WD-html-in-xml-19981205

Reformulating HTML in XML

W3C Working Draft *5th December 1998*

This version draft:
http://www.w3.org/TR/1998/WD-html-in-xml-19981205
Latest version:
http://www.w3.org/TR/WD-html-in-xml
Also available for local browsing as a Zipped archive
Editors:
Dave Raggett <dsr@w3.org>, W3C on assignment from HP
Frank Boumphrey <bckman@ix.netcom.com>, HTML Writers Guild,
Murray Altheim <altheim@eng.sun.com>, Sun Microsystems,
Ted Wugofski <ted.wugofski@otmp.com>, Over the Moon Productions.

Abstract

This working draft reformulates HTML 4.0 as an XML application and defines the corresponding namespaces. Document profiles are introduced as a basis for interoperability guarantees for different subsets or supersets of HTML in an increasingly heterogeneous environment. Rather than restate the semantics of HTML 4.0, these are defined by the W3C Recommendation for HTML 4.0 unless otherwise overridden in this specification. Compatibility with existing HTML browsers is possible by following a small set of guidelines.

Status of this document

This working draft may be updated, replaced or rendered obsolete by other W3C documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". This is work in progress and does not imply endorsement by the W3C membership.

This document has been produced as part of the W3C HTML Activity, and is intended for early discussion in a process leading to drafting a Proposed Recommendation on reformulating HTML as an application of

XML. The goals of the HTML Working Group *(members only)* are discussed in the HTML charter *(members only)*.

Table of Contents

1 Why choose to reformulate HTML as an application of XML?

This section will explain why W3C is switching to XML for the next generation of HTML and how this transition will be achieved in a way that offers immediate benefits to content providers, although at first there will few browsers that support XML.

1.1 What is XML?

XML is an acronym for the eXtensible Markup Language, a subset of the ISO standard: Standard Generalized Markup Language (SGML). SGML is a language for describing markup languages, particularly those used in electronic document exchange, document management, and document publishing. HTML is an example of a language defined in SGML.

SGML has been around for since the middle 1980's and has remained quite stable. Much of this stability comes from the fact that the language is feature-rich and flexible. This flexibility, however, has led to a level of complexity that inhibits adoption across the great number and diversity of platforms attached to the World Wide Web.

HTML addressed this problem by specifying a limited set of tags for specifying relatively simple documents. In addition to simplifying the document structure, HTML added support for hypertext and multimedia.

Since HTML's invention, there has been rapid invention of new tags for use within HTML (as a standard) and for adapting HTML to vertical, highly specialized, markets. This has led to compatibility problems for content (documents) across different platforms which is limiting

HTML's usage in a rapidly evolving environment with an increasingly heterogeneous mix of software and platforms.

XML was introduced as a means of regaining the power and flexibility of SGML without SGML's complexity. XML is a simplified subset of SGML that retains SGML's more commonly used features and removes many of those features that are complex and costly to implement.

2.2 Modularizing HTML

Modularizing HTML is the notion of specifying well-defined sets of HTML tags that can be mixed and matched by product designers. For example, a "table module" would contain the elements and attributes necessary to support tables and a "list module" would contain the elements and attributes necessary to support lists.

The reason for modularizing HTML is to make it economically feasible for content developers to delivery content on a greater number and diversity of platforms.

Over the last couple of years, many specialized markets have begun looking to HTML as a content language. There is a great movement afoot for using HTML across increasingly diverse computing platforms. Currently there is activity to move HTML onto mobile devices (handheld computers, portable phones, etc.), television devices (digital televisions, tv-based web browsers, etc.), and appliances (fixed function devices). Each of these devices has different requirements and constraints.

Reformulating HTML in XML gives product developers the tools with which they can extend or subset HTML to address the perceived needs of their customers. However, this does not solve the needs of the content community for conformance.

