XPath 简明教程

资料来源：W3School

目 录

0 XPath 教程

XPath 是一门在 XML 文档中查找信息的语言。XPath 可用来在 XML 文档中对元素和属性进行遍历。

XPath 是 W3C XSLT 标准的主要元素，并且 XQuery 和 XPointer 同时被构建于 XPath 表达之上。

因此，对 XPath 的理解是很多高级 XML 应用的基础。

1 XPath 简介

XPath 是一门在 XML 文档中查找信息的语言。XPath 用于在 XML 文档中通过元素和属性进行导航。

在学习之前应该具备的知识：

在您继续学习之前，应该对下面的知识有基本的了解：

- HTML / XHTML
- XML / XML 命名空间

如果您希望首先学习这些项目，请在我们的 首页 访问这些教程。

1.1 什么是 XPath?

- XPath 使用路径表达式在 XML 文档中进行导航
- XPath 包含一个标准函数库
- XPath 是 XSLT 中的主要元素
- XPath 是一个 W3C 标准

1.2 XPath 路径表达式

XPath 使用路径表达式来选取 XML 文档中的节点或者节点集。这些路径表达式和我们在常规的电脑文件系统中看到的表达式非常相似。

1.3 XPath 标准函数

XPath 含有超过 100 个内建的函数。这些函数用于字符串值、数值，日期和时间比较、节点和 QName 处理、序列处理、逻辑值等等。

1.4 XPath 在 XSLT 中使用

XPath 是 XSLT 标准中的主要元素。如果没有 XPath 方面的知识，您就无法创建 XSLT 文档。

您可以在我们的《XSLT 教程》中阅读更多的内容。

XQuery 和 XPointer 均构建于 XPath 表达式之上。XQuery 1.0 和 XPath 2.0 共享相同的数据模型，并支持相同的函数和运算符。

您可以在我们的《XQuery 教程》中阅读更多有关 XQuery 的知识。

1.5 XPath 是 W3C 标准

XPath 于 1999 年 11 月 16 日 成为 W3C 标准。

XPath 被设计供 XSLT、XPointer 以及其他 XML 解析软件使用。

您可以在我们的《W3C 教程》中阅读更多有关 XPath 标准的信息。

2 XPath 节点

在 XPath 中，有七种类型的节点：元素、属性、文本、命名空间、处理指令、注释以及文档节点（或成为根节点）。

2.1 XPath 术语

2.1.1 节点（Node）

在 XPath 中，有七种类型的节点：元素、属性、文本、命名空间、处理指令、注释以及文档（根）节点。XML 文档是被作为节点树来对待的。树的根被称为文档节点或者根节点。

请看下面这个 XML 文档：

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>

<book>

  <title lang="en">Harry Potter</title>

  <author>J K. Rowling</author>

  <year>2005</year>

  <price>29.99</price>
```

```
</book>

</bookstore>
```

上面的 XML 文档中的节点例子：

```
<bookstore> （文档节点）

<author>J K. Rowling</author> （元素节点）

lang="en" （属性节点）
```

2.1.2 基本值（或称原子值，Atomic value）

基本值是无父或无子的节点。

基本值的例子：

```
J K. Rowling

"en"
```

2.1.3 项目（Item）

项目是基本值或者节点。

2.2 节点关系

2.2.1 父（Parent）

每个元素以及属性都有一个父。

在下面的例子中，book 元素是 title、author、year 以及 price 元素的父：

```
<book>

  <title>Harry Potter</title>

  <author>J K. Rowling</author>

  <year>2005</year>
```

```
  <price>29.99</price>
</book>
```

2.2.2 子（Children）

元素节点可有零个、一个或多个子。

在下面的例子中，title、author、year 以及 price 元素都是 book 元素的子：

