XMQ/HTMQ — see XML/HTML in a new light

by Fredrik Öhrström 2024-06-07 https://libxmq.org

Rationale XMQ offers a visual appearance for config files that is more readable and writeable for humans compared to XML. The visual appearance of XML is a contributing factor to why programmers rarely pick XML as the initial configuration file format for newly developed software. However after some time of development the configuration file has grown in complexity and you need full XML capabilities, but by then it is too late to switch to XML. XMQ solves this problem.

```
config {
    name   = Restore
    driver = /drivers/r.sh
    id     = 90234578
    cron   = '30 08 10 06 *'
    url    = https://a.b.c/api?x=2
}
```

```
<config>
  <name>Restore</name>
  <driver>/drivers/r.sh</driver>
  <id>90234578</id>
  <cron>30 08 10 06 *</cron>
  <url>https://a.b.c/api/r?x=2</url>
</config>
```

Tooling XMQ permits multiple root nodes and the tooling offers an implicit root node when loading the config file. If the first element is not that root node, it will be added. This means that the initial config file can be as simple as this:

```
name   = Restore
driver = /work/drivers/restore.sh
id     = 90234578
cron   = '30 08 10 06 *'
fetch  = https://a.b.c/api/restore
```

```
XMQDoc *doc = xmqNewDoc();
xmqParseFile(doc, "/etc/config.xmq", "config", 0);
name = xmqGetString(doc, "/config/name");
id = xmqGetLong(doc, "/config/id");
```

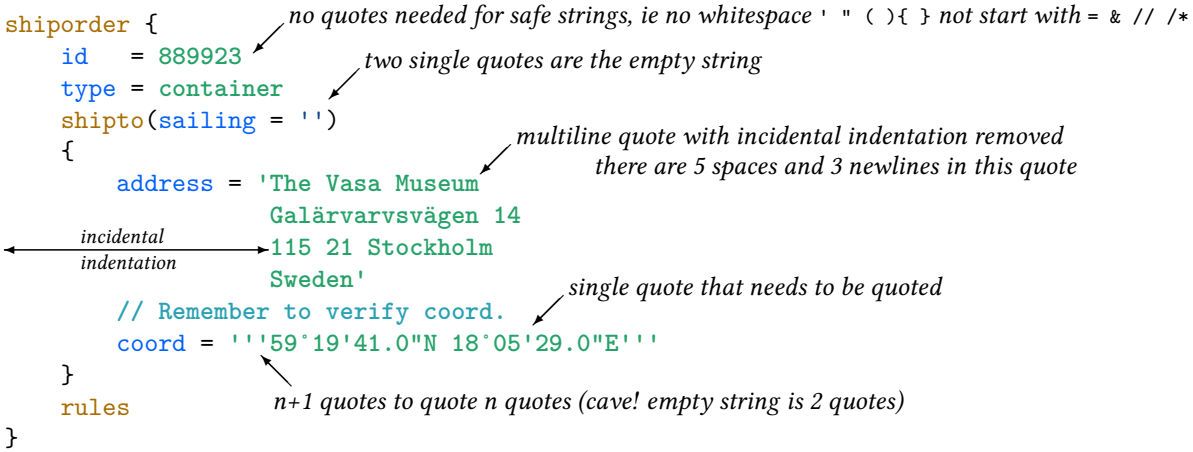
The standalone xmq tool can convert from such a root-less config file to XML like this:

```
xmq /etc/config.xmq add-root config to-xml > config.xml
```

The same tool can read and write any XML/HTML file since XMQ is 100% compatible and can pretty print the XMQ/HTMQ (terminal, browser, tex) and perform other functions on the DOM.

```
xmq work.xml
xmq work.xml browse
xmq page.htmq to-html > page.html
xmq data.xml transform work.xslq to-text > report.txt
xmq index.html delete //style delete //script page
```

Whitespace It is always possible to pretty print XMQ because whitespace is either *separating* or *content*. The content whitespace must always be quoted and is passed to the application, which then distinguishes between significant and insignificant.