Modularizing HTML provides a means for product designers to specify which elements are supported by a device using standard building blocks and standard methods for specifying which building blocks are used.

These modules serve as "points of conformance" for the content community. The content community can now target the installed base that supports a certain collection of modules, rather than worry about the installed base that supports this permutation of HTML elements or that permutation of HTML elements.

The use of standards is critical for modularized HTML to be successful on a large scale. It is not economically feasible for content developers

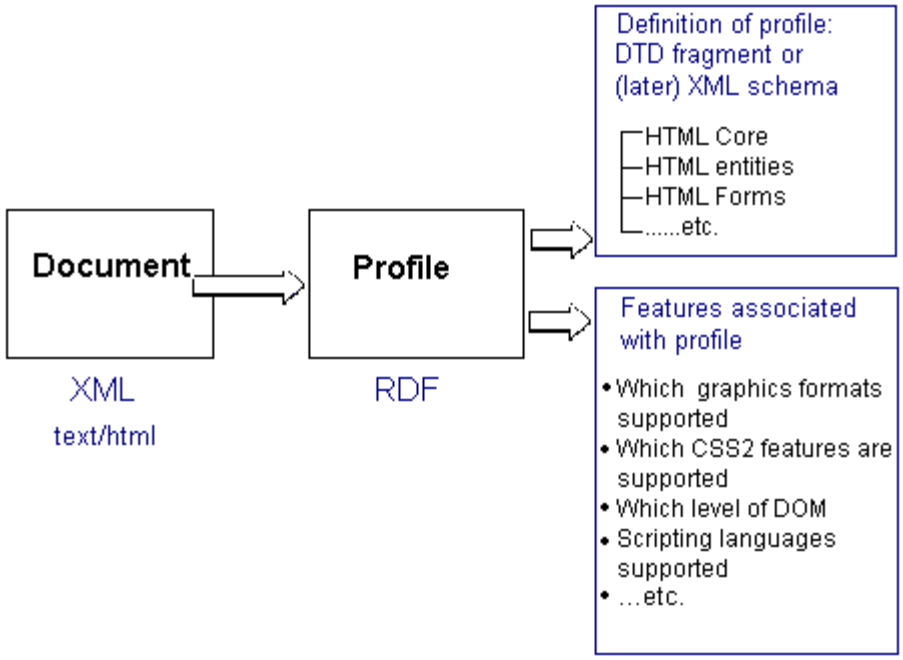
(authors) to tailor content to each and every permutation of HTML elements. By specifying a standard, either software processes can autonomously tailor content to a device, or the device can automatically load the software required to process a module.

XML provides the tools necessary for defining a modularized HTML language and for specifying how modules are defined, declared, and combined into meaningful systems.

1.3 Document Profiles

The document profile specifies the syntax and semantics of documents. Conformance to the document profile provides the basis for interoperability guarantees. The profile spells out which data formats are supported (e.g. which image formats can be used), levels of scripting and style sheet support, and so on. Further details are given below. The document profile is expressed in W3C's Resource Description Framework (RDF).

The document schema specifies the syntax of documents that conform to a document profile. This specification uses the Document Type Definition (DTD) syntax of XML 1.0 as its schema syntax, but use of alternative schema languages are possible within the profile framework. The syntax is specified in terms of which HTML modules are used as well as additional modules for other XML tag sets, e.g. for representing chemical formulae, mathematics, musical notation and vector graphics. W3C expects such modules to be developed by a range of organizations interested in sharing specialized kinds of information.



This section is intended to give a glimpse of the kinds of information that document profiles will cover. The details will be fleshed out in a separate specification. W3C is very interested in learning more about the requirements for document profiles from other groups.

Document profiles consist of assertions written in RDF that define the minimal support expected of user agents, and provide the basis for interoperability guarantees. We expect to use the RDF schema language to formalize document profiles.

