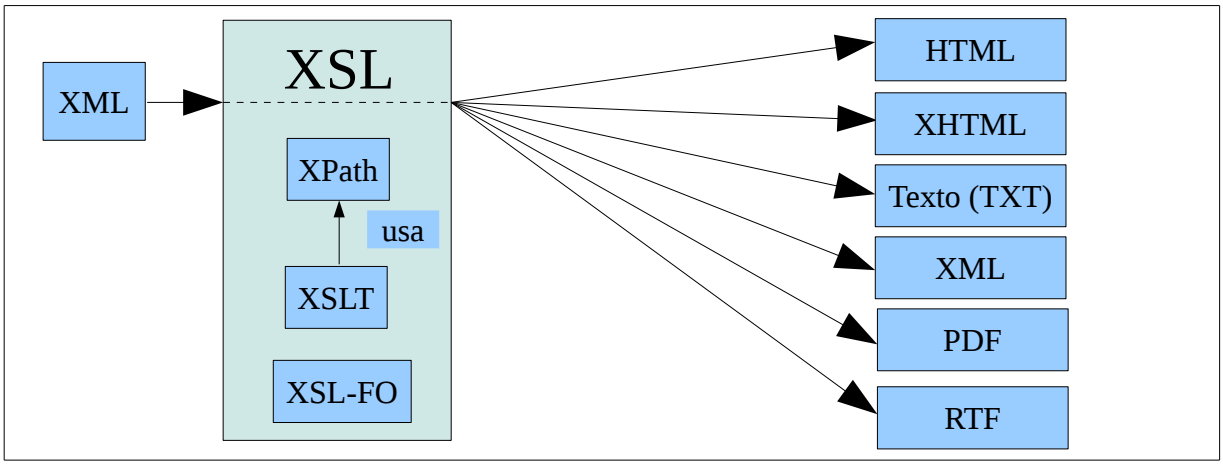
U.D. 12 XPATH

1. Introducción



Las siglas ***XSL*** significan ***eXtensible Stylesheet Language*** (Lenguaje extensible de hojas de estilo). Este lenguaje nació con la intención de generar hojas de estilo para ***XML***. Más tarde se ha convertido en una herramienta mucho más amplia para transformarlos a diversos formatos.

XSL es un conjunto de tres estándares para la transformación de documentos ***XML*** (***XSLT***, ***XPath*** y ***XSL-FO***):

- ***XPath*** es un lenguaje para seleccionar elementos dentro del documento fuente ***XML***, similar a la función que cumplen los selectores ***CSS***, pero mucho más potente, pues permite seleccionar también valores de atributos e incluso nombres de atributos y elementos.
- ***XSLT*** (***XSL Transformations***) es un lenguaje basado en ***XML*** para describir el documento de salida a partir de un documento ***XML*** de entrada
- ***XSL-FO*** (***Formating Object)*** permite realizar transformaciones más complejas para generar documentos ***PDF*** o ***RTF***.

2. XPath: consultas sobre XML

XPath (***XML Path***) nos permite seleccionar partes de un documento ***XML***. Es un lenguaje declarativo, basado en cadenas de expresiones, que no tiene sintaxis ***XML.***