The shiporder only has content whitespace in the address and in the coord.

COMPACT Every XMQ file can be printed in compact form on a single line where separating whitespace between tokens is minimized and the actual newlines are escaped.

```
shiporder{id=889923 type=container shipto(sailing=''){address=('The Vasa
Museum'&#10;'Galärvarvsvägen 14'&#10;'115 21 Stockholm'&#10;'Sweden')/*Re
member to verify coord.*/coord='''59°19'41.0"N 18°05'29.0"E'''}rules}
```

XMQ can always switch between a compact and a pretty printed layout. Compact XMQ is useful for log files where each appended line can be a complete and valid XMQ document.

JSON There is an easy to understand mapping from JSON to XMQ/XML. The xpath to select every what below is `"/_/todos/_/what"` the underscores represents the missing object types in JSON.

```
_ {
    todos(A)
    {
        _ {
            id   = 5
            what = 'Solve a cube'
        }
        _ {
            id   = 6
            what = 'Bake pastries'
        }
    }
    id(S) = 827309
}
```

```
{
    "todos":
    [
        {
            "id": 5,
            "what": "Solve a cube"
        },
        {
            "id": 6,
            "what": "Bake pastries"
        }
    ],
    "id": "827309",
}
```

XSLT It is easier to write XSLQ transforms since the content whitespace is visible. XSLT files can often not be pretty printed because this would introduce unwanted whitespace in the output.

```
xsl:stylesheet(version    = 1.0
               xmlns:xsl = http://www.w3.org/1999/XSL/Transform)
{
    xsl:template(match = _/todos)
    {
        html {
            body {
                table(border = 1)
                {
                    xsl:for-each(select = _)
                    {
                        tr {
                            td {
                                xsl:value-of(select = what)
                            }
                        }
                    }
                }
            }
        }
    }
    xsl:template(match = total)
}
```

The Q in XMQ/HTMQ/XSLQ/XSQ does not mean anything. It is merely an available file suffix.