The basic idea is to be able to make a number of assertions to the effect:

- A guarantee on the longevity of the document profile. The longevity specifies the date and time until which the document profile may be safely cached before checking for an update.
- A URI for the document schema that defines the document syntax. The document schema is expressed as an XML document type definition or in the XML Schema language being developed by W3C. The schema defines the syntax as a composition of tag sets (modules), as described below.
- Machine interpretable representation of semantic constraints which further constrain the space of valid documents, e.g. anchors can't be nested, label elements must contain one and only one field within their content etc. Hopefully the need to express these constraints will be lessened by further work on

XML (XML schemas and XML data).

- The list of content types data formats that a conforming user agent is required to support.
- Additional assertions covering details of these formats, e.g. which CSS features the author expects to be supported, and which libraries are needed for ECMAScript and for Java.
- Assertions describing the kinds of devices for which the document profile is a good match. See the following section on device profiles.
- A link to a human readable description of the document profile
- Additional information about who has defined the profile and when (attribution and copyright).

The document profile can be used by servers to establish whether the server has a version of a document suitable for delivery to a user agent with a given device profile. Sometimes this may involve transformation, either to a more restricted document profile or even to a device specific document format such as WML for cellphones.

1.4 Device profiles

Separate work at W3C is looking at how to use RDF to define device profiles which specify the capabilities of browsers as well as user preferences. This will allow servers to select the appropriate variant of a document to deliver to the browser, perhaps by transforming the content, based upon the match between the device profile of the browser and the document profile of the document.

1.5 Transforming mark-up to make it suitable for different devices

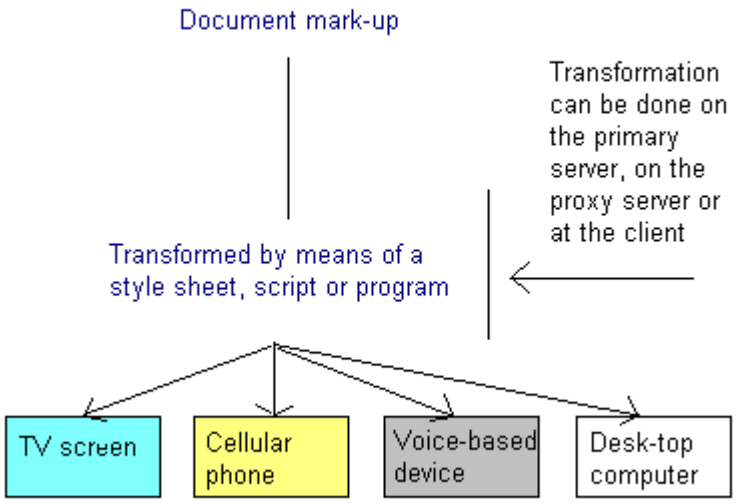
Document profiles and device profiles should greatly simplify tuning mark-up to match the needs of different devices. When the set of HTML features supported by a class of devices can be precisely anticipated, transformational software can re-purpose the markup in a simple and reliable fashion.

For example, consider a mobile phone that doesn't support scripts, style sheets or images. The server strips these out of documents before sending them to the phone, speeding page display and reducing connection charges. The server can do this by comparing the

device profile and the document profile for the documents in question, to determine what to strip out.

Transformations may be applied by the author working on the website, at a proxy server under the control of an ISP, or in the browser itself.

Work on the next generation of HTML will seek to encourage the development of authoring an related management tools that reduce the cost of creating content that may be easily re-purposed for different user agents operating on different devices.



2 Voyager

2.1 What is Voyager?

Voyager is the code name for HTML reformulated as an application of XML. Voyager specifies document profiles as XML namespaces each with their own web address (URI). The HTML Working Group will specify a set of Voyager Document Profiles for use in particular domains (such as mobile and television). For example, the "HTML Strict" Profile would contain the modules that, in general, correspond to the HTML 4 Strict DTD. The profile specifies the syntax of conforming documents in terms of a combination of syntax modules. For example, the "Table" module would contain the elements and attributes associated with HTML tables.

It is expected that non-W3C entities (companies, consortiums, other standards organizations) will specify Platforms. Platforms consist of

Voyager Profiles, platform-specific technologies, constraints, and usage requirements. For example, a digital television standards organization might specify a "DTV" Platform that contains the Television Profile, Java virtual machine, and a constrained set of allowable plug-ins.

2.2 Goals for Voyager

Here is a rather more formal statement of the goals for Voyager:

1. Voyager must be easy and straightforward to use over the Internet.
2. Voyager documents must be XML documents that can be validated according to the Voyager specification.
3. Voyager documents may be identified to applications as either "text/xml" or "text/html". Applications are under no obligation to process a document identified as "text/xml" beyond constraints specified in the XML 1.0 specification.
4. Voyager document capabilities must be extensible in a standard, well-defined way that is consistent with the Voyager specification.
5. If a real-time return channel is available, Voyager conforming user-agents must identify their capabilities to the server at the time a request for a document is made, and Voyager documents must be clearly marked with a profile in order to allow the server to deliver the appropriate document content to the client.
6. It must be possible to create Voyager documents without the use of specialized tools.
7. Voyager must maintain interoperability with associated XML-based specifications defined by W3C.
8. The Voyager specification must be written formally, concisely, and in a way that is subject to as little divergent interpretation by implementers as possible.
9. The Voyager specification will define Voyager documents in terms of coherent sets of elements as a step towards modularity and extensibility.
10. Voyager must be specified in such a way as to provide backwards compatibility with previous versions of HTML to the greatest extent that is consistent with the goals stated above.

2.3 Normative Definition of Voyager

Voyager documents may be labelled as either "***text/html***" or as "***text/xml***". The former allows user agents to interpret the content as HTML and to apply the semantics specific to HTML. By following a few simple guidelines, Voyager documents will be rendered without

problems on existing browsers. This is important as it provides for a smooth transition. HTML user agents can distinguish Voyager documents by the presence of the ***xmlns*** attribute on the ***html*** element. This attribute provides a URI designating both the namespace and the document profile, which can be used for interoperability guarantees and document validation.

Example:

```
<html xmlns="http://www.w3.org/Profiles/voyager-strict">
  <head>
    <title>Frobnostication</title>
  </head>
  <body>
    <p>Moved to <a href="http://www.frob.com/">www.frob.com</a>.</p>
  </body>
</html>
```

Voyager documents labeled as "***text/xml***" may be processed by generic XML processors. Such processors have no *a priori* knowledge of HTML, so a style sheet is needed if the document needs to be rendered. The standard XML linking mechanisms should be used as these become standardized. Guidelines for delivering Voyager documents as text/xml are given below.

2.3.1 Voyager documents must be well formed XML

As defined by the XML 1.0 specification. Note that the XML 1.0 specification constrains the behavior of user agents when encountering violations of well-formedness constraints (see section 1.2 Terminology):

> "Once a fatal error is detected, however, the processor must not continue normal processing (i.e., it must not continue to pass character data and information about the document's logical structure to the application in the normal way)."

2.3.2 The xmlns attribute must be used to designate the document profile

The xmlns attribute must be used on the html element to designate the document profile. When Voyager documents are delivered as text/html, the presence of the xmlns attribute implies that the contents of the html element are written in well formed XML and **must** be processed as per the XML 1.0 specification.

2.3.3 Tags and attributes must be in lower case

Voyager documents use lower case for all HTML tags and attributes. This is necessitated by the fact that XML is case-sensitive so that <li> and <LI> are considered to be different tags.

2.3.4 End Tags are Required

End tags are required as a consequence of Voyager being an XML application.

