References

- XQuery 1.0: An XML Query Language
  - [www.w3.org/TR/xquery/](http://www.w3.org/TR/xquery/)
- XML Query Use Cases
  - [www.w3.org/TR/xmlquery-use-cases](http://www.w3.org/TR/xmlquery-use-cases)
- Xquery Interpreter:
  - [http://prometheus2.isi.edu:9001/xquery.html](http://prometheus2.isi.edu:9001/xquery.html)
- Advice:
  - Case matters
  - Write your queries incrementally!
    - It is hard to debug entire queries
Example XML Document: bib.xml

```xml
<bib>
  <book year="1994">
    <title>TCP/IP Illustrated</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>
  <book year="1992">
    <title>Advanced Programming in the Unix environment</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>
</bib>
```
Example XML Document: bib.xml

<book year="2000">
    <title>Data on the Web</title>
    <author><last>Abiteboul</last><first>Serge</first></author>
    <author><last>Buneman</last><first>Peter</first></author>
    <author><last>Suciu</last><first>Dan</first></author>
    <publisher>Morgan Kaufmann Publishers</publisher>
    <price>39.95</price>
</book>

<book year="1999">
    <title>The Economics of Technology and Content for Digital TV</title>
    <editor><last>Gerbarg</last><first>Darcy</first><affiliation>CITI</affiliation></editor>
    <publisher>Kluwer Academic Publishers</publisher>
    <price>129.95</price>
</book>
</bib>
Xquery Overview

- Xquery is an expression language
  - Every statement evaluates to some result
    - `let $x := 5 let $y := 6 return 10*$x+$y`
      - Evaluates to 56

- Primitive types
  - Number, boolean, strings, dates, times, durations, and XML types
Nodes and Expressions

- Various functions create or return nodes
  - Document function reads an XML file
    - `doc("http://www.isi.edu/info-agents/courses/iiweb/bib.xml")`
    - We will use `doc("bib.xml")` throughout, but you must use the expansion to run the demo
  - Element constructor creates a node:
    - `<doc><par>Blah Blah</par></doc>`
  - Use curly braces to embed XQuery expressions inside an element constructor
Path Expressions

- Xquery uses path expressions from Xpath (a W3C standard)
- Let $b := \text{doc("bib.xml")}$
  return <result>{$b/bib/book}</result>
- /book selects the child elements named book
- /book/author selects the author elements of the top-level book elements
Path Expressions (cont.)

- //book
  - returns all book elements that appear anywhere in the document
- //book[author/last = “Stevens”]
  - all book elements with author = “Hunter”
- //book[@year > 1999]
  - book elements with attribute year > 1999
- //book[@pages]
  - all book elements that have a pages attribute
Advanced Path Expressions

- `//book/(author | editor)` – returns all author or editor elements from current node
- `(//book | //collection)[publisher = $pub]` – books for collections where publisher equals $pub variable
- `(//book)[1]/title/text()` – returns the text nodes of the first book element
FLWOR Expressions

- For/Let, Where, Order by, Result Expressions

```html
<html>{
  let $d := doc("bib.xml")/bib
  for $b in $d/book
  where $b/@year > 1998
  order by $b/publisher
  return <book>{$b/title, $b/price, $b/publisher}</book>
}
</html>
```
Projection

Return the names of all authors of books

let $d := \text{doc("bib.xml")}$
return $\langle \text{result}\rangle\{d/bib/book/author\}\rangle$
The same query can also be written as a for loop:

```
for $bk in doc("bib.html")/bib/book return
  for $aut in $bk/author return $aut
```

= 

```
<author><last>Stevens</last><first>W.</first></author>
<author><last>Stevens</last><first>W.</first></author>
<author><last>Abiteboul</last><first>Serge</first></author>
<author><last>Buneman</last><first>Peter</first></author>
<author><last>Suciu</last><first>Dan</first></author>
```
Selection

- Return the titles of all books published before 1997

\[\text{bib/book[@year < "1997"]/title} = \]

<title>TCP/IP Illustrated</title>
<title>Advanced Programming in the Unix environment</title>
Selection (cont.)

- Return the titles of all books published before 1997:

```
/bib/book[@year < "1997"]/title
```

```xml
define $bk in doc("bib.xml")/bib/book
  where $bk/@year < "1997"
  return $bk/title
=

<title>TCP/IP Illustrated</title>
<title>Advanced Programming in the Unix environment</title>
```
Selection (cont.)

- Return book with the title “Data on the Web”

/bib/book[title = "Data on the Web"]

= 

    <book year="2000">
        <title>Data on the Web</title>
        <author><last>Abiteboul</last><first>Serge</first></author>
        <author><last>Buneman</last><first>Peter</first></author>
        <author><last>Suciu</last><first>Dan</first></author>
        <publisher>Morgan Kaufmann Publishers</publisher>
        <price>39.95</price>
    </book>
Selection (cont.)

- Return the price of the book “Data on the Web”
  
  
  /bib/book[title = "Data on the Web"]/price

  =

  <price> 39.95</price>

How would you return the book with a price of $39.95?
Return the book with a price of $39.95
for $bk in doc("bib.xml")/bib/book
    where $bk/price = " 39.95"
    return $bk

= 

<book year="2000">
  <title>Data on the Web</title>
  <author><last>Abiteboul</last><first>Serge</first></author>
  <author><last>Buneman</last><first>Peter</first></author>
  <author><last>Suciu</last><first>Dan</first></author>
  <publisher>Morgan Kaufmann Publishers</publisher>
  <price>39.95</price>
</book>
Construction

- Return year and title of all books published before 1997

```xml
for $bk in doc("bib.xml")/bib/book
  where $bk/@year < "1997"
  return <book>{ $bk/@year, $bk/title }</book>
```

= 

```xml
<book year="1994">
  <title>TCP/IP Illustrated</title>
</book>

<book year="1992">
  <title>Advanced Programming in the Unix environment</title>
</book>
```
Grouping

- Return titles for each author

for $author in distinct-values(/bib/book/author/last) return

<author name={ $author/text() }>
  { /bib/book[author/last = $author]/title }
</author>

= 

<author name="Stevens">
  <title>TCP/IP Illustrated</title>
  <title>Advanced Programming in the Unix environment</title>
</author>

<author name="Abiteboul">
  <title>Data on the Web</title>
</author>

…
Join

Return the books that cost more at amazon than fatbrain
Let $amazon := doc(http://www.amazon.com/books.xml),
Let $fatbrain := doc(http://www.fatbrain.com/books.xml)
For $am in $amazon/books/book,
   $fat in $fatbrain/books/book
Where $am/isbn = $fat/isbn
   and $am/price > $fat/price
Return <book>{ $am/title, $am/price, $fat/price }<book>
Define function reverse ($items)
{
  let $count := count($items)
  for $i in 0 to $count
    return $items[$count - $i]
}
Reverse(1 to 5)
Note: (1 to 5) = (1, 2, 3, 4, 5)
Example Query 1

```xml
<bib>
  { for $b in doc("bib.xml")/bib/book
    where $b/publisher = "Addison-Wesley" and $b/@year > 1991
    return <book year={$b/@year }>
      { $b/title } 
    </book> } 
</bib>

What does this do?
Result Query 1

<bib>
  <book year="1994"/>
    <title>TCP/IP Illustrated</title>
  </book>
  
  <book year="1992"/>
    <title>Advanced Programming in the Unix environment</title>
  </book>
</bib>
Example Query 2

```xml
<results>
{ for $b in doc("bib.xml")/bib/book,
  $t in $b/title,
  $a in $b/author/last
  return
    <result>
      { $t }
      { $a }
    </result>
  }
</results>
```
Example Query 3

```xml
<books-with-prices>
{
    for $b in doc("http://www.bn.com/bib.xml")//book,
        $a in doc("http://www.amazon.com/reviews.xml")//entry
    where $b/title = $a/title
    return
        <book-with-prices>
            { $b/title }
            <price-amazon>{ $a/price/text() }</price-amazon>
            <price-bn>{ $b/price/text() }</price-bn>
        </book-with-prices>
}
</books-with-prices>
```
<books-with-prices>
  <book-with-prices>
    <title>TCP/IP Illustrated</title>
    <price-amazon>65.95</price-amazon>
    <price-bn>65.95</price-bn>
  </book-with-prices>
  <book-with-prices>
    <title>Advanced Programming in the Unix environment</title>
    <price-amazon>65.95</price-amazon>
    <price-bn>65.95</price-bn>
  </book-with-prices>
  <book-with-prices>
    <title>Data on the Web</title>
    <price-amazon>34.95</price-amazon>
    <price-bn>39.95</price-bn>
  </book-with-prices>
</books-with-prices>
Example Query 4

```xml
<bib>
    { for $b in doc("bib.xml")//book
      where $b/publisher = "Addison-Wesley" and $b/@year > "1991"
      order by $b/title
      return <book> { $b/@year } { $b/title } </book>
    }
</bib>
```
Example Result 4

<bib>
  <book year="1992">
    <title>Advanced Programming in the Unix environment</title>
  </book>
  <book year="1994">
    <title>TCP/IP Illustrated</title>
  </book>
</bib>