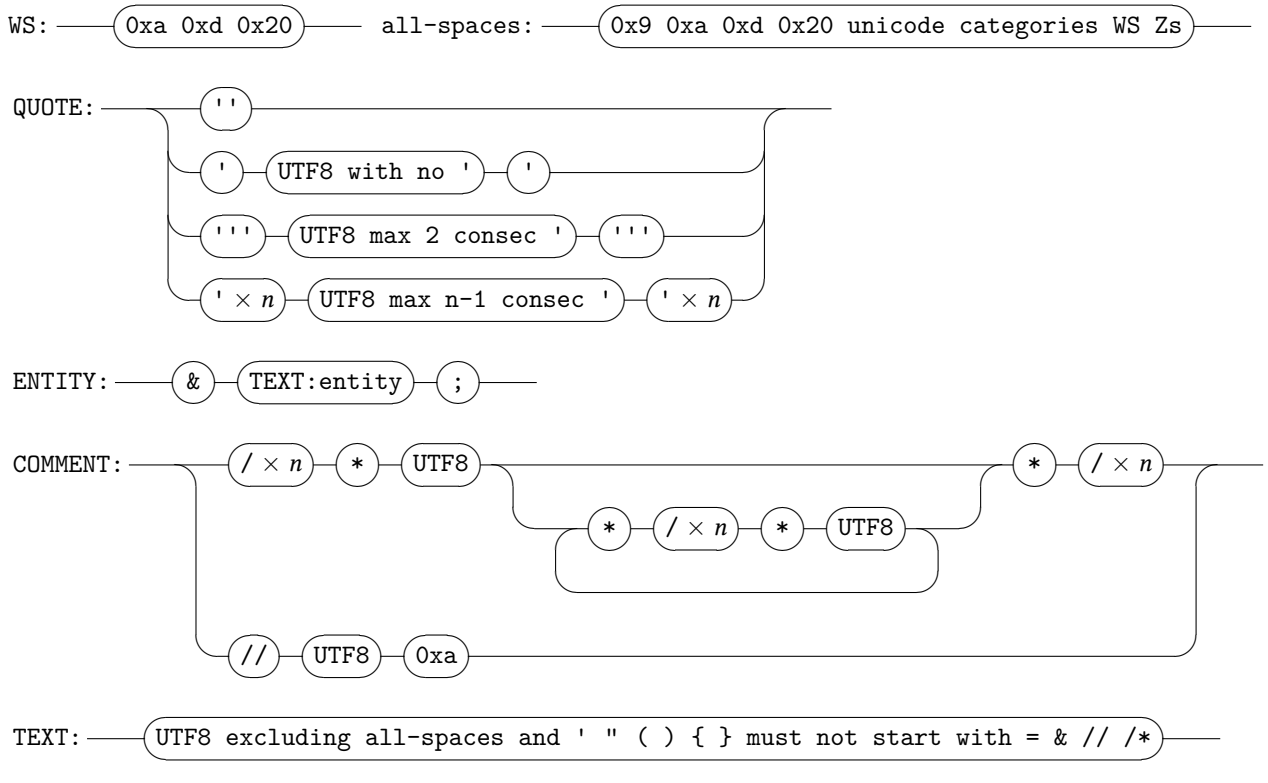
```
xmq todos.json select /_/todos/_/what
what = 'Bake pastries'
what = 'Solve a cube'
```

```
xmq todos.json transform todos.xslq
html {
    body {
        table(border = 1)
        {
            tr {
                td = 'Solve a cube'
            }
            tr {
                td = 'Bake pastries'
            }
        }
    }
}
```

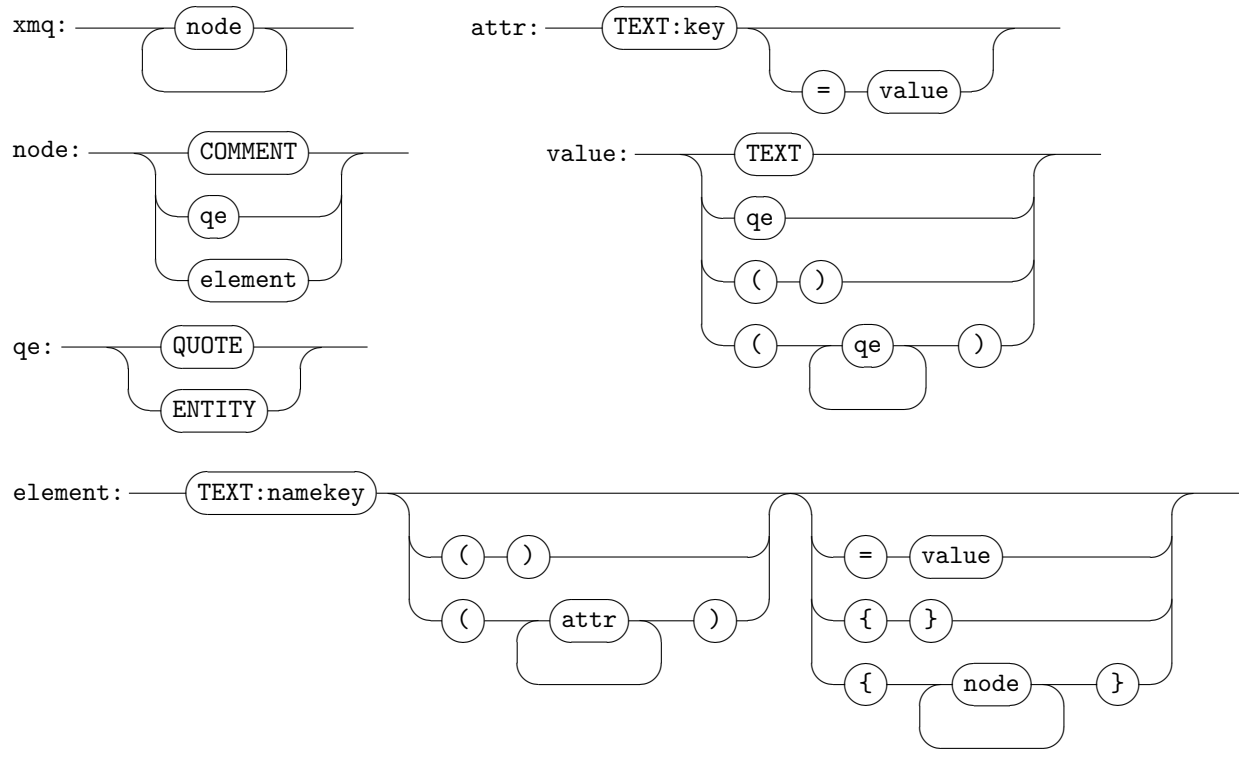
Specification for XMQ by Fredrik Öhrström

