Reading XML: Introducing XPath

Andrew Walker

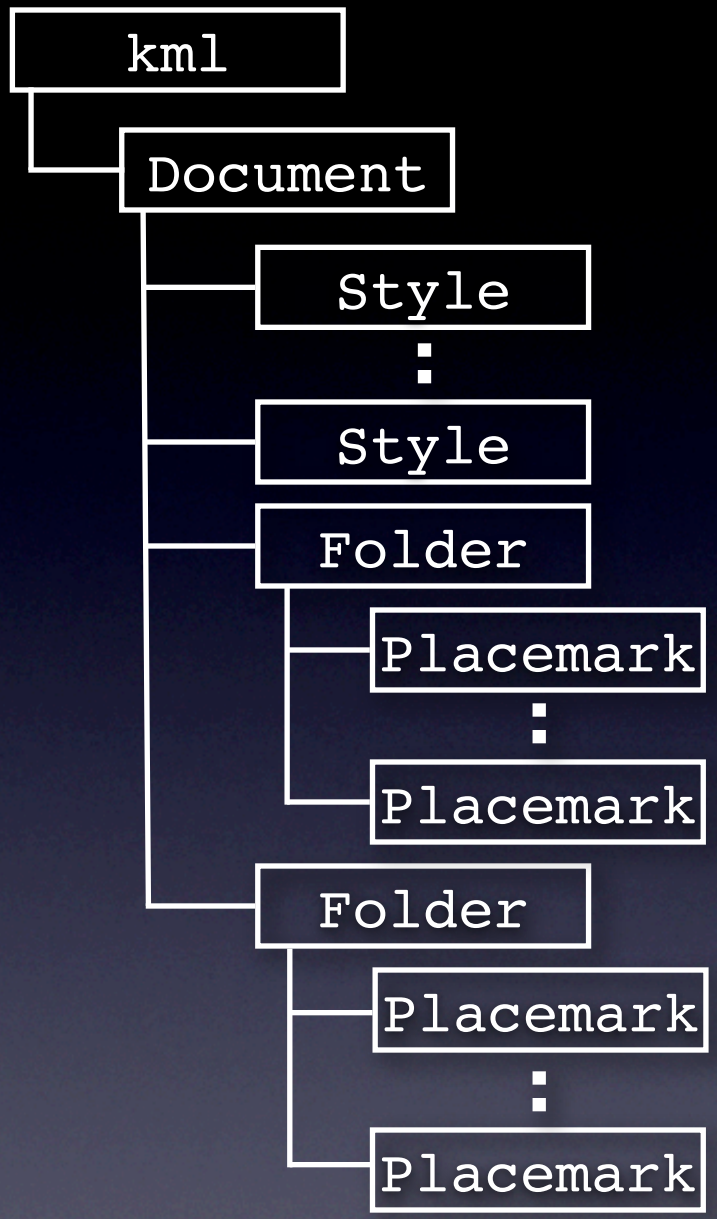
Why?

- Makes writing analysis scripts for your XML encoded results easy.
- Is used by many other XML technologies. XSLT, XPointer etc.
- A Fortran XPath API is in development.
- It helps you design your XML documents.

```
#! /usr/bin/perl -w
$num = '[+-]?\d+\.?\d*';
while (<>) {
  if (/temperature = ($num)/) {
    $temp = $1;
  } elsif (/pressure = ($num)/) {
    $pres = $1;
  }
}
...
```

```
#! /usr/bin/perl -w

use XML::XPath;

my $xp = XML::XPath
  ->new(filename => $ARGV[0]);

$temp = $xp
  -> find("//temperature");

$pres = $xp
  -> find("//pressure");

...
```

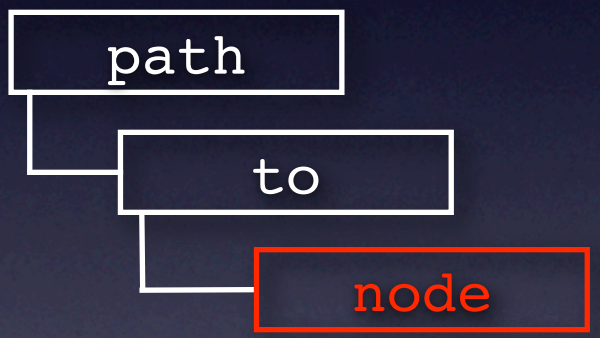
/XPath/queries/are/like/paths




```
<kml>
    <Document>
        <Style/>
        <Style/>
        <Folder>
          <Placemark/>
          <Placemark/>
        </Folder>
        <Folder>
          <Placemark/>
          <Placemark/>
        </Folder>
    </Document>
</kml>
```