```
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

2.2.3 同胞（Sibling）

拥有相同的父的节点

在下面的例子中，title、author、year 以及 price 元素都是同胞：

```
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

2.2.4 先辈（Ancestor）

某节点的父、父的父，等等。

在下面的例子中，title 元素的先辈是 book 元素和 bookstore 元素：

```
<bookstore>
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
</bookstore>
```

2.2.5 后代（Descendant）

某个节点的子，子的子，等等。

在下面的例子中，bookstore 的后代是 book、title、author、year 以及 price 元素：

```
<bookstore>
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
</bookstore>
```

3 XPath 语法

XPath 使用路径表达式来选取 XML 文档中的节点或节点集。节点是通过沿着路径 (path) 或者步 (steps) 来选取的。

3.1 XML 实例文档

我们将在下面的例子中使用这个 XML 文档。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
<book>
  <title lang="eng">Harry Potter</title>
  <price>29.99</price>
</book>
<book>
  <title lang="eng">Learning XML</title>
  <price>39.95</price>
</book>
</bookstore>
```

3.2 选取节点

XPath 使用路径表达式在 XML 文档中选取节点。节点是通过沿着路径或者 step 来选取的。

下面列出了最有用的路径表达式：

表达式	描述
nodename	选取此节点的所有子节点
/	从根节点选取
//	从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置
.	选取当前节点
..	选取当前节点的父节点
@	选取属性

实例

在下面的表格中，我们已列出了一些路径表达式以及表达式的结果：

路径表达式	结果
bookstore	选取 bookstore 元素的所有子节点
/bookstore	选取根元素 bookstore 注释：假如路径起始于正斜杠(/)，则此路径始终代表到某元素的绝对路径！
bookstore/book	选取所有属于 bookstore 的子元素的 book 元素。
//book	选取所有 book 子元素，而不管它们在文档中的位置。
bookstore//book	选择所有属于 bookstore 元素的后代的 book 元素，而不管它们位于 bookstore 之下的什么位置。
//@lang	选取所有名为 lang 的属性。

3.3 谓语（Predicates）

谓语用来查找某个特定的节点或者包含某个指定的值的节点。

谓语被嵌在方括号中。

实例

在下面的表格中，我们列出了带有谓语的一些路径表达式，以及表达式的结果：

路径表达式	结果
/bookstore/book[1]	选取属于 bookstore 子元素的第一个 book 元素。
/bookstore/book[last()]	选取属于 bookstore 子元素的最后一个 book 元素。
/bookstore/book[last()-1]	选取属于 bookstore 子元素的倒数第二个 book 元素。
/bookstore/book[position()<3]	选取最前面的两个属于 bookstore 元素的子元素的 book 元素。
//title[@lang]	选取所有拥有名为 lang 的属性的 title 元素。
//title[@lang='eng']	选取所有 title 元素，且这些元素拥有值为 eng 的 lang 属性。
/bookstore/book[price>35.00]	选取所有 bookstore 元素的 book 元素，且其中的 price 元素的值须大于 35.00。
/bookstore/book[price>35.00]/title	选取所有 bookstore 元素中的 book 元素的 title 元素，且其中的 price 元素的值须大于 35.00。

3.4 选取未知节点

XPath 通配符可用来选取未知的 XML 元素。

通配符	描述
*	匹配任何元素节点
@*	匹配任何属性节点
node()	匹配任何类型的节点

实例

在下面的表格中，我们列出了一些路径表达式，以及这些表达式的结果：

路径表达式	结果
/bookstore/*	选取 bookstore 元素的所有子节点
//*	选取文档中的所有元素
//title[@*]	选取所有带有属性的 title 元素。

3.5 选取若干路径

通过在路径表达式中使用“|”运算符，您可以选取若干个路径。

实例

在下面的表格中，我们列出了一些路径表达式，以及这些表达式的结果：

路径表达式	结果
//book/title | //book/price	选取所有 book 元素的 tilte 和 price 元素。
//title | //price	选取所有文档中的 title 和 price 元素。
/bookstore/book/title | //price	选取所有属于 bookstore 元素的 book 元素的 title 元素，以及文档中所有的 price 元素。

4 XPath Axes（坐标轴）

4.1 XML 实例文档

我们将在下面的例子中使用此 XML 文档：

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>

<book>

  <title lang="eng">Harry Potter</title>

  <price>29.99</price>

</book>

<book>

  <title lang="eng">Learning XML</title>

  <price>39.95</price>

</book>

</bookstore>
```

4.2 XPath 轴

轴可定义某个相对于当前节点的节点集。

轴名称	结果
ancestor	选取当前节点的所有先辈（父、祖父等）
ancestor-or-self	选取当前节点的所有先辈（父、祖父等）以及当前节点本身
attribute	选取当前节点的所有属性
child	选取当前节点的所有子元素。
descendant	选取当前节点的所有后代元素（子、孙等）。
descendant-or-self	选取当前节点的所有后代元素（子、孙等）以及当前节点本身。
following	选取文档中当前节点的结束标签之后的所有节点。
namespace	选取当前节点的所有命名空间节点

parent	选取当前节点的父节点。
preceding	选取文档中当前节点的开始标签之前的所有节点。
preceding-sibling	选取当前节点之前的所有同级节点。
self	选取当前节点。

4.3 位置路径表达式

位置路径可以是绝对的，也可以是相对的。
绝对路径起始于正斜杠(/)，而相对路径不会这样。在两种情况中，位置路径均包括一个或多个步，每个步均被斜杠分割：

绝对位置路径：

```
/step/step/...
```

相对位置路径：

```
step/step/...
```

每个步均根据当前节点集之中的节点来进行计算。

步（step）包括：

轴（axis）

定义所选节点与当前节点之间的树关系

节点测试（node-test）

识别某个轴内部的节点

零个或者更多谓语（predicate）

更深入地提炼所选的节点集

步的语法：

```
轴名称::节点测试[谓语]
```

实例

例子	结果
child::book	选取所有属于当前节点的子元素的 book 节点

attribute::lang	选取当前节点的 lang 属性
child::*	选取当前节点的所有子元素
attribute::*	选取当前节点的所有属性
child::text()	选取当前节点的所有文本子节点
child::node()	选取当前节点的所有子节点
descendant::book	选取当前节点的所有 book 后代
ancestor::book	选择当前节点的所有 book 先辈
ancestor-or-self::book	选取当前节点的所有 book 先辈以及当前节点（假如此节点是 book 节点的话）
child::*/child::price	选取当前节点的所有 price 孙。

5 XPath 运算符

XPath 表达式可返回节点集、字符串、逻辑值以及数字。

5.1 XPath 运算符

下面列出了可用在 XPath 表达式中的运算符：

运算符	描述	实例	返回值
|	计算两个节点集	//book | //cd	返回所有带有 book 和 ck 元素的节点集
+	加法	6 + 4	10
-	减法	6 - 4	2
*	乘法	6 * 4	24
div	除法	8 div 4	2
=	等于	price=9.80	如果 price 是 9.80，则返回 true。 如果 price 是 9.90，则返回 fasle。
!=	不等于	price!=9.80	如果 price 是 9.90，则返回 true。 如果 price 是 9.98，则返回 fasle。
<	小于	price<9.80	如果 price 是 9.00，则返回 true

			如果 price 是 9.98，则返回 fasle
<=	小于或等于	price<=9.80	如果 price 是 9.00，则返回 true。 如果 price 是 9.90，则返回 fasle。
>	大于	price>9.80	如果 price 是 9.90，则返回 true。 如果 price 是 9.80，则返回 fasle。
>=	大于或等于	price>=9.80	如果 price 是 9.90，则返回 true。 如果 price 是 9.70，则返回 fasle。
or	或	price=9.80 or price=9.70	如果 price 是 9.80，则返回 true。 如果 price 是 9.50，则返回 fasle。
and	与	price>9.00 and price<9.90	如果 price 是 9.80，则返回 true。 如果 price 是 8.50，则返回 fasle。
mod	计算除法的余数	5 mod 2	1

6 XPath 实例

在本节，让我们通过实例来学习一些基础的 XPath 语法。

6.1 XML 实例文档

我们将在下面的例子中使用这个 XML 文档：

"books.xml"：

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>

<book category="COOKING">

  <title lang="en">Everyday Italian</title>

  <author>Giada De Laurentiis</author>

  <year>2005</year>

  <price>30.00</price>

</book>

<book category="CHILDREN">
```

```
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>
<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>
```

