XS infoset

Draft 1.0
April 5, 2005
Copyright 2005 Keepsake, SPRL[i]
Copyright 2010-2015 Marvell International

1 Introduction

An information set (infoset) is an abstract description of XML documents defined by the W3C (http://www.w3.org/TR/2004/REC-xml-infoset-20040204).

XS grammars define templates to convert XML documents into ECMAScript instances and vice versa. Applications can parse and serialize directly the text of XML documents into and from ECMAScript instances.

Some XML standards require the text of XML documents to adopt conventions beyond the rules of the XML syntax. For instance XML Signatures require the canonicalization of the text of XML documents.

To help applications to adopt such conventions, XS allows to parse and serialize the infoset of XML documents into and from ECMAScript instances.

XS can also scan infoset from text and print infoset to text. Applications can of course then define their own way to convert infoset into text and vice versa.

2 Example

Let us define a grammar:

```
<object name="user" pattern="user">
    <string name="name" pattern="@name"/>
    <chunk name="password" pattern="."/>
</object>
<object name="computer" pattern="/computer">
    <array name="users" contents="user" pattern="."/>
</object>
```

Let us build a document:

```
buffer = '<?xml version="1.0" encoding="UTF-8"?>'
buffer += '<computer>'
buffer +=     '<user name="Flore">VHJpcGVsIFdlc3RtYWxsZQ==</user>'
buffer += '</computer>'
```

The document can be parsed directly from text with:

```
root = xs.parse(buffer)
```

The result is a tree of instances of prototypes defined by the grammar:

```
root ==
{   // prototype: computer,
    users: [
        {   // prototype: user,
            name: "Flore",
            // password: chunk with host data
```

```
        }
    ]
}
```

Vice versa, the document can be serialized directly into text with:

```
buffer = xs.serialize(root)
```

Similarly, the document can be serialized into an infoset with:

```
document = xs.infoset.serialize(root)
```

The result is an infoset. It is a tree of objects, arrays and strings that follow the abstract description defined by the W3C:

```
document ==
{   // prototype: xs.infoset.document,
    children: [
        {   // prototype: xs.infoset.element,
            // parent: document,
            name: "computer",
            children: [
                {   // prototype: xs.infoset.element,
                    // parent: document.children[0],
                    name: "user",
                    attributes: {
                        {   // prototype: xs.infoset.attribute,
                            // parent: document.children[0].children[0],
                            name: "name",
                            value: "Flore"
                        }
                    }
                    children: [
                        {   // prototype: xs.infoset.cdata,
                            // parent: document.children[0].children[0],
                            value: "VHJpcGVsIFdlc3RtYWxsZQ=="
                        }
                    ]
                }
            ]
        }
    ],
    // element: document.children[0],
    encoding: "UTF-8",
    version: "1.0"
}
```

Vice versa the document can be parsed from an infoset with:

```
root = xs.infoset.parse(document)
```

3 Prototypes

An information set is a tree of information items. The prototypes of the information items instances are defined by XS.

3.1 xs.infoset.document

The XML document is represented by the document item. It is the root of the tree and has the following properties:

children
The array of element, cdata or pi items contained by the document. Must contain one and only one element item.

element
The element item of the document.

encoding
The encoding of the document. Usually "UTF-8".

version
The version of XML of the document. Usually "1.0".

3.2 xs.infoset.element

XML elements are represented by element items. They are the nodes of the tree and have the following properties:

parent
The element or document item that contains the element.

children
The array of element, cdata or pi items contained by the element. Undefined if there is no content.

name
The name of the element.

namespace
The namespace of the element. Undefined if there is no namespace.

prefix
The prefix of the element. Undefined if there is no prefix.

attributes
The array of attributes items, neither named nor prefixed by "xmlns", contained by the element. Undefined if there is no such attribute.

xmlnsAttributes
The array of attributes items named or prefixed by "xmlns" contained by the element. Undefined if there is no such attribute.

instance
If the infoset is serialized without the xs.infoset.SERIALIZE_NO_REFERENCE flag, a reference to the instance that has been serialized into the element.

3.3 xs.infoset.attribute

XML attributes are represented by attribute items. They have the following properties:

parent
The element that contains the attribute.

name
The name of the attribute.

namespace
The namespace of the attribute. Undefined if there is no namespace.

prefix
The prefix of the attribute. Undefined if there is no prefix.

value
The value of the attribute.

3.4 xs.infoset.cdata

Character data are represented by cdata items. They have the following properties:

parent
The element or document item that contains the character data.

value

The content of the character data.

3.5 xs.infoset.pi

Processing instructions are represented by pi items. They have the following properties:
parent
The element or document item that contains the processing instruction.
name
The name of the processing instruction.
namespace
The namespace of the processing instruction. Undefined if there is no namespace.
prefix
The prefix of the processing instruction. Undefined if there is no prefix.
value
The content of the processing instruction.

3.6 xs.infoset.comment

Comments are represented by comment items. They have the following properties:
parent
The element or document item that contains the comment.
value
The content of the comment.

3.7 __xs__infoset

If the infoset is parsed without the xs.infoset.PARSE_NO_REFERENCE flag and if an instance is parsed from an element item, a reference to the element item is available to the instance in its __xs__infoset property.

4 Features

Besides the prototypes of the information items, the features related to the information set are also available through the xs.infoset prototype.

