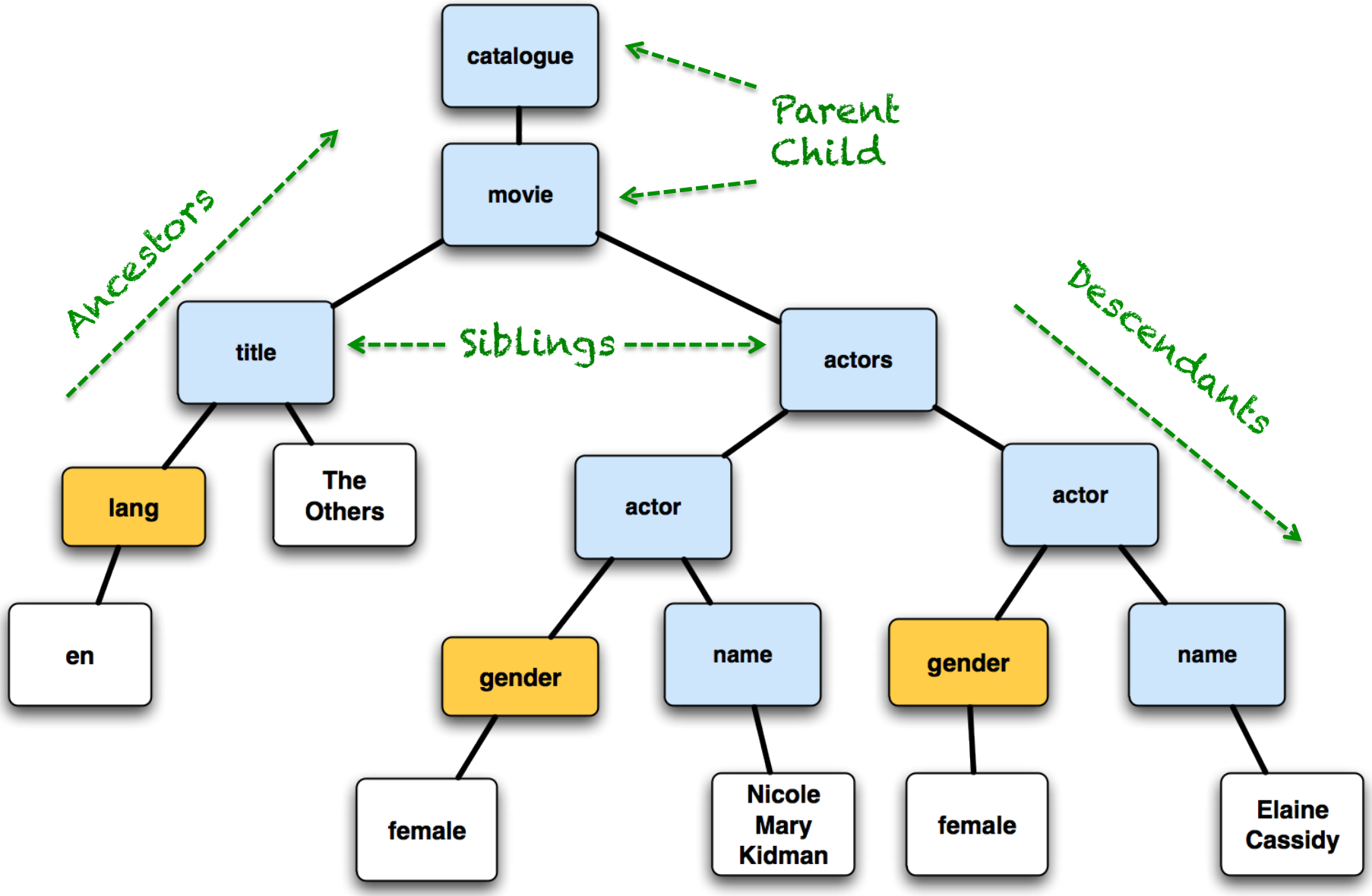
XPath

- XPath uses path expressions to select nodes or node-sets in an XML document
- It is used in XSLT (XSL Transformation)
- XPath expressions are built from
  - Location steps (one or more)
  - Predicates (one or more)

