Cheatography

# XQuery Cheat Sheet

by BenHuf via cheatography.com/163984/cs/34363/

## Overview

XQuery is ***the*** language for querying XML data

XQuery for XML is like SQL for databases

XQuery is built on XPath expressions

XQuery is supported by all major databases

XQuery is a W3C Recommendation

XQuery is a language for finding and extracting elements and attributes from XML documents.

Here is an example of what XQuery could solve:

"Select all CD records with a price less than $10 from the CD collection stored in cd_catalog.xml"

## FLWOR Expressions

| | |
|---|---|
| For | Selects a sequence of nodes |
| Let | Binds a sequence to a variable |
| Where | Filters the nodes |
| Order by | Sorts the nodes |
| Return | What to return (gets evaluated once for every node) |

**Example:**

```
for $x in doc("bo oks.xm l")/ boo kst ore /book
where $x/pri ce>30
order by $x/title
return $x/title
```

## The For Clause

## The For Clause (cont)

```
return <te st> x={$x} and y={$y} </t est>
```

**Returns**

```
<te st>x=10 and y=100< /te st>
<te st>x=10 and y=200< /te st>
<te st>x=20 and y=100< /te st>
<te st>x=20 and y=200< /te st>
```

## The Let Clause

**This**

```
let $x := (1 to 5)
return <te st> {$x }</ tes t>
```

**Returns**

```
<te st>1 2 3 4 5</ tes t>
```

The let clause allows variable assignments and it avoids repeating the same expression many times. The let clause does not result in iteration.

## The Where Clause

**This**

where $x/price>30 and $x/price<100

**Returns** Nodes only where the price is between 30 and 100

The where clause is used to specify one or more criteria for the result.

## The order by Clause

**This**

```
for $x in doc("bo oks.xm l")/ boo kst ore /book
order by $x/@ca tegory, $x/title
return $x/title
```

**Returns**

```
<title lang="e n">Harry Potter </t itl e>
<title lang="e n">E veryday Italia n</ title>
<title lang="e n">L earning XML</t itl e>
<title lang="e n">X Query Kick Start< /ti tle>
```

The order by clause is used to specify the sort order of the result.

To loop a specific number of times in a for clause, you may use the **to** keyword.

**This**

```
for $x in (1 to 5)
return <te st> {$x }</ tes t>
```

**Returns**

```
<te st> 1</ tes t>
<te st> 2</ tes t>
<te st> 3</ tes t>
<te st> 4</ tes t>
<te st> 5</ tes t>
```

To count the iteration use the **at** keyword

**This**

```
for $x at $i in doc("bo oks.xm l")/ boo kst ore /bo o
k/ title
return <bo ok> {$i}. {data( $x) }</ boo k>
```

**Returns**

```
<bo ok>1. Everyday Italia n</ boo k>
<bo ok>2. Harry Potter </b ook>
<bo ok>3. XQuery Kick Start< /bo ok>
<bo ok>4. Learning XML</b ook>
```

it is also allowed with more than one expression in the for clause. Use comma to separate each in expression.

**This**

```
for $x in (10,20), $y in (100,200)
```



Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Cheatography

# XQuery Cheat Sheet

by BenHuf via cheatography.com/163984/cs/34363/

## The Return Clause

**This**

```
for $x in doc("bo oks.xm l")/ boo kst ore /book
return $x/title
```

**Returns**

```
<title lang="e n">E veryday Italia n</ tit le>
<title lang="e n">Harry Potter </t itl e>
<title lang="e n">X Query Kick Start< /ti tle>
<title lang="e n">L earning XML</t itl e>
```

The return clause specifies what is to be returned.

## XQuery Basics

| | | |
|---|---|---|
| doc() | doc("books.xml") | Used to open a file |
| Path Expressions | doc("books.xml")**/bookstore/book/title** | Used to navigate through elements in an XML document |
| Predicates | doc("books.xml")/bookstore/book**[price<30]** | Used to limit the extracted data |

## XQuery Basic Syntax Rules

XQuery is case-sensitive

XQuery elements, attributes, and variables must be valid XML names

An XQuery string value can be in single or double quotes

An XQuery variable is defined with a $ followed by a name, e.g. $bookstore

XQuery comments are delimited by (: and :)

```
(: XQuery Comment :)
```

## Path Expressions vs. FLWOR Expressions

Path Expression

```
doc("bo oks.xm l")/ boo kst ore /bo ok[ pri ce>
 30] /title
```

FLWOR Expression

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
return $x/title
```

These expressions yield the same result.

## XQuery Comparisons

General Comparisons: =, !=, <, <=, >, >=

```
$books tor e// book/@q > 10
```

Returns true if any q attributes have a value greater than 10

Value Comparisons: eq, ne, lt, le, gt, ge

```
bookst ore //b ook/@q gt 10
```

Returns true if there is only one q attribute returned by the expression and its value is greater than 10. If more than one 1 is returned, and error occurs

## Examples of Function Calls

**Example 1: In an element**

