

The Big 3 Oh!

Adam Retter

adam@existdb.org

...new

XPath

XSLT

XQuery

W3C[®]





Hey... You cant count!





XQuery 3.0



**XQuery 3.0 was
a bit quicker!**

**Just 5 years?
1.0 was 7+ years!**



New Shiny Things

Group BY

Annotations

Switch

try/catch

Windowing

HoF



Group By

- Introduces Grouping into FLWOR
 - Group on Expressions(s)
 - Group on Collation

```
let $max := 3
for $d in //bar-tab
where $d/price > $max
group by $dn := $d/drink-name
return
```

```
<drink name="{ $dn }">
```

```
    Number of these costing more than { $max } € is
```

```
{count($d)}
```

```
</drink>
```



Group By - Results

```
<drink name="hefeweissen naturub">
```

Number of these costing more than 3 3 3 € is 200

```
</drink>
```

The result is maybe not what you expected?!?

- Bindings that are not ‘grouped upon’ are sequences in the result.
 - Use `distinct-value(...)`
 - Externalise the binding before the FLOWR



Switch Expression

- Switch on an Expression
- Case statements may also have Expressions
- Note - All Expressions are atomized!

```
switch(//drink/@name)  
  case "beer" return  
    "Yes Please"  
  case "brandy" return  
    "No Thanks!"  
  case "jägermeister" return  
    "More beer first!"  
  default return  
    "Yes"
```



Windowing

- A 'view' onto the bound sequence
 - Extract/filter
 - Paging
 - Sub-sequences
 - Calculations
 - etc.
- *Very very* powerful
 - I am still learning...
 - You need to know this to win iPads!



Tumbling Window

- Each Window tumbles over the next
 - i.e. They never overlap

for tumbling window \$w in (2, 4, 6, 8, 10, 12, 14)

start at \$s when fn:true()

only end at \$e when \$e - \$s eq 2

return

```
<window>{ $w }</window>
```

```
<window>2 4 6</window>
```

```
<window>8 10 12</window>
```



Window bindings

- 9 Variables that can be bound
 - Window
 - Start:
 - Item
 - Position
 - Preceding Item (previous)
 - Following Item (next)
 - End:
 - Item
 - Position
 - Preceding Item (previous)
 - Following Item (next)



Sliding Window

- Each Window slides along the bound sequence
 - i.e. They can overlap

for sliding window \$w in (2, 4, 6, 8, 10, 12, 14)

start at \$s when fn:true()

only end at \$e when \$e - \$s eq 2

return

```
<window>{ $w }</window>
```

```
<window>2 4 6</window>
```

```
<window>4 6 8</window>
```

```
<window>6 8 10</window>
```

```
<window>8 10 12</window>
```

```
<window>10 12 14</window>
```



Annotations

- Can be applied to:
 - Prolog
 - Variable
 - Function (inline and declared)
- Enabled Private Functions

```
declare
%private
function($a) {
    <hello/>
};
```

```
declare
%public
function($a) {
    <hello/>
};
```

```
declare
%sql:get("name", "select name from people where id = 2657;")
function($name) {
```



try/catch

- Handle (and recover) from errors by catching dynamic and type errors
- Catch any or multiple errors in one step
- 7 variables that can be bound, containing error information

```
try {  
    $x cast as xs:integer  
}  
catch * {  
    0  
}
```



try/catch

```
try {  
  $x cast as xs:integer  
}  
catch err:FORG0001 | err:XPTY0004 {  
  <error>  
    <code>{$err:code}</code>  
    <value>{$err:value}</value>  
    <description>{$err:description}</description>  
    <module>{$err:module}</module>  
    <where>  
      <line>{$err:line-number}</line>  
      <col>{$err:column-number}</col>  
    </where>  
  </error>  
}
```



Higher Order Functions

- Invoke functions dynamically

```
let $f := function-lookup(xs:QName("local:my-func"), 2)
return
    $f("hello", "world")
```

- Inline functions

```
let $f := function($name as xs:string, $value as xs:string) {
    element { $name } {
        $value
    }
} return
    $f("hello", "world")
```



Higher Order Functions

- Pass functions to functions

```
declare function local:say-hello($hello-fn as function()) {  
    <hello>{$hello-fn()}</hello>  
};  
let $f := function() {  
    concat("bob at ", current-date())  
} return  
    local:say-hello($f)
```

- Partial Function Application

```
let $drinks-fn := function($name as xs:string,  
$drink as xs:string) {  
    concat($name, " drinks ", $drink)  
} return  
    let $beer-drinker := $drinks-fn(?, "Beer")  
return  
    $beer-drinker("adam")
```



XQuery 1.0



made me feel like...



But, with
XQuery 3.0...

