SAXONICA.COM
XSLT AND XQUERY PROCESSING

Multi-User Interaction using Client-Side XSLT 2.0

Michael H. Kay
O'Neil D. Delpratt

xmlprague

Client-Side XSLT 1.0

- XSLT 1.0 now available in nearly all browsers
  - but took a while to get there
- Still a "Web 1.0" architecture – static content
- Not widely used
  - issues around 100% interoperability
  - issues around XML support by browsers
  - limitations of XSLT 1.0
  - still need to do the interesting stuff in Javascript
  - fashion

Saxon-CE

- XSLT 2.0 in the browser
- Java ⇒Javascript cross-compilation using GWT
- XSLT extensions to handle
  - user interaction events
  - animation events
  - access to the Javascript browser environment

Progress

- First prototype shown at XML Prague 2011

- General release product since June 2012
- Version 1.1 will be open source

This Talk

- Two demonstrations
  - Technical documentation browser
  - Two-player chess game over Twitter
- Purpose:
  - demonstrate possibilities for application design
  - show how XSLT handles
    - user interaction
    - server communication
  - illustrate benefits of using XSLT over Javascript

Documentation Browser

- Documents stored on server in XML
  - mainly, “abstract” XHTML 5.0
  - custom XML formats for Javadoc and for function specs
- No server-side component
- Single-page application using hash-bang URIs

User Interface Features

- Persistent and visible #! URIs
- Intra-site linking
- Table of Contents
- Search

Template for handling TOC clicks

```
<xsl:template match="*" mode="handle-itemclick">
    <xsl:variable name="ids" select="(., ancestor::li)/@id" as="xs:string*"/>
    <xsl:variable name="new-hash" select="string-join($ids, '/')"/>
      <xsl:choose>
        <xsl:when test="@class eq 'open'">
          <ixsl:set-attribute name="class" select="'closed'"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:sequence select="js:disableScroll()"/>
          <xsl:choose>
            <xsl:when test="f:get-hash() eq $new-hash">
              <xsl:variable name="new-class" select="f:get-open-class(@class)"/>
              <ixsl:set-attribute name="class" select="$new-class"/>
              <xsl:if test="empty(ul)">
                <xsl:call-template name="process-hashchange"/>
              </xsl:if>
            </xsl:when>
            <xsl:otherwise>
              <xsl:sequence select="f:set-hash($new-hash)"/>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:otherwise>
      </xsl:choose>
  </xsl:template>
```

Navigating to a new page

```
<xsl:function name="f:set-hash">
  <xsl:param name="hash"/>
  <ixsl:set-property name="location.hash" select="concat('!',$hash)"/>
</xsl:function>
```

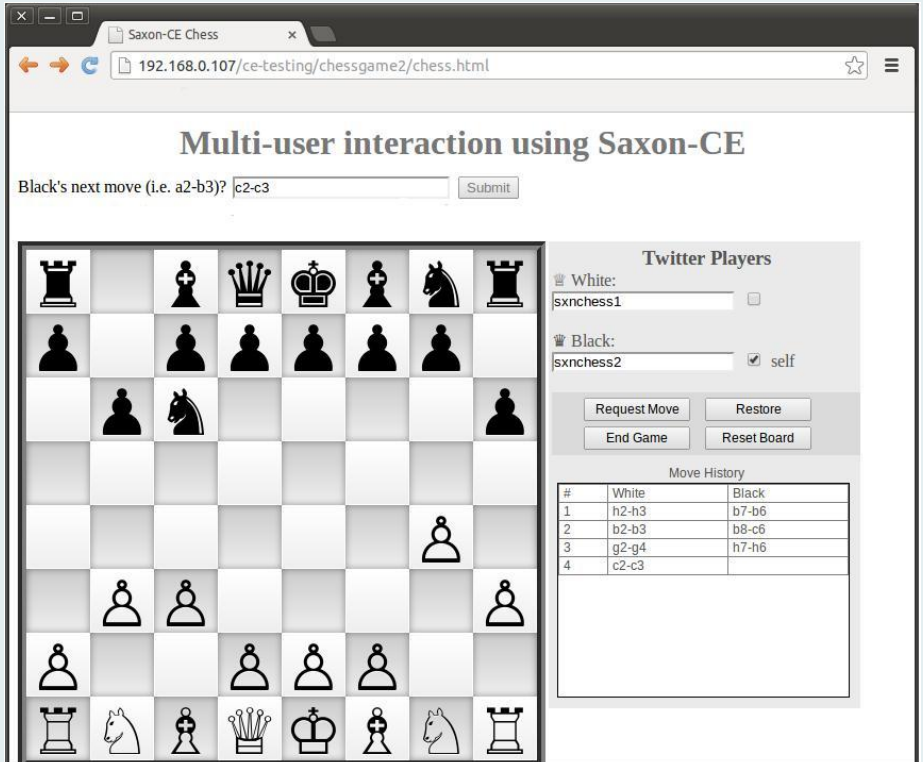
- Sets the Javascript property location.hash
- Updates the title bar
- Updates the history (so the back button works)
- Note: side-effects!
  - Uses a Pending Update List like XQuery Update

Searching

- Entry in search box fires a template rule
- Template rule uses fn:contains() to get a list of links to all the selected elements
- Results navigation activates these links
- Search term highlighting uses Javascript extension functions

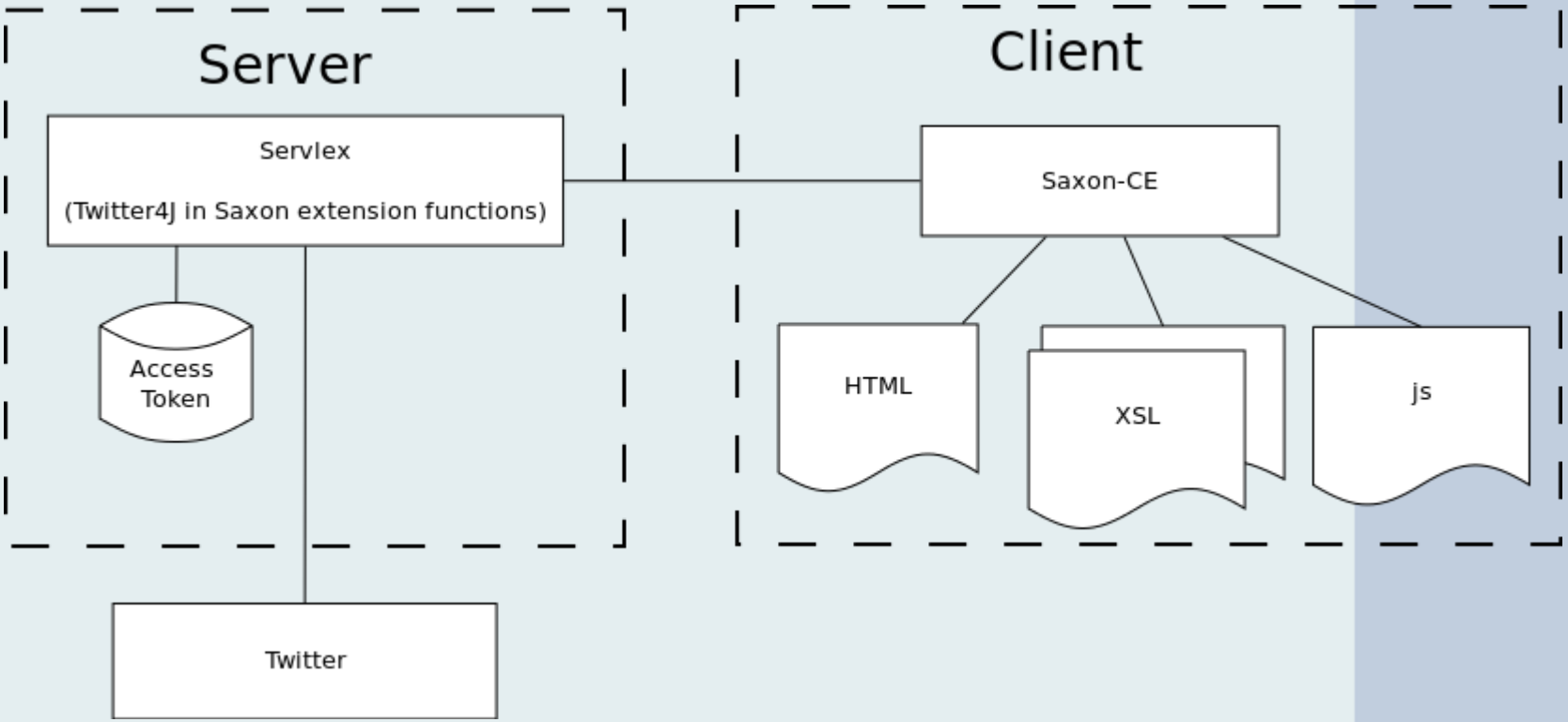
Multi-player Chess Game

- Two player game
- Communication over twitter
- Interactive user interface
- The chess game logic



SAXONICA.COM
XSLT AND XQUERY PROCESSING

Chess Game Architecture



Chess Game Demonstration

Server-side Component

- Proxy communication with Twitter
- Implemented in Servlex
  - Handles HTTP requests
    - No game knowledge
    - Handles OAuth authentication and proxy messaging
  - Twitter4J as a Saxon extension function
    - OAuth data persisted

http://192.168.0.2/servlex/chess/updateStatus?**user=johnWhite**&**status=@maryBlack %20RNBQ**
e2-e4%20p:1

Client-Side Component

- Saxon-CE XSLT 2.0 processor for browser
- Single HTML page
- Stylesheets
  - Renders chess board
  - Handles user input
  - Validates moves
  - Send moves over twitter via proxy
  - Receives twitter timeline

Chess game logic

```
<xsl:template match="div[@data-piece='knight']" mode="is-valid-move"
                as="element(move-test)">
   <xsl:param name="moveFrom" as="xs:integer"/>
   <xsl:param name="moveTo" as="xs:integer"/>
   <xsl:param name="board" as="element(div)+"/>
   <xsl:variable name="destinationAvailable"
                 select="not($board[$moveTo]/@data-colour = @data-colour)"/>
   <xsl:variable name="rowDistance" as="xs:integer"
                 select="f:row($moveTo) - f:row($moveFrom)"/>
   <xsl:variable name="columnDistance" as="xs:integer"
                 select="f:column($moveTo) – f:column($moveFrom)"/>
   <xsl:variable name="is-valid" as="xs:boolean"
                 select="$destinationAvailable and abs($rowDistance) *
                         abs($columnDistance) = 2"/>
   <move-test is-valid="{if ($is-valid) then 'yes' else 'no'}"/>
</xsl:template>
```

IsValidMove function

```
<xsl:function name="f:isValidMove" as="element(piece-move)">
   <xsl:param name="piece" as="element(div)"/>
   <xsl:param name="moveFrom" as="xs:integer"/>
   <xsl:param name="moveTo" as="xs:integer"/>
   <xsl:param name="board" as="element(div)*"/>
   <piece-move>
      <xsl:choose>
      <xsl:when test="$moveFrom = $moveTo">
         ...
      </xsl:when>
      <xsl:when test="$board[$moveFrom][self::empty]">
         ...
      </xsl:when>
      <xsl:when test="not($board[$moveTo][div/@data-piece eq 'empty' or
                          div/@data-colour != $piece/@data-colour])">
         ...
      </xsl:when>
      <xsl:otherwise>
         <xsl:variable name="move-test" as="element(move-test)">
         <xsl:apply-templates select="$piece" mode="is-valid-move">
            <xsl:with-param name="moveFrom" select="$moveFrom"/>
            <xsl:with-param name="moveTo" select="$moveTo"/>
            <xsl:with-param name="board" select="$board"/>
        </xsl:apply-templates>
     ...
```

Conclusion

- Client-side XSLT 2.0 can be used to create rich Web 2.0 applications
  - handling the user interaction
  - handling access to services on the web
  - and of course XML rendition
- Good cross-browser interoperability
- Plays well with Javascript but eliminates most JS coding
- Coming soon: open source

URLs

Documentation browser:
http://www.saxonica.com/documentation9.4-demo/index.html

Chess game demo:
http://dev.saxonica.com/ce-testing/chessgame/chess.html