2.3.5 Attribute Minimization

XML doesn't support attribute minimization. As a result, attributes such as *compact* and *checked* must be written in full. This is correct:

```
<dl compact="compact">
```

But, the following is not permitted:

```
<dl compact>
```

2.3.6 Script and Style elements

In Voyager the script and style elements are declared as having #PCDATA content. This means entities such as < and & will be expanded by the XML processor to < and & respectively. You can avoid this by wrapping the script statements within a CDATA marked section, e.g.

```
<script>
  <![CDATA[
     ... unescaped script content ...
  ]]>
</script>
```

CDATA sections are recognized by the XML processor and appear as nodes in the Document Object model, see section 1.3 of the DOM level 1 specification.

2.3.7 Title and Base elements

The title element must be placed at the start of the head element's content and followed by the base element if present. These restrictions are work arounds for differences between XML and SGML. For example, this is ok:

```
<html xmlns="http://www.w3.org/Profiles/voyager-strict">
  <head>
    <title>Frobnostication</title>
    <style type="text/css">
```

```
        body {
            margin-left:10%;
            margin-right: 10%;
            font-family: sans-serif;
          }
        h1 { margin-left: -8%; }
        h2 { margin-left: -5%; }
        h3,h4,h5,h6 { margin-left: -3%; }
      </style>
    </head>
    <body>
      ...
    </body>
  </html>
```

but the following is not, since the title element doesn't appear at the start of the contents of the head element.

```
<html xmlns="http://www.w3.org/Profiles/voyager-strict">
  <head>
    <style type="text/css">
      body {
          margin-left:10%;
          margin-right: 10%;
          font-family: sans-serif;
        }
      h1 { margin-left: -8%; }
      h2 { margin-left: -5%; }
      h3,h4,h5,h6 { margin-left: -3%; }
    </style>
    <title>Frobnostication</title>
  </head>
  <body>
    ...
  </body>
</html>
```

2.3.8 Anchor elements

To allow XML processors to recognize hypertext links, the 'a' tag should be declared with the xml:link attribute.

```
<!ATTLIST    a    xml:link  CDATA    #FIXED  "simple">
```

2.3.9 Empty Elements

Because Voyager documents are written in XML, empty tags must end with **/>**.

2.4 Processing Model

Voyager documents are processed in several steps:

1. Decoding the network encoding of character data. This results in

a stream of Unicode characters.
2. Tokenization as per XML 1.0
3. Parsing as per XML 1.0. This step results in a parse tree that can be accessed and manipulated via the W3C Document Object Model (DOM).
4. Optional validation against the document schema.
5. Formatting. This step applies style sheets and other semantics specified by HTML, e.g. for forms and applets, to produce a hierarchy of formatting objects.
6. Rendering of formatting objects.

Browsers typically run these steps concurrently to allow documents to be rendered incrementally as data is received from the network.

2.5 Whitespace handling

HTML 4.0 rules for the treatment of whitespace are extended in Voyager to attribute values. In particular, to strip leading and trailing white space, and to map sequences of one or more white space characters (including line breaks) to a single inter-word space (an ASCII space character for western scripts). See section 3.3.3 of the XML 1.0 specification.

2.6 Converting existing content to Voyager

HTML Tidy is W3C sample code providing a means to automatically convert existing web content to Voyager. It can cope with a wide range of markup errors, and offers a means to help realize a smooth transition for HTML.

2.7 Voyager's relationship to other W3C initiatives

The Voyager specification is being written with the intention that it be fully interoperable with other Web projects currently underway at W3C, especially other XML-based specifications. Because of its modular, component-based architecture, Voyager is heavily dependent on the work of other W3C Working Groups in order to fulfill the goals established in the HTML Charter *(members only)*. Voyager represents only one piece of the web-publishing puzzle - obviously, other Working Groups have a role to play as well. The following section outlines some of the key related areas.

Namespaces
: This specification defines three namespaces based upon HTML4 Further work is planned to associate namespaces with document profiles as the basis for describing interoperability guarantees. At

the time of writing "Namespaces in XML" is being reviewed by W3C members as a Proposed Recommendation.

