A look at Widex Requirements

Dave Raggett <dsr@w3.org>

A look at Widex Requirements

Drilling down on the requirements

- A look at some proposals for remote DOM updates
  - http://www.w3.org/TR/rex/
  - http://xmldb-org.sourceforge.net/xupdate/
  - draft-ietf-simple-xml-patch-ops-02.txt
- Requirements for DOM updates
- Requirements for DOM events

Remote Events for XML (REX)

- W3C working draft - 2nd February 2006
- Defined by task forced composed of members from CDF and SVG working groups
- Initiated at the request of OMA and W3C members
- Designed to support streaming of updates to SVG for use on mobile devices

Remote Events for XML (REX)

- Uses DOM3 Mutation Events in reverse to cause changes instead of indicating such changes
- Uses XPath to identify where changes are to be applied
- Each <rex></rex> element can contain one or more events
- REX fully supports XML namespaces
- Events can be specified to apply at a future time

REX examples

Setting an attribute

```
<rex xmlns='http://www.w3.org/2006/rex'>
  <event target='id("spot")/@fetch'
   name='DOMAttrModified' newValue='ball'/>
</rex>
```

REX examples

Inserting an element

```
<rex xmlns='http://www.w3.org/2006/rex'>
  <event target='/html/body/table[2]'
   name='DOMNodeInserted' position='7'>
    <tr xmlns='http://www.w3.org/1999/xhtml'>
      <td>Rover</td>
      <td>Alpine Labrador</td>
      <td class='food'>bone</td>
    </tr>
  </event>
</rex>
```

Inserts new row at position 7 in 2nd table. Element previously at position 7 will now be at position 8.

REX examples

Removing several elements

```
<rex xmlns='http://www.w3.org/2006/rex'>
  <event target='/svg/circle[@class="poodle-stylist-location"]'
   name='DOMNodeRemoved'/>
</rex>
```

REX examples

Replacing an element

```
<rex xmlns='http://www.w3.org/2006/rex'>
  <event target='id("femur")'
   name='DOMNodeRemoved'>
    <bone xmlns='http://example.org/BoneML'
     xml:id='tibia'>
      <taste>good</taste>
      <smell>excellent</smell>
      <solidity>medium</solidity>
      <availability>common</availability>
    </bone>
  </event>
</rex>
```

REX examples

Updating character data directly

```
<rex xmlns='http://www.w3.org/2006/rex'>
  <event target='/svg:svg/svg:g[2]/svg:tspan[7]/text()'
   name='DOMCharacterDataModified' newValue='Hello World!'/>
</rex>
```

REX examples

Replacing an entire document

```
<rex xmlns='http://www.w3.org/2006/rex'>
  <event target='/' name='DOMNodeRemoved'>
    <svg xmlns='http://www.w3.org/2000/svg'
     viewBox='0 0 300 400'>
      <defs>
        <!-- ... --->
      </defs>
      <g>
        <rect x='42' y='27' width='100' height='200' fill='red'
        <!-- ... --->
      </g>
    </svg>
  </event>
</rex>
```

REX Processing Model

- Processing must stop upon encountering well-formedness errors
- Each event element must be mapped into a DOM 3 Event object according to its type
- The target path must be resolved into a set of nodes
  - For small devices and interoperability, need well defined subset of XPath, including predicates and functions
- The event(s) must then be applied to each such node (subject to the timestamp attribute)
- Implementation sets the bubbles/cancelable fields based upon the event type
- timestamp must be set to zero if not provided by event

Note that REX doesn't provide a means to extract nodes from one part of the tree and copy them to another, but this could be a future extension.

XUpdate

- Devised by XML:DB and available on SourceForge
  - Requirements, last release November 24, 2000
  - XUpdate Working Draft, last release September 14, 2000
  - Reference implementation available
- Uses XPath for targeting nodes
- xupdate:modifications is the root element and may contain
  - xupdate:insert-before, xupdate:insert-after, xupdate:append, xupdate:remove, xupdate:rename, xupdate:variable, xupdate:value-of and xupdate:if
- Insertion elements can contain
  - xupdate:element, xupdate:attribute, xupdate:text, xupdate:processing-instruction and xupdate:comment
- Lacks a well defined processing model
- Lacks timestamp mechanism, and all actions apply immediately

XUpdate examples

Given

```
<addresses version="1.0">
  <address id="1">
    <fullname>Andreas Laux</fullname>
    <born day='1' month='12' year='1978'/>
    <town>Leipzig</town>
    <country>Germany</country>
  </address>
</addresses>
```

This will insert a new address element after the first:

```
<xupdate:modifications version="1.0"
         xmlns:xupdate="http://www.xmldb.org/xupdate">
  <xupdate:insert-after select="/addresses/address[1]" >
    <xupdate:element name="address">
      <xupdate:attribute name="id">2</xupdate:attribute>
      <fullname>Lars Martin</fullname>
      <born day='2' month='12' year='1974'/>
      <town>Leizig</town>
      <country>Germany</country>
    </xupdate:element>
  </xupdate:insert-after>
</xupdate:modifications>
```

XUpdate and the use of variables

Named variables can used as macros for path expressions:

```
<xupdate:modifications version="1.0"
        xmlns:xupdate="http://www.xmldb.org/xupdate">

  <!-- pick out name of town from first address -->
  <xupdate:variable name="town"
        select="/addresses/address[0]/town"/>

  <!-- and append new address with that town -->
  <xupdate:append select="/addresses">
    <xupdate:element name="address">
      <xupdate:value-of select="$town"/>
    </xupdate:element>
  </xupdate:append>
</xupdate:modifications>
```

In principle, REX could use value-of for copying sub-trees.

XML Patch

- Internet Draft: draft-ietf-simple-xml-patch-ops-02.txt (6 March 2006)
- Add, replace and remove operations, using XPath to target nodes
- Intended for patch operations on remote XML documents as an alternative to sending the entire document
  - e.g. for XML documents representing presence information (PIDF)

Example:

```
<p:diff xmlns="urn:ietf:params:xml:ns:xxx"
        xmlns:y="urn:ietf:params:xml:ns:yyy"
        xmlns:p="urn:ietf:params:xml:ns:diff">
  <p:add sel="doc/elem[@a='foo']">
    <child id="ert4773">
      <y:node/>
    </child>
  </p:add>
  <p:replace sel="doc/note/text()">Patched doc</p:replace>
  <p:remove sel="*/elem[@a='bar']/y:child" ws="both"/>
  <p:add sel="*/elem[@a='bar']" type="@b">new attr</p:add>
</p:diff>
```

Widex requirements for remote updates

For DOM Updates, the following points seem appropriate:

- There should be a well defined processing model
- XPath expressions should be used to identify the nodes on which a given operation is to be applied
- Updates should be able to identify nodes in designated XML namespaces
- Updates may be associated with a timestamp indicating the time when the update is to be applied
- Updates need to cover elements, attributes and text nodes

Widex requirements for remote updates

- Updates may support operations on other kinds of DOM nodes such as processing instructions and comments
- Updates should allow for copying DOM subtrees
- There should be a means for packaging multiple updates into a single message
- Updates should identify the session and the DOM tree to which they apply
  - May be implicit in the connection

Widex requirements for remote events

- Identify the session
  - The session should be identified by a URI as this provides the basis for being able to make statements about sessions which can be temporary or persistent. The URI doesn't need to be dereferencable.
    - May be implicit in the connection
- Identify the Widex renderer
  - Each Widex server can have a relationship with multiple Widex renders. The renderer should be identified by a URI as this provides the basis for being able to make statements about the renderer. The URI doesn't need to be dereferencable.
    - May be implicit in the connection
- Identify the DOM tree
  - A Widex renderer may have multiple DOM trees. This identifier is also used by DOM Updates to indicate which tree to update via an XPath function that maps it to the root node of the tree.
    - May be implicit in the connection

Widex requirements for remote events

- Identify the event target
  - This is in accordance with the DOM event model. For serialization, the target can be identified via an XPath expression.
- Identify the event type
  - For DOM2 this is a DOM string, but for DOM3, it is a string within an identified namespace.
- Support namespaces on event types
  - This is needed for use with DOM3.
  - Use QName for event type, e.g. geo:LocationUpdate
  - Or specify namespace in separate attribute/element

Widex requirements for remote events

- Include a timestamp for when the event was raised
  - This allows the Widex server to compare the times of events from a given Widex renderer.
  - A candidate data type is the number of milliseconds since 1 January 1970 00:00:00 GMT, as computed by the Widex renderer.
  - Easy to implement on a wide range of platforms, e.g. using the ECMAScript Date object's getTime() function.
  - Interoperable with timestamps in REX
  - Each Widex renderer and server is likely to report different times, and this should be taken into account when designing applications.
  - Is there a need to support Widex renderers that are unable to provide such timestamps? If so, how is the absence of the timestamps to be indicated?

Widex requirements for remote events

- Indicate how to serialize DOM events as XML
  - The W3C DOM specs define DOM events in terms of IDL and there needs to be a mapping from this into XML for interoperable exchange of DOM events.
- Allow for new kinds of event types and payloads
- Allow for packaging multiple events as a single message
  - Efficient transmission of events may necessitate the means to package several events into a single message, and there should be a means to represent this packaging in XML.
- Provide a means to indicate which events should be remoted
  - A Widex renderer may internally generate a large variety of DOM events, not all of which are relevant to the associated Widex server. The set of relevant event types may be covered by prior agreement or identified when the Widex renderer is bound to the Widex

server. For efficiency, it should be possible to identify event types both individually and by named sets or supersets of event types. A precedent is given by VoiceXML which features a hierarchical event naming mechanism.

Other kinds of updates

Some User Interface updates go beyond updating the markup:

- Which node has the input focus?
- The position of the caret within a text input field?
- The text selection?

These can be set via events sent from the Widex server to the Widex renderer.

Should these events be part of the Widex specification or left to the application layer?

Suggestions for next steps

- Following further discussion, update the draft requirements
- Prepare draft specification(s), seeking to align with REX
- What's missing from REX?
  - Identification of session, DOM tree and Widex Renderer
  - Means to copy subtrees
  - Support for remoting arbitrary DOM events
  - Treatment of session parameters for session initiation
  - Bindings to specific protocols, e.g. SIP and HTTP
- One or two specifications?
- How to work with W3C REX task force?

Extending REX with value-of

Wrapping an existing element with another

```
<rex:rex xmlns:rex='http://www.w3.org/2006/rex'>
  <event target='id("femur")' name='DOMNodeRemoved'>
    <leg>
      <rex:value-of select='.'/>
    </leg>
  </rex:event>
</rex:rex>
```