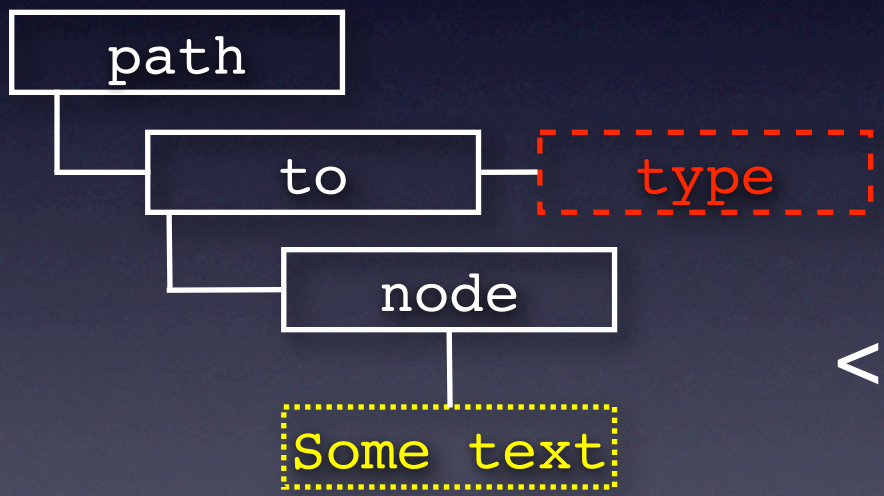
Elements are nodes

/path/to/node




```
<path>
    <to>
        <node/>
    </to>
<path>
```

Everything is a node




```
<path>
    <to type='foo'>
        <node>Some text</node>
    </to>
<path>
```

comments

elements

attributes

the root node

Types of node

text nodes

processing instructions

namespace nodes

/path/to/node/@attribute

- / -- the root node
- elementName -- an element
- @attributeName -- an attribute
- text() -- a text node

/text/to/node/text()

boolean: true or false

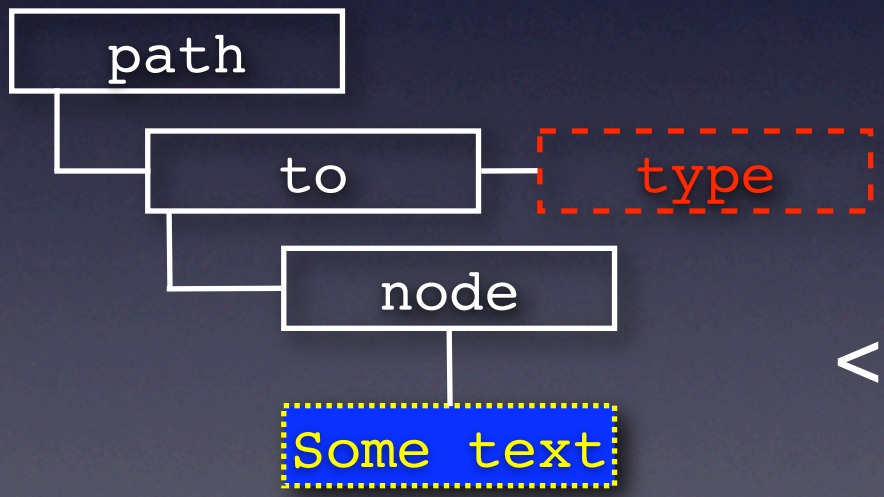
string: ‘some text in quotes’

Data types - What can be used or returned?