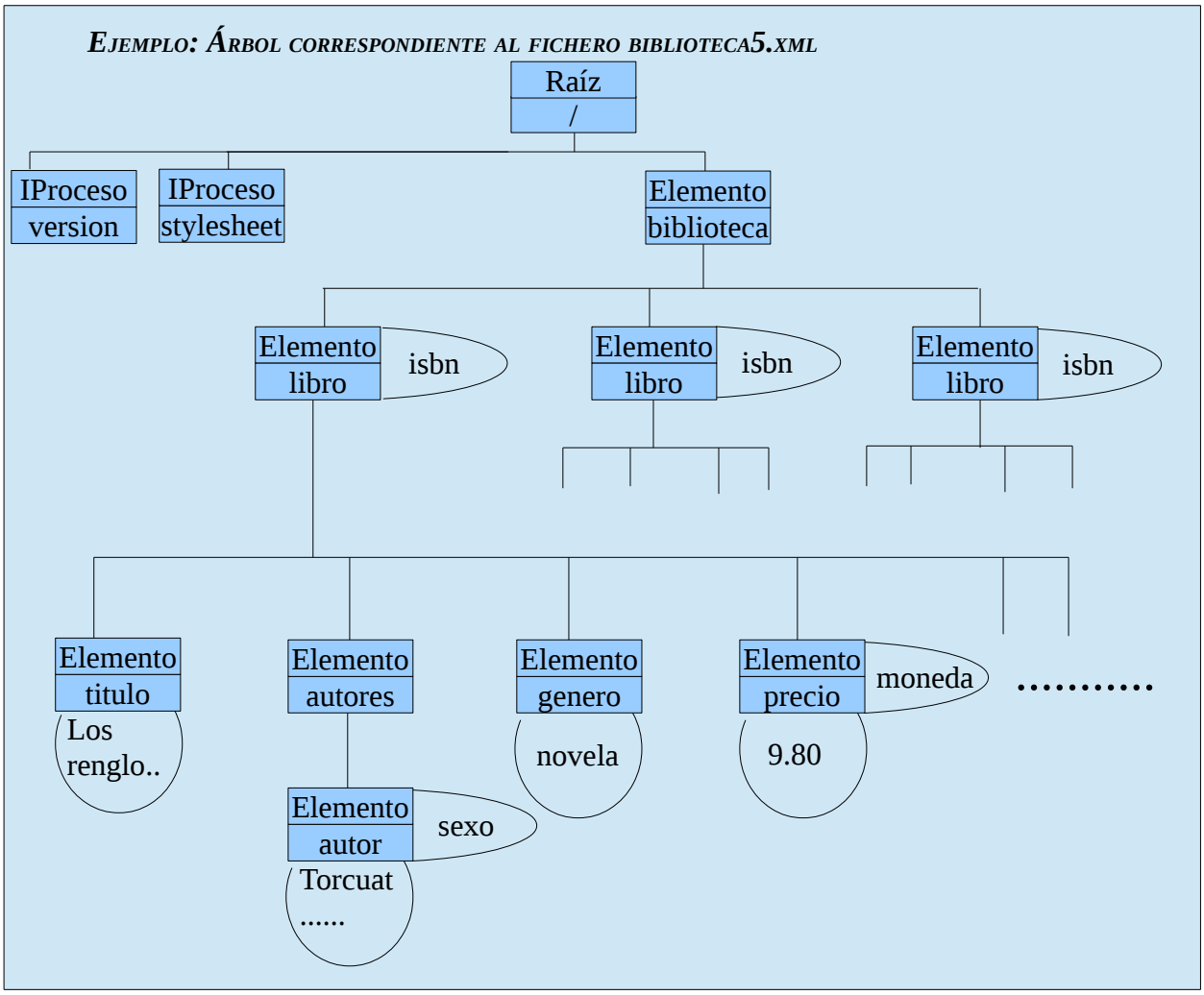
XPath considera al documento fuente ***XML*** como si estuviese estructurado en forma de árbol. En los nodos de ese árbol se aloja todo el contenido del documento (incluidos los comentarios y las instrucciones de proceso). Mediante las expresiones ***XPath*** podemos hacer referencia a cualquier elemento del árbol, o sea, a cualquier elemento del documento base.

Entiéndase que ***XPath*** sólo hace eso: referenciar partes del documento. De su posterior procesamiento se encargará ***XSLT***. ***XPath***, básicamente, proporciona la sintaxis adecuada para seleccionar uno más nodos de un documento ***XML***.

2.1 El árbol del documento

El árbol del documento fuente puede tener nodos de 7 tipos distintos:

- ***Raíz***: nodo raíz del árbol (que no debe confundirse con el elemento raíz del documento ***XML***). El nodo raíz (root node) no tiene ningún texto asociado; podemos asimilarlo al nombre del fichero.



- ***Instrucciones de proceso***: recuérdese que las instrucciones de proceso se encuentran en el prólogo de un documento ***XML*** y van encerradas entre los símbolos **<?** y **?>**
- ***Espacios de nombres***
- ***Elementos***: los nodos de elementos incluyen etiqueta de apertura, contenido y etiqueta de cierre.
- ***Atributos***
- ***Texto***
- ***Comentarios***

2.2 Valor de una expresión XPath

Una expresión ***XPath*** puede devolver uno de los siguientes valores: un nodo o conjunto de nodos, un valor booleano, un número o una cadena de caracteres.

- Un *nodo* del árbol o *conjunto de nodos* del árbol

La expresión	*Devuelve*
`/biblioteca/libro[1]/titulo`	el elemento ***titulo*** del primer ***libro***
`/biblioteca/libro[2]/autores/autor`	los elementos ***autor*** del segundo ***libro***

- Un *valor booleano* (true o false)

La expresión	*Devuelve*
`/biblioteca/libro[1]/@isbn="978-84-09349-7"`	***true***

- Un *número*

  (*cuidado: tenemos que estar situados en el árbol en algún libro****)***

La expresión	*Devuelve*
`count(autores/autor)`	el número de hijos ***autor*** que tiene el elemento ***autores***

- ***Una cadena de caracteres***

La expresión	*Devuelve*
`/biblioteca/libro[1]/@isbn`	978-84-09349-7

Nota: Eclipse ofrece un analizador de expresiones ***XPath*** en ***Ventana/Mostrar vista/XPath*** que podemos usar para comprobar las expresiones anteriores. Cuando escribimos una expresión válida el resultado se marca en el documento.

2.3 Tipos de expresiones

Este árbol nos recuerda la estructura de directorios y archivos que los sistemas operativos utilizan en los dispositivos de almacenamiento, así que no es casualidad que las expresiones ***XPath*** se parezcan al ***pathname*** o ruta de un archivo. También aquí hablamos de rutas absolutas y relativas, existiendo además un tercer tipo que hemos llamado independientes:

- ***Absolutas***: Camino completo de la raíz al objetivo. Comienzan con el símbolo (*/*)

```
/biblioteca/libro/autores/autor
```

- ***Relativas***: Describen el camino desde el nodo actual (también llamado contexto) hasta nuestro objetivo

```
autores/autor  (estamos suponiendo que el contexto es libro)
```

- ***Independientes***: Indican el objetivo sin importar en qué parte del árbol se encuentran. Siempre comienzan con doble barra (*//*)

```
//autor
```

Ejemplos:

expresión	hace referencia a
`titulo`	nodo ***titulo*** (supuesto que el nodo actual es ***libro***)
`/`	raíz del árbol (que no es el elemento raíz)
`//titulo`	nodo ***titulo*** (da igual cual sea el nodo actual)
`/biblioteca/libro/@isbn`	atributo ***isbn*** de ***libro***, hijo de ***biblioteca***, hijo de la ***raíz***

2.4 Comodines

También en ***XPath*** podemos usar las siguientes referencias y comodines habituales:

Expresión	*Significado*
.	nodo actual o contexto
..	nodo padre del actual
/	nodo raíz del árbol
*****	cualquier elemento
@*	cualquier atributo
node()	cualquier nodo

Nota: ***** referencia a los nodos que son elementos, ***node()*** referencia a todos los nodos: elementos y no elementos (como instrucciones de procesamiento y comentarios)

2.5 Operadores en XPath

Los operadores que usa ***XPath*** son:

- ***Operadores aritméticos:***
  - + (suma)
  - - (resta)
  - ***** (producto)
  - ***div*** (división, obsérvese que la barra / tiene otros usos en ***XPath***)
  - ***mod*** (operación resto, también llamada módulo)
- ***Operadores relacionales:***
  - < (menor que)
  - > (mayor que)
  - <= (menor o igual que)
  - >= (mayor o igual que)
  - = (igual)
  - ***!***= (no igual)
- ***Operadores booleanos:***
  - ***and*** (Y lógico)
  - ***or*** (O lógico aplicado a predicados)
  - ***not*** (negación lógica)
  - | (O lógico aplicado a expresiones)

Particularidades de los operadores de igualdad

El significado de usar los operadores igual (=) o no igual (***!***=) contra un conjunto de nodos es el siguiente: el resultado es verdadero si en este conjunto de nodos existe algún nodo que cumpla la igualdad o no igualdad respectivamente. Como consecuencia del criterio usado, hay que observar que se produce la paradoja de que el conjunto de nodos ***puede ser igual y no igual al mismo tiempo***.

EJEMPLO*: *Igualdad y desigualdad en XPath

La siguiente expresión devuelve ***true*** porque existe un nodo cuyo ***isbn*** es igual al indicado:

```
/biblioteca/libro/@isbn = “978-84-09349-7”
```

pero la siguiente expresión también devuelve ***true*** porque también existen nodos cuyo ***isbn*** no es igual al indicado:

```
/biblioteca/libro/@isbn != “978-84-09349-7”
```

En general, diremos que al comparar un conjunto de nodos contra otro dato (booleano, numérico o cadena) la comparación será verdadera si y sólo si hay un nodo del conjunto tal que el resultado de hacer la comparación es verdadero.

Si los dos objetos a comparar son conjuntos de nodos, entonces la comparación será verdadera si y sólo si hay un nodo en el primer conjunto de nodos y un nodo en el segundo conjunto de nodos tales que el resultado de realizar la comparación de los ***valores de cadena***[1] de los dos nodos es verdadero.

Particularidades del operador resta

Dado que XML permite guiones en nombres, el operador ***resta*** necesitará ser precedido por un espacio en blanco para distinguirlo del guión.

- ***libro-bolsillo*** se evalúa como un conjunto de nodos hijo llamados ***libro-bolsillo***
- ***libro - bolsillo*** se evalúa como la diferencia entre el resultado de convertir en número el valor de cadena del primer elemento hijo ***libro*** y el resultado de convertir en número el valor de cadena del primer hijo ***bolsillo***.

Particularidades del operador O lógico

Este operador (***or***) se puede usar tanto entre predicados como entre expresiones ***XPath***: en el caso de aplicarse a expresiones se usa el símbolo barra vertical ***(|)***; en el caso de aplicarse a predicados se usa la palabra ***"or"***

EXPRESIÓN	*VALOR*
//libro/titulo | //libro/precio	Selecciona todos los elementos ***titulo*** y los elementos ***precio*** hijos de los elementos ***libro.***
//titulo | //precio	Selecciona todos los elementos ***titulo*** y ***precio*** del documento.
/biblioteca/libro/titulo | //precio	Selecciona todos los elementos ***titulo*** de elementos ***libro*** y todos los ***precios*** del documento.
/biblioteca/libro/titulo [@lang or @edicion]	Selecciona todos los elementos ***titulo*** de elementos

1 El ***valor de cadena*** de un nodo es el resultado de convertir ese nodo en una cadena.

	libro con un atributo ***lang*** o ***edicion***.

Particularidades de los operadores relacionales

Como es bien sabido, dentro de un documento ***XML*** los símbolos mayor y menor que (><) tienen un significado muy definido. No podemos, por tanto, usarlos como operadores relacionales directamente, sino que hemos de recurrir a sus entidades de caracteres:

- ***<*** (abreviatura de less than) para el símbolo <
- ***>*** (abreviatura de greater than) para el símbolo >

La expresión `libro[position()<3]` habrá que escribirla como `libro[position()<3]`

NOTA: No obstante, a lo largo de este tema usaremos los caracteres *"prohibidos"* para una mejor legibilidad. En Eclipse si se aceptan los símbolos < , >.

Precedencia

La precedencia de los operadores en ***XPath*** sigue el siguiente orden. Como siempre, para alterar esta precedencia tenemos la posibilidad de usar paréntesis:

- <=, <, >=, >
- =, ***!***=
- ***and***
- ***or***

2.6 Predicados

Los predicados nos permiten concretar nodos específicos dentro de un conjunto. Se escriben entre corchetes e incluyen números, funciones o expresiones lógicas, conformando una condición que deben cumplir los nodos elegidos.

EJEMPLO1:

expresión	hace referencia a
`libro[2]`	el segundo ***libro***
`libro[last()]`	el último ***libro***
`libro[last()-1]`	el penúltimo ***libro***
`libro[position()<3]`	los dos primeros ***libro***
`comidas/plato[@precio]`	todos los ***platos*** con un atributo ***precio***
`comidas/plato[precio]`	todos los ***platos*** con un elemento ***precio***
`*[@atbto="valor"]`	todos los elementos con el atributo ***atbto*** y valor

	indicado
`/biblioteca/libro[@isbn]/titulo`	el ***titulo*** de todos los ***libros*** que tengan un atributo ***isbn***
`//autor[@sexo="femenino"]`	las autoras de ***sexo femenino***
`//capitulo[not(@public)]`	todos los capítulos que ***no*** tengan el atributo ***public***
`//autor[2]`	los ***autor*** que sean 2º hijo de su padre

EJEMPLO2: PREDICADOS ANIDADOS

Un predicado puede contener una expresión más compleja que a su vez contenga predicados:

`//capitulo[parrafo/*[@ref]]`	selecciona todos los ***capitulo*** que tengan un ***párrafo*** que a su vez tenga algún elemento hijo con el atributo ***ref***

EJEMPLO3: VARIOS PREDICADOS CONSECUTIVOS

Cuando dos o más predicados se suceden uno tras otro el conjunto de nodos inicial se filtra por el primer predicado para generar un nuevo conjunto de nodos; este nuevo conjunto de nodos es entonces filtrado usando el segundo predicado, y así sucesivamente.

`//capitulo[@href][@public="si"]`	De entre todos los ***capitulo*** se seleccionan los que tenga un atributo ***href*** y de este conjunto sólo nos quedamos con los que tengan un atributo ***public*** con valor ***si***

El orden en que aparecen los predicados, a veces, es importante:

```
autor[@sexo='masculino'][1]
```

selecciona el primer ***autor*** que tenga un atributo ***sexo*** con valor ***masculino*** (el primer predicado selecciona a todos los autores varones y el segundo actúa sobre esa preselección devolviendo primero de ellos)

```
autor[1][@sexo="masculino"]
```

selecciona el primer ***autor*** si tiene un atributo ***sexo*** con valor ***masculino*** (el primer predicado selecciona el autor que es primer hijo y el segundo actúa sobre esa preselección (de un solo autor) devolviéndolo si es varón)

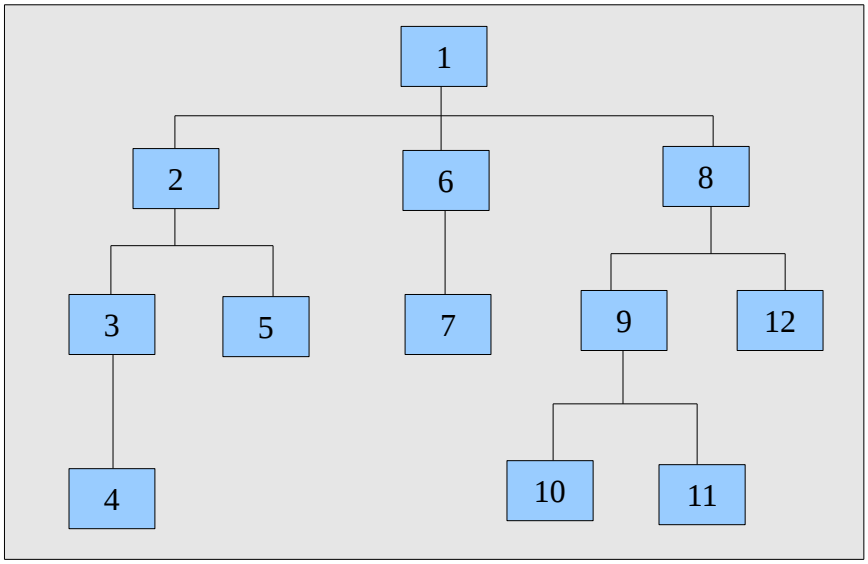
NOTA: ***Uso de comillas.*** Las expresiones ***XPath*** siempre aparecerán encerradas entre comillas cuando las usemos como parte de una instrucción ***XSL***, así que cuando dentro de la propia expresión vuelve a haber comillas tenemos que anidarlas como siempre que esto ocurre en algún lenguaje de marcas: si las comillas externas son simples, las internas tendrán que ser dobles y viceversa. Un ejemplo de esto es la siguiente línea de un archivo ***XSL***, que incluye una expresión ***XPath***: `<xsl:value-of select='precio[@moneda="dolar"]' />`

2.7 El orden del documento: Primero en Profundidad

Cuando el resultado de una expresión sea un conjunto de elementos probablemente nos interesará saber como se ordenan entre sí. ***XPath*** siempre recorre el árbol (o un subárbol) usando el orden llamado ***PRIMERO EN PROFUNDIDAD***, que establece:

- ***Primero***: el nodo raíz
- ***Segundo***: cada uno de los hijos de la raíz comenzando por la izquierda y aplicando para cada uno de ellos recursivamente el orden ***PRIMERO EN PROFUNDIDAD***.

En el siguiente ejemplo se han numerado los nodos siguiendo este orden y puede observarse por qué se denomina ***PRIMERO EN PROFUNDIDAD***:



Sin embargo, cuando el resultado de una expresión ***XPath*** sea un conjunto de atributos no existe un orden preestablecido.

2.8 Funciones

Funciones sobre un conjunto de nodos:

- ***last ()*** devuelve la posición del último nodo de la lista procesada.

```
/biblioteca/libro/last()
```

  devolverá el número 3, puesto que hay 3 ***libros*** hijos de ***biblioteca***.

- ***position ()*** devuelve la posición en la lista de nodos, del nodo que se está procesando en cada momento.
- ***count (node-set)*** devuelve el número de nodos que componen el conjunto.
- ***id ("valor")*** devuelve los nodos cuyo identificador[2] sea ***valor***.

```
id('foo')/child::libro[position()=5]
```

  selecciona el quinto ***libro***, hijo del elemento con identificador único ***foo***

2 ***IDENTIFICADORES ÚNICOS***: Los nodos elemento pueden tener un identificador único ***(ID)***. Este es el valor del atributo que se declara en el ***DTD*** como de tipo ***ID***. No puede haber dos elementos en un documento con el mismo ***ID***. Si un documento no tiene ***DTD***, entonces ningún elemento del documento tendrá ***ID***.

- ***local-name (node-set?)*** devuelve la parte local del primer nodo del conjunto o la cadena vacía, si el conjunto es vacío o no tiene un nombre expandido[3].
- ***namespace-uri (node-set?)*** devuelve la ***uri*** del espacio de nombres del primer nodo o cadena vacía, si el conjunto es vacío o no tiene un nombre expandido.
- ***name (node-set?)*** devuelve el nombre expandido del primer nodo.