XML Linking
: Future versions of HTML will exploit the work on linking and addressing being carried out by the XML Linking working group *(members only)*.

XML Fragments
: The XML Fragments working group *(members only)* is developing ways for allowing fragments of XML documents to be delivered without the need to send the enclosing document. This capability is likely to be important for future use of HTML.

Resource Description Language (RDF)
: RDF will be used as the basis for document profiles. The RDF schema language will be used to formalize document profiles.

Document Object Model (DOM)
: This provides the basis for programmatic access to HTML documents. DOM level 1 is a W3C Recommendation.

Style Sheets (CSS and XSL)
: These provide the means to control the rendering of HTML documents as well as a means for transforming documents. Document profiles will provide the means to specify which style sheet features can be relied on for user agents conforming to the profile.

Synchronized Multimedia
: W3C work in this area will allow HTML documents to be used as part of multimedia presentations.

Vector Graphics
: The Scaleable Vector Graphics working group *(members only)* is developing an XML format for graphics for integration with HTML documents.

Mathematics
: Ensuring that mathematical content written in MathML can be seamlessly integrated with HTML.

Accessibility
: The Web Accessability Initiative is working to ensure that HTML content can be accessible to all.

Internationalization
: Making sure that HTML documents meet the needs of all languages.

Web enhanced Television Broadcasts
: Development of document profiles for use with television broadcasts incorporating Web content.

Mobile Access to the Web
: The development of document profiles for use with mobile devices.

3 Compatibility Guidelines Summary

3.1 Existing HTML Browsers

This section summarizes design guidelines for authors who wish to render Voyager documents on existing HTML browsers:

- Label the content as "***text/html***" to ensure that it is recognised as HTML and interpreted with the appropriate semantics.
- Don't use processing instructions as these are rendered on some older browsers.
- Include a space before the trailing / and > of empty elements, e.g.
, <hr /> and .
- Use external style sheets if your style sheet uses **<** or **&** or **]]>**
- Use external scripts if your script uses **<** or **&** or **]]>**
- Avoid using the <object> element as this is a common source of incompatibilities
- Avoid line breaks and multiple white space characters within attribute values. These are handled inconsistently by browsers.
- Make sure all attributes are written out in full. If you want to use the *compact* attribute, for instance on the dl element, you will need to write it as `<dl compact="compact">` and not as `<dl compact>`. This also applies to the *checked* attribute used for form fields
- Use the ***lang*** attribute when specifying the language of an element.
- Don't use elements other than those in HTML 3.2 or 4.0

3.2 Generic XML Processors

A different set of concerns relates to authoring HTML documents for use with generic XML processors that don't know the HTML specific semantics.

- Label the content as "***text/xml***" to ensure it gets handled via the XML content handler
- Use the xml processing instruction if the character encoding for

the document is other than UTF-8 or UTF-16, e.g:

```
<?xml version="1.0" encoding="EUC-JP"?>
```

- Avoid the use of character entity names other than the ones built into the XML 1.0 specification (< & and >). For instance use   rather than   for non-breaking spaces.

- Style sheets are referenced by means of a special processing instruction, for example:

```
<?xml-stylesheet href="mystyle.css" type="text/css"?>
```

- Use a document type declaration if you need to include entity declarations or would like to be able to validate the document against the DTD.

- Use XML Linking attributes with the <a> and <img> elements so that the XML processor will treat these appropriately.

- Avoid the use of elements such as frameset, frame, object, form and applet that can't be properly handled via generic XML processors

- IN XML, a URI that ends with "#name" refers to an element with an ***id*** attribute of that name, and not a name= attribute. Many existing HTML clients don't support the use of id attributes in this way, so if you want to be able to process the document on HTML clients, you may wish to supply both id and name values, e.g.

```
<a id="foo" name="foo"> ... </a>
```

- Use the ***xml:lang*** attribute when specifying the language of an element. Use both xml:lang and the HTML lang attribute if you also want to be able to process the document on HTML processors.