number: -24.451

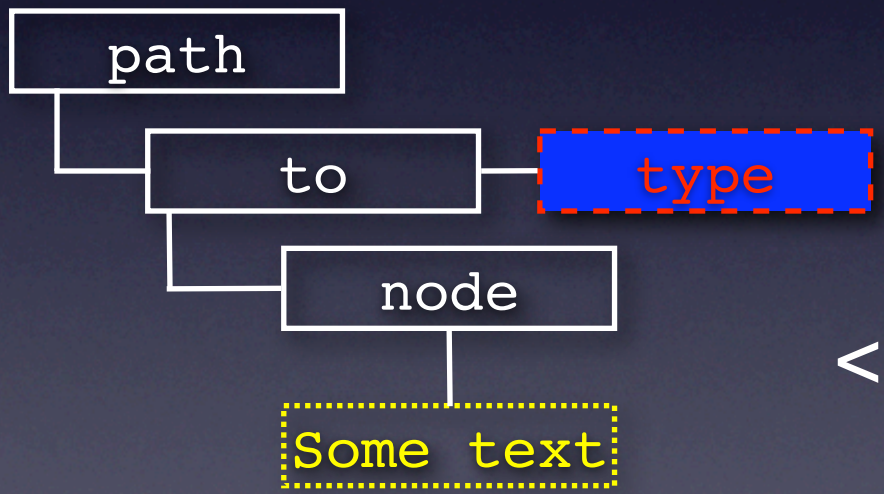
nodeset: one or more nodes

/path/to/node/text()
gives string: ‘Some text’



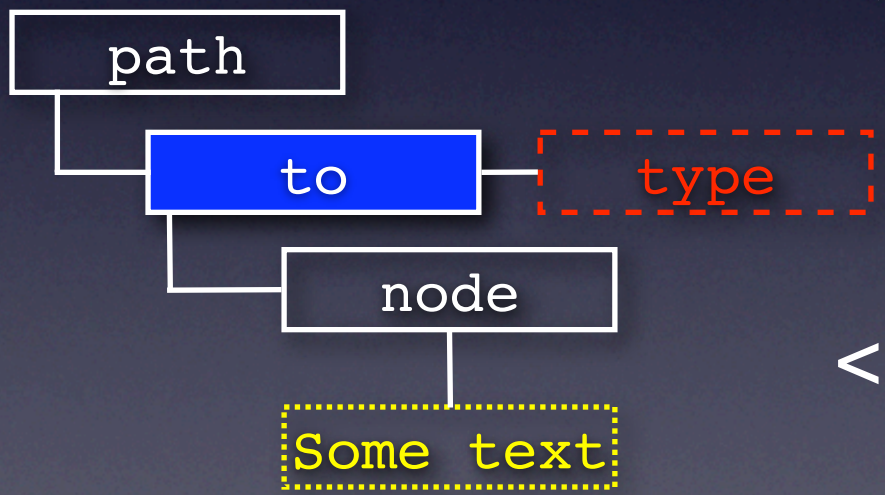

```
<path>
    <to type='foo'>
        <node>Some text</node>
    </to>
<path>
```

/path/to/@type gives string: ‘foo’




```
<path>
    <to type='foo'>
        <node>Some text</node>
    </to>
<path>
```

/path/to
gives: a nodeset




```
<path>
    <to type='foo'>
        <node>Some text</node>
    </to>
<path>
```

string(anytype)