Funciones sobre cadenas

- ***string (objeto?)*** devuelve el valor del argumento en una cadena, conocido como ***valor de cadena*** . Si es un conjunto de nodos, transforma en cadena el primer nodo. Si es un número devuelve un ***string*** con la secuencia de dígitos (o ***NaN*** o ***Infinity*** o ***-Infinity***). Si es un ***boolean*** devuelve las cadenas ***"false"*** o ***"true"***
- ***concat (cadena1, cadena2, ..., cadenaN)*** devuelve el ***string*** resultado de concatenar todos los argumentos. Cada una de ellos puede ser un texto entre comillas o una expresión ***XPath***.
- ***starts-with (string, string)*** devuelve ***true*** o ***false*** dependiendo de que la primera cadena empiece o no con la segunda. Si la segunda es cadena vacía, devuelve ***true***.
- ***contains (string, string)*** devuelve ***true*** o ***false*** dependiendo de que la primera cadena contenga a la segunda cadena.
- ***substring-before (string, string)*** devuelve la subcadena de la primera cadena argumento que precede a la primera aparición de la segunda cadena argumento.
- ***substring-after (string, string)*** devuelve la subcadena de la primera cadena argumento que sigue a la primera aparición de la segunda cadena argumento.
- ***substring (cadena, comienzo, longitud?)*** devuelve la parte de la cadena que se especifica en los argumentos. Hasta el final si se omite la longitud (el primer carácter de la cadena ocupa la posición 1).
- ***string-length (string?)*** devuelve el número de caracteres del argumento (o del contexto convertido a cadena).
- ***normalize-space (string?)*** devuelve el argumento, eliminando espacios al inicio y final y sustituyendo varios espacios (o tab o retornos...) a uno solo.
- ***translate (string, string, string)*** devuelve el primer argumento, pero sustituyendo las apariciones del 2º argumento por el 3º argumento. Las sustituciones se entienden carácter a carácter, o sea, el primer carácter de 2º argumento por el primer carácter de 3º argumento y así sucesivamente. Los que no tengan correspondencia (porque 2º argumento sea más largo que 3º) se eliminan. Los que tengan 2 o más correspondencias (porque en 2º argumento se repita algún carácter) toman la primera sustitución.
-

3 El ***nombre expandido*** de un elemento es el nombre completo formado por su espacio de nombres (en forma de ***URI***) y la etiqueta (que sería la parte local del nombre). Los nodos que pueden tener un nombre expandido son los correspondientes a elementos y atributos.

Funciones booleanas

- ***boolean (objeto)*** convierte el valor de los argumentos a verdadero o falso. Aplicada a un conjunto de nodos, devuelve ***true*** si no es vacío. Un número será ***true*** cuando no sea 0 ni ***NaN***. Un ***string*** será ***true*** si no es cadena vacía.
- ***not (objeto)*** devuelve ***true*** si el argumento es ***false*** y viceversa.
- ***true ()*** devuelve ***true***.
- ***false ()*** devuelve ***false***.
- ***lang (string)*** devuelve ***true*** si el lenguaje indicado en el argumento concuerda (o es un sublenguaje) con el lenguaje que corresponde al nodo contexto (establecido con ***xsl:lang***) y en caso contrario devuelve ***false***.

Funciones sobre números

- ***number (objeto?)*** convierte el argumento en un número (o ***NaN*** si no es posible). Si el argumento es booleano devuelve 1 (para ***true***) ó 0 (para ***false***); Si el argumento es un nodo, primero se transforma en ***string*** y posteriormente en número.
- ***sum (node-set)*** devuelve la suma de convertir a número todos los nodos del conjunto.
- ***floor (nº)*** devuelve el número entero inmediatamente menor al argumento.
- ***ceiling (nº)*** devuelve el número entero inmediatamente mayor al argumento.
- ***round (nº)*** redondea el número del argumento al entero más próximo, por arriba o por abajo. Si ambos están a igual distancia devuelve el mayor.

NOTA: Podemos encontrar más detalles sobre estas funciones en la especificación oficial de la ***W3C*** para ***XPath*** como la de ***W3Schools.com***.

Funciones sobre tipos de nodos

- ***node ()*** devuelve todos los nodos de cualquier tipo. Así para seleccionar todos los nodos hijo del tipo ***párrafo***, se escribe:

```
/parrafo/node()
```

- ***processing-instruction ()*** devuelve las instrucciones de proceso que hay en el nodo, mostrándolas completas con la sintaxis original del documento.
- ***text ()*** devuelve el texto del contenido del nodo. El nodo completo incluye etiquetas de apertura y cierre, pero con ***text()*** seleccionamos exclusivamente el contenido de texto.
- ***comment ()*** devuelve los comentarios del nodo. En este caso incluye los símbolos de inicio y final del comentario: <***!***-- -->

*NOTA**: orden del documento.*** Un nodo elemento puede tener tres tipos de hijos: elementos, atributos y espacios de nombres. En el orden del documento se consideran primero los espacios de

nombres sin un orden preestablecido entre ellos, en segundo lugar los atributos, sin orden preestablecido entre ellos y en tercer lugar los elementos ordenados primero en profundidad.