在您的浏览器中查看此 "books.xml" 文件。

6.2 节点选取

我们将使用微软的 XML DOM 对象来载入 XML 文档，并使用 selectNodes() 函数从 XML 文档选取节点：

```
set xmlDoc=CreateObject("Microsoft.XMLDOM")
```

```
xmlDoc.async="false"

xmlDoc.load("books.xml")

xmlDoc.selectNodes(路径表达式)
```

选取所有的 book 节点

下面的这个例子选取了 bookstore 元素下所有的 book 节点：

```
xmlDoc.selectNodes("/bookstore/book")
```

假如您正在使用 IE 5 或更高的版本，可以亲自试一试。

选取第一个 book 节点

下面的例子仅选取 bookstore 元素下第一个 book 节点：

```
xmlDoc.selectNodes("/bookstore/book[0]")
```

假如您正在使用 IE 5 或更高的版本，可以亲自试一试。

注释：IE 5 和 6 会把 [0] 作为第一个节点来执行，但是根据 W3C 的标准，应该使用 [1] ！！

注释：这个问题在 IE 6 SP2 中被纠正了！

选取 price

下面的例子从所有的 price 节点选取文本：

```
xmlDoc.selectNodes("/bookstore/book/price/text()")
```

假如您正在使用 IE 5 或更高的版本，可以亲自试一试。

选取价格高于 35 的 price 价格

下面的例子会选取所有价格高于 35 的 price 节点：

```
xmlDoc.selectNodes("/bookstore/book[price>35]/price")
```

假如您正在使用 IE 5 或更高的版本，可以亲自试一试。

选取价格高于 35 的 title 节点

下面的例子会选取所有价格高于 35 的 title 节点：

```
xmlDoc.selectNodes("/bookstore/book[price>35]/title")
```

假如您正在使用 IE 5 或更高的版本，可以亲自试一试。

7 XPath、XQuery 以及 XSLT 函数

7.1 存取函数

名称	说明
fn:node-name(node)	返回参数节点的节点名称。
fn:nilled(node)	返回是否拒绝参数节点的布尔值。
fn:data(item.item,...)	接受项目序列，并返回原子值序列。
fn:base-uri() fn:base-uri(node)	返回当前节点或指定节点的 base-uri 属性的值。
fn:document-uri(node)	返回指定节点的 document-uri 属性的值。

7.2 错误和跟踪函数

名称	说明
fn:error() fn:error(error) fn:error(error,description) fn:error(error,description,error-object)	例子：error(fn:QName('http://example.com/test', 'err:toohigh'), 'Error: Price is too high') 结果：向外部处理环境返回 http://example.com/test#toohigh 以及字符串 "Error: Price is too high"。
fn:trace(value,label)	用于对查询进行 debug。