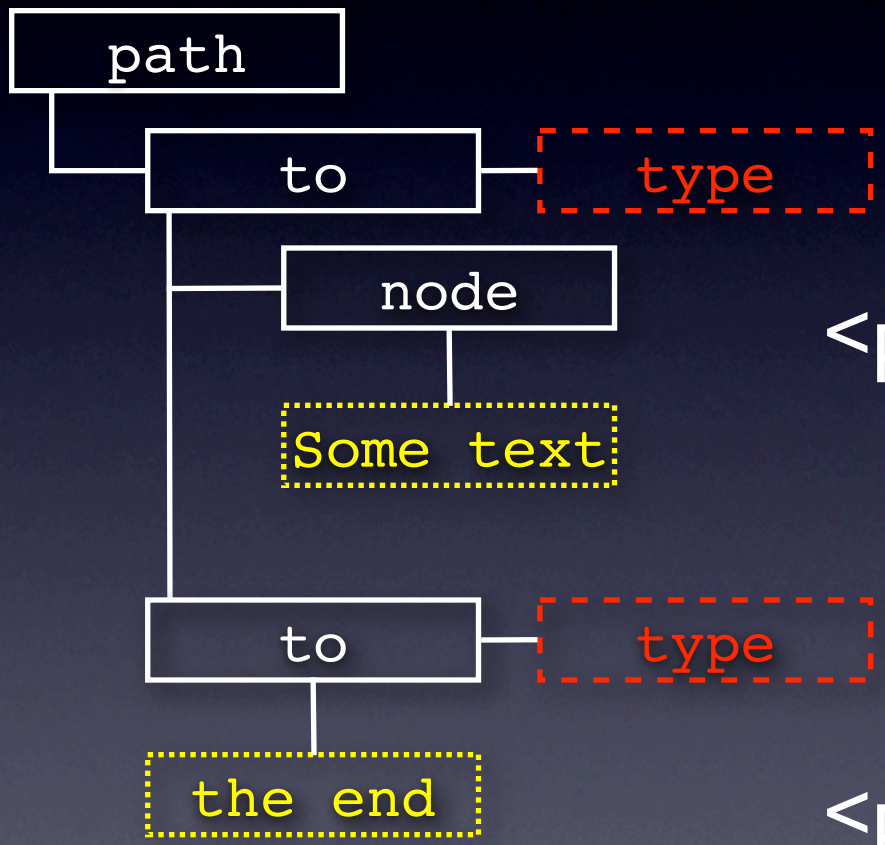
normalize-space(string)

(Some) functions

count(nodeset)

namespace-uri(nodeset)

count(/path/to)
gives: number 2




```
<path>
    <to type='foo'>
        <node>Some text</node>
    </to>
    <to> type='bar'>the end</to>
<path>
```

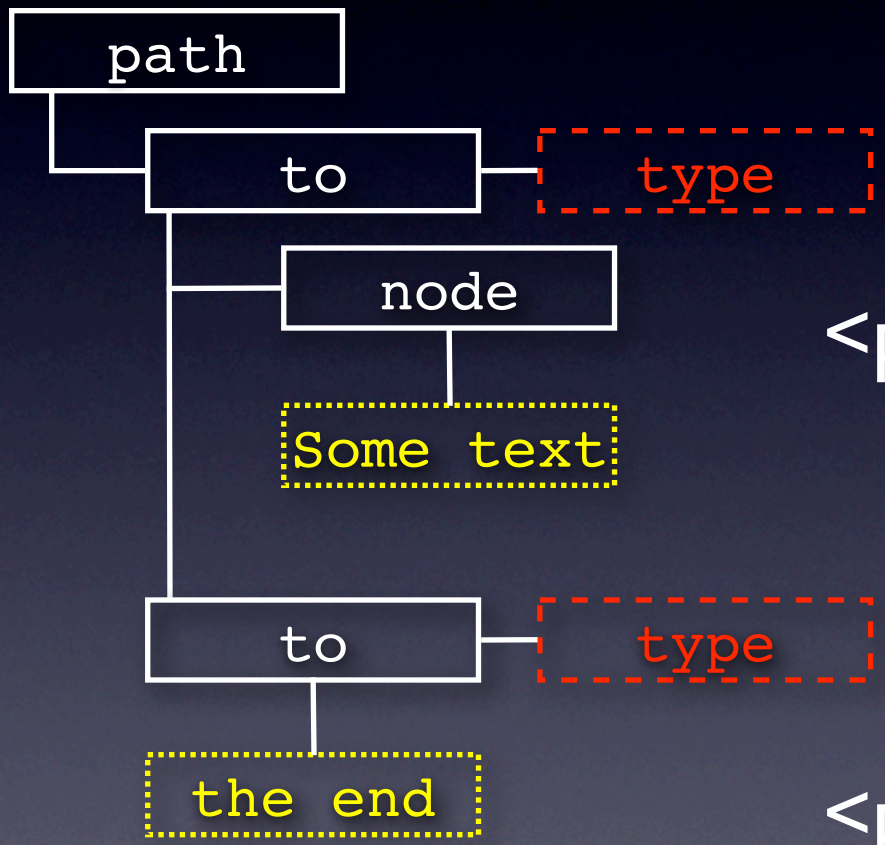
Predicates

Path matches query if predicate is true

/path/to[@type='foo']

