NAME

XML::LibXML::DOM − XML::LibXML DOM Implementation

DESCRIPTION

XML::LibXML provides a lightweight interface to *modify* a node of the document tree generated by the XML::LibXML parser. This interface follows as far as possible the DOM Level 3 specification. In addition to the specified functions, XML::LibXML supports some functions that are more handy to use in the perl environment.

One also has to remember, that XML::LibXML is an interface to libxml2 nodes which actually reside on the C−Level of XML::LibXML. This means each node is a reference to a structure which is different from a perl hash or array. The only way to access these structures' values is through the DOM interface provided by XML::LibXML. This also means, that one *can't* simply inherit an XML::LibXML node and add new member variables as if they were hash keys.

The DOM interface of XML::LibXML does not intend to implement a full DOM interface as it is done by XML::GDOME and used for full featured application. Moreover, it offers an simple way to build or modify documents that are created by XML::LibXML's parser.

Another target of the XML::LibXML interface is to make the interfaces of libxml2 available to the perl community. This includes also some workarounds to some features where libxml2 assumes more control over the C−Level that most perl users don't have.

One of the most important parts of the XML::LibXML DOM interface is that the interfaces try to follow the DOM Level 3 specification (<http://www.w3.org/TR/DOM−Level−3−Core/>) rather strictly. This means the interface functions are named as the DOM specification says and not what widespread Java interfaces claim to be the standard. Although there are several functions that have only a singular interface that conforms to the DOM spec XML::LibXML provides an additional Java style alias interface.

Moreover, there are some function interfaces left over from early stages of XML::LibXML for compatibility reasons. These interfaces are for compatibility reasons *only*. They might disappear in one of the future versions of XML::LibXML, so a user is requested to switch over to the official functions.

Encodings and XML::LibXML's DOM implementation

See the section on Encodings in the *XML::LibXML* manual page.

Namespaces and XML::LibXML's DOM implementation

XML::LibXML's DOM implementation is limited by the DOM implementation of libxml2 which treats namespaces slightly differently than required by the DOM Level 2 specification.

According to the DOM Level 2 specification, namespaces of elements and attributes should be persistent, and nodes should be permanently bound to namespace URIs as they get created; it should be possible to manipulate the special attributes used for declaring XML namespaces just as other attributes without affecting the namespaces of other nodes. In DOM Level 2, the application is responsible for creating the special attributes consistently and/or for correct serialization of the document.

This is both inconvenient, causes problems in serialization of DOM to XML, and most importantly, seems almost impossible to implement over libxml2.

In libxml2, namespace URI and prefix of a node is provided by a pointer to a namespace declaration (appearing as a special xmlns attribute in the XML document). If the prefix or namespace URI of the declaration changes, the prefix and namespace URI of all nodes that point to it changes as well. Moreover, in contrast to DOM, a node (element or attribute) can only be bound to a namespace URI if there is some namespace declaration in the document to point to.

Therefore current DOM implementation in XML::LibXML tries to treat namespace declarations in a compromise between reason, common sense, limitations of libxml2, and the DOM Level 2 specification.

In XML::LibXML, special attributes declaring XML namespaces are often created automatically, usually when a namespaced node is attached to a document and no existing declaration of the namespace and prefix is in the scope to be reused. In this respect, XML::LibXML DOM implementation differs from the DOM Level 2 specification according to which special attributes for declaring the appropriate XML namespaces should not be added when a node with a namespace prefix and namespace URI is created.

Namespace declarations are also created when XML::LibXML::Document's **createElementNS()** or **createAttributeNS()** function are used. If the a namespace is not declared on the documentElement, the namespace will be locally declared for the newly created node. In case of Attributes this may look a bit confusing, since these nodes cannot have namespace declarations itself. In this case the namespace is internally applied to the attribute and later declared on the node the attribute is appended to (if required).

The following example may explain this a bit:

```
my $doc = XML::LibXML->createDocument;
my $root = $doc->createElementNS( "", "foo" );
$doc->setDocumentElement( $root );

my $attr = $doc->createAttributeNS( "bar", "bar:foo", "test" );
$root->setAttributeNodeNS( $attr );
```

This piece of code will result in the following document:

```
<?xml version="1.0"?>
<foo xmlns:bar="bar" bar:foo="test"/>
```

The namespace is declared on the document element during the **setAttributeNodeNS()** call.

Namespaces can be also declared explicitly by the use of XML::LibXML::Element's **setNamespace()** function. Since 1.61, they can also be manipulated with functions **setNamespaceDeclPrefix()** and **setNamespaceDeclURI()** (not available in DOM). Changing an URI or prefix of an existing namespace declaration affects the namespace URI and prefix of all nodes which point to it (that is the nodes in its scope).

It is also important to repeat the specification: While working with namespaces you should use the namespace aware functions instead of the simplified versions. For example you should *never* use **setAttribute()** but **setAttributeNS()**.

AUTHORS

Matt Sergeant, Christian Glahn, Petr Pajas

VERSION

2.0134

COPYRIGHT

LICENSE