Fall 2007 **CSC 560:** Management of XML Data Alexander Dekhtyar

XQuery, Part III
Built-in Functions

Built-in Functions in XQuery

Input and Control Functions

Name, inputs	Output	Effect
`collection(xs:string)`	`node()*`	retreives all docs in given collection
`doc(xs:string)`	`node()`	retreives specified document
`doc-available(xs:string)`	`xs:boolean`	checks if document is available
`document-uri(node())`	`xs:anyURI`	URI of a document
`root(node()`	`node)?`	returns document node of the argument

Name, inputs	Output	Effect
`deep-equal(item()*,item()*`	`xs:boolean`	deep equality comparison
`distinct-values(xs:anyAtomicType*)`	`xs:anyAtomicType`	sequence of unique atomic values from arg
`error(xs:Qname?, item*)`	none	stop execution with an error message
`index-of(xs:anyAtomicType*, xs:anyAtomicType)`	`xs:integer`	returns index of the second arg in the first arg
`insert-before(item()*,xs:integer, item()*)`	`item()*`	insert third arg at given position into first arg
`matches(xs:string, xs:string, xs:string)`	`xs:boolean`	matches a regular expression
`subsequence(item()*, xs:double, xs:double)`	`item()*`	extracts a subsequence
`unordered(item()*)`	`item()*`	treat sequence as bag

Arithmetic Functions

Name, inputs	Output	Effect
`abc(numeric)`	matching `numeric`	absolute value
`ceiling(numeric)`	matching `numeric`	ceiling value
`floor(numeric)` matching `numeric`	floor value	
`round(numeric)`	`numeric`	rounds the number
`round-half-to-even(numeric)`	`numeric`	financial rounding

Aggregate Functions

Name, inputs	Output	Effect
`count(item()*`	`xs:integer`	number of items in sequence
`last()`	`xs:integer`	number of items in the current context
`max(xs:anyAtomicType*)`	`xs:anyAtomicType`	maximum item in the sequence
`min(xs:anyAtomicType*)`	`xs:anyAtomicType`	minimum item in the sequence
`sum(xs:anyAtomicType*)`	`xs:anyAtomicType`	total value of items in sequence

String Functions

Name, inputs	Output	Effect
`compare(xs:string, xs:string)`	-1, 0, 1	-1 if first arg < second arg, 0 if they are the same 1 if first arg > second arg
`concat(xs:anyAtomicType,...,xs:anyAtomicType)`	`xs:string`	concatenation
`contains(xs:string,xs:string)`	`xs:boolean`	`true` if second arg is a substring of the first arg
`ends-with(xs:string,xs:string)`	`xs:boolean`	checks if first arg ends with the second arg
`lower-case(xs:string)`	`xs:string`	convert to lowercase
`upper-case(xs:string)`	`xs:string`	convert to uppercase
`name(node()`	`xs:string`	returns name of the node
`node-name(node())`	`xs:QName`	returns qualified name of node
`normalize-space(xs:string)`	`xs:string`	normalizes whitespace
`replace(xs:string, xs:string, xs:string)`	`xs:string`	replaces 2nd arg with 3d arg in 1st arg
`starts-with(xs:string,xs:string)`	`xs:boolean`	checks if first arg has second arg as prefix
`string-join(xs:string*,xs:string)`	`xs:string`	joins strings from first arg, uses 2nd arg between them
`string-length(xs:string)`	`xs:integer`	returns length of string
`substring(xs:string, xs:double,xs:double)`	`xs:string`	extracts a substring
`substring-after(xs:string, xs:string)`	`xs:string`	extracts a substring after the first occurence of 2nd arg
`substring-after(xs:string, xs:string)`	`xs:string`	extracts a substring before the first occurence of 2nd arg
`tokenize(xs:string,xs:string)`	`xs:string*`	breaks input string into tokens
`translate(xs:string, xs:string, xs:string)`	`xs:string`	replaces chars of 2nd arg with chars of 3d arg in 1st arg

Conversion Functions

Name, inputs	Output	Effect
`data(item()*)`	`anyAtomicType`	extract typed values
`number(xs:anyAtomicType)`	`xs:double`	converts input into number
`string(item()?)`	xs:string	converts input to string

Boolean and sequence Functions

Name, inputs	Output	Effect
`boolean(item()*)`	`xs:boolean`	returns effective boolean value.
`empty(item()*)`	`xs:boolean`	checks if a sequence is empty
`exactly-one(item()*`	`xs:boolean`	checks if a sequence has exactly one item
`one-or-more(item()*`	`item()+`	returns the argument if it is not empty, otherwise, error
`zero-to-one(item()*`	`item()?`	returns the argument if it is not a compound sequence
`position()`	`xs:integer`	position of current context in the context sequence
`remove(item()*, xs:integer)`	`item()*`	removes one item from sequence
`exists(item()*`	`xs:boolean`	checks if a sequence is not empty
`false()`	`xs:boolean`	returns `false`
`true()`	`xs:boolean`	returns `true`
`not(item()*`	`xs:boolean`	boolean negation

Effective Boolean Value. Each atomic object in XQuery data model has an effective boolean value. The rules are:

- Empty sequences are `false`.

- Empty strings are `false`.
- 0 is `false`.
- sequence of more than one item raises an error.
- Non-empty strings are `true`.
- non-zero numbers are `true`.
- nodes (element, attribute) are `true`.
- path expressions are `true` if they evaluate to a nonempty element, `false` if they evaluate to an empty sequence.

Date/Time Functions

Name, inputs	Output	Effect
`current-date()`	`xs:date`	returns current date
`current-dateTime()`	`xs:dateTime`	returns current date and time
`current-time()`	`xs:time`	returns current time
`dateTime(xs:date,xs:time`	`xs:dateTime`	construct date-time value from date and time
`day-from-date(xs:date)`	`xs:integer`	returns the day portion of the date
`day-from-dateTime(xs:dateTime)`	`xs:integer`	returns the day portion of the date-time
`days-from-duration(xs:duration)`	`xs:integer`	returns the number of days from duration
`hours-from-dateTime(xs:dateTime)`	`xs:integer`	returns hours portion of day-time value
`hours-from-duration(xs:duration)`	`xs:integer`	returns hours portion of duration value
`hours-from-time(xs:time)`	`xs:integer`	returns hours portion of time value
`minutes-from-dateTime(xs:dateTime)`	`xs:integer`	returns minutes portion of dateTime value
`minutes-from-duration(xs:duration)`	`xs:integer`	returns minutes portion of duration value
`minutes-from-time(xs:time)`	`xs:integer`	returns minutes portion of time value
`months-from-date(xs:date)`	`xs:integer`	returns months portion of date value
`months-from-dateTime(xs:dateTime)`	`xs:integer`	returns months portion of dateTime value
`months-from-duration(xs:duration)`	`xs:integer`	returns months portion of duration value
`seconds-from-dateTime(xs:dateTime)`	`xs:integer`	returns seconds portion of dateTime value
`seconds-from-duration(xs:duration)`	`xs:integer`	returns seconds portion of duration value
`seconds-from-time(xs:time)`	`xs:integer`	returns seconds portion of time value
`year-from-date(xs:date)`	`xs:integer`	returns year portion of date value
`year-from-dateTime(xs:dateTime)`	`xs:integer`	returns year portion of dateTime value
`years-from-duration(xs:duration)`	`xs:integer`	returns years portion of duration value

Other Functions

Name, inputs	Output	Effect
`id(xs:string*, node())`	`element()*`	converts IDs to elements with these IDs (dereferencing)
`idref(xs:string*, node())`	`node()*`	retrieves nodes that contain given IDREFS