Included in square brackets in query

/path/to[@type='bar']/text()
gives: string 'the end'




```
<path>
    <to type='foo'>
        <node>Some text</node>
    </to>
    <to> type='bar'>the end</to>
<path>
```

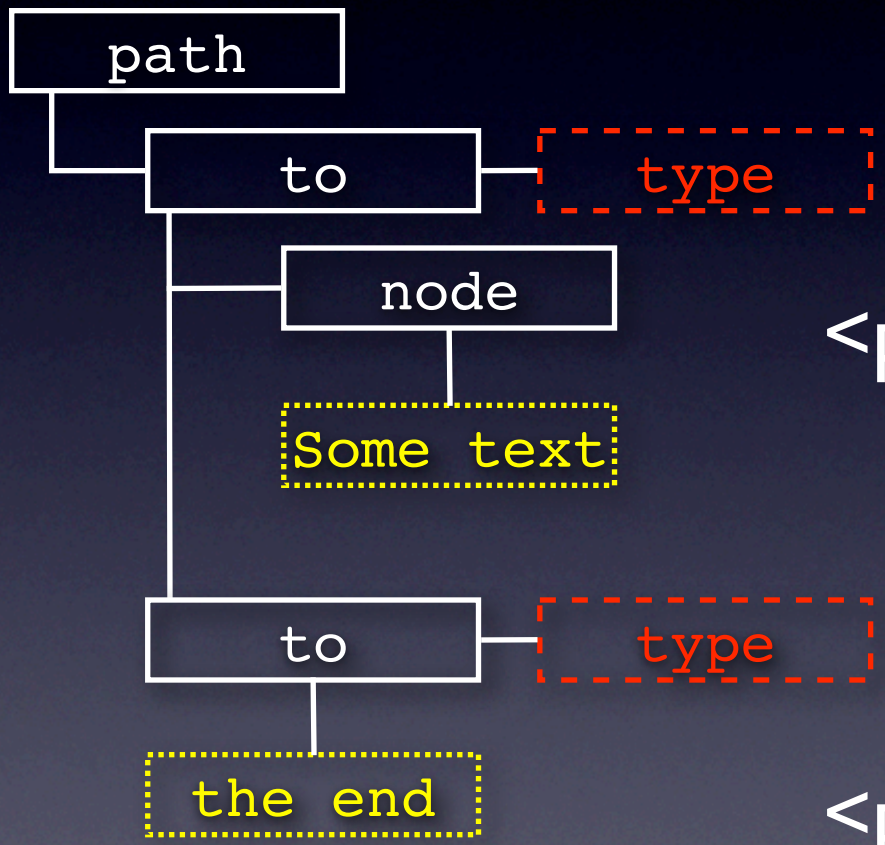
XPath in python

```
import lxml.etree

docRoot = lxml.etree.parse(source="monty.xml")

answer = docRoot.xpath("/film/@name")


print answer
```