7.3 有关数值的函数

名称	说明
fn:number(arg)	返回参数的数值。参数可以是布尔值、字符串或节点集。 例子：number('100') 结果：100
fn:abs(num)	返回参数的绝对值。 例子：abs(3.14) 结果：3.14 例子：abs(-3.14) 结果：3.14
fn:ceiling(num)	返回小于 num 参数的最小整数。 例子：ceiling(3.14) 结果：4
fn:floor(num)	返回不大于 num 参数的最大整数。 例子：floor(3.14) 结果：3
fn:round(num)	把 num 参数舍入为最接近的整数。 例子：round(3.14) 结果：3
fn:round-half-to-even()	例子：round-half-to-even(0.5) 结果：0 例子：round-half-to-even(1.5) 结果：2 例子：round-half-to-even(2.5) 结果：2

7.4 攸关字符串的函数

名称	说明
fn:string(arg)	返回参数的字符串值。参数可以是数字、逻辑值或节点集。 例子：string(314) 结果："314"
fn:codepoints-to-string(int,int,...)	根据代码点序列返回字符串。

	例子：codepoints-to-string(84, 104, 233, 114, 232, 115, 101) 结果：'Thérèse'
fn:string-to-codepoints(string)	根据字符串返回代码点序列。 例子：string-to-codepoints("Thérèse") 结果：84, 104, 233, 114, 232, 115, 101
fn:codepoint-equal(comp1,comp2)	根据 Unicode 代码点对照，如果 comp1 的值等于 comp2 的值，则返回 true。(http://www.w3.org/2005/02/xpath-functions/collation/codepoint)，否则返回 false。
fn:compare(comp1,comp2) fn:compare(comp1,comp2,collation)	如果 comp1 小于 comp2，则返回 -1。如果 comp1 等于 comp2，则返回 0。如果 comp1 大于 comp2，则返回 1。（根据所用的对照规则）。 例子：compare('ghi', 'ghi') 结果：0
fn:concat(string,string,...)	返回字符串的拼接。 例子：concat('XPath ','is ','FUN!') 结果：'XPath is FUN!'
fn:string-join((string,string,...),sep)	使用 sep 参数作为分隔符，来返回 string 参数拼接后的字符串。 例子：string-join(('We', 'are', 'having', 'fun!'), ' ') 结果：' We are having fun! ' 例子：string-join(('We', 'are', 'having', 'fun!')) 结果：'Wearehavingfun!' 例子：string-join((), 'sep') 结果：''
fn:substring(string,start,len) fn:substring(string,start)	返回从 start 位置开始的指定长度的子字符串。第一个字符的下标是 1。如果省略 len 参数，则返回从位置 start 到字符串末尾的子字符串。 例子：substring('Beatles',1,4) 结果：'Beat' 例子：substring('Beatles',2) 结果：'eatles'
fn:string-length(string) fn:string-length()	返回指定字符串的长度。如果没有 string 参数，则返回当前节点的字符串值的长度。 例子：string-length('Beatles') 结果：7
fn:normalize-space(string) fn:normalize-space()	删除指定字符串的开头和结尾的空白，并把内部的所有空白序列替换为一个，然后返回结果。如果没

	有 string 参数，则处理当前节点。 例子：normalize-space(' The XML ') 结果：'The XML'
fn:normalize-unicode() fn:upper-case(string)	把 string 参数转换为大写。 例子：upper-case('The XML') 结果：'THE XML'
fn:lower-case(string)	把 string 参数转换为小写。 例子：lower-case('The XML') 结果：'the xml'
fn:translate(string1,string2,string3)	把 string1 中的 string2 替换为 string3。 例子：translate('12:30','30','45') 结果：'12:45' 例子：translate('12:30','03','54') 结果：'12:45' 例子：translate('12:30','0123','abcd') 结果：'bc:da'
fn:escape-uri(stringURI,esc-res)	例子：escape-uri("http://example.com/test#car", true()) 结果："http%3A%2F%2Fexample.com%2Ftest#car" 例子：escape-uri("http://example.com/test#car", false()) 结果："http://example.com/test#car" 例子：escape-uri ("http://example.com/~bébé", false()) 结果："http://example.com/~b%C3%A9b%C3%A9"
fn:contains(string1,string2)	如果 string1 包含 string2，则返回 true，否则返回 false。 例子：contains('XML','XM') 结果：true
fn:starts-with(string1,string2)	如果 string1 以 string2 开始，则返回 true，否则返回 false。 例子：starts-with('XML','X') 结果：true
fn:ends-with(string1,string2)	如果 string1 以 string2 结尾，则返回 true，否则返回 false。 例子：ends-with('XML','X') 结果：false
fn:substring-before(string1,string2)	返回 string2 在 string1 中出现之前的子字符串。 例子：substring-before('12/10','/') 结果：'12'
fn:substring-after(string1,string2)	返回 string2 在 string1 中出现之后的子字符串。

