NAME

XML::LibXML::Attr – XML::LibXML Attribute Class

SYNOPSIS

```
use XML::LibXML;
# Only methods specific to Attribute nodes are listed here,
# see the XML::LibXML::Node manpage for other methods

$attr = XML::LibXML::Attr->new($name [,$value]);
$string = $attr->getValue();
$string = $attr->value;
$attr->setValue( $string );
$node = $attr->getOwnerElement();
$attr->setNamespace($nsURI, $prefix);
$bool = $attr->isId;
$string = $attr->serializeContent;
```

DESCRIPTION

This is the interface to handle Attributes like ordinary nodes. The naming of the class relies on the W3C DOM documentation.

METHODS

The class inherits from XML::LibXML::Node. The documentation for Inherited methods is not listed here.

Many functions listed here are extensively documented in the DOM Level 3 specification (<http://www.w3.org/TR/DOM-Level-3-Core/>). Please refer to the specification for extensive documentation.

new

```
$attr = XML::LibXML::Attr->new($name [,$value]);
```

Class constructor. If you need to work with ISO encoded strings, you should *always* use the `createAttribute` of XML::LibXML::Document.

getValue

```
$string = $attr->getValue();
```

Returns the value stored for the attribute. If undef is returned, the attribute has no value, which is different of being `not specified`.

value

```
$string = $attr->value;
```

Alias for ***getValue()***

setValue

```
$attr->setValue( $string );
```

This is needed to set a new attribute value. If ISO encoded strings are passed as parameter, the node has to be bound to a document, otherwise the encoding might be done incorrectly.

getOwnerElement

```
$node = $attr->getOwnerElement();
```

returns the node the attribute belongs to. If the attribute is not bound to a node, undef will be returned. Overwriting the underlying implementation, the *parentNode* function will return undef, instead of the owner element.

setNamespace

```
$attr->setNamespace($nsURI, $prefix);
```

This function tries to bound the attribute to a given namespace. If `$nsURI` is undefined or empty, the function discards any previous association of the attribute with a namespace. If the namespace was not

previously declared in the context of the attribute, this function will fail. In this case you may wish to call **setNamespace()** on the ownerElement. If the namespace URI is non-empty and declared in the context of the attribute, but only with a different (non-empty) prefix, then the attribute is still bound to the namespace but gets a different prefix than `$prefix`. The function also fails if the prefix is empty but the namespace URI is not (because unprefixed attributes should by definition belong to no namespace). This function returns 1 on success, 0 otherwise.

isId

```
$bool = $attr->isId;
```

Determine whether an attribute is of type ID. For documents with a DTD, this information is only available if DTD loading/validation has been requested. For HTML documents parsed with the HTML parser ID detection is done automatically. In XML documents, all ''xml:id'' attributes are considered to be of type ID.

serializeContent($docencoding)

```
$string = $attr->serializeContent;
```

This function is not part of DOM API. It returns attribute content in the form in which it serializes into XML, that is with all meta-characters properly quoted and with raw entity references (except for entities expanded during parse time). Setting the optional `$docencoding` flag to 1 enforces document encoding for the output string (which is then passed to Perl as a byte string). Otherwise the string is passed to Perl as (UTF-8 encoded) characters.

AUTHORS

Matt Sergeant, Christian Glahn, Petr Pajas

VERSION

2.0134

COPYRIGHT

LICENSE