Namespaces

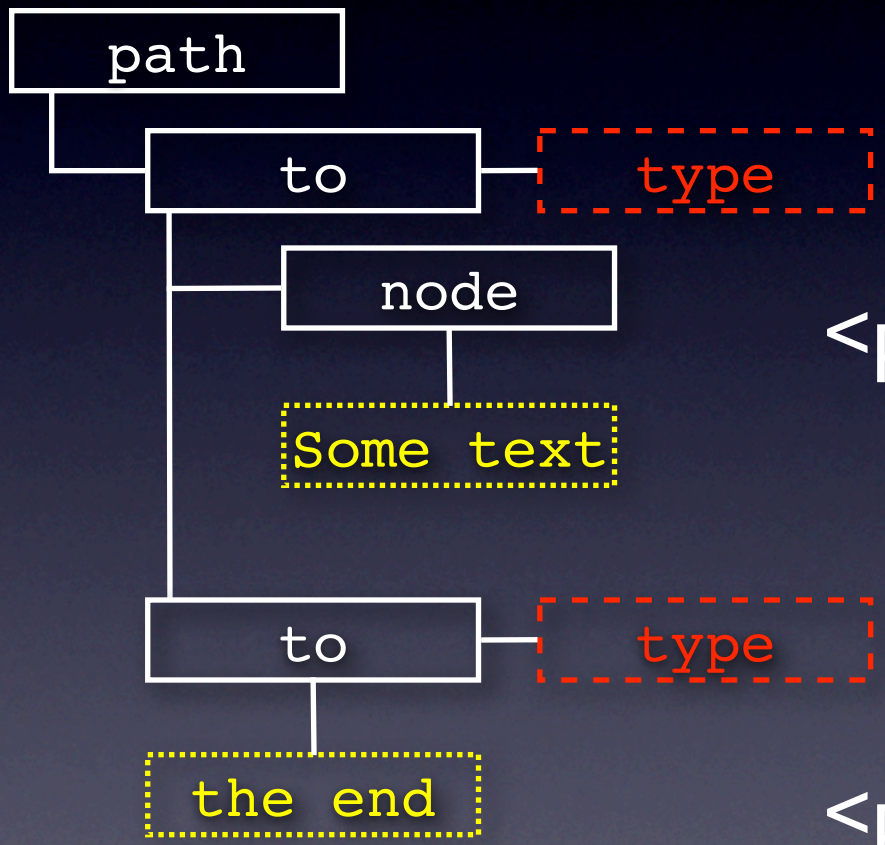
- Create a dictionary relating namespace URIs to local namespace prefix.
- Pass dictionary to XPath function.
- Include prefix before all elements separated by a colon.
- No inheritance in query!

count(/ns:path/ns:to)
gives: number 2




```
<path xmlns='name1'>
    <to type='foo'>
        <node>Some text</node>
    </to>
    <to> type='bar'>the end</to>
<path>
```

count(/ns:path/ns2:to) gives: number 1




```
<path xmlns='name1'>
    <to type='foo' xmlns='name2'>
        <node>Some text</node>
    </to>
    <to> type='bar'>the end</to>
<path>
```

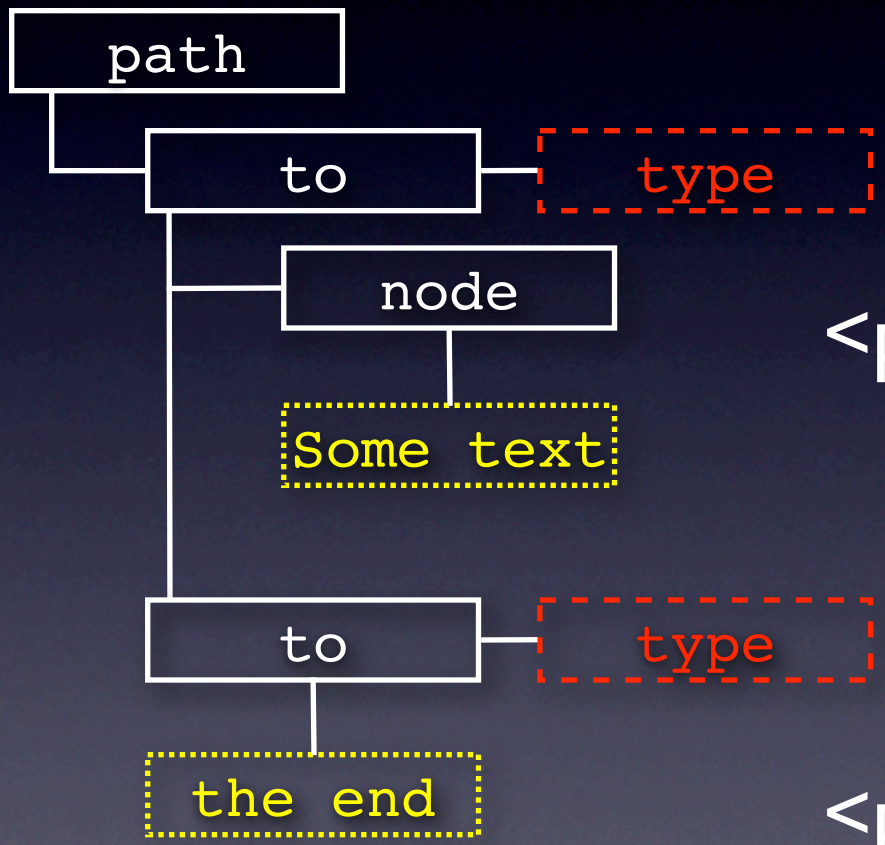
Namespaces in python

```
import lxml.etree

namespaces =
  {'q':'http://www.example.com/quotes',
   'f':'http://www.example.com/films'}

docRoot = lxml.etree.parse(source="monty.xml")

answer = docRoot.xpath
   ("/q:film/@name", namespaces)

print answer
```

Delimiters

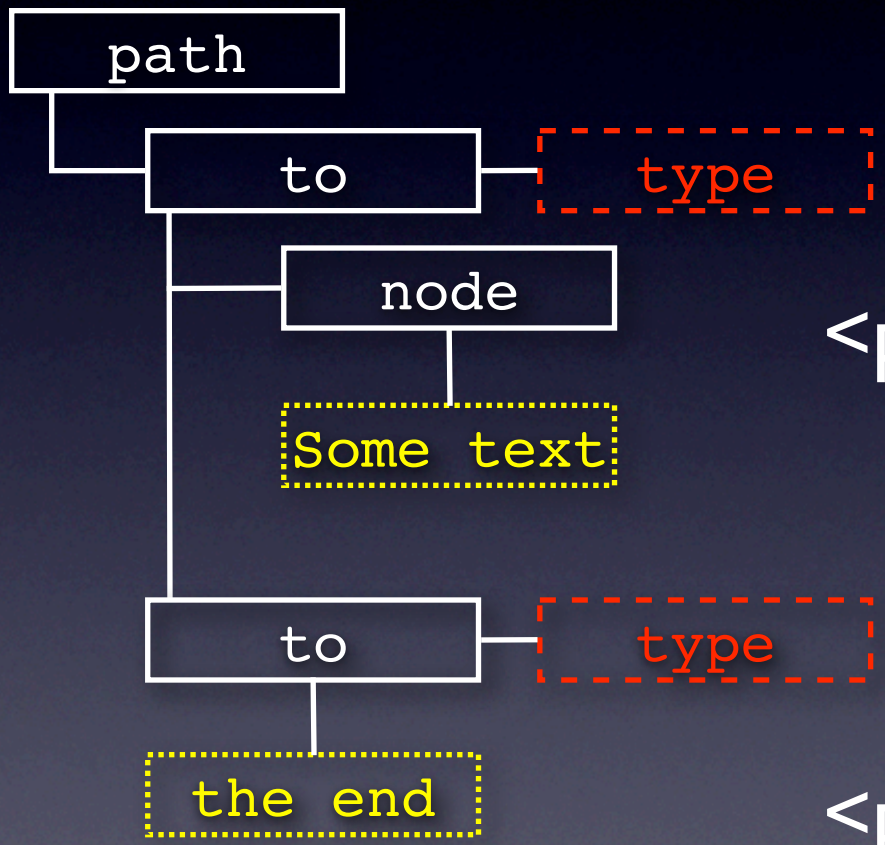
- / -- separates location steps
- // -- descendent or self
- .. -- parent
- . -- self
- [ ] -- predicate
- @ -- attribute

count(//ns:to)
gives: number 2




```
<path xmlns='name1'>
    <to type='foo'>
        <node>Some text</node>
    </to>
    <to> type='bar'>the end</to>
<path>
```

//ns:to/ns:node/text()
gives: string ‘Some text’




```
<path xmlns=’name1’>
    <to type=’foo’>
        <node>Some text</node>
    </to>
    <to> type=’bar’>the end</to>
<path>
```

Relative paths

```
import lxml.etree

docRoot = lxml.etree.parse(source="films.xml")

nodes = docRoot.xpath("//film[@type='comedy']")


for node in nodes:
  answer = node.xpath("name/text()")
  print answer
```

numeric valued predicates
and the document order

more functions

What have I not covered

unusual nodes

the axis

XPath APIs

- Java: standard API provided by javax.xml.xpath as of Java 5
- Perl: CPAN module XML::XPath
- Python: lxml.xpath http://codespeak.net/lxml
- C and bindings to other languages: libxml at http://xmlsoft.org/