	例子：substring-after('12/10','/') 结果：'10'
fn:matches(string,pattern)	如果 string 参数匹配指定的模式，则返回 true，否则返回 false。 例子：matches("Merano", "ran") 结果：true
fn:replace(string,pattern,replace)	把指定的模式替换为 replace 参数，并返回结果。 例子：replace("Bella Italia", "l", "*") 结果：'Be**a Ita*ia' 例子：replace("Bella Italia", "l", "") 结果：'Bea Itaia'
fn:tokenize(string,pattern)	例子：tokenize("XPath is fun", "\s+") 结果：("XPath", "is", "fun")

7.5 针对 anyURI 的函数

名称	说明
fn:resolve-uri(relative,base)	

7.6 关于布尔值的函数

名称	说明
fn:boolean(arg)	返回数字、字符串或节点集的布尔值。
fn:not(arg)	首先通过 boolean() 函数把参数还原为一个布尔值。如果该布尔值为 false，则返回 true，否则返回 true。 例子：not(true()) 结果：false
fn:true()	返回布尔值 true。 例子：true() 结果：true
fn:false()	返回布尔值 false。 例子：false() 结果：false

7.7 有关持续时间、日期和时间的函数

日期、时间、持续时间的组件提取函数

名称	说明
fn:dateTime(date,time)	把参数转换为日期和时间。
fn:years-from-duration(datetimedur)	返回参数值的年份部分的整数，以标准词汇表示法来表示。
fn:months-from-duration(datetimedur)	返回参数值的月份部分的整数，以标准词汇表示法来表示。
fn:days-from-duration(datetimedur)	返回参数值的天部分的整数，以标准词汇表示法来表示。
fn:hours-from-duration(datetimedur)	返回参数值的小时部分的整数，以标准词汇表示法来表示。
fn:minutes-from-duration(datetimedur)	返回参数值的分钟部分的整数，以标准词汇表示法来表示。
fn:seconds-from-duration(datetimedur)	返回参数值的分钟部分的十进制数，以标准词汇表示法来表示。
fn:year-from-dateTime(datetime)	返回参数本地值的年部分的整数。 例子：year-from-dateTime(xs:dateTime("2005-01-10T12:30-04:10")) 结果：2005
fn:month-from-dateTime(datetime)	返回参数本地值的月部分的整数。 例子：month-from-dateTime(xs:dateTime("2005-01-10T12:30-04:10")) 结果：01
fn:day-from-dateTime(datetime)	返回参数本地值的天部分的整数。 例子：day-from-dateTime(xs:dateTime("2005-01-10T12:30-04:10")) 结果：10
fn:hours-from-dateTime(datetime)	返回参数本地值的小时部分的整数。 例子：hours-from-dateTime(xs:dateTime("2005-01-10T12:30-04:10"))