4 Voyager Modules

Voyager is more than the reformulation of HTML in XML. Voyager modularizes HTML into a collection of tag sets. These tag sets are building blocks which developers may use to build innovative products with World Wide Web connectivity. More importantly, these tag sets serve as design points of conformance for the content community.

The modules defined below are a first attempt at defining a

reasonable set of modules that balance the needs of product developers (for small and flexible building blocks) and for content developers (for a few building blocks with few permutations). Where greater subsetting is desired, product developers are encouraged to consider server or proxy-based transformational software that provides full module support to the content community and a smaller specification for the delivery platform.

4.1 Base Module

The Base Module specifies basic Voyager data types and content models, together with the minimal set of elements that a Voyager Profile must include. Specifically, the Base Module contains the **html, head, title, base, meta, link, body, h1-6, p, br, a, bdo, span,** and **div** elements.

4.2 Transitional Module

The Transitional Module specifies those elements that are in the HTML 4.0 Transitional Profile but excluded from the HTML 4.0 Strict Profile. Specifically, the Transitional Module contains the **basefont, font, center, s** and **u** elements. It also contains the definitions for the presentational attributes such as *border, align,* and *noshade*.

4.3 Style Module

The style module specifies the **style** element, *style* attribute, and the use of the html **link** element for linking to style sheets.

4.4 Script Module

The script module specifies the **script** and **noscript** element.

4.5 Font Module

The Font Module specifies font-related elements that are found in the HTML 4.0 Strict Profile: **tt, b, i, big,** and **small**.

4.6 Phrase Module

The Phrase Module specifies phrasal elements that provide domain specific information above and beyond the intent of the author. Specifically, the Phrase Module contains the **abbr, acronym, address, blockquote, q, cite, code, dfn, kbd, samp,** and **var** elements.

4.7 Inflection Module

The Inflection Module specifies phrasal elements that do not provide domain specific information but provide a hint of the intent of the author. Specifically, the Inflection Module contains the **em, pre, strong, sub, sup,** and **hr** elements.

4.8 Editor Module

The Editor Module specifies document editing-related elements. Specifically, the Editor Module contains the **del** and **ins** elements.

4.9 List Module

The List Module specifies list-related elements. Specifically, the List Module contains the **dl, dt, dd, ul, ol,** and **li** elements.

4.10 Forms Module

The Forms Module specifies the HTML 4.0 forms-related elements. Specifically, the Forms Module contains the **form, input, textarea, select, optgroup, option, label, button, fieldset, legend,** and **isindex** elements.

4.11 Table Module

The Table Module specifies the table-related elements. Specifically, the Table Module contains the **table, caption, col, colgroup, thead, tbody, tfoot, tr, th,** and **td** elements.

4.12 Image Module

The Image Module contains the **img** element. Some low-end systems support images but not image maps.

4.13 Image Map Module

The Image Map Module contains the **map,** and **area** elements for use with the Image Module.

4.14 Object Module

The Object Module specifies the object-related elements. Specifically, the Object Module contains the **object,** and **param**.

4.15 Applet Module

The Applet Module contains the **applet,** and **param** elements and is used when the profile supports Java applets.

4.16 Frames Module

The Frames Module specifies the HTML 4.0 frame-related elements. Specifically, the Frames Module contains the **frameset, frame, iframe,** and **noframes** elements.