The XML Tree


catalogue
Root node
movie
Element node
Attribute node
title
actors
lang
The Others
actor
actor
en
gender
name
gender
name
female
Nicole Mary Kidman
female
Elaine Cassidy
Atomic values

The XML Tree


catalogue
movie
title
actors
lang
The Others
actor
actor
en
gender
name
gender
name
female
Nicole Mary Kidman
female
Elaine Cassidy
Parent Child
Ancestors
Siblings
Descendants

XPath: Selecting Nodes

actor : selects all nodes with the name "actor"

/catalogue : selects the root element catalogue

Note: if the path starts with a slash (/) it always represents an absolute path to an element!

. : selects the current node

.. : selects the parent of the current node

actors/actor : selects all actor elements that are children of actors

//actor : selects all actor elements no matter where they are in the document

actors//name : selects all name elements that are descendant of the actors element, no matter where they are under the actors element

//@lang : selects all attributes that are named lang

XPath: Predicates

- Predicates are used to select a specific node or a node that contains a specific value.
- Predicates are always embedded in square brackets.

XPath: Predicates

actors/actor[1]: selects the first actor element that is the child of actors

actors/actor[last()] : selects the last actor element that is the child of actors

//title[@lang] : selects all title elements that have an attribute named lang

//title[@lang='en'] : selects all title elements that have an attribute named lang with the value 'en'

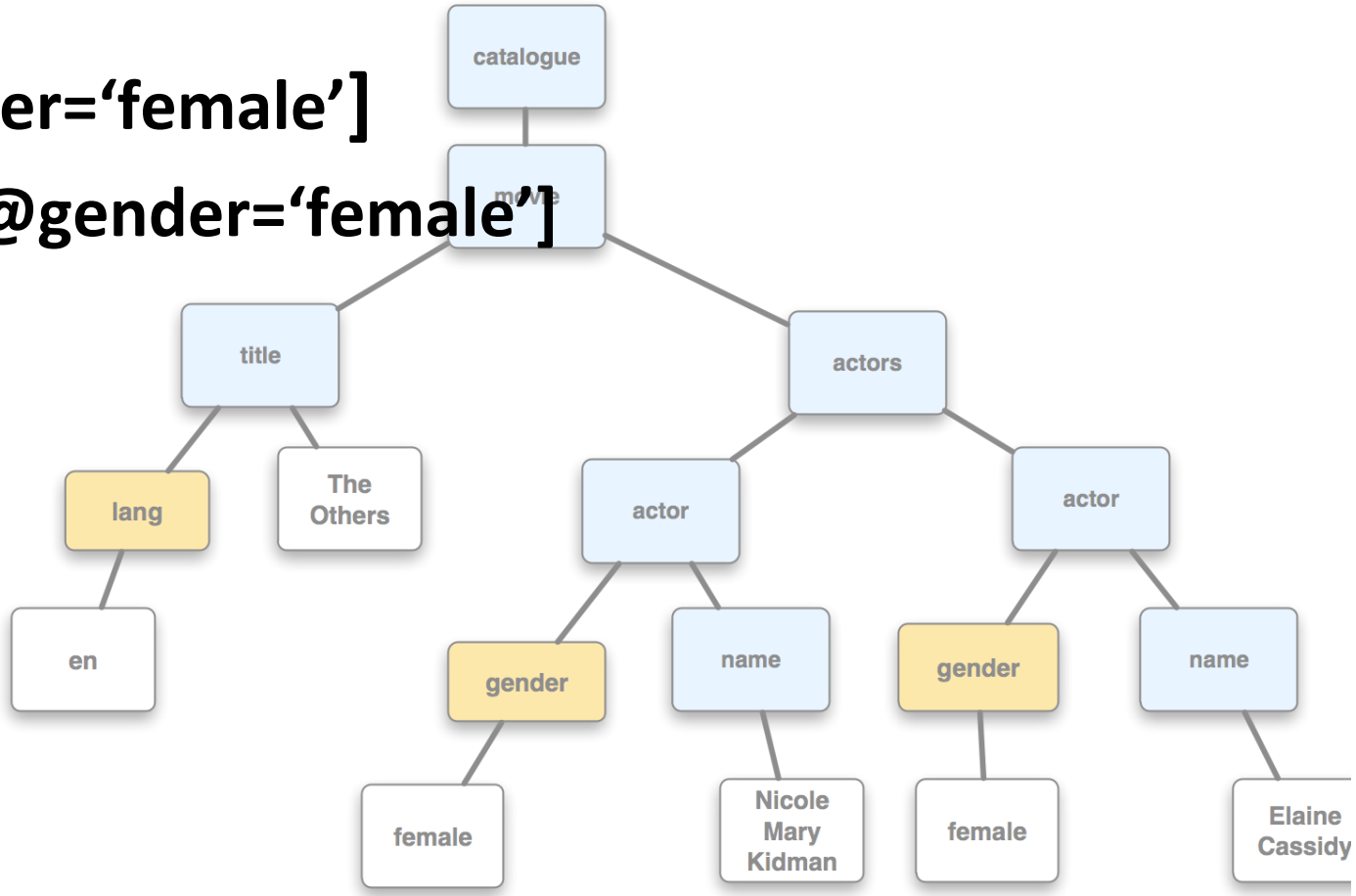
/bookstore/book[price>35.00]: Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00