	结果：12
fn:minutes-from-dateTime(datetime)	返回参数本地值的分钟部分的整数。 例子：minutes-from-dateTime(xs:dateTime("2005-01-10T12:30-04:10")) 结果：30
fn:seconds-from-dateTime(datetime)	返回参数本地值的秒部分的十进制数。 例子：seconds-from-dateTime(xs:dateTime("2005-01-10T12:30:00-04:10")) 结果：0
fn:timezone-from-dateTime(datetime)	返回参数的时区部分，如果存在。
fn:year-from-date(date)	返回参数本地值中表示年的整数。 例子：year-from-date(xs:date("2005-04-23")) 结果：2005
fn:month-from-date(date)	返回参数本地值中表示月的整数。 例子：month-from-date(xs:date("2005-04-23")) 结果：4
fn:day-from-date(date)	返回参数本地值中表示天的整数。 例子：day-from-date(xs:date("2005-04-23")) 结果：23
fn:timezone-from-date(date)	返回参数的时区部分，如果存在。
fn:hours-from-time(time)	返回参数本地值中表示小时部分的整数。 例子：hours-from-time(xs:time("10:22:00")) 结果：10
fn:minutes-from-time(time)	返回参数本地值中表示分钟部分的整数。 例子：minutes-from-time(xs:time("10:22:00")) 结果：22
fn:seconds-from-time(time)	返回参数本地值中表示秒部分的整数。 例子：seconds-from-time(xs:time("10:22:00")) 结果：0
fn:timezone-from-time(time)	返回参数的时区部分，如果存在。
fn:adjust-dateTime-to-timezone(datetime, timezone)	如果 timezone 参数为空，则返回没有时区的 dateTime。否则返回带有时区的 dateTime。
fn:adjust-date-to-timezone(date,timezone)	如果 timezone 参数为空，则返回没有时区的 date。否则返回带有时区的 date。

fn:adjust-time-to-timezone(time,timezone)	如果 timezone 参数为空，则返回没有时区的 time。否则返回带有时区的 time。

7.8 与 QNames 相关的函数

名称	说明
fn:QName()	
fn:local-name-from-QName()	
fn:namespace-uri-from-QName()	
fn:namespace-uri-for-prefix()	
fn:in-scope-prefixes()	
fn:resolve-QName()	

7.9 关于节点的函数

名称	说明
fn:name() fn:name(nodeset)	返回当前节点的名称或指定节点集中的第一个节点。
fn:local-name() fn:local-name(nodeset)	返回当前节点的名称或指定节点集中的第一个节点 - 不带有命名空间前缀。
fn:namespace-uri() fn:namespace-uri(nodeset)	返回当前节点或指定节点集中第一个节点的命名空间 URI。
fn:lang(lang)	如果当前节点的语言匹配指定的语言，则返回 true。 例子：Lang("en") is true for ... 例子：Lang("de") is false for ...
fn:root() fn:root(node)	返回当前节点或指定的节点所属的节点树的根节点。通常是文档节点。

7.10 有关序列的函数

7.10.1 一般性的函数

名称	说明
fn:index-of((item,item,...),searchitem)	返回在项目序列中等于 searchitem 参数的位置。 例子：index-of ((15, 40, 25, 40, 10), 40) 结果：(2, 4) 例子：index-of (("a", "dog", "and", "a", "duck"), "a") Result (1, 4) 例子：index-of ((15, 40, 25, 40, 10), 18) 结果：()
fn:remove((item,item,...),position)	返回由 item 参数构造的新序列 - 同时删除 position 参数指定的项目。 例子：remove(("ab", "cd", "ef"), 0) 结果：("ab", "cd", "ef") 例子：remove(("ab", "cd", "ef"), 1) 结果：("cd", "ef") 例子：remove(("ab", "cd", "ef"), 4) 结果：("ab", "cd", "ef")
fn:empty(item,item,...)	如果参数值是空序列，则返回 true，否则返回 false。 例子：empty(remove(("ab", "cd"), 1)) 结果：false
fn:exists(item,item,...)	如果参数值不是空序列，则返回 true，否则返回 false。 例子：exists(remove(("ab"), 1)) 结果：false
fn:distinct-values((item,item,...),collation)	返回唯一不同的值。 例子：distinct-values((1, 2, 3, 1, 2)) 结果：(1, 2, 3)
fn:insert-before((item,item,...),pos,inserts)	返回由 item 参数构造的新序列 - 同时在 pos 参数指定位置插入 inserts 参数的值。 例子：insert-before(("ab", "cd"), 0, "gh") 结果：("gh", "ab", "cd") 例子：insert-before(("ab", "cd"), 1, "gh") 结果：("gh", "ab", "cd") 例子：insert-before(("ab", "cd"), 2, "gh")