```
<na me> {up per -ca se( $bo okt itl e)} </n ame>
```

**Example 2: In the predicate of a path expression**

```
doc("bo oks.xm l")/ boo kst ore /bo ok[ sub str ing
1, 5)= 'Ha rry']
```

**Example 3: In a let clause**

```
let $name := (subst rin g($ boo kti tle ,1,4))
```

A call to a function can appear where an expression may appear.

## User-Defined Functions

If you cannot find the XQuery function you need, you can write your ow functions can be defined in the query or in a separate library.

**Syntax**

```
declare function prefix :fu nct ion _na me( $pa rame
as return Dat atype
{
 ...fun ction code here...
};
```

**Example: Declared in the Query**

```
declare function local: min Pri ce($p as xs:dec ima
imal?)
as xs:dec imal?
{
let $disc := ($p * $d) div 100
return ($p - $disc)
};
```

**How to call the function**




Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

## User-Defined Functions (cont)

```
<mi nPr ice >{l oca l:m inP ric e($ boo k/p ric e,$ boo k/i sc o
un t)} </m inP ric e>
```

**Note:**
Use the declare function keyword.
The name of the function must be prefixed.
The data type of the parameters are mostly the same as the data types defined in XML Schema.
The body of the function must be surrounded by curly braces.

## XQuery Terminology

| | |
|---|---|
| Nodes | In XQuery there are 7 kinds: element, attribute, text, namespace, processing-instruction, comment, and document (root) |
| Atomic Values | Atomic values are nodes with no children or parent |
| Items | Items refer to nodes and atomic values |
| Parent | Each element and attribute has one parent |
| Children | Element nodes may have 0, 1, or more children. |
| Siblings | Nodes that have the same parent |
| Ancestors | A node's parent, parent's parent, etc. |
| Descendants | A node's children, children's children, etc. |

## XQuery Conditional Expressions

If-Then-Else expressions are allowed in XQuery

```
for $x in doc("bo oks.xm l")/ boo kst ore /book
return if ($x/@c ate gor y="c hil dre n")
then <ch ild >{d ata ($x /ti tle )}< /ch ild>
else <ad ult >{d ata ($x /ti tle )}< /ad ult>
```

**Note:** "if-then-else" syntax: parentheses around the if expression are required. else is required, but it can be just else ().

## Returning HTML

```
This
for $x in doc("bo oks.xm l")/ boo kst ore /book
where $x/pri ce>30
order by $x/title
return $x/title
```

**Yields**

```
<ul>
<li ><title lang="e n">E veryday Italia n</ tit -
le> </l i>
<li ><title lang="e n">Harry Potter </t itl e>
< /li>
<li ><title lang="e n">L earning XML</t itl e>
< /li>
<li ><title lang="e n">X Query Kick Start< /ti -
tle ></ li>
</u l>
====== === === === === === === === === === === -
=== === === ======
```

**This**

```
<ul>
{
for $x in doc("bo oks.xm l")/ boo kst ore /bo -
ok/ title
order by $x
return <li >{d ata ($x )}< /li>
}
</u l>
```

**Yeilds**

```
<ul>
<li >Ev eryday Italia n</ li>
<li >Harry Potter </l i>
<li >Le arning XML</l i>
<li >XQuery Kick Start< /li>
</u l>
```

**Note:** To eliminate the title element and show only data inside the title element use data($x)



Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

## Adding Elements and Attributes to the Result

**This**

```
<ht ml>
<bo dy>
<h1 >Bo oks tor e</ h1>
<ul>
{
for $x in doc("bo oks.xm l")/ boo kst ore /book
order by $x/title
return <li >{d ata ($x /ti tle)}. Category:
{data( $x/ @ca teg ory )}< /li>
}
</u l>
</b ody>
</h tml>
```

**Yields**

```
<ht ml>
<bo dy>
<h1 >Bo oks tor e</ h1>
<ul>
<li >Ev eryday Italian. Category: COOKIN G</ li>
<li >Harry Potter. Category: CHILDR EN< /li>
<li >Le arning XML. Category: WEB</l i>
<li >XQuery Kick Start. Category: WEB</l i>
</u l>
</b ody>
</h tml>
====== === === === === === === === === === ======
```

**This**

```
<ht ml>
<bo dy>
<h1 >Bo oks tor e</ h1>
<ul>
{
```

## Adding Elements and Attributes to the Result (cont)

```
> for $x in doc("books.xml")/bookstore/book
order by $x/title
return <li class="{data($x/@category)}">{data($x/title)}</li>
}
</ul>
</body>
</html>
```

**Yields**

```
<html>
<body>
<h1>Bookstore</h1>
<ul>
<li class="COOKING">Everyday Italian</li>
<li class="CHILDREN">Harry Potter</li>
<li class="WEB">Learning XML</li>
<li class="WEB">XQuery Kick Start</li>
</ul>
</body>
</html>
```



Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com