4.1 xs.infoset.parse (document, [[flags], prototype, ...])

To parse an infoset, call xs.infoset.parse. The first parameter must be an instance of xs.infoset.document. The second parameter is optional, its default value is xs.infoset.PARSE_DEFAULT. Pass a combination of

- xs.infoset.PARSE_NO_ERROR to get no error,
- xs.infoset.PARSE_NO_WARNING to get no warnings,
- xs.infoset.PARSE_NO_REFERENCE to get no reference from the parsed instances to element items.

The remaining parameters are also optional, it is the list of prototypes defined by a grammar with a root pattern that the application accepts to parse. Pass no prototype to accept all of them. The result is an instance of a prototype defined by a grammar with a root pattern.

When parsing element, attribute and pi items, xs use their namespace property, if any, but ignore their prefix property.

4.2 xs.infoset.serialize(instance, [flag])

To serialize an infoset, call xs.infoset.serialize. The first parameter must be the instance of a prototype defined by a grammar with a root pattern. The second parameter is optional, its default value is xs.infoset.SERIALIZE_DEFAULT. Pass

xs.infoset.SERIALIZE_NO_REFERENCE to have no reference from the element items to the serialized instances. The result is an instance of xs.infoset.document.

4.3 xs.infoset.scan(string)

To scan the text of an XML document into an infoset, call xs.infoset.scan. The parameter must be a string. The result is an instance of xs.infoset.document.

4.4 xs.infoset.print(document, [flag])

To print the text of an XML document from an infoset, call xs.infoset.print. The first parameter must be an instance of xs.infoset.document. The second parameter is optional, its default value is xs.infoset.PRINT_DEFAULT. Pass xs.infoset.PRINT_NO_COMMENT to skip the comment items. The result is a string.

The result adopts the lexical and syntactical conventions of canonical XML:

- There is no XML declaration.
- No line break, no space, no tab is added or removed between tags.
- Empty elements have a start tag and an end tag.
- Attributes are separated by one space, without space between the name, the '=' and the value. The value is delimited by double quotes.
- Attributes are sorted, the first key is the namespace, the second key is the name, xmlns attributes precede non xmlns attributes.
- In the value of an attribute, the characters '&', '<', '"', '\t', '\n' and '\r' are escaped with the entities & > " 	 ;
 and
- In the value of a cdata, the characters '&', '<', '>', and '\r' are escaped with the entities & > < and
- The name and value of a processing instruction are separated by one space.
- In the value of a processing instruction or a comment, no character is escaped.
- Processing instructions or comments before the document element are followed by a line break.
- Processing instructions or comments after the document element are preceded by a line break.

5 Applications

The prototypes of the information items defined by XS have no behavior. Applications can override them to give them the behavior they want, for instance to process a scanned or serialized infoset.

```
<object name="application">
    <object name="infoset" prototype="xs.infoset">
        <object name="document" prototype="xs.infoset.document">
            <function name="process">
                // change document
                var array, c, i;
                array = this.children;
                if (array)
                    for (c = array.length, i = 0; i < c; i++)
                        array[i].process();
            </function>
        </object>
        <object name="element" prototype="xs.infoset.element">
            <function name="process">
                // change element
                var array, c, i;
                array = this.xmlnsAttributes;
                if (array)
                    for (c = array.length, i = 0; i < c; i++)
```

```
                    array[i].process();
                array = this.attributes;
                if (array)
                    for (c = array.length, i = 0; i < c; i++)
                        array[i].process();
                array = this.children;
                if (array)
                    for (c = array.length, i = 0; i < c; i++)
                        array[i].process();
            </function>
        </object>
        <object name="attribute" prototype="xs.infoset.attribute">
            <function name="process">
                // change attributes
            </function>
        </object>
        <object name="cdata" prototype="xs.infoset.cdata">
            <function name="process">
                // change cdata
            </function>
        </object>
        <object name="pi" prototype="xs.infoset.pi">
            <function name="process">
                // change pi
            </function>
        </object>
        <object name="comment" prototype="xs.infoset.comment">
            <function name="process">
                // change comment
            </function>
        </object>
        <function name="scan" params="string">
            var result = xs.infoset.scan.call(this, string);
            result.process();
            return result;
        </function>
        <function name="serialize" params="root">
            var result = xs.infoset.serialize.call(this, root);
            result.process();
            return result;
        </function>
    </object>
</object>
```

Notice that the overriding prototypes are defined with the same name in an instance of xs.infoset. When the application will call application.infoset.scan or application.infoset.serialize, XS will build the results with instances of the overriding prototypes.

6 Remarks

There are differences between the ways XS and the W3C define an infoset. Mostly, for the sake of brevity and clarity, some prototypes and properties have different names.

6.1 DTD

There are no prototypes and no properties for DTD related information items since no feature use them. Applications can however define them, they will be ignored.

6.2 Processing instruction

In XS, processing instructions can have a namespace and a prefix. It has been useful for some documents. In most cases the target of the pi item defined by the W3C will be the name of the pi item defined by XS.

6.3 Synthetic infoset

In XS all namespaces and prefixes are defined by the root, a.k.a. the document element. The in-scope namespaces property and the namespace information item are omitted. If necessary applications can build them from the xmlnsAttributes property of the element property of the document item and from the namespace properties of the element, attribute and pi items.

---

[i] This document is part of xslib.

xslib is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

xslib is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with xslib; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA