JChemTidy: A Tool for Converting Chemical Web Document Collections to an XHTML Representation

Georgios V. Gkoutos,[a] Philip R. Kenway,[b] and Henry S. Rzepa*[a,]

[a] Department of Chemistry, Imperial College of Science, Technology and Medicine, London, SW7 2AY.
[b] Merck Sharp and Dohme Research Laboratories, Neuroscience Research Centre, Terlings Park, Harlow, Essex, CM20 2QR.

Received July 26, 2000

Ⓦ **This paper contains enhanced objects available on the Internet at http://pubs.acs.org/journals/jcisd8.**

A robot-based procedure is described for traversing a collection of hyperlinked documents written in HTML and converting these to the XML-compliant and well-formed XHTML representation. Transcluded chemical content invoked using <embed> or <applet> HTML calls are converted to the XHTML recommended <object> form. Additional attributes such as title or derived chemical attributes such as a SMILES descriptor are added to improve the indexing of the resulting document collection. Conformance tests for the popular Web browsers are reported.

INTRODUCTION

Since 1993, the World-Wide Web system has rapidly evolved in two key areas of chemical relevance. Standards defining the document markup language HTML have progressed, not always entirely consistently, through five versions. The latest version (XHTML 1.0) is based on the XML meta-language[1] and was designed to be interoperable with other members of the XML family such as Chemical Markup Language[1]. Currently however, the majority of Web-based chemical documents are expressed in older versions of HTML, and many documents may not conform to any explicit version. HTML was also never designed to carry chemical information, and this has resulted in the deployment of a variety of mechanisms for expressing or transcluding chemical content into Web pages, these varying widely in their degree of re-useability or chemical transformability. At one extreme are the bit-mapped images (GIF, JPEG), whilst the other is represented by inclusion of a variety of data formats defined by appropriate chemical MIME types[2] and expressed on static browser pages by invoking either browser plugins or Java applets. In addition to these HTML-based mechanisms, various procedures are used for dynamically processing chemical information based on either client-based scripts (e.g. JavaScript) or server-side requests (so-called CGI processes interfacing to remote databases) or server-side includes.

An ever-present concern which we address in the current article has been how to achieve self-consistency, quality, indexability and interoperability of the chemical content in a simple, low cost and reliable manner. In particular, mechanisms for preserving our ability to transform this content into future open Web-based standards such as XML, whilst avoiding over-dependence on specific closed and proprietary server-based solutions, need to be developed. Such transformations are seen as an essential feature of achieving the data immortality which must be a prerequisite of long term archival of Web-based chemical information.

Web-based chemical content is expressed using two predominant mechanisms: via bit map images (the <img> element) and by transclusion of external data files via one of the four HTML elements: <a>, <script>, <applet> and <embed>. Of these four HTML elements, <applet> and <embed> were never regarded as formal standards but were widely adopted from 1995 onwards because of support for them in Netscape, the then dominant browser. Unfortunately, this promoted the creation of further non inter-operating chemical data islands in many HTML documents. Use of <embed> is particularly unfortunate since this element is neither a well formed data container (the element not being closed with a </embed>), nor can it be validated in the sense that each attribute should be formally defined in a DTD (document type description). This is because the attributes of this element are only specified by the particular plug-in software used to support <embed>, and these vary according to the plug-in used.

We have previously reported[3] increasing browser support for the <object> element, which should be formally used to replace the <applet> and <embed> elements (now both formally deprecated in the XHTML standard). In the present article, we describe a largely automated mechanism for converting a chemically oriented Web site from the unvalidated and potentially ill formed (but parsable) "legacy" HTML codes to well-formed XHTML, and in the process converting elements such as <applet> and <embed> to <object> form.

PROCEDURES

We chose a robot-based mechanism for traversing the hyperlinks in a HTML collection starting from the so called remote root document specified by a full URL. The htdig indexing robot[4] has publically available source codes and most importantly has support for inserting an external parser for processing documents retrieved by the traversing process. The robot approach has the advantage that direct access to

10.1021/ci000396y CCC: $20.00
Published on Web 03/26/2001

Table 1. JChemTidy Classes

JChemTidy Class name	Description
Htdigfrontmain.class	External parser called from Htdig robot. Calls corresponding classes according to MIME type.
HtmlConvert.class	Constructs directories, calls JTidy, writes converted HTML file, reports errors.
HtmlConvertHref.class	Searches HTML document for anchors, reads file suffix, assigns MIME type, calls appropriate ExecProcess*Href.class or read*Href.class, passes anchor URL to external process or class, formats according to Chart 3.
ExecProcess*Href.class (*= Mol, xyz, pdb)	Evaluates SMILES string using external process, creates title string
read*Href.class (*= pdb, Mol, xyz)	Evaluates Molecular Formula from contents of local file, creates id string
HtmlConvertEmbed.class	Searches HTML document for <embed>, creates new object, assigns MIME from file suffix, calls ExecProcess(*)Href.class, passes anchor URL to external classes, formats as <object> according to Charts 2 and 3.
HtmlConvertApplet.class	Searches HTML document for <applet>, creates new object, assigns MIME from file suffix, calls ExecProcess(*)Href.class, passes anchor URL to external classes, formats as <object> according to Charts 2 and 3.

Chart 1

non XHTML Code	XHTML Converted Code
`<embed border="0" bgcolor="white" src="chair.pdb" name="chair" width="250" height="250" display3d="ball&stick" > <embed type="application/x-spt" width="12" height="12" button="push" target="chair" script="select *; color cpk; spacefill off; wireframe on">`	`<object data="chair.pdb" width="250" height="250" > <param name="bgcolor" value="white"/> <param name="name" value="chair"/> <param name="display3d" value="ball&stick"/> </object> <object type="application/x-spt" width="12" height="12"> <param name="button" value="push"/> <param name="target" value="chair"/> <param name="script" value="select *; color cpk; spacefill off; wireframe on"/> </object>`

Ⓦ The underlined material in this chart represents links which are available in the HTML version of this article, which is avialable on the Internet at http://pubs.acs.org.

the remote site is not required and that documents not specified in the overall document tree are not included. The procedure involved the following stages.

A first pass using htdig is used to produce a file containing the URL locators for the complete document collection, using a built in HTML parser which can follow hyperlinks. This built in parser is reasonably robust in that it can normally reliably follow the hyperlink tree of a collection even if the HTML describing this is not (within limits) well formed. An exception to the link following is when the document references files via a <form> declaration, requiring the user to select a file from a pull-down menu. In principle, all documents declared within a <form> should also be declared in a <link> statement to allow the complete tree to be traversed.

A second pass of htdig is made, but this time no document traversing is done, and instead each document is specified as a root document as defined by the URL file created in the first pass. The default parser is also reset from the internal HTML to an external parser termed JChemTidy written by us.

JChemTidy[5] is a collection of Java classes (Table 1) originating from two sources. The first set is called JTidy and is a Java version of the Tidy program developed by Raggett[6]. The remaining classes, written by us, handle the required chemical functionality, including operations such as processing the chemical data files, deriving properties such as molecular formulae and requesting derived information such as SMILES strings.

JTidy is a robust HTML parser which has been written to cope with a wide variety of older standards (and non-standards) and which can also cope with some badly formed HTML constructs, such as an unclosed <p> container. JTidy can be configured to output well formed XHTML, with an option to break each new instance of an HTML element onto a new line. This feature will be used to tokenise the HTML into elements, the values of any attributes of each element, and the contents of the element container. The tokenised output of JTidy is then passed back to our JChemTidy classes, which we use to transform specific elements such as <applet> and <embed> to the <object> element. This operation is not (currently) handled by JTidy itself. Whilst the structure of <applet> maps trivially onto <object>, the <embed> element requires mapping of specific attributes to <param> declarations (Chart 1). The attributes shown in this chart relate to those supported by the Chime browser plugin,[7] although other plugins such as Chem3D[8] can also be mapped. The mapping would be done as a superset of both, since unsupported attributes are simply ignored. The <img> element is not transformed, since parsing a low resolution bit map image to give chemical semantics is possible only with the greatest of difficulty. Although the GIF and PNG formats do allow invisible data fields to be included in the binary format, these are rarely used for chemical purposes.At this stage, one has an option to add redundancy not present in the original HTML (but not yet additional or transformed information). A significant drawback of the original <applet> or <embed> mechanism is that neither can default to the other. For example, pages which invoke <embed> for display using a browser plugin such as Chime or Chem3D cannot be viewed if the recipient does not have such a plugin installed. Conversely, pages which invoke <applet> cannot be viewed if the recipient has Java disabled. The result of these transforms can be

Chart 2

Plugin to Nested Object

```
<object data="models/codein.mol"
width="320" height="240"
title="object displayed using browser plugin" >
<param name="display3d" value="spacefill" />
<param name="bgcolor" value="white" />
<param name="rotationbars" value="true" />
     <object  classid="java:RenderBasic.class"
     width="320" height="240"
     codetype="application/java"
     codebase="http://..."
     archive="chemsymphony.lite.zip"
     standby="Loading Java"
     title="Object displayed using Java">
     <param name="model" value="models/codein.mol" />
         <object id="image" type="image/jpeg"
         data="vib.jpg" width="418" height="325"
         title="Display using image only">
         Text displayed if neither object
         nor image can be displayed
         </object>
     </object>
</object>
```

Applet to Nested Object

```
<object classid="java:RenderBasic.class"
width="320" height="240"
codetype="application/java"
codebase="http://..."
archive="chemsymphony.lite.zip"
standby="Loading Java"
title="Object displayed using Java">
<param name="model" value="models/codein.mol" />
    <object data="models/codein.mol"
    width="320" height="240"
    title="object displayed using browser plugin">
    <param name="display3d" value="spacefill" />
    <param name="bgcolor" value="white" />
    <param name="rotationbars" value="true" />
         <object id="image" type="image/jpeg"
         data="vib.jpg" width="418" height="325"
         title="Display using image only">
         Text displayed if neither object
         nor image can be displayed
         </object>
    </object>
</object>
```

Ⓦ The underlined material in this chart represents links which are available in the HTML version of this article, which is available on the Internet at http://pubs.acs.org.

included in the <object> tree (Chart 2). This example shows how a molfile format chemical data file originally referenced via an <embed> declaration can be presented to a browser. The first option for display is via an appropriate plugin preinstalled by the user into the browser. If this does not exist, then appropriate Java code is referenced, in this example the ChemSymphony Lite system.[9] Since security considerations do not allow the Java code to be acquired from a separate HTTP site, the appropriate code must be mounted on the HTTP server as part of the conversion process. Since there is not necessarily any available mapping from plugin to Java functionality, we only attempt to include such redundancy for the most common chemical data types, with corresponding chemical MIME such as chemical/x-pdb, chemical/x-mdl-mol, chemical/x-jcamp-dx and chemical/x-xyz for which both plugin and applet support is widely available. We also note that whilst all the original attributes can be mapped to a corresponding <param> declaration, not all these options will necessarily be supported in the alternate mode of presentation. For example, a <param name="script" value="select *; wireframe on"/> script declaration may not be honoured by a Java applet and will simply be ignored. We do note that ChemSymphony support for such Rasmol scripts has been developed.[10] It is also possible that the attribute name declared in the original HTML document (e.g. name="display3d") may not have a direct correspondence with the equivalent feature supported in the applet. In this case however, it would be possible to define correspondence if it exists from a configuration file.

A similar method must be used for handling data files originally declared via an <applet> element. Thus a criticism of the original implementation of this element is that data files may be declared using any arbitrary naming scheme. We therefore currently only attempt to provide an alternate mode for <applet> display by converting a limited number of known forms to the equivalent <object> syntax (e.g. chart 2).

A further pass through JTidy is used to remove some browser dependent features and also if desired to parse purely stylistic elements such as <font> to the formal CSS2 stylesheet declarations. The resulting XHTML document is therefore well formed, valid, and can carry a clearer separation of marked up content and general presentational attributes. No attempt to achieve this separation for the chemical content is made. Thus stylistic forms such as ball and stick or spacefill rendering of a molecule remains defined via specific <param> declarations rather than via CSS2 stylesheet entries.

The final stage involves using JChemTidy to recreate the original directory structure from the information provided with the URL list, and write the converted documents, together with unparsed files transcluded with the original documents into this directory. We note that JTidy will reject documents with HTML constructs that are considered too ambiguous to reliably parse, and these will be flagged for manual attention.

To validate the entire process, htdig is then used in its indexing capacity. An index is built of the document collection before and after conversion, and searches performed on both indices. Clearly, the number of hits from each index per search should match, although the relative ranking of these hits can change slightly. For example, XHTML must declare a <title> container and if none exists in the original HTML then the first occurence of the highest ranking <h> header can be taken. This difference can result in slightly different weights for the two indices, resulting in slightly differing rankings.

RESULTS AND DISCUSSION

Conformance Tests[5]. The procedure was tested using a test set of HTML-based documents.[5] Invocation of chemical data sets is handled by browsers via the XHTML <object> and <param> elements. The contents of <object> can be passed to either a Java applet or a browser plugin, and <object> itself can be made to cascade such that if the first declared method is not available via the browser, then the second is attempted, and so forth. As we noted previously, not all versions of all browsers handle this process correctly. We set out the typical behaviour of five different browsers (Table 2).[5] Only one browser (iCab) fully supports all the tests collected in Table 2. The widely used Internet Explorer version 5 supports the <object> element, but any associated <param> declarations are discarded, including data files

Table 2. Browser Support for <object> Tag[a]

Browser Test	Browser display							
	Internet Explorer 5.5/5.0		Netscape 4.75		Netscape 6.0		Opera 4.02	iCab 2.1
	Windows	MacOS	Windows	MacOS	Windows	MacOS	Windows	MacOS
Two Objects from embed (Chart 1)	Displays molecule/ignores <param> and second object	Displays molecule/ignores <param> and second object	Displays molecule/honours <param>and second object	Displays molecule/honours <param>, ignores second object	Attempts to display	Error in molecule display	Error in molecule display	Displays molecule/ honours <param> and second object
Single Object from applet	Displays molecule using Java	Ignores applet	Displays molecule using Java	Displays molecule using Java	Displays Molecule using Java	Displays Molecule using Java	Attempts to display	Displays molecule using Java
Single Object from image	Displays image	Displays image	Ignores image	Ignores image	Displays image	Displays image	Displays image	Displays image
Nested Object (Scheme 2)	Displays molecule (ignores <param>) using plugin + Java + image	Outlines all objects, displays image	Displays molecule/honours <param>, using plugin	Displays molecule/honours <param>, using plugin	Displays molecule/honours <param>, using plugin	Error in molecule display	Error locating plugin directory	Displays molecule/honours <param> using plugin
Nested Object, plugin disabled	Outlines all objects, displays cached Java[b] + image	Displays image	Displays molecule/honours <param> using Java	Displays molecule/honours <param> using Java	Displays molecule/honours <param> using Java	Displays molecule/honours <param> using Java	Displays text	Displays molecule/honours <param> using Java
Nested Object, plugin, Java disabled	Outlines all objects, displays image	Displays image	Displays text	Displays text	Displays image	Displays image	Displays text	Displays image
Nested Object, plugin, Java, Images disabled	Outlines all objects, displays image	Displays text	Displays text	Displays text	Displays text	Displays text	Displays text	Displays image

[a] Browser versions as of October 2000. [b]Only displays Java if previously cached.

Ⓦ The underlined material in this table represents links which are available in the HTML version of this article, which is available on the Internet at http://pubs.acs.org.

Table 3. Site Statistics Using JChemTidy

Sites indexed and converted	Total number of documents retrieved	Approximate time of conversion	Approximate time of indexing	Total number of chemical files retrieved	Total number of HTML files failed to be converted
http://www.chem.ox.ac.uk/mom/	553	40min	19 min	222	0
http://www.chm.bris.ac.uk/motm/	60	9 min	4 min	30	0
http://www.ch.ic.ac.uk/motm/	128	8 min	3 min	64	0

Ⓦ The underlined material in this table represents links which are available in the HTML version of this article, which is available on the Internet at http://pubs.acs.org.

passed to a Java applet by this mechanism. Rather than invoking only the first supported object, this browser also displays outlines for all the nested objects. The Netscape browser at both version 4.7 and 6.0 (beta at the time of evaluation) does correctly handle both the <object> and <param> although there were significant issues in the support of plugins such as Chime or Chem3D with the version 6.0 preview release. Opera 4.0 is a relatively new browser, which although claiming HTML4.0/XHTML conformance, was not successful in supporting the <object> element, although the deprecated <embed> is functional.

We next developed a set of tests which evaluate the mapping of <embed> attributes to <param> elements as part of the conversion to <object> (Table 2). Again we note that the iCab browser passes all these tests. In particular, we note that one commonly used attribute passed to the Chime browser plugin is the Rasmol script value. This is correctly resolved using both iCab and Netscape 4.7. Given the very large amount of material presented in this form, this was an especially important case to succeed.[10]

Performance Aspects. For successful automation, the process described above must be capable of handling a reasonably large number of documents. The htdig robot itself can handle up to e.g. 1,000,000 in around two days, depending of course on network bandwidth. Our post-processing additions invariably slow down this process. To establish the characteristics, we tested three representative sites comprising the "Molecules of the month" collections (Table 3). These reveal that approximately 15-40 documents per minute can be processed in the first pass of htdig, whilst the second pass using JTidy and JChemTidy takes about twice that time. A medium sized site with around 15,000 documents would on this basis take around 1 day. We also note that each of the three sites referenced a significant number of chemical files for transclusion using the above conversions. No documents were found on these sites that could not be processed.

Specific Issues with HTML. Whilst in general, the JTidy software handles all HTML parsing errors, several specific instances and exceptions are noteworthy.

Elements Outside the HTML Specification. By default, JTidy will not process any document containing unknown elements. An example might be e.g. the use of <author>...</author> in the header of a formal article. Such elements have to be individually specified as new-inline-tags in a master JTidy configuration file.

Processing the & Character. This character is used in HTML to declare other character entities, but is also used as part of the Chime browser plugin to define a ball&stick style for the display3D attribute. Occurances of this character are replaced by &, a form which is supported by Chime.

SMILES/SMIRKS Descriptors[11]. Characters such as [,], , +, -, (,), #, @, ., \, ; and / can be found in SMILES molecular descriptors and can be safely embedded as the value of a <param> title attribute. Further characters such as > and & are used in SMIRKS reaction descriptors. These will be converted to entity declarations by JTidy and would again have to be recognised as such by any software which makes use of them.

Rasmol Scripts. The so-called Rasmol script is often used in conjunction with the Chime plug-in to annotate a molecular structure. It can either be referenced from a file. For a small script, it is often convenient to include the script as the value of the script attribute of the <embed> element. Because Rasmol scripts were derived independently of HTML, a conflict can arise from the use of the > or < operators for atom or group selection. The quoted entity equivalent (i.e. >) needs to be used instead. A more serious problem arises out of the use of line breaks as separators between distinct Rasmol script commands, which the JTidy program will automatically remove. Currently, such scripts will have to be manually corrected.

Derived Chemical Attributes. The XHTML standard provides for more attributes of the <object> and <a> element then may be present in the <embed>, <applet>, <a> and elements. Three generic attributes in particular are considered useful.

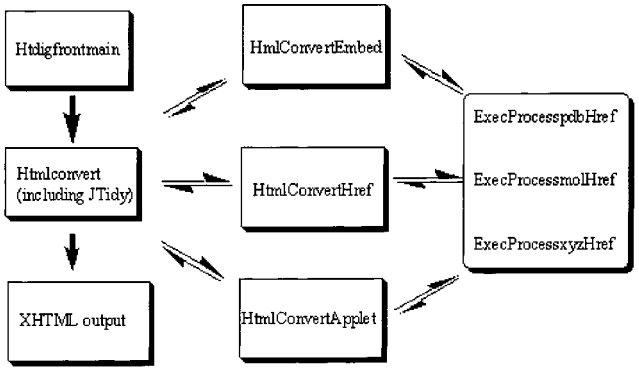
The type="MIME-type specification" is often not explicitly declared but is retrieved from the server and is used to resolve the browser plugin which will be used to display the object. This information is normally (but not invariably) derived on the server from the extension of the data file name and the server MIME configuration file and would be useful to uniquely identify the type of e.g. chemical data declared to the object. Since this information could be of use in subsequent transformations of the document collection, its inclusion as a declared <object> attribute is a valuable operation. The attribute id="unique-identifier" is used to uniquely identify an element within a document. The title attribute provides the valuable function of mapping onto browser "mouse-overs" to provide additional information about the object and serves an equally useful purpose of providing keywords for an indexing agent. These three components, which are rarely declared in original documents, would need to be optionally derived for use in XHTML. The MIME type has a unique value, whilst the other two can have arbitrary definitions, although chemical significance is clearly desirable.

We implemented one scheme for adding chemical information in this manner (Scheme 1). The elements <applet> <embed> <a> are identified through the main JChemTidy program that calls the corresponding set of java classes.

The contents of the element are then tokenised and each token is searched for the attribute that contains the data file i.e. <embed src="...">, <applet data="...">, <a href="...">

The data file suffix is retrieved and compared with the contents of the MIME configuration file derived from the first pass of htdig to see if it is a known chemical file type.

Scheme 1



If true, this information is associated with the type attribute of each element.

The value of the type attribute is then passed on to a new class along with the location path of the file, which was derived from the relative URL addresses. Because this program follows URL links, a copy of the chemical data file can be automatically created in the directory tree.

A specific Java class is defined to evaluate e.g. a molecular formula by parsing the contents of two specific data files; type="chemical/x-mdl-molfile" and type="chemical/x-xyz"

A Java class is used to pass a request to an external program (we used the Daylight toolkit) to evaluate a SMILES[11] string for these specific classes of data file. Other external processes could of course be substituted or added.[12] An alternative to calling an external process is to invoke another Java class. For example, the JME (Java Molecular Editor) provides public methods which can be called to convert a Molfile to e.g. a unique SMILES descriptor.

The derived SMILES string and molecular formula are then associated with the title attribute of the XHTML document. These values could also be used to create a unique ID for the <object>. Ideally, this ID should be globally unique, but currently no readily available mechanism exists to implement this.

The result of these operations is shown in Chart 3. We note in particular that the title attribute now provides chemically significant information which can be used in subsequent indexing operations.[13] For example, by entering the unique search string "Molecule-object", an immediate count of all transcluded molecules could be obtained. Further subdivision according to the Type attribute could be achieved, to enable identification of how many are defined in any particular data format, and more specific searches for e.g. the unique SMILES descriptors can be made. Other types of information such as spectral data, VRML models etc can be extracted.

Conversion of Other Types of HTML Content. The process described above relates to purely static HTML content. However, it is quite common to define HTML content dynamically. This can be done in two ways.

The entire document collection can be created from a request made to a closed database system, and wrapped with HTML when the user request to view the document is made. Our robot system cannot access such data. Here however, the onus is on the maintainer of the database to provide support for XHTML conforming browsers. We note in this regard however a tendency to use a technique known as "browser sniffing" in conjunction with dynamic HTML

Chart 3

Original Element	Element with chemically derived attributes
<object data="chair.mol" width="250" height="250"> <param name="name" value="chair"/> . . <object>	<object type="chemical/x-mdl-molfile" id="C3H7B1O2" title="Molecule-object, Molfile-format, SMILES: CO[BH2]OC=C" data="chair.mol" width="250" height="250"> <param name="name" value="chair"/> . . </object>
boat transition state	boat transition state
<object data="ms.jdx" width="250" height="250"> <object>	<object type="chemical/x-jcamp-dx" id="A1" title="Spectrum-object, JCAMP format, Acetophene" data="ms.jdx" width="250" height="250"> </object>

creation. This involves issuing a HTTP request to the browser for identification. Such a system is potentially expensive to maintain since it may have to cater for an increasingly large population of slightly differing browsers and their various versions on various operating systems, as illustrated by the diversity in Table 2. We would suggest that the task of creating a consistent XHTML-based document collection from a Web site which can be used for further transformation into other members of the XML family, such as CML (Chemical Markup Language), MathML or for future archival purposes is highly desirable.[1]

Another common method for dynamically creating HTML display pages is to use e.g. JavaScript to write HTML directly to the browser window. Here, the HTML syntax is defined within the syntax of JavaScript, where in practice it is essentially inaccessible. Although in theory appropriate interpreters could be included within the htdig external parser method described above, converting this in a reliably error free manner to output e.g. XHTML and <object> declarations would be both non trivial and expensive. This particular issue is in fact handled more elegantly using XML, where appropriate transforms of the XML defined content are handled by XSLT transforms rather than by JavaScript.[1] In our current system, therefore, any <script> elements are left unaltered in the HTML document, and, hence, included without modification in the resulting XHTML.

CONCLUSIONS

During a period of seven years up to 2000, it is estimated more than 1 billion documents have been published on Web servers, of which perhaps 5% are science related. We have described here a partial mechanism for largely automating the conversion of such document collections to allow the HTML used to be standardised at the current XHTML 1.0 level, and to introduce mechanisms to achieve more systematic separation of the content of the documents from the style in which it is presented to the user. The use of the <object> element as a single structured mechanism for invoking chemical display software at the browser would reduce the dependence on often very specific software combinations that authors have often assumed in the past. By ensuring the resulting transformed documents are XML compliant, further transformations using appropriate XML tools are enabled. In the next article in this series,[13] some of these transformations will be illustrated.

Acknowledgment. One of us (GVG) thanks Merck Sharp and Dohme and the EPSRC for the award of a scholarship.

REFERENCES AND NOTES

(1) Murray-Rust, P.; Rzepa, H.S. *J. Chem. Inf. Comp. Sci.*, **1999**, *39*, 928. For examples of how XML can be used in a browser environment, see Murray-Rust, P.; Rzepa, H.S.; Wright, M. and Zara, S. *Chem. Commun*, 2000, 1471.
(2) Rzepa, H.S; Murray-Rust, P.; Whitaker, B. J. *J. Chem. Inf. Comp. Sci*., **1998**, *38*, 976-982.
(3) Gkoutos, G. V.; Rzepa, H.S.; Wright, M. *Internet J. Chem.*, **2000**, *3*, article 7. See http://www.ijc.com/articles/2000v3/7/
(4) Scherpbier, A. *et al*, htdig: http://www.htdig.org/
(5) JChemTidy validation tests are available on-line via http://www.ch.ic.ac.uk/chemime/tests/object/browsers/. An XHTML validation service is available at http://validator.w3.org/check/
(6) Raggett, D. http://www.w3.org/People/Raggett/tidy/. The JTidy version is due to Quick, A. http://www3.sympatico.ca/ac.quick/
(7) Miller, M. A.; Martz, E.; Maffett, T. *Abstr. Pap. Am. Chem. S*. **1997**, *214*, 22-COMP.
(8) See Hinchliffe, A. *Elec. J. Theor. Chem.*, **1997**, *2* 215-217; Brecher, J.S, *Chimia*, **1999**, *52*, 658.
(9) Krassavine, A. *Chimia*, **1999**, *52*, 668-672.
(10) Martz, E. *FASEB J.*, **1997**, *11* A850-A850. Suppl.
(11) Weininger, D. *J. Chem. Inf. Comput. Sci.*, **1988**, *28*, 31.
(12) For examples of such external calls, see Rzepa, H.S.; Tonge, A.P. *J. Chem. Inf. Comp. Sci.*, **1998**, *38*, 1048-1053; Tonge, A. P.; Rzepa, H. S.; Yoshida, H. *J. Chem. Inf. Comp. Sci.*, **1999**, *39*, 483-490.
(13) Gkoutos, G. V.; Kenway, P. R.; Rzepa, H. S. *New. J. Chem.*, **2001**, *25*, in press.

CI000396Y