5 Namespaces for strict, loose and frameset profiles

This specification defines XML namespaces for three profiles corresponding to each of the HTML 4.0 strict, transitional and frameset DTDs, reformulating them according to the XML 1.0 specification.

http://www.w3.org/Profiles/voyager-strict
: Namespace for use with documents converted from HTML 4.0 strict. These documents must conform to the Voyager strict DTD

http://www.w3.org/Profiles/voyager-loose
: Namespace for documents converted from HTML 4.0 transitional (*aka* loose), which includes a number of presentational elements and attributes. These documents must conform to the Voyager loose DTD

http://www.w3.org/Profiles/voyager-frameset
: Namespace for documents converted from HTML 4.0 frameset, which is used for documents acting as frame sets. These documents must conform to the Voyager frameset DTD

The languages defined by the Voyager document type definitions and associated rules form a normative part of this specification. It has been placed in a separate file for the convenience of those people who wish to print the specification, except for the document type definitions

6 Acknowledgements

HTML Working Group Chair
: Steven Pemberton <steven.pemberton@cwi.nl>, CWI

With help from:
: Daniel Austin, CNET
: John Burger, Mitre
: Angus Davis, Netscape
: Andrew Donho, IBM

Jon Gnaegy, Apple
Klaus Hofrichter, GMD
Philipp Hoschka, W3C
Masayasu Ishikawa, W3C
Peter King, Unwired Planet
Paula Klante, Jet Form
Kenneth Lee, Citibank
Shin'ichi Matsui, W3C/Panasonic
Shane McCarron, Open Group
Ann Navarro, HTML Writers Guild, Inc.
Zach Nies, Quark
Robert Pernett, Lotus
Patrick Schmitz, Microsoft
Robert Sutor, IBM
Chris Wilson, Microsoft
Dan Zigmond, WebTV
Warner ten Kate, Philips

7 References

HTML 4.0
HTML 4.0 Specification 18 December 1997, revised 24 April 1998. Dave Raggett, Arnaud Le Hors, Ian Jacobs. This is available at: http://www.w3.org/TR/REC-html40

XML 1.0
Extensible Markup Language (XML) 1.0 Specification 10 February 1998, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen. This is available at: http://www.w3.org/TR/REC-xml

CSS2
Cascading Style Sheets, level 2 (CSS2) Specification 12 May 1998, Bert Bos, Håkon Wium Lie, Chris Lilley, Ian Jacobs. This is available at: http://www.w3.org/TR/REC-CSS2

Associating stylesheets with XML documents
Describes a means for a stylesheet to be associated with an XML document by including one or more processing instructions with a target of xml-stylesheet in the document's prolog. This is available at: http://www.w3.org/TR/WD-xml-stylesheet

Namespaces in XML
XML namespaces provide a simple method for qualifying names used in Extensible Markup Language documents by associating them with namespaces identified by URI. At the time of writing the work is at Proposed Recommendation status and can be found at: http://www.w3.org/TR/PR-xml-names

XML Linking Language (XLink)
Specifies constructs that may be inserted into XML resources to describe links between objects. It uses XML syntax to create

structures that can describe the simple unidirectional hyperlinks of today's HTML as well as more sophisticated multi-ended and typed links. This is available at: http://www.w3.org/TR/WD-xlink

DOM Level one.

Document Object Model (DOM) Level 1 Specification, Vidur Apparao, *et al.* This is available at: http://www.w3.org/TR/REC-DOM-Level-1

URI (Web addresses, including URLs and URNs)

"RFC2396: Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, L. Masinter, August 1998. This supercedes RFC1738 and RFC1808. Available at http://info.internet.isi.edu:80/in-notes/rfc/files/rfc2396.txt.

HTML Tidy

This is a tool for detecting and correcting a wide range of markup errors prevalent in HTML. It can also be used as a tool for converting existing HTML content to be well formed XML. Tidy is being made available on the same terms as other W3C sample code, i.e. free for any purpose, and entirely at your own risk.

Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiation

Describes a method for using the Resource Description Format (RDF) to create a general, yet extensible framework for describing user preferences and device capabilities. Servers can exploit this to customize the service or content provided. This document is available at: http://www.w3.org/TR/NOTE-CCPP/