XPath: Example

- All female actors in the movies in the movie catalogue:

//name[@gender='female']

//actor/name[@gender='female']



XPath Axes

- An axis defines a node-set **relative to the current node**
  - E.g. ancestor-or-self, following-sibling or preceding sibling
- Location Path Expression:
  - Can be absolute (starts with a '/') or relative
  - Consists of one or several steps
- A step: axisname::nodetest[predicate]
- Examples:
  - child::actor selects all actor elements that are children of current node
  - child::*/child::actor selects all actor grandchildren of the current node

Resources on Course Website

Tutorials

- XML
- XML Schema
- XPath
- XSLT
- RELAX NG (OASIS)
- DOM

More at W3C Schools Home.

Important Specifications

- **URI:** Uniform Resource Identifier (URI): Generic Syntax. (2005) RFC 3986
- **http:** Hypertext Transfer Protocol -- HTTP/1.1. (1999) RFC 2616
- **Web Architecture:** Architecture of the World Wide Web, Volume One. (2004) W3C Recommendation
- **HTML:** HTML 4.01 Specification. (1999) W3C Recommendation
- **XHTML:** XHTML 1.0 - A Reformulation of HTML 4 in XML 1.0. (2000/2002) W3C Recommendation
- **XML:** Extensible Markup Language (XML) 1.0 (Fifth Edition) (2008) W3C Recommendation
- **XML Namespaces:** Namespaces in XML. (2009) W3C Recommendation
- **XML Schema** (3 parts):
  - XML Schema Part 0: Primer Second Edition. (2004) W3C Recommendation
  - W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures. (2012) W3C Recommendation
  - W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. (2012) W3C Recommendation
- **XPath:** XML Path Language (XPath) 2.0 (Second Edition). (2010) W3C Recommendation
- **XSLT:** XML Transformations.,(1999) W3C Recommendation
- **RELAX NG:** RELAX-NG Home Page. (2011)

Next Week:

- Continuation of Topic Web Architecture: Structured Formats
  - XML Manipulation (XSLT), JSON/YAML
- Homework 3:
  - due on Thursday 9/20 11:59PM