Input must be valid UTF8 (0x9 | 0xa | 0xd | [20-d7ff] | [e000-fffd] | [10000-10ffff])
CRLF pairs (0xd 0xa) and standalone CR (0xd) are treated as LF (0xa) when parsing.

LEXER



PARSER



RULES

```
    TEXT:namekey TEXT:key and TEXT:entity
r1. Must start with a letter or underscore.
r2. Cannot start with the letters xml (or XML, or Xml, etc).
r3. Can contain letters, digits, hyphens, underscores and periods.
r4. Can contain a single colon separating the TEXT into two parts, each following r1,r2,r3.

    Three permitted exceptions to rule r1 and r3.
r5. A single !DOCTYPE before the first element and ?pi elements and &#..; entities.

    If quoted content contains at least one newline then:
r6. Leading (and ending) WS will be trimmed to newlines only, leaving one out.
r7. All spaces before a newline are removed.
r8. Incidental indentation (some spaces after a newline) is removed.
r9. The indent to be removed is the minimum source code indentation
    for text within the block where empty lines are ignored.
r10. The first line is prefixed with spaces if the following lines
     have lesser source code indentation.

     TEXT:namekey is (only for syntax highlighting) either a name or a key.
r11. it is a key if '=' follows immediately (ie no attributes), otherwise it is a name.

A quote with only spaces and a single newline is equivalent to the empty string.
The element age=123 is shorthand for age{'123'}
```

EXAMPLES

```
car {
    // An example structure.
    regnr = 'ABC 123'
    color = red
    img   = /www/y.png
    tag   = <car>
}
```

```
<car>
  <!-- An example structure. -->
  <regnr>ABC 123</regnr>
  <color>red</color>
  <img>/www/y.png</img>
  <tag>&lt;car&gt;</tag>
</car>
```

```
div(id = 32) {
    h1 = Welcome!
    'Rest here weary
     traveller:'
    a(href = https://a.b.c) {
        img(url = /img/i.png)
        'Click here!'
    }
}
```

```
<div id="32">
  <h1>Welcome!</h1>
    Rest here weary
traveller:<a href="https://a.b.c">↩
↪<img url="/img/i.png">Click here!</a>
</div>
```

Html cannot be pretty printed with newlines here,
whereas xmq can be pretty printed without introducing whitespace.

CORNERS

```
Explicit spaces: abc = '    '    abc {'    '}
Spaces surrounding newline: abc = (' ' #10; ' ')   abc {' ' #10; ' '}
Value with leading/ending quotes: x = ( #39; 'quoted quote' #39; ) or:
'''
'quoted quote'
'''
or: x { #39; 'quoted quote' #39; }
A single newline: abc = #10;       abc { #10; }
or: abc = '


          '
```