	结果：("ab", "gh", "cd") 例子：insert-before(("ab", "cd"), 5, "gh") 结果：("ab", "cd", "gh")
fn:reverse((item,item,...))	返回指定的项目的颠倒顺序。 例子：reverse(("ab", "cd", "ef")) 结果：("ef", "cd", "ab") 例子：reverse(("ab")) 结果：("ab")
fn:subsequence((item,item,...),start,len)	返回 start 参数指定的位置返回项目序列，序列的长度由 len 参数指定。第一个项目的位置是 1。 例子：subsequence(($item1, $item2, $item3,...), 3) 结果：($item3, ...) 例子：subsequence(($item1, $item2, $item3, ...), 2, 2) 结果：($item2, $item3)
fn:unordered((item,item,...))	依据实现决定的顺序来返回项目。

7.10.2 测试序列容量的函数

名称	说明
fn:zero-or-one(item,item,...)	如果参数包含零个或一个项目，则返回参数，否则生成错误。
fn:one-or-more(item,item,...)	如果参数包含一个或多个项目，则返回参数，否则生成错误。
fn:exactly-one(item,item,...)	如果参数包含一个项目，则返回参数，否则生成错误。

7.10.3 Equals, Union, Intersection and Except

名称	说明
fn:deep-equal(param1,param2,collation)	如果 param1 和 param2 与彼此相等（deep-equal），则返回 true，否则返回 false。

7.10.4 合计函数

名称	说明

fn:count((item,item,...))	返回节点的数量。
fn:avg((arg,arg,...))	返回参数值的平均数。例子：avg((1,2,3)) 结果：2
fn:max((arg,arg,...))	返回大于其它参数的参数。例子：max((1,2,3)) 结果：3 例子：max(('a', 'k')) 结果：'k'
fn:min((arg,arg,...))	返回小于其它参数的参数。例子：min((1,2,3)) 结果：1 例子：min(('a', 'k')) 结果：'a'
fn:sum(arg,arg,...)	返回指定节点集中每个节点的数值的总和。

7.10.5 生成序列的函数

名称	说明
fn:id((string,string,...),node)	Returns a sequence of element nodes that have an ID value equal to the value of one or more of the values specified in the string argument
fn:idref((string,string,...),node)	Returns a sequence of element or attribute nodes that have an IDREF value equal to the value of one or more of the values specified in the string argument
fn:doc(URI)	
fn:doc-available(URI)	如果 doc() 函数返回文档节点，则返回 true，否则返回 false。
fn:collection() fn:collection(string)	

7.11 上下文函数

名称	说明
fn:position()	返回当前正在被处理的节点的 index 位置。 例子：//book[position()<=3] 结果：选择前三个 book 元素
fn:last()	返回在被处理的节点列表中的项目数目。 例子：//book[last()] 结果：选择最后一个 book 元素
fn:current-dateTime()	返回当前的 dateTime（带有时区）。

fn:current-date()	返回当前的日期（带有时区）。
fn:current-time()	返回当前的时间（带有时区）。
fn:implicit-timezone()	返回隐式时区的值。
fn:default-collation()	返回默认对照的值。
fn:static-base-uri()	返回 base-uri 的值。

W3School TIY

TRY IT YOURSELF （请在下面的文本框中编辑您的代码，然后单击此按钮测试结果。）

编辑您的代码：

```
<html>
<body>
```

查看结果: