# XHTML™ 2.0

## W3C Working Draft 22 July 2004

This version:
http://www.w3.org/TR/2004/WD-xhtml2-20040722
Latest version:
http://www.w3.org/TR/xhtml2
Previous version:
http://www.w3.org/TR/2003/WD-xhtml2-20030506
Diff-marked version:
xhtml2-diff.html
Editors:
Jonny Axelsson, Opera Software
Beth Epperson, Netscape/AOL
Masayasu Ishikawa, W3C
Shane McCarron, Applied Testing and Technology
Ann Navarro, WebGeek, Inc.
Steven Pemberton, CWI (HTML Working Group Chair)

This document is also available in these non-normative formats: Single XHTML file [p.1] , PostScript version, PDF version, ZIP archive, and Gzip'd TAR archive.

Copyright ©2004 W3C® (MIT, ERCIM , Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

---

## Abstract

XHTML 2 is a general-purpose markup language designed for representing documents for a wide range of purposes across the World Wide Web. To this end it does not attempt to be all things to all people, supplying every possible markup idiom, but to supply a generally useful set of elements.

## Status of This Document

*This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.*

This document is the sixth public Working Draft of this specification. It should in no way be considered stable, and should not be normatively referenced for any purposes whatsoever. This version includes an early implementation of XHTML 2.0 in RELAX NG [RELAXNG [p.225] ], but *does not* include the implementations in DTD or XML Schema form. Those will be included in subsequent versions, once the content of this language stabilizes. This version also *does not* address the issues revolving around the use of [XLINK [p.227] ] by XHTML 2.0. Those issues are being worked independent of the evolution of this specification. Those issues should, of course, be resolved as quickly as possible, and the resolution will be reflected in a future draft. Finally, the Working Group is actively working to resolve many of the issues that have been submitted by the public, but not all of them have been addressed yet. If your particular issue has not yet been addressed, please be patient - there are many issues, and some are more complex than others.

Formal issues and error reports on this specification shall be submitted to www-html-editor@w3.org (archive). It is inappropriate to send discussion email to this address. Public discussion may take place on www-html@w3.org (archive). To subscribe send an email to www-html-request@w3.org with the word *subscribe* in the subject line.

This document has been produced by the W3C HTML Working Group (*members only*) as part of the W3C HTML Activity. The goals of the HTML Working Group are discussed in the HTML Working Group charter.

This document was produced under the 24 January 2002 CPP as amended by the W3C Patent Policy Transition Procedure. The Working Group maintains a public list of patent disclosures relevant to this document; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) with respect to this specification should disclose the information in accordance with section 6 of the W3C Patent Policy.

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than "work in progress."

# Quick Table of Contents

1. Introduction
2. Terms and Definitions . . . . . 15
3. Conformance Definition
4. The XHTML 2.0 Document Type . . . . . 23
5. Module Definition Conventions . . . . . 25
6. XHTML Attribute Collections . . . . . 29
7. XHTML Document Module . . . . . 31
8. XHTML Structural Module . . . . . 35
9. XHTML Text Module . . . . . 45
10. XHTML Hypertext Module . . . . . 53
11. XHTML List Module . . . . . 55

12. XHTML Core Attributes Module . . . . . 61
13. XHTML Hypertext Attributes Module . . . . . 63
14. XHTML I18N Attribute Module . . . . . 69
15. XHTML Bi-directional Text Attribute Module . . . . . 71
16. XHTML Edit Attributes Module . . . . . 73
17. XHTML Embedding Attributes Module . . . . . 75
18. XHTML Image Map Attributes Module . . . . . 77
19. XHTML Metainformation Attributes Module . . . . . 81
20. XHTML Metainformation Module . . . . . 87
21. XHTML Object Module . . . . . 97
22. Ruby Module . . . . . 109
23. XHTML Scripting Module . . . . . 111
24. XHTML Style Attribute Module . . . . . 115
25. XHTML Style Sheet Module . . . . . 117
26. XHTML Tables Module . . . . . 121
27. XForms Module . . . . . 145
28. XML Events Module . . . . . 147
A. Changes from XHTML 1.1 . . . . . 149
B. XHTML 2.0 RELAX NG Definition . . . . . 151
C. XHTML RELAX NG Module Implementations . . . . . 155
D. XHTML 2.0 Schema . . . . . 205
E. XHTML Schema Module Implementations . . . . . 207
F. XHTML 2.0 Document Type Definition . . . . . 209
G. XHTML DTD Module Implementations . . . . . 211
H. Stylesheet for XHTML 2 . . . . . 213
I. List of Elements . . . . . 219
J. List of Attributes . . . . . 221
K. Cross-reference Index . . . . . 223
L. References . . . . . 225
M. Acknowledgements . . . . . 229

# List of Issues

1. [PR #7336] Identifying XHTML version in ansence of DTDs
2. [PR #7442] xml:id
3. Need a normative definition for the version attribute
4. [PR #6554] [XHTML2] Comments on WD 5: make white space preservation part of the standard values for the code and kbd elements?
5. [PR #7642] Efficient param's
6. [PR #634] Fw: Character entities in HTML and MathML
7. [PR #670] Entity management: do we still need it?
8. [PR #671] Character entities: do we still need them?

# Full Table of Contents

1. Introduction
    1.1. What is XHTML 2?
        1.1.1. Design Aims
        1.1.2. Backwards compatibility
        1.1.3. XHTML 2 and Presentation
        1.1.4. XHTML 2 and Linking
    1.2. Major Differences with XHTML 1
    1.3. What are the XHTML 2 Modules?
2. Terms and Definitions . . . . . 15
3. Conformance Definition
    3.1. Document Conformance
        3.1.1. Strictly Conforming Documents
    3.2. XHTML Family User Agent Conformance
4. The XHTML 2.0 Document Type . . . . . 23
    4.1. Issues . . . . . 24
5. Module Definition Conventions . . . . . 25
    5.1. Module Structure . . . . . 25
    5.2. Abstract Module Definitions . . . . . 25
    5.3. Syntactic Conventions . . . . . 25
    5.4. Content Types . . . . . 26
    5.5. Attribute Types . . . . . 27
6. XHTML Attribute Collections . . . . . 29
    6.1. Issues . . . . . 30
7. XHTML Document Module . . . . . 31
    7.1. The html element . . . . . 31
    7.2. The head element . . . . . 32
    7.3. The title element . . . . . 32
    7.4. The body element . . . . . 33
8. XHTML Structural Module . . . . . 35
    8.1. The address element . . . . . 37
    8.2. The blockcode element . . . . . 37
    8.3. The blockquote element . . . . . 38
    8.4. The div element . . . . . 38
    8.5. The heading elements . . . . . 39
    8.6. The p element . . . . . 40
    8.7. The pre element . . . . . 41
    8.8. The section element . . . . . 42
    8.9. The separator element . . . . . 42
9. XHTML Text Module . . . . . 45
    9.1. The abbr element . . . . . 46
    9.2. The cite element . . . . . 46

9.3. The code element . . . . . 47
9.4. The dfn element . . . . . 47
9.5. The em element . . . . . 47
9.6. The kbd element . . . . . 48
9.7. The I element . . . . . 48
9.8. The quote element . . . . . 49
9.9. The samp element . . . . . 49
9.10. The span element . . . . . 50
9.11. The strong element . . . . . 50
9.12. The sub element . . . . . 50
9.13. The sup element . . . . . 51
9.14. The var element . . . . . 51
9.15. Issues . . . . . 52
10. XHTML Hypertext Module . . . . . 53
10.1. The a element . . . . . 53
11. XHTML List Module . . . . . 55
11.1. Definition lists: the dl, di, dt, and dd elements . . . . . 57
11.2. The nl element . . . . . 58
11.3. The ol, and ul elements . . . . . 58
11.4. The li element . . . . . 59
11.5. The label element . . . . . 59
12. XHTML Core Attributes Module . . . . . 61
12.1. Core Attribute Collection . . . . . 61
13. XHTML Hypertext Attributes Module . . . . . 63
13.1. Hypertext Attribute Collection . . . . . 63
14. XHTML I18N Attribute Module . . . . . 69
14.1. I18N Attribute Collection . . . . . 69
15. XHTML Bi-directional Text Attribute Module . . . . . 7.1
15.1. Bi-directional Text Collection . . . . . 7.1
15.1.1. Inheritance of text direction information . . . . . 7.1
15.1.2. The effect of style sheets on bidirectionality . . . . . 7.2
15.1.3. 15.1.1. Inheritance of text direction information . . . . . 7.1
15.1.4. 15.1.2. The effect of style sheets on bidirectionality . . . . . 7.2
16. XHTML Edit Attributes Module . . . . . 7.3
16.1. Edit Collection . . . . . 7.3
17. XHTML Embedding Attributes Module . . . . . 7.5
17.1. Embedding Attribute Collection . . . . . 7.5
18. XHTML Image Map Attributes Module . . . . . 7.7
18.1. Image Map Attribute Collection . . . . . 7.7
19. XHTML Metainformation Attributes Module . . . . . 81
19.1. Metadata Attribute Collection . . . . . 81
19.2. Properties . . . . . 84
19.3. Mapping Lexical Content . . . . . 85

20. XHTML Metainformation Module . . . . . 87
20.1. Literals and Resources . . . . . 88
20.2. Document Properties . . . . . 88
20.2.1. Literals . . . . . 88
20.2.2. Resources . . . . . 89
20.2.3. Making Use of External Lists of Properties . . . . . 90
20.3. Properties of Other Resources . . . . . 90
20.3.1. Resources Within the Containing Document . . . . . 90
20.3.2. External Resources . . . . . 91
20.4. Chaining Metadata . . . . . 91
20.5. The link element . . . . . 93
20.5.1. Forward and reverse links . . . . . 93
20.5.2. Links and search engines . . . . . 94
20.6. The meta element . . . . . 95
20.6.1. meta and search engines . . . . . 96
21. XHTML Object Module . . . . . 97
21.1. The object element . . . . . 97
21.1.1. Defining terminology . . . . . 98
21.1.2. Basic Information for Object Handlers . . . . . 98
21.1.3. Rules for processing objects . . . . . 99
21.2. The param element . . . . . 103
21.2.1. Referencing object data . . . . . 105
21.2.2. Object element declarations and instantiations . . . . . 106
21.3. The standby element . . . . . 107
21.4. Issues . . . . . 107
22. Ruby Module . . . . . 109
23. XHTML Scripting Module . . . . . 111
23.1. The script element . . . . . 111
23.1.1. Rules for processing scripts . . . . . 112
23.1.2. Specifying the scripting language . . . . . 112
23.1.3. Declaration of a scripting language . . . . . 112
23.1.4. Dynamic modification of documents . . . . . 113
24. XHTML Style Attribute Module . . . . . 115
24.1. Style Attribute Collection . . . . . 115
25. XHTML Style Sheet Module . . . . . 117
25.1. The style element . . . . . 117
25.1.1. External style sheets . . . . . 118
25.1.2. Preferred and alternate style sheets . . . . . 118
25.1.3. Specifying external style sheets . . . . . 119
26. XHTML Tables Module . . . . . 121
26.1. The caption element . . . . . 122
26.1.1. Caption Clipping . . . . . 122
26.2. The col and colgroup elements . . . . . 123

26.2.1. Calculating the number of columns in a table . . . . . 124
26.3. The summary element . . . . . 125
26.4. The table element . . . . . 125
26.4.1. Visual Rendering . . . . . 126
26.4.2. Table directionality . . . . . 127
26.4.3. Table rendering by non-visual user agents . . . . . 127
26.4.4. Sample table . . . . . 135
26.5. The tbody element . . . . . 136
26.6. The td and th elements . . . . . 137
26.6.1. Cells that span several rows or columns . . . . . 139
26.7. The thead and tfoot elements . . . . . 142
26.8. The tr element . . . . . 143
27. XForms Module . . . . . 145
27.1. Forms Attribute Collection . . . . . 145
28. XML Events Module . . . . . 147
28.1. Events . . . . . 147
A. Changes from XHTML 1.1 . . . . . 149
B. XHTML 2.0 RELAX NG Definition . . . . . 151

B.0.1. RELAX NG XHTML 2.0 Driver . . . . . 151
C. XHTML RELAX NG Module Implementations . . . . . 155
C.1. XHTML Module Implementations . . . . . 155
C.1.1. Attribute Collections . . . . . 155
C.1.2. Document . . . . . 156
C.1.3. Structural . . . . . 158
C.1.4. Text . . . . . 163
C.1.5. Hypertext . . . . . 168
C.1.6. List . . . . . 168
C.1.7. Core Attributes . . . . . 172
C.1.8. Hypertext Attributes . . . . . 173
C.1.9. I18N Attribute . . . . . 175
C.1.10. Bi-directional Text Attribute . . . . . 176
C.1.11. Edit Attributes . . . . . 176
C.1.12. Embedding Attributes . . . . . 177
C.1.13. Image Map Attributes . . . . . 178
C.1.14. Metainformation Attributes . . . . . 179
C.1.15. Metainformation . . . . . 181
C.1.16. Object . . . . . 182
C.1.17. Scripting . . . . . 184
C.1.18. Style Attribute . . . . . 185
C.1.19. Style Sheet . . . . . 185
C.1.20. Tables . . . . . 186
C.2. XHTML RELAX NG Support Modules . . . . . 190

C.2.1. Datatypes . . . . . 191
C.2.2. Events . . . . . 194
C.2.3. Param . . . . . 195
C.2.4. Caption . . . . . 196
C.3. RELAX NG External Modules . . . . . 197
C.3.1. Ruby . . . . . 197
C.3.2. Ruby Driver for Full Ruby Markup . . . . . 200
C.3.3. XML Events . . . . . 200
C.3.4. XML Schema instance . . . . . 202
D. XHTML 2.0 Schema . . . . . 205
E. XHTML Schema Module Implementations . . . . . 207
F. XHTML 2.0 Document Type Definition . . . . . 209
F.1. Issues . . . . . 209
G. XHTML DTD Module Implementations . . . . . 211
G.1. XHTML Modular Framework . . . . . 211
G.2. XHTML Module Implementations . . . . . 211
G.3. XHTML DTD Support Modules . . . . . 211
H. Stylesheet for XHTML 2 . . . . . 213
I. List of Elements . . . . . 219
J. List of Attributes . . . . . 221
K. Cross-reference Index . . . . . 223
L. References . . . . . 225
L.1. Normative References . . . . . 225
L.2. Informative References . . . . . 227
M. Acknowledgements . . . . . 229

# 1. Introduction

This section is *informative*.

## 1.1. What is XHTML 2?

XHTML 2 is a general purpose markup language designed for representing documents for a wide range of purposes across the World Wide Web. To this end it does not attempt to be all things to all people, supplying every possible markup idiom, but to supply a generally useful set of elements, with the possibility of extension using the `class` attribute on the `span` and `div` elements in combination with stylesheets, and attributes from the metadata attributes collection.

### 1.1.1. Design Aims

In designing XHTML 2, a number of design aims were kept in mind to help direct the design. These included:

- As generic XML as possible: if a facility exists in XML, try to use that rather than duplicating it.
- Less presentation, more structure: use stylesheets for defining presentation.
- More usability: within the constraints of XML, try to make the language easy to write, and make the resulting documents easy to use.
- More accessibility: some call it 'designing for our future selves' – the design should be as inclusive as possible.
- Better internationalization: since it is a World Wide Web.
- More device independence: new devices coming online, such as telephones, PDAs, tablets, televisions and so on mean that it is imperative to have a design that allows you to author once and render in different ways on different devices, rather than authoring new versions of the document for each type of device.
- Less scripting: achieving functionality through scripting is difficult for the author and restricts the type of user agent you can use to view the document. We have tried to identify current typical usage, and include those usages in markup.

### 1.1.2. Backwards compatibility

Because earlier versions of HTML were special-purpose languages, it was necessary to ensure a level of backwards compatibility with new versions so that new documents would still be usable in older browsers. However, thanks to XML and stylesheets, such strict element-wise backwards compatibility is no longer necessary, since an XML-based browser, of which at the time of writing means more than 95% of browsers in use, can process new markup languages without having to be updated. Much of XHTML 2 works already in existing browsers; much, but not all: just as when forms and tables were added to HTML, and people had to wait for new version of browsers before being able to use the new facilities, some parts of XHTML 2, such as XForms and XML Events, still require user agents that understand that functionality.

### 1.1.3. XHTML 2 and Presentation

The very first version of HTML was designed to represent the structure of a document, not its presentation. Even though presentation-oriented elements were later added to the language by browser manufacturers, HTML is at heart a document structuring language. XHTML 2 takes HTML back to these roots, by removing all presentation elements, and subordinating all presentation to stylesheets. This gives greater flexibility, greater accessibility, more device independence, and more powerful presentation possibilities, since stylesheets can do more than the presentational elements of HTML ever did.

### 1.1.4. XHTML 2 and Linking

The original versions of HTML relied upon built-in knowledge on the part of User Agents and other document processors. While much of this knowledge had to do with presentation (see above), the bulk of the remainder had to do with the relationships between documents — so called "linking".

A variety of W3C and other efforts, most notably [XLINK [p.227] ], attempted to create a grammar for defining the characteristings of linking. Unfortunately, these grammars all fall short of the requirements of XHTML. While the community continues in its efforts to create a comprehensive grammar, the HTML Working Group has defined a grammar that is sufficient for its needs. The linking constructs in this document are described in terms of this grammar, [HLINK [p.227] ].

Note that by describing the linking characteristics using [HLINK [p.227] ], this document is NOT requiring that implementations support the HLINK constructs or grammar, just as it does not require implementations support [CSS2 [p.225] ] even though some presentation aspects of this document are defined using CSS. Instead, the document relies upon this grammar as a formalism for ensuring that the characteristics of inter-document "links" can be described in a consistent manner, and thus implemented consistently.

## 1.2. Major Differences with XHTML 1

XHTML 2 is designed to be recognizable to the HTML and XHTML 1 author, while correcting errors and insufficiencies identified in earlier versions of the HTML family, and taking the opportunity to make improvements.

The most visible changes are the following:

- More structuring possibilities:
  - Sections and headings: in previous versions of HTML a document's structure had to be inferred from the various levels of headings in the document; this was particularly a problem when authors misused the heading elements for visual effects. XHTML 2 lets you explicit markup the document structure with the section [p.42] element, and its related header element h [p.39] .
  - Separators: in previous versions of HTML, the `hr` element was used to separate

sections of a text from each other. In retrospect, the name `hr` (for *horizontal rule*) was badly chosen, because an `hr` was neither necessarily horizontal (in vertical text it was vertical), nor necessarily a rule (books often use other typographical methods to represent separators, such as a line of three asterisks, and stylesheets can be used to give you this freedom). In order to emphasize its structuring nature, and to make it clearer that it has no essential directionality, `hr` has been renamed separator [p.42] .

- Line breaks: in previous versions of HTML, the `br` element was used to add micro-structure to text, essentially breaking a piece of text into several 'lines'. This micro-structure is now made explicit in XHTML 2 with the l [p.48] element, which encloses the text to be broken. Amongst other advantages, this gives more presentational opportunities, such as the ability to automatically number lines, or to color alternate lines differently.
- Paragraph structure: in earlier versions of HTML, a p [p.40] element could only contain simple text. It has been improved to bring it closer to what people perceive as a paragraph, now being allowed to include such things as lists and tables.

- Navigation lists: Part of the design of XHTML 2 has been to observe existing use of HTML and identify what is perceived as missing, for instance by use of scripting to achieve ends not supported directly in HTML. One obvious component of very many HTML pages is the 'navigation list', consisting of a collection of links to other parts of the site, presented vertically, horizontally, or as a drop-down menu. To support this type of usage, XHTML 2 introduces the navigation list element nl [p.58] , which codifies such parts of documents, and allows different presentational idioms to be applied. An additional advantage is for assistive technologies, that can allow the user to skip such elements.

- Images: the HTML `img` element has many shortcomings: it only allows you to specify a single resource for an image, rather than offering the fallback opportunities of the object [p.97] element; the only fallback option it gives is the `alt` text, which can only be plain text, and not marked up in any way; the `longdesc` attribute which allows you to provide a long description of the image is difficult to author and seldom supported.

  XHTML 2 takes a completely different approach, by taking the premise that all images have a long description and treating the image and the text as equivalents. In XHTML 2 *any* element may have a src [p.75] attribute, which specifies a resource (such as an image) to load instead of the element. If the resource is unavailable (because of network failure, because it is of a type that the browser can't handle, or because images have been turned off) then the element is used instead. Essentially the `longdesc` has been moved into the document, though this behavior also mimicks the fallback behavior of the object [p.97] element. (To achieve the tooltip effect that some browsers gave with the `alt` attribute, as in HTML4 you use the title [p.62] attribute).

- Type: in HTML4, the type [p.76] attribute when referring to an external resource was purely a hint to the user agent. In XHTML 2 it is no longer a hint, but specifies the type(s) of resource the user agent must accept.
- Tables: the content model of tables has been cleaned up and simplified, while still allowing the same functionality.
- Bi-directional text: rather than use an explicit element to describe bi-directional override, new values have been added to the dir [p.71] attribute that allow bi-directional override on

any element.

- Edit: rather than use explicit `ins` and `del` elements to mark changes in a document, an attribute edit [p.73] may be used on any element for the same purpose.
- Linking: In HTML 3, only a [p.53] elements could be the source and target of hyperlinks. In HTML 4 and XHTML 1, any element could be the target of a hyperlink, but still only a [p.53] elements could be the source. In XHTML 2 any element can now also be the source of a hyperlink, since href [p.63] and its associated attributes may now appear on any element. So for instance, instead of `<li><a href="home.html">Home</a></li>`, you can now write `<li href="home.html">Home</li>`. Even though this means that the a [p.53] element is now strictly-speaking unnecessary, it has been retained.
- Metadata: the meta [p.95] and link [p.93] elements have been generalized, and their relationship to RDF [RDF [p.227] ] described. Furthermore, the attributes on these two elements can be more generally applied across the language.

- Events: event handling in HTML was restricted in several ways: since the event names were hard-wired in the language (such as `onclick`), the only way to add new events was to change the language; many of the events (such as `click`) were device-specific, rather than referring to the intent of the event (such as activating a link); you could only handle an event in one scripting language — it was not possible to supply event handlers in the same document for several different scripting languages.

  XHTML 2 uses XML Events [XMLEVENTS [p.226] ] to specify event handling, giving greater freedom in the ability to handle events.

- Forms: HTML Forms were introduced in 1993, before the advent of the e-commerce revolution. Now with more than a decade of experience with their use, they have been thoroughly overhauled and updated to meet the needs of modern forms, in the shape of XForms [XFORMS [p.226] ], which are an integral part of XHTML 2.
- Ownership where due: since HTML 4 was a standalone application, it defined many things which no longer need to be defined now that it is an XML application. For instance the definitions of whitespace are given by XML for input, and CSS for output; similarly, the definition of values of the media [p.117] attribute are relegated to the relevant stylesheet language.

## 1.3. What are the XHTML 2 Modules?

XHTML 2 is a member of the XHTML Family of markup languages. It is an XHTML Host Language as defined in XHTML Modularization. As such, it is made up of a set of XHTML Modules that together describe the elements and attributes of the language, and their content model. XHTML 2 updates many of the modules defined in XHTML Modularization 1.0 [XHTMLMOD [p.226] ], and includes the updated versions of all those modules and their semantics. XHTML 2 also uses modules from Ruby [RUBY [p.226] ], XML Events [XMLEVENTS [p.226] ], and XForms [XFORMS [p.226] ].

The modules defined in this specification are largely extensions of the modules defined in XHTML Modularization 1.0. This specification also defines the semantics of the modules it includes. So, that means that unlike earlier versions of XHTML that relied upon the semantics defined in HTML 4 [HTML4 [p.227] ], all of the semantics for XHTML 2 are defined either in this specification or in the specifications that it normatively references.

Even though the XHTML 2 modules are defined in this specification, they are available for use in other XHTML family markup languages. Over time, it is possible that the modules defined in this specification will migrate into the XHTML Modularization specification.

# 2. Terms and Definitions

This section is *normative*.

While some terms are defined in place, the following definitions are used throughout this document. Familiarity with the W3C XML 1.0 Recommendation [XML [p.226] ] is highly recommended.

abstract module
: a unit of document type specification corresponding to a distinct type of content, corresponding to a markup construct reflecting this distinct type.

content model
: the declared markup structure allowed within instances of an element type. XML 1.0 differentiates two types: elements containing only element content (no character data) and mixed content (elements that may contain character data optionally interspersed with child elements). The latter are characterized by a content specification beginning with the "#PCDATA" string (denoting character data).

deprecated
: a feature marked as deprecated is in the process of being removed from this recommendation. Portable applications should not use features marked as deprecated.

document model
: the effective structure and constraints of a given document type. The document model constitutes the abstract representation of the physical or semantic structures of a class of documents.

document type
: a class of documents sharing a common abstract structure. The ISO 8879 [SGML [p.226] ] definition is as follows: "a class of documents having similar characteristics; for example, journal, article, technical manual, or memo. (4.102)"

document type definition (DTD)
: a formal, machine-readable expression of the XML structure and syntax rules to which a document instance of a specific document type must conform; the schema type used in XML 1.0 to validate conformance of a document instance to its declared document type. The same markup model may be expressed by a variety of DTDs.

driver
: a generally short file used to declare and instantiate the modules of a DTD. A good rule of thumb is that a DTD driver contains no markup declarations that comprise any part of the document model itself.

element
: an instance of an element type.

element type
: the definition of an element, that is, a container for a distinct semantic class of document content.

entity
: an entity is a logical or physical storage unit containing document content. Entities may be composed of parseable XML markup or character data, or unparsed (i.e., non-XML, possibly non-textual) content. Entity content may be either defined entirely within the

document entity ("internal entities") or external to the document entity ("external entities"). In parsed entities, the replacement text may include references to other entities.

entity reference
: a mnemonic string used as a reference to the content of a declared entity (e.g., "&" for "&", "<" for "<", "©" for "©".)

facilities
: Facilities are elements, attributes, and the semantics associated with those elements and attributes.

generic identifier
: the name identifying the element type of an element. Also, element type name.

hybrid document
: A hybrid document is a document that uses more than one XML namespace. Hybrid documents may be defined as documents that contain elements or attributes from hybrid document types.

instantiate
: to replace an entity reference with an instance of its declared content.

markup declaration
: a syntactical construct within a DTD declaring an entity or defining a markup structure. Within XML DTDs, there are four specific types: entity declaration defines the binding between a mnemonic symbol and its replacement content; element declaration constrains which element types may occur as descendants within an element (see also content model); attribute definition list declaration defines the set of attributes for a given element type, and may also establish type constraints and default values; notation declaration defines the binding between a notation name and an external identifier referencing the format of an unparsed entity.

markup model
: the markup vocabulary (i.e., the gamut of element and attribute names, notations, etc.) and grammar (i.e., the prescribed use of that vocabulary) as defined by a document type definition (i.e., a schema) The markup model is the concrete representation in markup syntax of the document model, and may be defined with varying levels of strict conformity. The same document model may be expressed by a variety of markup models.

module
: an abstract unit within a document model expressed as a DTD fragment, used to consolidate markup declarations to increase the flexibility, modifiability, reuse and understanding of specific logical or semantic structures.

modularization
: an implementation of a modularization model; the process of composing or de-composing a DTD by dividing its markup declarations into units or groups to support specific goals. Modules may or may not exist as separate file entities (i.e., the physical and logical structures of a DTD may mirror each other, but there is no such requirement).

modularization model
: the abstract design of the document type definition (DTD) in support of the modularization goals, such as reuse, extensibility, expressiveness, ease of documentation, code size, consistency and intuitiveness of use. It is important to note that a modularization model is only orthogonally related to the document model it describes, so that two very different modularization models may describe the same document type.

parameter entity
: an entity whose scope of use is within the document prolog (i.e., the external subset/DTD or internal subset). Parameter entities are disallowed within the document instance.

parent document type
: A parent document type of a hybrid document is the document type of the root element.

tag
: descriptive markup delimiting the start and end (including its generic identifier and any attributes) of an element.

# 3. Conformance Definition

This section is *normative*.

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119 [p.225] ].

## 3.1. Document Conformance

In this document, the use of the word 'schema' refers to any definition of the syntax of XHTML 2, regardless of the definition language used.

### 3.1.1. Strictly Conforming Documents

A strictly conforming XHTML 2.0 document is a document that requires only the facilities described as mandatory in this specification. Such a document must meet all the following criteria:

1. The document must conform to the constraints expressed in the schemas in Appendix B - XHTML 2.0 RELAX NG Definition [p.151] , Appendix D - XHTML 2.0 Schema [p.205] and Appendix F - XHTML 2.0 Document Type Definition [p.209] .

2. The local part of the root element of the document must be `html`.

3. The start tag of the root element of the document must explicitly contain an `xmlns` declaration for the XHTML 2.0 namespace [XMLNS [p.226] ]. The namespace URI for XHTML 2.0 is defined to be `http://www.w3.org/2002/06/xhtml2`.

   The start tag must also contain an xsi:schemaLocation [p.32] attribute. The schema location for XHTML 2.0 is defined to be `TBD`.

   An example root element might look like:

   ```
   <html xmlns="http://www.w3.org/2002/06/xhtml2"
         xml:lang="en"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2002/06/xhtml2 TBD"
   >
   ```

4. There must be a DOCTYPE declaration in the document prior to the root element. If present, the public identifier included in the DOCTYPE declaration must reference the DTD found in Appendix F [p.211] using its Public Identifier. The system identifier may be modified appropriately.

```
<!DOCTYPE
 html PUBLIC "-//W3C//DTD XHTML 2.0//EN"
 "TBD">
```

Here is an example of an XHTML 2.0 document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 2.0//EN"
    "TBD">
<html xmlns="http://www.w3.org/2002/06/xhtml2"
          xml:lang="en"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2002/06/xhtml2 TBD"
>
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>Moved to <a href="http://vlib.org/">vlib.org</a>.</p>
  </body>
</html>
```

Note that in this example, the XML declaration is included. An XML declaration like the one above is not required in all XML documents. XHTML document authors are strongly encouraged to use XML declarations in all their documents. Such a declaration is required when the character encoding of the document is other than the default UTF-8 or UTF-16 and no encoding was determined by a higher-level protocol.

## 3.2. XHTML Family User Agent Conformance

A conforming user agent must meet all of the following criteria:

1. The user agent must parse and evaluate an XHTML 2 document for well-formedness. If the user agent claims to be a validating user agent, it must also validate documents against a referenced schema according to [XML [p.226] ].
2. When the user agent claims to support facilities defined within this specification or required by this specification through normative reference, it must do so in ways consistent with the facilities' definition.
3. When a user agent processes an XHTML 2 document as generic XML, it shall only recognize attributes of type `ID` (e.g., the `id` attribute on most XHTML 2 elements) as fragment identifiers.
4. If a user agent encounters an element it does not recognize, it must continue to process the content of that element.
5. If a user agent encounters an attribute it does not recognize, it must ignore the entire attribute specification (i.e., the attribute and its value).
6. If a user agent encounters an attribute value it doesn't recognize, it must use the default attribute value.
7. If it encounters an entity reference (other than one of the predefined entities) for which the user agent has processed no declaration (which could happen if the declaration is in the external subset which the user agent hasn't read), the entity reference should be rendered

as the characters (starting with the ampersand and ending with the semi-colon) that make up the entity reference.

8. When rendering content, user agents that encounter characters or character entity references that are recognized but not renderable should display the document in such a way that it is obvious to the user that normal rendering has not taken place.

9. White space is handled according to the rules of [XML [p.226] ]. All elements preserve whitespace.

   The user agent must use the definition from CSS for processing white space characters [CSS3-TEXT [p.225] ].

10. In the absence of a style-sheet, including user agents that do not process stylesheets, visual presentation should be as if the user agent used the CSS stylesheet specified in Appendix H.

# 4. The XHTML 2.0 Document Type

This section is *normative*.

The XHTML 2.0 document type is a fully functional document type with rich semantics. It is a collection of XHTML-conforming modules (most of which are defined in this specification). The Modules and their elements are listed here for information purposes, but the definitions in their base documents should be considered authoritative. In the on-line version of this document, the module names in the list below link into the definitions of the modules within the relevant version of the authoritative specification.

Document Module [p.31]
`body`, `head`, `html`, `title`
Structural Module [p.35]
`address`, `blockcode`, `blockquote`, `div`, `h`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `p`, `pre`, `section`, `separator`
Text Module [p.45]
`abbr`, `cite`, `code`, `dfn`, `em`, `kbd`, `l`, `quote`, `samp`, `span`, `strong`, `sub`, `sup`, `var`
Hypertext Module [p.53]
`a`
List Module [p.55]
`dl`, `dt`, `dd`, `label`, `nl`, `ol`, `ul`, `li`
Core Attributes Module [p.61]
`class`, `id`, and `title` attributes
Hypertext Attributes Module [p.63]
`href`, `hreftype`, `cite`, `target`, `rel`, `rev`, `access`, `nextfocus`, `prevfocus`, and `xml:base` attributes
Internationalization Attribute Module [p.69]
`xml:lang` attribute
Bi-directional Text Module [p.71]
`dir` attribute
Edit Attributes Module [p.73]
`edit`, and `datetime` attributes
Embedding Attributes Module [p.75]
`src`, and `type` attributes
Image Map Attributes Module [p.77]
`usemap`, `ismap`, `shape`, and `coords` attributes
Metainformation Attributes Module [p.87]
`about`, `content`, `datatype`, `property`, `rel`, `resource`, `restype`, and `rev` attributes
Metainformation Module [p.87]
`meta`, `link`
Object Module [p.97]
`object`, `param`, `standby`

Scripting Module [p.111]
`noscript, script`
Style Attribute Module [p.115]
`style` attribute
Stylesheet Module [p.117]
`style` element
Tables Module [p.121]
`caption, col, colgroup, summary, table, tbody, td, tfoot, th, thead, tr`

XHTML 2.0 also uses the following externally defined modules:

Ruby Annotation Module [RUBY [p.226] ]
`ruby, rbc, rtc, rb, rt, rp`
XForms Module [XFORMS [p.226] ]
`action, alert, bind, case, choices, copy, delete, dispatch, extension, filename, group, help, hint, input, insert, instance, item, itemset, label, load, mediatype, message, model, output, range, rebuild, recalculate, refresh, repeat, reset, revalidate, secret, select, select1, send, setfocus, setindex, setvalue, submission, submit, switch, textarea, toggle, trigger, upload`, and `value` elements, and `repeat-model, repeat-bind, repeat-nodeset, repeat-startindex`, and `repeat-number` attributes
XML Events Module [XMLEVENTS [p.226] ]
`listener` element, and `defaultAction, event, handler, objserver, phase, propagate`, and `target` attributes in the [XMLEVENTS [p.226] ] namespace

An implementation of this document type as a RELAX NG grammar is defined in Appendix B [p.151] , as an XML Schema in Appendix D [p.205] , and as a DTD in Appendix F [p.209] .

## 4.1. Issues

Identifying XHTML version in ansence of DTDs PR #7336
State: Suspended
Resolution: Defer
User: None

**Notes:**
BAE F2F: for the present DTD's are required for entity resolution. This is a tricky issue, and the working group needs to resolve it quickly. We are asking for input from the Hypertext Coordination Group and others in our quest to sort it out.

# 5. Module Definition Conventions

This section is *normative*.

This document defines a variety of XHTML modules and the semantics of those modules. This section describes the conventions used in those module definitions.

## 5.1. Module Structure

Each module in this document is structured in the following way:

- An abstract definition [p.25] of the module's elements, attributes, and content models, as appropriate.
- A sub-section for each element in the module; These sub-sections contain the following components:
  - A brief description of the element,
  - A definition of each attribute or attribute collection [p.29] usable with the element, and
  - A detailed description of the behavior of the element, if appropriate.

  *Note that attributes are fully defined only the first time they are used in each module. After that, only a brief description of the attribute is provided, along with a link back to the primary definition.*

## 5.2. Abstract Module Definitions

An abstract module is a definition of an XHTML module using prose text and some informal markup conventions. While such a definition is not generally useful in the machine processing of document types, it is critical in helping people understand what is contained in a module. This section defines the way in which XHTML abstract modules are defined. An XHTML-conforming module is *not required* to provide an abstract module definition. However, anyone developing an XHTML module is encouraged to provide an abstraction to ease in the use of that module.

## 5.3. Syntactic Conventions

The abstract modules are not defined in a formal grammar. However, the definitions do adhere to the following syntactic conventions. These conventions are similar to those of XML DTDs, and should be familiar to XML DTD authors. Each discrete syntactic element can be combined with others to make more complex expressions that conform to the algebra defined here.

element name
: When an element is included in a content model, its explicit name will be listed.

content set
: Some modules define lists of explicit element names called *content sets*. When a content set is included in a content model, its name will be listed.

`expr ?`
: Zero or one instances of expr are permitted.

`expr +`
: One or more instances of expr are required.

`expr *`
: Zero or more instances of expr are permitted.

`a , b`
: Expression `a` is required, followed by expression `b`.

`a | b`
: Either expression a or expression b is required.

`a - b`
: Expression a is permitted, omitting elements in expression b.

parentheses
: When an expression is contained within parentheses, evaluation of any subexpressions within the parentheses take place before evaluation of expressions outside of the parentheses (starting at the deepest level of nesting first).

extending pre-defined elements
: In some instances, a module adds attributes to an element. In these instances, the element name is followed by an ampersand (`&`).

defining required attributes
: When an element requires the definition of an attribute, that attribute name is followed by an asterisk (`*`).

defining the type of attribute values
: When a module defines the type of an attribute value, it does so by listing the type in parentheses after the attribute name.

defining the legal values of attributes
: When a module defines the legal values for an attribute, it does so by listing the explicit legal values (enclosed in quotation marks), separated by vertical bars (`|`), inside of parentheses following the attribute name. If the attribute has a default value, that value is followed by an asterisk (`*`). If the attribute has a fixed value, the attribute name is followed by an equals sign (`=`) and the fixed value enclosed in quotation marks.

## 5.4. Content Types

Abstract module definitions define minimal, atomic content models for each module. These minimal content models reference the elements in the module itself. They may also reference elements in other modules upon which the abstract module depends. Finally, the content model in many cases requires that text be permitted as content to one or more elements. In these cases, the symbol used for text is PCDATA. This is a term, defined in the XML 1.0 Recommendation, that refers to processed character data. A content type can also be defined as EMPTY, meaning the element has no content in its minimal content model.

## 5.5. Attribute Types

In some instances, it is necessary to define the types of attribute values or the explicit set of permitted values for attributes. The following attribute types (defined in the XML 1.0 Recommendation) are used in the definitions of the abstract modules:

| Attribute Type | Definition |
|---|---|
| CDATA | Character data |
| ID | A document-unique identifier |
| IDREF | A reference to a document-unique identifier |
| IDREFS | A space-separated list of references to document-unique identifiers |
| NMTOKEN | A name composed of only name tokens as defined in XML 1.0 [XML [p.226] ]. |
| NMTOKENS | One or more white space separated NMTOKEN values |
| NUMBER | Sequence consisting only digits ([0-9]) |
| PCDATA | Processed character data |

In addition to these pre-defined data types, XHTML Modularization defines the following data types and their semantics (as appropriate):

| Data type | Description |
|---|---|
| Charset | A character encoding, as per [RFC2045 [p.225] ]. |
| ContentTypes | A list of media ranges with optional accept parameters, as defined in section 14.1 of [RFC2616 [p.226] ] as the field value of the accept request header. |
| Coordinates | Comma separated list of Length [p.28] s used in defining areas. |
| Datetime | Date and time information, as defined by the type `dateTime` in [XMLSCHEMA [p.226] ]. |
| HrefTarget | Name used as destination for results of certain actions, with legal values as defined by NMTOKEN [p.27] . |
| LanguageCode | A language code, as per [RFC3066 [p.226] ]. |
| LanguageCodes | A comma-separated list of language ranges with optional accept parameters, as defined in section 14.5 of [RFC2616 [p.226] ] as the field value of the Accept-Language request header. |
| Length | The value may be either in pixels or a percentage of the available horizontal or vertical space. Thus, the value "50%" means half of the available space. |
| LocationPath | A location path as defined in [XPATH [p.227] ]. |
| MediaDesc | A comma-separated list of media descriptors as described by [CSS2 [p.225] ]. The default is `all`. |
| Number | One or more digits |
| QName | An [XMLNS [p.226] ]-qualified name. See QName for a formal definition. |
| Text | Arbitrary textual data, likely meant to be human-readable. |
| URI | A Uniform Resource Identifier Reference, as defined by the type anyURI in [XMLSCHEMA [p.226] ]. |
| URIs | A space-separated list of URIs as defined above. |

# 6. XHTML Attribute Collections

This section is *normative*.

Many of the modules in this document define the required attributes for their elements. The elements in those modules may also reference zero or more attribute collections. Attribute collections are defined in their own modules, but the meta collection "Common" is defined in this section. The table below summarizes the attribute collections available.

| Collection | Module | Description |
|---|---|---|
| Core [p.61] | Core Attributes Module [p.61] | Basic attributes used to identify and classify elements and their content. |
| I18N [p.69] | Internationalization Attribute Module [p.69] | Attribute to identify the language of an elements and its contents. |
| Bi-directional [p.71] | Bi-directional Text Collection [p.71] | Attributes used to manage bi-directional text. |
| Edit [p.73] | Edit Attributes Module [p.73] | Attributes used to annotate when and how an element's content was edited. |
| Embedding [p.75] | Embedding Attributes Module [p.75] | Attributes used to embed content from other resources within the current element. |
| Events [p.147] | XML Events Module [p.147] | Attributes that allow associating of events and event processing with an element and its contents. |
| Hypertext [p.63] | Hyptertext Attributes Module [p.63] | Attributes that designate characteristics of links within and among documents. |
| Map [p.77] | Image Map Attributes Module [p.77] | Attributes for defining and referencing client-side image maps. |
| Metainformation [p.81] | Metainformation Attributes [p.81] | Attributes that allow associating of elements with metainformation about those elements |
| Style [p.115] | Style Attribute Module [p.115] | An attribute for associating style information with an element and its contents. |
| Common | Attribute Collections Module [p.29] | A meta-collection of all the other collections, including the Core [p.61] , I18N [p.69] , Events [p.147] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , Style [p.115] , Bi-directional [p.71] , and Hypertext [p.63] attribute collections. |

Implementation: RELAX NG [p.155]

Each of the attributes defined in an XHTML attribute collection is available when its corresponding module is included in an XHTML Host Language or an XHTML Integration Set (see [XHTMLMOD [p.226] ]). In such a situation, the attributes are available for use on elements that are NOT in the XHTML namespace when they are referenced using their namespace-qualified identifier (e.g., `xhtml:id`). The semantics of the attributes remain the same regardless of whether they are referenced using their qualified identifier or not. If both the qualified and non-qualified identifier for an attribute are used on the same XHTML namespace element, the behavior is unspecified.

## 6.1. Issues

xml:id PR #7442
State: Suspended
Resolution: Defer
User: None

**Notes:**
If xml:id becomes stable document in time for use in this document, we will migrate to its use.

# 7. XHTML Document Module

This section is *normative*.

The Document Module defines the major structural elements for XHTML. These elements effectively act as the basis for the content model of many XHTML family document types. The elements and attributes included in this module are:

| Elements | Attributes | Minimal Content Model |
|---|---|---|
| html [p.31] | Common [p.29] , version [p.32] (CDATA [p.27] ), xmlns (URI [p.28] = "http://www.w3.org/2002/06/xhtml2"), xsi:schemaLocation [p.32] (URIs [p.28] = "http://www.w3.org/2002/06/xhtml2 TBD") | head [p.32] , body [p.33] |
| head [p.32] | Common [p.29] | title [p.32] |
| title [p.32] | Common [p.29] | (PCDATA \| Text [p.45] )* |
| body [p.33] | Common [p.29] | ( Heading [p.37] \| Structural [p.37] \| List [p.55] )* |

This module is the basic structural definition for XHTML content. The `html` element acts as the root element for all XHTML Family Document Types.

Note that the value of the xmlns declaration is defined to be "http://www.w3.org/2002/06/xhtml2". Also note that because the xmlns declaration is treated specially by XML namespace-aware parsers [XMLNS [p.226] ], it is legal to have it present as an attribute of each element. However, any time the xmlns declaration is used in the context of an XHTML module, whether with a prefix or not, the value of the declaration shall be the XHTML namespace defined here.

Implementation: RELAX NG [p.156]

## 7.1. The html element

After the document type declaration, the remainder of an XHTML document is contained by the html [p.31] element.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

version = CDATA [p.27]
: The value of this attribute specifies which XHTML Family document type governs the current document. The format of this attribute value is unspecified.

  Need a normative definition for the version attribute

  The version attribute needs a machine processable format so that document processors can reliably determine that the document is an XHTML Family conforming document.

xsi:schemaLocation = URIs [p.28]
: This attribute allows the specification of a location where an XML Schema [XMLSCHEMA [p.226] ] for the document can be found. The syntax of this attribute is defined in xsi_schemaLocation. The behavior of this attribute in XHTML documents is defined in Strictly Conforming Documents.

## 7.2. The head element

The head [p.32] element contains information about the current document, such as its title, that is not considered document content. The default presentation of the head is not to display it; however that can be overridden with a stylesheet for special purpose use. User agents may however make information in the head [p.32] available to users through other mechanisms.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Example:

```
<head>
    <title>My Life</title>
</head>
```

## 7.3. The title element

Every XHTML document **must** have a title [p.32] element in the head [p.32] section.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

The title [p.32] element is used to identify the document. Since documents are often consulted out of context, authors should provide context-rich titles. Thus, instead of a title such as "Introduction", which doesn't provide much contextual background, authors should supply a title such as "Introduction to Medieval Bee-Keeping" instead.

For reasons of accessibility, user agents must always make the content of the title [p.32] element available to users. The mechanism for doing so depends on the user agent (e.g., as a caption, spoken).

Example:

```
<title>A study of population dynamics</title>
```

## 7.4. The body element

The body of a document contains the document's content. The content may be processed by a user agent in a variety of ways. For example by visual browsers it can be presented as text, images, colors, graphics, etc., an audio user agent may speak the same content, and a search engine may create an index prioritized according to where text appears.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Example:

```
<body id="theBody">
    <p>A paragraph</p>
</body>
```

# 8. XHTML Structural Module

This section is *normative*.

This module defines all of the basic text container elements, attributes, and their content models that are structural in nature.

| Element | Attributes | Minimal Content Model |
|---|---|---|
| address [p.37] | Common [p.29] | (PCDATA \| Text [p.45] )* |
| blockcode [p.37] | Common [p.29] | (PCDATA \| Text [p.45] \| Heading [p.37] \| Structural [p.37] \| List [p.55] )* |
| blockquote [p.38] | Common [p.29] | (PCDATA \| Text [p.45] \| Heading [p.37] \| Structural [p.37] \| List [p.55] )* |
| div [p.38] | Common [p.29] | (PCDATA \| Flow [p.37] )* |
| h [p.39] | Common [p.29] | (PCDATA \| Text [p.45] )* |
| h1 [p.39] | Common [p.29] | (PCDATA \| Text [p.45] )* |
| h2 [p.39] | Common [p.29] | (PCDATA \| Text [p.45] )* |
| h3 [p.39] | Common [p.29] | (PCDATA \| Text [p.45] )* |
| h4 [p.39] | Common [p.29] | (PCDATA \| Text [p.45] )* |
| h5 [p.39] | Common [p.29] | (PCDATA \| Text [p.45] )* |
| h6 [p.39] | Common [p.29] | (PCDATA \| Text [p.45] )* |
| p [p.40] | Common [p.29] | (PCDATA \| Text [p.45] \| List [p.55] \| blockcode [p.37] \| blockquote [p.38] \| pre [p.41] )* |
| pre [p.41] | Common [p.29] | (PCDATA \| Text [p.45] )* |
| section [p.42] | Common [p.29] | (PCDATA \| Flow [p.37] )* |
| separator [p.42] | Common [p.29] | EMPTY |

The content model for this module defines some content sets:

Heading
h [p.39] | h1 [p.39] | h2 [p.39] | h3 [p.39] | h4 [p.39] | h5 [p.39] | h6 [p.39]
Structural
address [p.37] | blockcode [p.37] | blockquote [p.38] | div [p.38] | p [p.40] | pre [p.41] | section [p.42] | separator [p.42] |
Flow
Heading [p.37] | Structural [p.37] | Text [p.45]

Implementation: RELAX NG [p.158]

## 8.1. The address element

The address [p.37] element may be used by authors to supply contact information for a document or a major part of a document such as a form. This element often appears at the beginning or end of a document.

*Attributes*

The Common [p.29] collection
A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Example:

```
<address href="mailto:webmaster@example.net">Webmaster</address>
```

## 8.2. The blockcode element

This element indicates that its contents are a block of "code" (see the code [p.47] element). This element is similar to the pre [p.41] element, in that whitespace in the enclosed text has semantic relevance. The whitespace should normally be included in visual renderings of the content.

Non-visual user agents are not required to respect extra white space in the content of a blockcode [p.37] element.

*Attributes*

The Common [p.29] collection
A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

The following example shows a code fragment:

```
<blockcode class="Perl">
sub squareFn {
    my $var = shift;

    return $var * $var ;
}
</blockcode>
```

Here is how this might be rendered:

```
sub squareFn {
    my $var = shift;

    return $var * $var ;
}
```

## 8.3. The blockquote element

This element designates a block of quoted text.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

This example formats an excerpt from "The Two Towers", by J.R.R. Tolkien, as a blockquote.

```
<blockquote cite="http://www.example.com/tolkien/twotowers.html">
<p>They went in single file, running like hounds on a strong scent,
and an eager light was in their eyes. Nearly due west the broad
swath of the marching Orcs tramped its ugly slot; the sweet grass
of Rohan had been bruised and blackened as they passed.</p>
</blockquote>
```

## 8.4. The div element

The div [p.38] element, in conjunction with the id [p.61] and class [p.61] attributes, offers a generic mechanism for adding extra structure to documents. This element defines no presentational idioms on the content. Thus, authors may use this element in conjunction with style sheets [p.117] , the xml:lang [p.69] attribute, etc., to tailor XHTML to their own needs and tastes.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

For example, suppose you wish to make a presentation in XHTML, where each slide is enclosed in a separate element. You could use a div [p.38] element, with a class [p.61] of `slide`:

```
<body>
    <h>The meaning of life</h>
    <p>By Huntington B. Snark</p>
    <div class="slide">
        <h>What do I mean by "life"</h>
        <p>....</p>
    </div>
    <div class="slide">
        <h>What do I mean by "mean"?</h>
        ...
    </div>
    ...
</body>
```

## 8.5. The heading elements

A heading element briefly describes the topic of the section it introduces. Heading information may be used by user agents, for example, to construct a table of contents for a document automatically.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

There are two styles of headings in XHTML: the numbered versions h1 [p.39] , h2 [p.39] etc., and the structured version h [p.39] , which is used in combination with the section [p.42] element.

There are six levels of numbered headings in XHTML with h1 [p.39] as the most important and h6 [p.39] as the least. The visual presentation of headers can render more important headings in larger fonts than less important ones.

Structured headings use the single h element, in combination with the section [p.42] element to indicate the structure of the document, and the nesting of the sections indicates the importance of the heading. The heading for the section is the one that is a child of the section element.

For example:

```
<body>
<h>This is a top level heading</h>
<p>....</p>
<section>
    <p>....</p>
    <h>This is a second-level heading</h>
    <p>....</p>
```

```
        <h>This is another second-level heading</h>
        <p>....</p>
    </section>
    <section>
        <p>....</p>
        <h>This is another second-level heading</h>
        <p>....</p>
        <section>
            <h>This is a third-level heading</h>
            <p>....</p>
        </section>
    </section>
```

These visual representation of these levels can be distinguished in a style sheet:

```
h {font-family: sans-serif; font-weight: bold; font-size: 200%}
section h {font-size: 150%} /* A second-level heading */
section section h {font-size: 120%} /* A third-level heading */
```

etc.

***Numbered sections and references***
*XHTML does not itself cause section numbers to be generated from headings. Style sheet languages such as CSS however allow authors to control the generation of section numbers.*

*The practice of skipping heading levels is considered to be bad practice. The series* `h1 h2 h1` *is acceptable, while* `h1 h3 h1` *is not, since the heading level* `h2` *has been skipped.*

## 8.6. The p element

The p [p.40] element represents a paragraph.

In comparison with earlier versions of HTML, where a paragraph could only contain inline text, XHTML2's paragraphs represent the conceptual idea of a paragraph, and so may contain lists, blockquotes, pre's and tables as well as inline text. They may not, however, contain directly nested p [p.40] elements.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Example:

```
<p>Payment options include:
<ul>
<li>cash</li>
<li>credit card</li>
<li>luncheon vouchers.</li>
</ul>
</p>
```

## 8.7. The pre element

The pre [p.41] element indicates that whitespace in the enclosed text has semantic relevance, and will normally be included in visual renderings of the content.

Note that *all* elements in the XHTML family preserve their whitespace in the document, which is only removed on rendering, via a stylesheet, according to the rules of CSS [CSS]. This means that in principle any elements may preserve or collapse whitespace on rendering, under control of a stylesheet. Also note that there is no normative requirement that the <pre> element be rendered in a monospace font (although this is the default rendering), nor that text wrapping be disabled.

Non-visual user agents are not required to respect extra white space in the content of a pre [p.41] element.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

The following example shows a preformatted verse from Shelly's poem *To a Skylark*:

```
<pre>
       Higher still and higher
         From the earth thou springest
       Like a cloud of fire;
         The blue deep thou wingest,
And singing still dost soar, and soaring ever singest.
</pre>
```

Here is how this might be rendered:

```
       Higher still and higher
         From the earth thou springest
       Like a cloud of fire;
         The blue deep thou wingest,
And singing still dost soar, and soaring ever singest.
```

## 8.8. The section element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

The section [p.42] element, in conjunction with the h [p.39] element, offers a mechanism for structuring documents into sections. This element defines content to be block-level but imposes no other presentational idioms on the content, which may otherwise be controlled from a style sheet.

*By representing the structure of documents explicitly using the section [p.42] and h [p.39] elements gives the author greater control over presentation possibilities than the traditional implicit structuring using numbered levels of headings. For instance, it is then possible to indicate the nesting of sections by causing a border to be displayed to the left of sections.*

Here is an example

```
<body>
<h>Events</h>
<section>
    <h>Introduction</h>
    <p>....</p>
    <h>Specifying events</h>
    <p>...</p>
    <section>
        <h>Attaching events to the handler</h>
        <p>...</p>
    </section>
    <section>
        <h>Attaching events to the listener</h>
        <p>...</p>
    </section>
    <section>
        <h>Specifying the binding elsewhere</h>
        <p>...</p>
    </section>
</section>
```

## 8.9. The separator element

The separator [p.42] element places a break in the document. The default style for this break is a horizontal line in Western languages.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Example:

```
<p>This is some lead in text</p>
<separator />
<p>This is some additional, but separate text.</p>
```

# 9. XHTML Text Module

This section is *normative*.

This module defines all of the basic text container elements, attributes, and their content models that are "inline level". Note that while the concept of "inline level" can be construed as a presentation aspect, in this case it is intended to only have a semantic meaning.

| Element | Attributes | Minimal Content Model |
|---|---|---|
| abbr | Common [p.29] | (PCDATA \| Text [p.45] )* |
| cite | Common [p.29] | (PCDATA \| Text [p.45] )* |
| code | Common [p.29] | (PCDATA \| Text [p.45] )* |
| dfn | Common [p.29] | (PCDATA \| Text [p.45] )* |
| em | Common [p.29] | (PCDATA \| Text [p.45] )* |
| kbd | Common [p.29] | (PCDATA \| Text [p.45] )* |
| l | Common [p.29] | (PCDATA \| Text [p.45] )* |
| quote | Common [p.29] | (PCDATA \| Text [p.45] )* |
| samp | Common [p.29] | (PCDATA \| Text [p.45] )* |
| span | Common [p.29] | (PCDATA \| Text [p.45] )* |
| strong | Common [p.29] | (PCDATA \| Text [p.45] )* |
| sub | Common [p.29] | (PCDATA \| Text [p.45] )* |
| sup | Common [p.29] | (PCDATA \| Text [p.45] )* |
| var | Common [p.29] | (PCDATA \| Text [p.45] )* |

The content model for this module defines a content set:

Text
: abbr [p.46] | cite [p.46] | code [p.47] | dfn [p.47] | em [p.47] | kbd [p.48] | l [p.48] | quote [p.49] | samp [p.49] | span [p.50] | strong [p.50] | var [p.51]

Implementation: RELAX NG [p.163]

## 9.1. The abbr element

The abbr [p.46] element indicates that a text fragment is an abbreviation (e.g., W3C, XML, Inc., Ltd., Mass., etc.); this includes acronyms.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

The content of the abbr [p.46] element specifies the abbreviated expression itself, as it would normally appear in running text. The title [p.62] attribute of these elements may be used to provide the full or expanded form of the expression. Such a title [p.62] attribute should be repeated each time the abbreviation is defined in the document.

Note that abbreviations often have idiosyncratic pronunciations. For example, while "IRS" and "BBC" are typically pronounced letter by letter, "NATO" and "UNESCO" are pronounced phonetically. Still other abbreviated forms (e.g., "URI" and "SQL") are spelled out by some people and pronounced as words by other people. When necessary, authors should use style sheets to specify the pronunciation of an abbreviated form.

Examples:

```
<abbr title="Limited">Ltd.</abbr>
<abbr title="Abbreviation">abbr.</abbr>
```

## 9.2. The cite element

The cite [p.46] element contains a citation or a reference to other sources.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

In the following example, the cite [p.46] element is used to reference the book from which the quotation is taken:

```
As <cite cite="http://www.example.com/books/the_two_towers"">Gandalf
the White</cite> said, <quote xml:lang="en-us">The hospitality of
your hall is somewhat lessened of late, Theoden King.</quote>
```

In this example, a separate specification is referenced:

```
More information can be found in <cite cite="http://www.w3.org/TR/REC-xml">[XML]</cite>.
```

## 9.3. The code element

The code [p.47] element contains a fragment of computer code.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Example:

```
The Pascal statement <code>i := 1;</code> assigns the
literal value one to the variable <var>i</var>.
```

## 9.4. The dfn element

The dfn [p.47] element contains the defining instance of the enclosed term.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Example:

```
An <dfn id="def-acronym">acronym</dfn> is a word formed
from the initial letters or groups of letters of words in a set phrase
or series of words.
```

## 9.5. The em element

The em [p.47] element indicates emphasis for its contents.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Example:

```
Do <em>not</em> phone before 9 a.m.
```

## 9.6. The kbd element

The kbd [p.48] element indicates input to be entered by the user.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Example:

```
To exit, type <kbd>QUIT</kbd>.
```

## 9.7. The l element

The l [p.48] element represents a semantic line of text (e.g., a line of verse or a line of computer code).

When visually represented, the l [p.48] element should start on a new line and have a line break at the end. Whether the line should wrap or not depends on styling properties of the element.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

*By retaining structure in text that has to be broken over lines, you retain essential information about its makeup. This gives you greater freedom with styling the content. For instance, line numbers can be generated automatically from the stylesheet if needed.*

*For instance, for a document with the following structure:*

```
<blockcode class="program">
<l>program p(input, output);</l>
<l>begin</l>
<l>   writeln("Hello world");</l>
<l>end.</l>
</blockcode>
```

*the following CSS stylesheet would number each line:*

```
.program { counter-reset: linenumber }

l:before {
    position: relative;
    left: -1em;
    counter-increment: linenumber;
    content: counter(linenumber);
}
```

## 9.8. The quote element

This element designates an inline text fragment of quoted text.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Visual user agents must not by default add delimiting quotation marks (as was the case for the `q` element in earlier versions of XHTML). It is the responsibility of the document author to add any required quotation marks, either directly in the text, or via a stylesheet.

The following example illustrates nested quotations with the quote [p.49] element.

```
<p>John said, <quote>"I saw Lucy at lunch, she told me
<quote>'Mary wants you
to get some ice cream on your way home.'</quote> I think I will get
some at Jen and Berry's, on Gloucester Road."</quote></p>
```

Here is an example using the cite [p.64] attribute:

```
Steven replied:
<quote cite="http://lists.example.org/2002/01.html">We quite agree</quote>
```

## 9.9. The samp element

The samp [p.49] element designates sample output from programs, scripts, etc.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Example:

```
On starting, you will see the prompt <samp>$ </samp>.
```

## 9.10. The span element

The span [p.50] element, in conjunction with the id [p.61] and class [p.61] attributes, offers a generic mechanism for adding structure to documents. This element imposes no presentational idioms on the content. Thus, authors may use this element in conjunction with style sheets [p.117] , the xml:lang [p.69] attribute, etc., to tailor XHTML to their own needs and tastes.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

For example, suppose you wish to mark all words in a document that need to be collected into an index. You could use a span [p.50] element, with a class [p.61] of `xref`:

```
<p>This operation is called
the <span class="xref">transpose</span>
or <span class="xref">inverse</span>.</p>
```

## 9.11. The strong element

The strong [p.50] element indicates higher importance for its contents.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

```
On <strong>Monday</strong> please put the rubbish out,
but <em>not</em> before nightfall!
```

## 9.12. The sub element

The sub [p.50] element indicates that its contents should regarded as a subscript.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] ,

and Hypertext [p.63]

For visual user agents this element would normally be rendered as a subscript of the text baseline, but on user agents where this is not possible (for instance teletype-like devices) other renderings may be used. For instance, `a<sub>i, j</sub>` that would be rendered as $a_{i, j}$ on a device that can render it so, might be rendered as `a[i, j]` on a simpler device.

Example:

```
H<sub>2</sub>O
```

# 9.13. The sup element

The sup [p.51] element indicates that its contents should be regarded as a super-script.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

For visual user agents this element would normally be rendered as a super-script of the text baseline, but on user agents where this is not possible (for instance teletype-like devices) other renderings may be used. For instance, `2<sup>n</sup>` that would be rendered as $2^{n}$ on a device that can render it so, might be rendered as $2\uparrow(n)$ on a simpler device.

Many scripts (e.g., French) require superscripts or subscripts for proper rendering. The sub [p.50] and sup [p.51] elements should be used to markup text in these cases.

Examples:

```
E = mc<sup>2</sup>
<span xml:lang="fr">M<sup>lle</sup> Dupont</span>
```

# 9.14. The var element

The var [p.51] element indicates an instance of a variable or program argument.

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Example:

```
The parameter <var>ncols</var> represents
the number of colors to use.
```

## 9.15. Issues

[XHTML2] Comments on WD 5: make white space preservation part of the standard values for the code and kbd elements? PR #6554
State: Approved
Resolution: Accepted
User: Agree

**Notes:**
All elements preserve space in XHTML 2, so the author has the opportunity on user agents that support CSS to specify whether that should be reflected on output. There will be an attribute that will pass this preservation on to presentation.

# 10. XHTML Hypertext Module

This section is *normative*.

The Hypertext Module provides an element that traditionally has been used in HTML to define hypertext links to other resources. Although all elements may now play the role of a hyperlink (using the href [p.63] attribute) or hyperlink anchor (using the id [p.61] attribute), this element remains for explicit markup of links, though is otherwise identical in semantics to the span [p.50] element.

This module supports the following element:

| Element | Attributes | Minimal Content Model |
|---|---|---|
| a [p.53] | Common [p.29] | (PCDATA \| Text [p.45] )* |

This module adds the a [p.53] element to the Text [p.45] content set of the Text [p.45] Module.

Implementation: RELAX NG [p.168]

## 10.1. The a element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

An a [p.53] element defines an anchor. Since hypertext attributes such as href [p.63] may be applied to any element, this element is not strictly necessary, being equivalent to a span [p.50] , but has been retained to allow the expression of explicit links.

```
<a href="http://www.w3.org/">The W3C Home Page</a>
```

***Note.*** *User agents are required to find anchors created by empty elements.*

# 11. XHTML List Module

This section is *normative*.

As its name suggests, the List Module provides list-oriented elements. Specifically, the List Module supports the following elements and attributes:

| Elements | Attributes | Minimal Content Model |
|---|---|---|
| dl [p.57] | Common [p.29] | label [p.59] ?, (( dt [p.57] \| dd [p.57] )+ \| di [p.57] +) |
| di [p.57] | Common [p.29] | ( dt [p.57] +, dd [p.57] *) |
| dt [p.57] | Common [p.29] | (PCDATA \| Text [p.45] )* |
| dd [p.57] | Common [p.29] | (PCDATA \| Flow [p.37] )* |
| label [p.59] | Common [p.29] | (PCDATA \| Text [p.45] )* |
| nl [p.58] | Common [p.29] | label [p.59] , li [p.59] + |
| ol [p.58] | Common [p.29] | label [p.59] ?, li [p.59] + |
| ul [p.58] | Common [p.29] | label [p.59] ?, li [p.59] + |
| li [p.59] | Common [p.29] , value [p.59] | (PCDATA \| Flow [p.37] )* |

This module also defines the content set List with the minimal content model (dl | nl | ol | ul)+ and adds this set to the Flow [p.37] content set of the Structural [p.35] Module.

Implementation: RELAX NG [p.168]

XHTML offers authors several mechanisms for specifying lists of information. All lists must contain one or more list elements. Lists may contain:

- Unordered information.
- Ordered information.
- Navigation information.
- Definitions.

The previous list, for example, is an unordered list, created with the ul [p.58] element:

```
<ul>
<li>Unordered information. </li>
<li>Ordered information. </li>
<li>Navigation information. </li>
<li>Definitions. </li>
</ul>
```

An ordered list, created using the ol [p.58] element, should contain information where order should be emphasized, as in a recipe:

1. Mix dry ingredients thoroughly.
2. Pour in wet ingredients.
3. Mix for 10 minutes.
4. Bake for one hour at 300 degrees.

Definition lists, created using the dl [p.57] element, generally consist of a series of term/definition pairs (although definition lists may have other applications). Thus, when advertising a product, one might use a definition list:

**Lower cost**
: The new version of this product costs significantly less than the previous one!

**Easier to use**
: We've changed the product so that it's much easier to use!

**Safe for kids**
: You can leave your kids alone in a room with this product and they won't get hurt (not a guarantee).

defined in XHTML as:

```
<dl>
<dt><strong>Lower cost</strong></dt>
<dd>The new version of this product costs significantly less than the
previous one!</dd>
<dt><strong>Easier to use</strong></dt>
<dd>We've changed the product so that it's much easier to
use!</dd>
<dt><strong>Safe for kids</strong></dt>
<dd>You can leave your kids alone in a room with this product and
they won't get hurt (not a guarantee).</dd>
</dl>
```

Lists may also be nested and different list types may be used together, as in the following example, which is a definition list that contains an unordered list (the ingredients) and an ordered list (the procedure):

**The ingredients:**
- 100 g. flour
- 10 g. sugar
- 1 cup water
- 2 eggs
- salt, pepper

**The procedure:**
1. Mix dry ingredients thoroughly.
2. Pour in wet ingredients.
3. Mix for 10 minutes.
4. Bake for one hour at 300 degrees.

**Notes:**
The recipe may be improved by adding raisins.

The exact presentation of the three list types depends on the user agent and/or the stylesheet in effect. Authors must not use lists purely as a means of indenting text. This is a stylistic issue and is properly handled by style sheets.

## 11.1. Definition lists: the dl , di , dt , and dd elements

*Attributes*

The Common [p.29] collection
A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Definition lists vary only slightly from other types of lists in that list items consist of two parts: a term and a description. The term is given by the dt element and is restricted to inline content. The description is given with a dd element that contains block-level content. The term and its definition can be grouped within a di [p.57] element to help clarify the relationship between a term and its definition(s).

Here is an example:

```
<dl>
  <di>
    <dt>Dweeb</dt>
    <dd>young excitable person who may mature
      into a <em>Nerd</em> or <em>Geek</em></dd>
  </di>

  <di>
    <dt>Hacker</dt>
    <dd>a clever programmer</dd>
  </di>

  <di>
    <dt>Nerd</dt>
    <dd>technically bright but socially inept person</dd>
  </di>

</dl>
```

Here is an example with multiple terms and descriptions:

```
<dl>
   <dt>Center</dt>
   <dt>Centre</dt>
   <dd> A point equidistant from all points
              on the surface of a sphere.</dd>
   <dd> In some field sports, the player who
              holds the middle position on the field, court,
              or forward line.</dd>
</dl>
```

Another application of dl [p.57] , for example, is for marking up dialogues, with each dt naming a speaker, and each dd containing his or her words.

## 11.2. The nl element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Navigation lists are intended to be used to define collections of selectable items for presentation in a "navigation" menu. A navigation list is required to start with a label [p.59] element that defines the label for the list.

This example illustrates the basic structure of a nested navigation list:

```
<nl>
   <label>Contents </label>
   <li href="#introduction">Introduction</li>
   <li>
      <nl>
          <label>Terms</label>
          <li href="#may">May</li>
          <li href="#must">Must</li>
          <li href="#should">Should</li>
      </nl>
   </li>
   <li href="#conformance">Conformance</li>
   <li href="#references">References</li>
   ...
</nl>
```

## 11.3. The ol , and ul elements

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] ,

and Hypertext [p.63]

Ordered and unordered lists are rendered in an identical manner except that user agents MUST by default number ordered list items. User agents may present those numbers in a variety of ways. Unordered list items are not numbered.

Both types of lists are made up of sequences of list items defined by the li [p.59] element.

This example illustrates the basic structure of a list.

```
<ul>
   <li> ... first list item...</li>
   <li> ... second list item...</li>
   ...
</ul>
```

## 11.4. The li element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

value = NUMBER [p.27]
: This attribute specifies the value to be used when determining the value of the enumerator on a list item within an ol [p.58] .

The li [p.59] element defines a list item within an ordered, unordered, or navigation list. When the href [p.63] attribute is defined, the contents of the list item become a selectable link, just as an a [p.53] element with an href [p.63] attribute would be.

## 11.5. The label element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

The label [p.59] element is used to define a label for a list. The contents of the label element are displayed as the title of a list (or sublist).

# 12. XHTML Core Attributes Module

This section is *normative*.

This module defines the Core [p.61] attribute collection.

## 12.1. Core Attribute Collection

class = NMTOKENS [p.27]
: This attribute assigns one or more class names to an element; the element may be said to belong to these classes. A class name may be shared by several element instances.

  The class [p.61] attribute can be used for different purposes in XHTML, for instance as a style sheet [p.117] selector (when an author wishes to assign style information to a set of elements), and for general purpose processing by user agents.

  For instance in the following example, the p [p.40] element is used in conjunction with the class [p.61] attribute to identify a particular type of paragraph.

```
<p class="note">
These programs are only available if you have purchased
the advanced professional suite.
</p>
```

  Style sheet rules can then be used to render the paragraph appropriately, for instance by putting a border around it, giving it a different background colour, or where necessary by not displaying it at all.

id = ID [p.27]
: The id [p.61] attribute assigns an identifier to an element. The id of an element must be unique within a document.

  The id [p.61] attribute has several roles in XHTML:

  - As a style sheet [p.117] selector.
  - As a target anchor [p.53] for hypertext links.
  - As a means to reference a particular element from a script [p.111] .
  - As the name of a declared object [p.97] element.
  - For general purpose processing by user agents (e.g. for identifying fields when extracting data from XHTML pages into a database, translating XHTML documents into other formats, etc.).

  As an example, the following headings are distinguished by their id [p.61] values:

```
<h id="introduction">Introduction</h>
<p>...</p>
<h id="events">The Events Module</h>
<p>...</p>
```

title = Text [p.28]

This attribute offers advisory information about the element for which it is set.

Values of the title [p.62] attribute may be used by user agents in a variety of ways. For instance, visual browsers should display the title as a "tool tip" (a short message that appears when the pointing device pauses over an object). Audio user agents may speak the title information in a similar context.

Example of the use of title [p.62] :

```
<a href="Jakob.html" title="Author biography">Jakob Nielsen</a>’s Alertbox for January 11, 1998
```

Implementation: RELAX NG [p.172]

# 13. XHTML Hypertext Attributes Module

This section is *normative*.

The Hypertext Attributes Module defines the Hypertext [p.63] attribute collection. This collection allows an element to be the start point of a hypertext link to a remote resource.

## 13.1. Hypertext Attribute Collection

href = URI [p.28]

This attribute specifies a URI that is actuated when the element is activated.

Actuation occurs as the default action of a [DOM [p.225] ] DOMActivate event for the element that the attribute occurs on.

Examples:

```
<abbr href="http://www.w3.org/" title="World Wide Web">WWW</abbr>
<li href="contents.xhtml">contents</li>
<a href="http://www.cwi.nl/~steven/amsterdam.html">Amsterdam</a>
<quote href="hamlet.xhtml#p2435">To be or not to be</quote>
<var href="#index_ninc">ninc</var>
```

In terms of [HLINK [p.227] ], the href [p.63] attribute is specified as:

```
<hlink namespace="http://www.w3.org/2002/06/xhtml2"
       locator="href"
       effectValue="replace"
       replacement="target"
       role="rel"
       mediaType="hreftype"
       actuateValue="onRequest"/>
```

hreftype = ContentTypes [p.28]

This attribute specifies the allowable content types of the relevant href [p.63] URI. At its most general, it is a comma-separated list of media ranges with optional accept parameters, as defined in section 14.1 of [RFC2616 [p.226] ] as the field value of the accept request header.

In its simplest case, this is just a media type, such as "image/png" or "application/xml", but it may also contain asterisks, such as "image/*" or "*/*", or lists of acceptable media types, such as "image/png, image/gif, image/jpeg".

The user agent must combine this list it with its own list of acceptable media types by taking the intersection, and then use the resulting list as the field value of the `accept` request header when requesting the resource using HTTP.

For instance, if the attribute specifies the value "image/png, image/gif, image/jpeg", but the user agent does not accept images of type "image/gif" then the resultant accept header would contain "image/png, image/jpeg".

A user agent should imitate similar behavior when using other methods than HTTP. For instance, when accessing files in a local filestore, `<p src="logo" type="image/png, image/jpeg">` might cause the user agent first to look for a file `logo.png`, and then for `logo.jpg`.

If this attribute is not present, "*/*" is used for its value.

For the current list of registered content types, please consult [MIMETYPES [p.225] ].

Example:

```
<p href="http://www.w3.org" hreftype="text/html,application/xhtml+xml">
   The W3C Home Page
</p>
```

hreflang = LanguageCodes [p.28]

This attribute specifies the base language of the resource designated by href [p.63] and may only be used when href [p.63] is specified. At its most general, it is a comma-separated list of language ranges with optional accept parameters, as defined in section 14.5 of [RFC2616 [p.226] ] as the field value of the Accept-Language request header.

In its simplest case, this is just a language code, such as "nl", but it may also contain variant specifications such as "en-gb".

The user agent must use this list as the field value of the `accept-language` request header when requesting the resource using HTTP.

If this attribute is not present, the user agent must use its default value of the accept-language request header.

Example:

```
<p href="http://www.w3.org/2003/06/semantictour-pressrelease" hreflang="fr">
   The press release in French
</p>
```

cite = URI [p.28]

The value of this attribute is a URI that designates a source document or message. This attribute is intended to give further information about the element’s contents (e.g., the source from which a quotation was borrowed, or the reason text was inserted or deleted). User Agents should provide a means for the user to access the further information. Example:

```
cite="comments.html"
```

In terms of [HLINK [p.227] ], the cite [p.64] attribute is specified as:

```
<hlink namespace="http://www.w3.org/2002/06/xhtml2"
       locator="cite"
       effectValue="new"
       actuateValue="onRequestSecondary"/>
```

target = HrefTarget [p.28]

This attribute identifies an environment that will act as the destination for a resource identified by a hyperlink when it is activated.

This specification does not define how this attribute gets used, since that is defined by the environment that the hyperlink is actuated in. See for instance XFrames [XFRAMES [p.227] ].

Example:

```
<a href="home.html" target="main">Home</a>
```

access = QName [p.28]

This attribute assigns an access name to an element which is used to allow faster access to important parts of a document, by allowing shortcuts to be bound to them.

Activating an access shortcut bound to an element gives focus to the element. The action that occurs when an element receives focus depends on the element. For example, when a user activates a link defined by the a [p.53] element, the user agent generally follows the link. When a user activates a radio button, the user agent changes the value of the radio button. When the user activates a text field, it allows input, etc.

Access names do not have to be unique in a document. Activating a shortcut takes the user to the next element with the bound-to name.

In this example, we assign the access name "contents" to a link. Using the binding assigned to this location takes the user to another document, in this case, a table of contents.

```
<p access="contents"
   href="http://example.com/specification/contents.html">
  Table of Contents
</p>
```

shortcuts

The invocation of access shortcuts, and the assignment of key or other bindings to them is not further defined here. A user agent should allow the user to access the user agent's list of recognized access points, to add to them, and to specify bindings for them.

nextfocus = IDREF [p.27]

This attribute specifies an IDREF of an element in the current document that will receive focus when the user requests that user agent navigate to the next element that can receive

focus.

The elements in a document that can be "activated" are called "focusable" elements. The sequence of focusable elements is called the document's navigation order. The navigation order defines the order in which elements will receive focus when navigated by the user. The navigation order may include elements nested within other elements.

When a document is first loaded, a user agent must do the following:

1. If a document is loaded using a URI that includes a fragment reference (such as `book.html#chapter5`)
    1. If the fragment reference identifies an element in the document, the user agent must ensure that navigation starts at the beginning of that element.
    2. If the referenced element is focusable, that element receives focus.
    3. If the fragment reference does not resolve in the document, the user agent must ensure navigation starts at the beginning of the document.
2. If there is no fragment reference when the document is loaded:
    1. If the root element of the document has a nextfocus [p.65] attribute, and the element referred to by the attribute is focusable, the element must receive focus. The used agent must ensure the beginning of the element is visible on the display.
    2. If the root element has no nextfocus [p.65] attribute, no element receives initial focus. The user agent must ensure navigation starts at the beginning of the document.
3. If the user has moved away from the initial navigation point of a document (e.g., through using page up and page down or by changing focus), refreshing the document should result in the user's navigation location being preserved.

In the event no element in the document has focus, when the user requests the next focusable element, that element must be the next focusable element forward from the current navigation point in document order. If there are no focusable elements before the end of the document, focus shifts to the first focusable element in document order. If a document has no focusable elements, the behavior is unspecified.

Once a focusable element in the document has focus, upon requesting that focus change to the next focusable element, the user agent must do the following:

1. If the focused element has a nextfocus [p.65] attribute that references a focusable element, focus is moved to that element.
2. Otherwise, focus shifts to the next focusable element in document order.
3. If there are no remaining focusable elements in document order, focus shifts to the first focusable element in document order.

Regardless of the way in which an element receives focus, if the element is not currently visible on the user agent's display, the display must be updated so that the element is visible.

The following example would allow the links to be navigated in column order (without the use of `nextfocus` they would be navigated in document, i.e. row, order):

```
<table>
<tr><td id="a" nextfocus="b">NW</td>
    <td id="c" nextfocus="d">NE</td></tr>
<tr><td id="b" nextfocus="c">SW</td>
    <td id="d">SE</td></tr>
</table>
```

***Navigation keys.*** *The actual key sequence that causes navigation or element activation depends on the configuration of the user agent (e.g., the "tab" key might be used for navigation and the "enter" key or "space" key used to activate a selected element).*

prevfocus = IDREF [p.27]

This attribute specifies an IDREF of an element in the current document that will receive focus when the user requests that user agent navigate to the previous element that can receive focus.

The elements in a document that can be "activated" are called "focusable" elements. The sequence of focusable elements is called the document's navigation order. The navigation order defines the order in which elements will receive focus when navigated by the user. The navigation order may include elements nested within other elements.

In the event no element in the document has focus, when the user requests the previous focusable element, that element must be the next focusable element backward from the current navigation point in document order. If there is no such focusable element back to the start of the document, focus shifts to the last focusable element in document order. If a document has no focusable elements, the behavior is unspecified.

Once a focusable element in the document has focus, upon requesting that focus change to the previous focusable element, the user agent must do the following:

1. If the focused element has a prevfocus [p.67] attribute that references a focusable element, focus is moved to that element.
2. Otherwise, focus shifts to the previous focusable element in document order.
3. If there are no previous focusable elements in document order, focus shifts to the last focusable element in document order.

Regardless of the way in which an element receives focus, for visual user agents, if the element is not currently visible on the user agent's display, the display must be updated so that the element is visible.

The following example would allow the links to be navigated in column order (without the use of `nextfocus` and `prevfocus` they would be navigated in document, i.e. row, order):

```
<table>
<tr>
    <td id="a" nextfocus="b"              >NW</td>
    <td id="c" nextfocus="d" prevfocus="b">NE</td></tr>
<tr>
    <td id="b" nextfocus="c" prevfocus="a">SW</td>
    <td id="d"               prevfocus="c">SE</td></tr>
</table>
```

***Navigation keys.*** *The actual key sequence that causes navigation depends on the configuration of the user agent (e.g., the "shift-tab" key might be used for navigation).*

xml:base = URI [p.28]

This attribute specifies the base URI from which to resolve relative URIs. It is normatively defined in [XMLBASE [p.226] ]. Any relative URI used on an element that uses this attribute, or on an element contained within an element that uses this attribute, must be resolved relative to the base URI defined by this attribute.

An element inherits URI base information according to the following order of precedence (highest to lowest):

- The xml:base [p.68] attribute set for the element itself.
- The closest parent element that has the xml:base [p.68] attribute set (i.e., the xml:base [p.68] attribute is inherited).
- The HTTP "Content-Location" header (which may be configured in a server).
- The location of the document itself.

Example:

```
See:
<ul xml:base="http://www.w3.org">
<li href="/" src="Icons/w3c_home">The W3C home page</li>
<li href="/TR">The W3C Technical Reports page</li>
<li href="/Markup">The HTML home page</li>
<li href="/Markup/Forms">The XForms home page</li>
</ul>
```

Implementation: RELAX NG [p.173]

# 14. XHTML I18N Attribute Module

This section is *normative*.

This module defines the I18N [p.69] attribute collection.

## 14.1. I18N Attribute Collection

xml:lang = LanguageCode [p.28]

This attribute specifies the base language of an element's attribute values and text content. It is defined normatively in [XML [p.226] ] section 2.12. The default value of this attribute is unspecified.

An element inherits language code information according to the following order of precedence (highest to lowest):

- The xml:lang [p.69] attribute set for the element itself.
- The closest parent element that has the xml:lang [p.69] attribute set (i.e., the xml:lang [p.69] attribute is inherited).
- The HTTP "Content-Language" header (which may be configured in a server).

In this example, the primary language of the document is French ("fr"). Within the first paragraph a single word is declared to be in English ("en"), after which the primary language returns to French. The following paragraph is declared to be in English and includes an embedded French word.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 2.0//EN"
   "TBD">
<html xmlns="http://www.w3.org/2002/06/xhtml2" xml:lang="fr">
<head>
   <title>Un document multilingue</title>
</head>
<body>
<p>En janvier, toutes les boutiques de Londres affichent des panneaux
<span xml:lang="en">SALE</span>, mais en fait ces magasins
sont bien propres!</p>
<p xml:lang="en">Everwhere I went in France, the bakeries proclaimed
<em xml:lang="fr">PAIN</em>!
Those poor French bakers!</p>
</body>
</html>
```

Implementation: RELAX NG [p.175]

# 15. XHTML Bi-directional Text Attribute Module

This section is *normative*.

The Bi-directional Text module defines the Bi-directional [p.71] attribute collection.

## 15.1. Bi-directional Text Collection

dir = "`ltr`|`rtl`|`lro`|`rlo`"

This attribute specifies the base direction of the element's text content. This direction overrides the inherent directionality of characters as defined in Unicode Standard Annex #9: The Bidirectional Algorithm [UAX9 [p.226] ], and defines directional properties of text as defined by CSS2 [CSS2 [p.225] ]. The default value of this attribute is user-agent dependent, but must be either `ltr` or `rtl`. Possible values are:

- `ltr`: Left-to-right text. The effect of this attribute is defined by the CSS2 rule:

```
*[dir="ltr"] { unicode-bidi: embed; direction: ltr}
```

- `rtl`: Right-to-left text. The effect of this attribute is defined by the CSS2 rule:

```
*[dir="rtl"] { unicode-bidi: embed; direction: rtl}
```

- `lro`: Left-to-right override. The effect of this attribute is defined by the CSS2 rule:

```
*[dir="lro"] { unicode-bidi: bidi-override; direction: ltr}
```

- `rlo`: Right-to-left override. The effect of this attribute is defined by the CSS2 rule:

```
*[dir="rlo"] { unicode-bidi: bidi-override; direction: rtl}
```

Example:

```
<p dir="ltr">
The Hebrew word for "Hebrew" is <span xml:lang="he">&#1506;&#1489;&#1512;&#1497;&#1514;</span>,
but since Hebrew letters have intrinsic right-to-left directionality,
I had to type the word starting from the letter "&#1506;",
i.e. <span xml:lang="he" dir="lro">&#1506;&#1489;&#1512;&#1497;&#1514;</span>.
</p>
```

### 15.1.3. 15.1.1. Inheritance of text direction information

The Unicode bidirectional algorithm requires a base text direction for text blocks. To specify the base direction of a block-level element, set the element's dir [p.71] attribute. The default value of the dir [p.71] attribute is "ltr" (left-to-right text).

When the dir [p.71] attribute is set for a block-level element, it remains in effect for the duration of the element and any nested block-level elements. Setting the dir [p.71] attribute on a nested element overrides the inherited value.

To set the base text direction for an entire document, set the dir [p.71] attribute on the html [p.31] element.

Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 2.0//EN"
   "TBD">
<html dir="rtl">
<head>
<title>...a right-to-left title...</title>
</head>
...right-to-left text...
<p dir="ltr">...left-to-right text...</p>
<p>...right-to-left text again...</p>
</html>
```

Inline elements, on the other hand, do not inherit the dir [p.71] attribute. This means that an inline element without a dir [p.71] attribute does **not** open an additional level of embedding with respect to the bidirectional algorithm. (Here, an element is considered to be inline if its default presentation is expressed in [CSS2 [p.225] ] as `display: inline`.)

## 15.1.4. 15.1.2. The effect of style sheets on bidirectionality

In general, using style sheets to change an element's visual rendering from block-level to inline or vice-versa is straightforward. However, because the bidirectional algorithm relies on the inline/block-level distinction, special care must be taken during the transformation.

When an inline element that does not have a dir [p.71] attribute is transformed to the style of a block-level element by a style sheet, it inherits the dir [p.71] attribute from its closest parent block element to define the base direction of the block.

When a block element that does not have a dir [p.71] attribute is transformed to the style of an inline element by a style sheet, the resulting presentation should be equivalent, in terms of bidirectional formatting, to the formatting obtained by explicitly adding a dir [p.71] attribute (assigned the inherited value) to the transformed element.

Implementation: RELAX NG [p.176]

# 16. XHTML Edit Attributes Module

This section is *normative*.

This module defines the Edit [p.73] attribute collection.

## 16.1. Edit Collection

edit = "`inserted`|`deleted`|`changed`|`moved`"

This attribute allows elements to carry information indicating how content has changed. Possible values:

- `inserted`: the content has been inserted
- `deleted`: the content has been deleted
- `changed`: the content has changed considerably, therefore making it not worth being marked up with values of `inserted` and `deleted`
- `moved`: the content has been moved from some other part of the document.

The default presentation for an element with `edit="deleted"` is `display: none` (in other words, it is not displayed) although an alternate style might display it as crossed through. The other three values cause no special presentation by default, though an alternate style might use background colors or other text decoration to indicate the changed text.

Example:

```
<p>I will do it
next <span edit="deleted">week</span><span edit="inserted">month</span>.</p>
```

datetime = Datetime [p.28]

The value of this attribute specifies the date and time when a change was made. Example:

```
datetime="2003-01-13T13:15:30Z"
```

Implementation: RELAX NG [p.176]

# 17. XHTML Embedding Attributes Module

This section is *normative*.

The Embedding Attributes module defines the Embedding [p.75] attribute collection.

This collection causes the contents of a remote resource to be embedded in the document in place of the element's content. If accessing the remote resource fails, for whatever reason (network unavailable, no resource available at the URI given, inability of the user agent to process the type of resource) the content of the element must be processed instead.

Note that this behavior makes documents far more robust, and gives much better opportunities for accessible documents than the `longdesc` attribute present in earlier versions of XHTML, since it allows the description of the resource to be included in the document itself, rather than in a separate document.

Examples:

```
<p src="holiday.png" type="image/png">
    <span src="holiday.gif" type="image/gif">
        An image of us on holiday.
    </span>
</p>

<table src="temperature-graph.png" type="image/png">
<caption>Average monthly temperature over the last 20 years</caption>
<tr><th>Jan</th><th>Feb</th><th>Mar</th><th>Apr</th><th>May</th><th>Jun</th>
    <th>Jul</th><th>Aug</th><th>Sep</th><th>Oct</th><th>Nov</th><th>Dec</th>
</tr>
<tr><td> 4</td><td> 2</td><td> 7</td><td> 9</td><td>13</td><td>16</td>
    <td>17</td><td>17</td><td>14</td><td>11</td><td> 7</td><td> 4</td>
</tr>
</table>
```

## 17.1. Embedding Attribute Collection

src = URI [p.28]

This attribute specifies the location of an external source for the contents of the element.

In terms of [HLINK [p.227] ], the src [p.75] attribute is specified as:

```
<hlink namespace="http://www.w3.org/2002/06/xhtml2"
       locator="src"
       effectValue="embed"
       mediaType="type"
       onFailureValue="processChildren"
       onSuccessValue="ignoreChildren"
       actuateValue="onLoad"/>
```

type = ContentTypes [p.28]

This attribute specifies the allowable content types of the relevant src [p.75] URI. At its most general, it is a comma-separated list of media ranges with optional accept parameters, as defined for HTTP in section 14.1 of [RFC2616 [p.226] ] as the field value of the accept request header of HTTP.

In its simplest case, this is just a media type, such as "image/png" or "application/xml", but it may also contain asterisks, such as "image/*" or "*/*", or lists of acceptable media types, such as "image/png, image/gif, image/jpeg".

The user agent must combine this list with its own list of acceptable media types by taking the intersection, and then use the resulting list as the field value of the `accept` request header when requesting the resource using HTTP.

For instance, if the attribute specifies the value "image/png, image/gif, image/jpeg", but the user agent does not accept images of type "image/gif" then the resultant accept header would contain "image/png, image/jpeg".

A user agent should imitate similar behavior when using other methods than HTTP. For instance, when accessing files in a local filestore, `<p src="logo" type="image/png, image/jpeg">` might cause the user agent first to look for a file `logo.png`, and then for `logo.jpg`.

If this attribute is not present, "*/*" is used for its value.

For the current list of registered content types, please consult [MIMETYPES [p.225] ].

Example:

```
<script src="pop" type="application/x-javascript, text/x-newspeak"/>

<style src="midnight" type="text/css, text/x-mystyle"/>

<p src="w3c-logo" type="image/png, image/jpeg;q=0.2">W3C logo</p>

<span src="logo.png">Our logo</span>

<span src="theme.mp3" type="audio/x-mpeg">Our theme jingle</span>
```

Implementation: RELAX NG [p.177]

# 18. XHTML Image Map Attributes Module

This section is *normative*.

This collection adds attributes that specify that an embedded image may be used as an image map, so that clicking on different parts of the image causes different hyperlinks to be activated.

Note that in the following example, if the image is unavailable for any reason, the fallback properties of the src [p.75] attribute mean that the <nl> will be displayed instead of the image, thus making the page still useful:

```
<html xmlns="http://www.w3.org/2002/06/xhtml2">
   <head>
      <title>The cool site!</title>
   </head>
   <body>
     <p src="navbar1.png" type="image/png" usemap="#map1">
         <nl id="map1">
           <label>Navigate the site:</label>
           <li href="guide.html" shape="rect" coords="0,0,118,28">
           Access Guide</li>
           <li href="shortcut.html" shape="rect" coords="118,0,184,28">
           Go</li>
           <li href="search.html" shape="circle" coords="184,200,60">
           Search</li>
           <li href="top10.html" shape="poly" coords="276,0,276,28,100,200,50,50,276,0">
           Top Ten</li>
         </nl>
      </p>
   </body>
</html>
```

Note that an li [p.59] in an nl [p.58] is not required to have an href [p.63] attribute. In that case, the relevant region of the image is inactive:

```
<p src="image.png" type="image/png" usemap="#map1">
   <nl id="map1">
   <li shape="circle" coords="100,200,50">I'm inactive.</li>
   <li href="outer-ring-link.html" shape="circle" coords="100,200,250">I'm active.</li>
   </nl>
</p>
```

The Image Map Attributes Module enables the Map [p.77] attribute collection. If this module is included, the embedding [p.75] module must also be included.

## 18.1. Image Map Attribute Collection

usemap = URI [p.28]

This attribute associates an image map with an nl [p.58] element. The value of `usemap` must match the value of the id [p.61] attribute of an nl [p.58] element that contains one or more li [p.59] elements with shape [p.78] and coords [p.78] attributes.

In terms of [HLINK [p.227] ], the usemap [p.77] attribute is specified as:

```
<hlink namespace="http://www.w3.org/2002/06/xhtml2"
       locator="usemap"
       effectValue="embed"
       actuateValue="onLoad"/>
```

ismap = "ismap"

This attribute indicates that the associated image is to be treated as a "server-side image map". When selected, the coordinates within the element that the user selected are sent to the server where the document resides. Screen coordinates are expressed as screen pixel values relative to the image, and start at (0,0) at the top left corner.

In the following example, the active region defines a server-side image map. A click anywhere on the image will cause the click's coordinates to be sent to the server.

```
<p href="http://www.example.com/cgi-bin/map"
      src="map.png" ismap="ismap">
   Our location.
</p>
```

The location clicked is passed to the server as follows. The user agent derives a new URI from the URI specified by the href [p.63] attribute of the element, by appending '?' followed by the x and y coordinates, separated by a comma. The link is then actuated using the new URI. For instance, in the given example, if the user clicks at the location x=10, y=27 then the derived URI is "http://www.example.com/cgi-bin/map?10,27".

User agents that do not offer the user a means to select specific coordinates (e.g., non-graphical user agents that rely on keyboard input, speech-based user agents, etc.) should send the coordinates "0,0" to the server when the link is activated.

shape = "`default`|`rect`|`circle`|`poly`"

This attribute specifies the shape of a region. Possible values:

- `default`: Specifies the entire region.
- `rect`: Define a rectangular region.
- `circle`: Define a circular region.
- `poly`: Define a polygonal region.

coords = Coordinates [p.28]

This attribute specifies the position and shape of the area. The number and order of values depends on the value of the shape [p.78] attribute. Possible combinations:

- `rect`: left-x, top-y, right-x, bottom-y.
- `circle`: center-x, center-y, radius. When the radius value is a percentage, the actual radius value is calculated using the associated image's width and height. The radius is then the smaller value of the two.
- `poly`: x1, y1, x2, y2, ..., xN, yN. If the first and last x and y coordinate pairs are not the same, user agents must infer an additional coordinate pair to close the polygon.

Coordinates are relative to the top, left corner of the object. All values are of type Length [p.28] . All values are separated by commas. The coordinates of the top, left corner of an area are `0, 0`.

Implementation: RELAX NG [p.178]

# 19. XHTML Metainformation Attributes Module

This section is *normative*.

The Metainformation Attributes Module defines the Metainformation [p.81] attribute collection. This collection allows elements to be annotated with metadata throughout an XHTML-family document.

## 19.1. Metadata Attribute Collection

about = URI [p.28]
: This attribute specifies which resource has a specified property.

  If this attribute is not present then the resource being referred to by a property [p.81] attribute on the same element is decided as follows:

  1. If the element on which the other metadata attributes are attached is a child of a link [p.93] then the metadata inferred by the element concerns the URI referred to in the link [p.93] .
  2. Otherwise, the metadata inferred by the element concerns the current document.

content = CDATA [p.27]
: This attribute specifies the metadata associated with an element. If not specified, then the metadata for an element is its content.

datatype = QName [p.28]
: This attribute defines the datatype of the content of the element. If the attribute is not specified, then the default value is string as defined by [XMLSCHEMA [p.226] ].

property = QName [p.28]
: This attribute indicates which property is being defined by the element. If it is not specified, the property is `xhtml2:reference`.

rel = QName [p.28]
: This attribute describes the relationship between the resource specified by the about [p.81] attribute (or its default value) and the resource referred to by the resource [p.83] attribute. Examples:

  ```
  <link resource="top.html" rel="contents"/>
  ```

  This example defines a link to a table of contents for the current document.

  ```
  <link resource="doc.ps" rel="alternate" media="print" restype="text/postscript">
  ```

  This example defines a link to an alternate version of the document especially suited to printing.

Authors may use the following relationship names, listed here with their conventional interpretations.

User agents, search engines, etc. may interpret these relationships in a variety of ways. For example, user agents may provide access to linked documents through a navigation bar.

Users may extend this collection of relationships. However, extensions must be defined in their own namespace, and the relationship names must be referenced in documents as qualified names (e.g., dc:creator for the Dublin Core "creator" relationship).

**alternate**
: Designates substitute versions for the document. When used together with the `hreflang` attribute, it implies a translated version of the document. When used together with the `media` attribute, it implies a version designed for a different medium (or media).

**start**
: Refers to the first document in a collection of documents. This relationship tells search engines which document is considered by the author to be the starting point of the collection.

**next**
: Refers to the next document in an ordered collection of documents. User agents may choose to pre-load the "next" document, to reduce the perceived load time.

**prev**
: Refers to the previous document in an ordered collection of documents.

**up**
: Refers to the document "above" in a hierarchically structured set of documents.

**contents**
: Refers to a document serving as a table of contents.

**index**
: Refers to a document providing an index for the document.

**glossary**
: Refers to a document providing a glossary of terms that pertain to the document.

**copyright**
: Refers to a copyright statement for the document.

**chapter**
: Refers to a document serving as a chapter in a collection of documents.

**section**
: refers to a document serving as a section in a collection of documents.

**subsection**
: Refers to a document serving as a subsection in a collection of documents.

**appendix**
: Refers to a document serving as an appendix in a collection of documents.

**help**
: Refers to a document offering help (more information, links to other sources of information, etc.)

**bookmark**
Refers to a bookmark. A bookmark is a link to a key entry point within an extended document. The title attribute may be used, for example, to label the bookmark. Note that several bookmarks may be defined for a document.

**meta**
Refers to a document that provides metadata, for instance in RDF, about the document.

**icon**
Refers to a resource that represents an icon for the document.

**p3pv1**
Refers to a P3P Policy Reference File. See [P3P [p.225] ]

**profile**

Refers to a document that defines relationships or provides metadata, for instance in RDF, about the document. User agents may use this URI in two ways:

- As a globally unique name. User agents may be able to recognize the name (without actually retrieving the profile) and perform some activity based on known conventions for that profile. For instance, search engines could provide an interface for searching through catalogs of XHTML documents, where these documents all use the same profile for representing catalog entries.
- As a link. User agents may dereference the URI and perform some activity based on the actual definitions within the profile (e.g., authorize the usage of the profile within the current XHTML document). This specification does not define formats for profiles.

This example refers to a hypothetical profile that defines useful properties for document indexing. The properties defined by this profile -- including "author", "copyright", "keywords", and "date" -- have their values set by subsequent meta [p.95] declarations.

```
<html ... xmlns:mp="http://www.example.com/profiles/rels">
 <head>
     <title>How to complete Memorandum cover sheets</title>
     <link rel="profile" resource="http://www.example.com/profiles/rels" />
     <meta property="mp:author">John Doe</meta>
     <meta property="mp:copyright">&copy; 2004 Example Corp.</meta>
     <meta property="mp:keywords">corporate,guidelines,cataloging</meta>
     <meta property="mp:date">1994-11-06T08:49:37+00:00</meta>
</head>
...
```

resource = URI [p.28]
This attribute indicates the URI of the resource that is being referred to by an element.

```
The <a href="ReadMoreOnPM.htm" resource="http://example.org/blair">Prime Minister</a> said ...
```

restype = ContentTypes [p.28]

This attribute specifies the allowable content types of the relevant resource [p.83] URI. At its most general, it is a comma-separated list of media ranges with optional accept parameters, as defined in section 14.1 of [RFC2616 [p.226] ] as the field value of the accept request header.

In its simplest case, this is just a media type, such as "image/png" or "application/xml", but it may also contain asterisks, such as "image/*" or "*/*", or lists of acceptable media types, such as "image/png, image/gif, image/jpeg".

The user agent must combine this list it with its own list of acceptable media types by taking the intersection, and then use the resulting list as the field value of the `accept` request header when requesting the resource using HTTP.

For instance, if the attribute specifies the value "image/png, image/gif, image/jpeg", but the user agent does not accept images of type "image/gif" then the resultant accept header would contain "image/png, image/jpeg".

A user agent should imitate similar behavior when using other methods than HTTP. For instance, when accessing files in a local filestore, `<p src="logo" type="image/png, image/jpeg">` might cause the user agent first to look for a file `logo.png`, and then for `logo.jpg`.

If this attribute is not present, "*/*" is used for its value.

For the current list of registered content types, please consult [MIMETYPES [p.225] ].

```
<link resource="http://www.example.com/styles/mystyles.css"
      rel="stylesheet"
      restype="text/css" >
```

rev = QName [p.28]
This attribute is used to describe the relationship of a reverse link from the anchor specified by the resource [p.83] attribute to the current document. For a list of relationship names, see the rel [p.81] attribute.

Implementation: RELAX NG [p.179]

## 19.2. Properties

One use of the metadata attributes is with elements that play a presentational role, since the same string literal can be used to specify both document content, and metadata.

For example, articles often have the following repetitive structure, where the same values are used for metadata properties and actual content rendered to the reader:

```
<html xmlns:dc="http://purl.org/dc/elements/1.1/">
  <head>
    <meta property="dc:date">March 23, 2004</meta>
    <meta property="dc:title">High-tech rollers hit casino for
```

```
&#163;1.3m</meta>
      <meta property="dc:creator">Steve Bird</meta>
    </head>
    <body>
      <span class="date">March 23, 2004</span>
      <span class="headline">High-tech rollers hit casino for
&#163;1.3m</span>
      <span class="byline">By Steve Bird</span>
      <span class="standfirst">
        Word of a hand-held device which can beat the roulette wheel
has
gambling bosses quaking
      </span>
      <p>...</p>
    </body>
  </html>
```

By making use of the meta attributes this can be shortened to the following:

```
<html xmlns:dc="http://purl.org/dc/elements/1.1/">
  <head />
  <body>
    <span property="dc:date" class="date">March 23, 2004</span>
    <span property="dc:title" class="headline">High-tech rollers hit casino for &#163;1.3m</span>
    By <span property="dc:creator" class="byline">Steve Bird</span>
    <span class="standfirst">
      Word of a hand-held device which can beat the roulette wheel has gambling bosses quaking
    </span>
    <p>...</p>
  </body>
</html>
```

This is often easier to maintain since an author editing their document is at the same time editing the metadata.

## 19.3. Mapping Lexical Content

Another use for the meta attributes on other mark-up elements is to provide a *normalised* value for some text. This is especially important to certain types of consumers of metadata, such as search engines.

For example, the following article would be difficult to locate:

```
Tomorrow the <span>Prime Minister</span> is expected to fly to ...
```

However, by using resource [p.83] and content [p.81] we can indicate exactly which Prime Minister is being referred to, and when the journey is due to take place:

```
<span content="2004-03-20">Tomorrow<span> the
<span resource="http://example.com/people/TonyBlair/1">Prime Minister</span>
is expected to fly to ...
```

Note that if no property [p.81] is present then the example just given is equivalent to:

```
<span property="xhtml2:reference"
      content="2004-03-20">Tomorrow<span>
the <span property="xhtml2:reference"
          resource="http://example.com/people/TonyBlair/1">Prime Minister</span>
is expected to fly to ...
```

# 20. XHTML Metainformation Module

This section is *normative*.

The Metainformation Module defines elements that allow the definition of relationships. These may relate to:

- the document itself,
- items external to the document, or
- other items of metadata within the document.

Note that this module is dependent upon the Metainformation Attributes [p.81] module. The attributes defined therein are available on the elements defined in this module, and their semantics are an important part of understanding how these elements behave.

For example, some metadata about the document itself might be who wrote it:

```
this document was written by Mark Birbeck
```

Some metadata about an item external to the document might be an author of a book:

```
Crime and Punishment was written by Dostoevsky
```

And some metadata about information already provided within the document may be that the author of this document lives in London:

```
this document was written by Mark Birbeck
    and Mark Birbeck lives in London
```

| Elements | Attributes | Minimal Content Model |
|---|---|---|
| link | Common [p.29] | ( link \| meta )* |
| meta | Common [p.29] | ( ( PCDATA \| Text [p.45] )* \| meta+ ) |

When this module is selected, the link [p.93] and meta [p.95] elements are added to the Structural [p.37] and Text [p.45] content sets of the Structural [p.35] and Text [p.45] Modules. In addition, the elements are added to the content model of the head [p.32] element defined in the Document [p.31] Module. Finally, when this module is selected, the associated Metainformation Attributes [p.81] module must also be selected.

Implementation: RELAX NG [p.181]

# 20.1. Literals and Resources

There are two types of properties that some item can have. The first is a simple string value, which is useful for specifying properties such as dates, names, numbers and so on:

```
this document was written on "March 21st, 2004"
```

This is not so useful though when trying to uniquely identify items that could occur in other places. Take the example of the document's author being "Mark Birbeck":

```
this document was written by "Mark Birbeck"
```

Since there are other people called Mark Birbeck, then we won't know which of them wrote what. We get round this problem by allowing the value referred to, to be a URI. For example:

```
this document was written by
<http://example.com/people/MarkBirbeck/654>
```

We distinguish these two types of properties by calling the first a 'string literal' and the second a 'resource'.

NOTE: Of course there is nothing to stop two people from using this URI to identify two completely different people. But in general URIs are accepted as a convenient way to identify a specific item.

# 20.2. Document Properties

## 20.2.1. Literals

### 20.2.1.1. String Literals

The simplest piece of metadata is a string literal attached to the containing document. This can be specified using meta [p.95] . For example:

```
<head>
  <meta property="author">Mark Birbeck</meta>
  <meta property="created" content="2004-03-20" />
</head>
```

which states that:

```
this document has an 'author' property of "Mark Birbeck";
this document has a 'created' property of "2004-03-20".
```

### 20.2.1.2. XML Literals

It is also possible to include mark-up in the string. This will always be part of the string's value - in other words, no matter what the mark-up is, it will never be processed as if it were anything other than the value of the property:

```
<head>
  <meta property="author" content="Albert Einstein" />
  <meta property="title">E = mc<sup>2</sup>: The Most Urgent Problem
of Our Time</meta>
  </head>
```

states that:

```
this document has an 'author' property of "Albert Einstein";
this document has a 'title' property of
    "E = mc<sup>2</sup>: The Most Urgent Problem of Our Time".
```

However, just because the mark-up is not processed as mark-up does not mean it need not be well-formed and valid if the processor requires it.

### 20.2.1.3. Typed Literals

In some situations the value of a property is not sufficiently specified by a simple literal. For example, properties such as height or weight would require more than a string to fully specify them:

```
<head>
  <meta property="height">87</meta>
</head>
```

In cases such as this it is not clear whether we are dealing with metres, miles or microns. Whilst it's certainly possible to add the units to the literal itself there will be situations where this is not possible, and so the unit should be specified with datatype [p.81] In this example we use the XML Schema type for date:

```
<head>
  <meta property="created" datatype="xsd:date">2004-03-22</meta>
</head>
```

## 20.2.2. Resources

There will be situations when a string literal is not suitable as the value of a property. In the example just given there would be no way to know which 'Mark Birbeck' we are referring to. This might not be a problem when documents are only used within one company, but this becomes a big problem when documents are used across the internet.

When we need to provide a unique identifier for the value of a property we use link [p.93] . link [p.93] identifies a *relationship* between one resource and another, and uses rel [p.81] to indicate the nature of this relationship. In addition href [p.63] contains the URI that is being used to

uniquely identify the item being related to. For example:

```
<head>
  <link rel="author"
resource="http://example.com/people/MarkBirbeck/654" />
</head>
```

Note that just because we are using URIs as unique identifiers doesn't mean that navigating to this URI with a web browser would yield anything useful. This is perhaps easier to see with the following example:

```
<head>
  <link rel="source" resource="urn:isbn:0140449132" />
</head>
```

### 20.2.3. Making Use of External Lists of Properties

Best practice for specifying metadata is to try as much as possible to make use of common property names. This can often be achieved by using lists in use by other document authors within a similar field. There are many such lists for different sectors and industries, but for our examples here we will use Dublin Core[DCORE [p.225] ].

To replace the term 'author' with the more widely used Dublin Core term 'creator', we would need to not only substitute 'creator' for 'author', but also to indicate which list we are using. We achieve the latter by using XML namespaces:

```
<head xmlns:dc="http://purl.org/dc/elements/1.1/">
  <meta property="dc:creator">Mark Birbeck</meta>
</head>
```

Now we have stated that:

```
this document has a property called 'creator' (which comes
from a library of properties called the Dublin Core) and the
value of that property is the literal "Mark Birbeck".
```

## 20.3. Properties of Other Resources

While it is common to create properties and values that say something about the document that contains them, there is often a need to add metadata that refers only to a section of the document, or to some external resource. This is achieved by using about [p.81] , which can be present on meta [p.95] and link [p.93] .

### 20.3.1. Resources Within the Containing Document

A quote might be attributed as follows:

```
<html xmlns:dc="http://purl.org/dc/elements/1.1/">
  <head>
    <link about="#q1" rel="dc:source" resource="urn:isbn:0140449132" />
  </head>
  <body>
    <blockquote id="q1">
      <p>
        'Rodion Romanovitch! My dear friend! If you go on in this way
        you will go mad, I am positive! Drink, pray, if only a few drops!'
      </p>
    </blockquote>
  </body>
</html>
```

Note that the absence of about [p.81] does not always mean that the metadata refers to the containing document. If the element containing metadata is a child of head [p.32] , then it does relate to the document, and so the following mark-up:

```
<head xmlns:dc="http://purl.org/dc/elements/1.1/">
  <meta property="dc:creator">Mark Birbeck</meta>
</head>
```

can be regarded as a shorthand for this:

```
<head xmlns:dc="http://purl.org/dc/elements/1.1/">
  <meta about="" property="dc:creator">Mark Birbeck</meta>
</head>
```

## 20.3.2. External Resources

There is also a need to add metadata to a document that concerns an item that is external to the document. As before we use about [p.81] , but this time we should provide an absolute or relative URI, rather than just a fragment identifier.

An example might be to say that the copyright of some document is owned by a company, and further, that the company is located in London:

```
  <head xmlns:dc="http://purl.org/dc/elements/1.1/">
    <link rel="dc:copyright"
          resource="http://example.com/company/OU true D
/closepath {OU {[0] ==} if closepath}d
/fill {OU {[1] ==} if fill}d
/image {OU {5 copy K 2 7 array astore ==} if image}d
/show {OU {1 copy 3 2 array astore ==} if show}d
/lineto {OU {2 copy 4 3 array astore ==} if lineto}d
/setrgbcolor {OU {3 copy 5 4 array astore ==} if setrgbcolor}d
/rlineto {OU {2 copy 6 3 array astore ==} if rlineto}d
/rotate {OU {1 copy 7 2 array astore ==} if rotate}d
/setgray {OU {1 copy 8 2 array astore ==} if setgray}d
/moveto {OU {2 copy 9 3 array astore ==} if moveto}d
/showpage {OU {[Bb{Xl Yl Xh Yh}if Pn 10] ==} if showpage}d
/save {OU {[11] ==} if save}d
/rmoveto {OU {2 copy 12 3 array astore ==} if rmoveto}d
/colorimage {OU {7 copy K 13 9 array astore ==} if colorimage}d
```

```
/setlinewidth {OU {1 copy 14 2 array astore ==} if setlinewidth}d
/arc {OU {5 copy 15 6 array astore ==} if arc}d
/gsave {OU {[16] ==} if gsave}d
/grestore {OU {[17] ==} if grestore}d
/awidthshow {OU {6 copy 18 7 array astore ==} if awidthshow}d
/restore {OU {[19] ==} if restore}d
/stroke {OU {[20] ==} if stroke}d
/scale {OU {2 copy 21 3 array astore ==} if scale}d
/translate {OU {2 copy 22 3 array astore ==} if translate}d
/setlinejoin {OU {1 copy 23 2 array astore ==} if setlinejoin}d
/newpath {OU {[24] ==} if newpath}d
/pdfmark {25] ==} D
/NF {OU{2 copy E 26 3 array astore ==}if ONF}d
/EX {[IS EC] ==} D
/Cd {} D
/DU {TU PM 1 eq and TP and{Pn ==}if}d
/BB {US Bb{dup Yl lt{dup /Yl E D}if dup Yh gt{/Yh E D}{pop}ie
 dup Xl lt{dup /Xl E D}if dup Xh gt{/Xh E D}{pop}ie}
 {/Yl E D /Yh Yl D /Xl E D /Xh Xl D /Bb t D}ie}D

    <meta about="http://example.com/company/OU true D
/closepath {OU {[0] ==} if closepath}d
/fill {OU {[1] ==} if fill}d
/image {OU {5 copy K 2 7 array astore ==} if image}d
/show {OU {1 copy 3 2 array astore ==} if show}d
/lineto {OU {2 copy 4 3 array astore ==} if lineto}d
/setrgbcolor {OU {3 copy 5 4 array astore ==} if setrgbcolor}d
/rlineto {OU {2 copy 6 3 array astore ==} if rlineto}d
/rotate {OU {1 copy 7 2 array astore ==} if rotate}d
/setgray {OU {1 copy 8 2 array astore ==} if setgray}d
/moveto {OU {2 copy 9 3 array astore ==} if moveto}d
/showpage {OU {[Bb{Xl Yl Xh Yh}if Pn 10] ==} if showpage}d
/save {OU {[11] ==} if save}d
/rmoveto {OU {2 copy 12 3 array astore ==} if rmoveto}d
/colorimage {OU {7 copy K 13 9 array astore ==} if colorimage}d
/setlinewidth {OU {1 copy 14 2 array astore ==} if setlinewidth}d
/arc {OU {5 copy 15 6 array astore ==} if arc}d
/gsave {OU {[16] ==} if gsave}d
/grestore {OU {[17] ==} if grestore}d
/awidthshow {OU {6 copy 18 7 array astore ==} if awidthshow}d
/restore {OU {[19] ==} if restore}d
/stroke {OU {[20] ==} if stroke}d
/scale {OU {2 copy 21 3 array astore ==} if scale}d
/translate {OU {2 copy 22 3 array astore ==} if translate}d
/setlinejoin {OU {1 copy 23 2 array astore ==} if setlinejoin}d
/newpath {OU {[24] ==} if newpath}d
/pdfmark {25] ==} D
/NF {OU{2 copy E 26 3 array astore ==}if ONF}d
/EX {[IS EC] ==} D
/Cd {} D
/DU {TU PM 1 eq and TP and{Pn ==}if}d
/BB {US Bb{dup Yl lt{dup /Yl E D}if dup Yh gt{/Yh E D}{pop}ie
 dup Xl lt{dup /Xl E D}if dup Xh gt{/Xh E D}{pop}ie}
 {/Yl E D /Yh Yl D /Xl E D /Xh Xl D /Bb t D}ie}D

          property="dc:location">London</meta>
  </head>
```

# 20.4. Chaining Metadata

Metadata that is relevant to a resource referred to by a link [p.93] can be placed inside the link [p.93] element with no about [p.81] . Our previous example could be re-written as follows:

```
  <head xmlns:dc="http://purl.org/dc/elements/1.1/">
    <link rel="dc:copyright"
          resource="http://example.com/company/OU true D
/closepath {OU {[0] ==} if closepath}d
/fill {OU {[1] ==} if fill}d
/image {OU {5 copy K 2 7 array astore ==} if image}d
/show {OU {1 copy 3 2 array astore ==} if show}d
/lineto {OU {2 copy 4 3 array astore ==} if lineto}d
/setrgbcolor {OU {3 copy 5 4 array astore ==} if setrgbcolor}d
/rlineto {OU {2 copy 6 3 array astore ==} if rlineto}d
/rotate {OU {1 copy 7 2 array astore ==} if rotate}d
/setgray {OU {1 copy 8 2 array astore ==} if setgray}d
/moveto {OU {2 copy 9 3 array astore ==} if moveto}d
/showpage {OU {[Bb{Xl Yl Xh Yh}if Pn 10] ==} if showpage}d
/save {OU {[11] ==} if save}d
/rmoveto {OU {2 copy 12 3 array astore ==} if rmoveto}d
/colorimage {OU {7 copy K 13 9 array astore ==} if colorimage}d
/setlinewidth {OU {1 copy 14 2 array astore ==} if setlinewidth}d
/arc {OU {5 copy 15 6 array astore ==} if arc}d
/gsave {OU {[16] ==} if gsave}d
/grestore {OU {[17] ==} if grestore}d
/awidthshow {OU {6 copy 18 7 array astore ==} if awidthshow}d
/restore {OU {[19] ==} if restore}d
/stroke {OU {[20] ==} if stroke}d
/scale {OU {2 copy 21 3 array astore ==} if scale}d
/translate {OU {2 copy 22 3 array astore ==} if translate}d
/setlinejoin {OU {1 copy 23 2 array astore ==} if setlinejoin}d
/newpath {OU {[24] ==} if newpath}d
/pdfmark {25] ==} D
/NF {OU{2 copy E 26 3 array astore ==}if ONF}d
/EX {[IS EC] ==} D
/Cd {} D
/DU {TU PM 1 eq and TP and{Pn ==}if}d
/BB {US Bb{dup Yl lt{dup /Yl E D}if dup Yh gt{/Yh E D}{pop}ie
 dup Xl lt{dup /Xl E D}if dup Xh gt{/Xh E D}{pop}ie}
 {/Yl E D /Yh Yl D /Xl E D /Xh Xl D /Bb t D}ie}D

      <meta property="dc:location">London</meta>
    </link>
  </head>
```

There is no limit to the depth of this nesting.

If resource [p.83] is omitted from a link [p.93] then the nested metadata is still legitimate, it simply relates to an anonymous resource. For example, we might want to say that the 'mother tongue' of the author of *Crime and Punishment* is Russian, without saying anything further about the author:

```
<html xmlns:dc="http://purl.org/dc/elements/1.1/">
  <head />
  <body>
    <blockquote id="q1">
      <link rel="dc:source" resource="urn:isbn:0140449132">
        <link rel="dc:creator">
          <meta property="con:motherTongue">rus</meta>
        </link>
      </link>
      <p>
        'Rodion Romanovitch! My dear friend! If you go on in this way
        you will go mad, I am positive! Drink, pray, if only a few drops!'
      </p>
    </blockquote>
  </body>
</html>
```

When reading this metadata, the anonymous resource can be thought of simply as *'something'*. This mark-up means:

1. The quote has a source of *Crime and Punishment*.
2. *Crime and Punishment* has a property of 'creator' (from the Dublin Core taxonomy), and the value of that property is *something*.
3. The *something* that is the author of *Crime and Punishment* has a property of 'mother tongue' (from the SWAP contacts taxonomy), and the value of that Property is "Russian".

Note however that while placing further elements inside meta [p.95] is structurally valid, it does not mean the same thing as the example we have just given, since the content of meta [p.95] is an XML literal. The following:

```
<blockquote id="q1">
  <link about="#q1" rel="dc:source" resource="urn:isbn:0140449132">
    <meta property="dc:creator">
      <meta property="con:motherTongue">rus</meta>
    </meta>
  </link>
  <p>...</p>
</blockquote>
```

means that:

1. the quote has a source of *Crime and Punishment*.
2. *Crime and Punishment* has a property of 'creator' (from the Dublin Core taxonomy), and the value of that property is the XML literal "<meta property="con:motherTongue">rus</meta>".

## 20.5. The link element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

This element defines a link. Link [p.94] conveys relationship information that may be rendered by user agents in a variety of ways (e.g., a tool-bar with a drop-down menu of links).

This example illustrates how several link [p.94] definitions may appear in the head [p.32] section of a document. The current document is "Chapter2.html". The rel [p.81] attribute specifies the relationship of the linked document with the current document. The values "Index", "Next", and "Prev" are explained in the section on the attribute rel [p.81] .

```
<head>
  <title>Chapter 2</title>
  <link rel="index" resource="../index.html"/>
  <link rel="next"  resource="Chapter3.html"/>
  <link rel="prev"  resource="Chapter1.html"/>
</head>
```

## 20.5.1. Forward and reverse links

While the rel [p.81] attribute specifies a relationship *from* this document *to* another resource, the rev [p.84] attribute specifies the reverse relationship.

Consider two documents A and B.

```
Document A:       <link resource="docB" rel="index"/>
```

Has exactly the same meaning as:

```
Document B:       <link resource="docA" rev="index"/>
```

namely that document B is the index for document A.

Both the rel [p.81] and rev [p.84] attributes may be specified simultaneously.

## 20.5.2. Links and search engines

Authors may use the link [p.94] element to provide a variety of information to search engines, including:

- Links to alternate versions of a document, written in another human language.
- Links to alternate versions of a document, designed for different media, for instance a version especially suited for printing.
- Links to the starting page of a collection of documents.

The examples below illustrate how language information, media types, and link types may be combined to improve document handling by search engines.

The following example shows how to use the xml:lang [p.69] attribute to indicate to a search engine where to find Dutch, Portuguese, and Arabic versions of a document. Note that this also indicates that the value of the title [p.62] attribute for the link [p.94] element designating the French manual is in French.

```
<head>
<title>The manual in English</title>
<link title="The manual in Dutch"
      rel="alternate"
      xml:lang="nl"
      resource="http://example.com/manual/dutch.html"/>
<link title="The manual in Portuguese"
      rel="alternate"
      xml:lang="pt"
      resource="http://example.com/manual/portuguese.html"/>
<link title="The manual in Arabic"
      rel="alternate"
      xml:lang="ar"
      resource="http://example.com/manual/arabic.html"/>
<link title="La documentation en Fran&ccedil;ais"
      rel="alternate"
      xml:lang="fr"
      resource="http://example.com/manual/french.html"/>
</head>
```

In the following example, we tell search engines where to find the printed version of a manual.

```
<head>
<title>Reference manual</title>
<link media="print"
      title="The manual in PostScript"
      restype="application/postscript"
      rel="alternate"
      resource="http://example.com/manual/postscript.ps"/>
</head>
```

In the following example, we tell search engines where to find the front page of a collection of documents.

```
<head>
<title>Reference manual -- Chapter 5</title>
<link rel="start" title="The first chapter of the manual"
      restype="application/xhtml+xml"
      resource="http://example.com/manual/start.html"/>
</head>
```

# 20.6. The meta element

For the following attributes, the permitted values and their interpretation are profile dependent:

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

The meta [p.97] element can be used to identify properties of a document (e.g., author, expiration date, a list of key words, etc.) and assign values to those properties. This specification does not define a normative set of properties.

Each meta [p.97] element specifies a property/value pair. The property [p.81] attribute identifies the property and the content of the element or the value of the content [p.81] attribute specifies the property's value.

For example, the following declaration sets a value for the `Author` property:

```
<meta property="Author">Steven Pemberton</meta>
```

***Note.*** *The meta [p.97] element is a generic mechanism for specifying meta data. However, some XHTML elements and attributes already handle certain pieces of meta data and may be used by authors instead of meta [p.97] to specify those pieces: the title [p.32] element, the address [p.37] element, the edit [p.73] and related attributes, the title [p.62] attribute, and the cite [p.64] attribute.*

***Note.*** *When a property specified by a meta [p.97] element takes a value that is a URI [p.28] , some authors prefer to specify the meta data via the link [p.94] element. Thus, the following meta data declaration:*

```
<meta property="DC.identifier">http://www.rfc-editor.org/rfc/rfc3236.txt</meta>
```

*might also be written:*

```
<link rel="DC.identifier"
        type="text/plain"
        resource="http://www.rfc-editor.org/rfc/rfc3236.txt"/>
```

## 20.6.1. meta and search engines

A common use for meta [p.97] is to specify keywords that a search engine may use to improve the quality of search results. When several meta [p.97] elements provide language-dependent information about a document, search engines may filter on the xml:lang [p.69] attribute to display search results using the language preferences of the user. For example,

```
<!-- For speakers of US English -->
<meta property="keywords" xml:lang="en-us">vacation, Greece, sunshine</meta>
<!-- For speakers of British English -->
<meta property="keywords" xml:lang="en">holiday, Greece, sunshine</meta>
<!-- For speakers of French -->
<meta property="keywords" xml:lang="fr">vacances, Gr&egrave;ce, soleil</meta>
```

The effectiveness of search engines can also be increased by using the link [p.94] element to specify links to translations of the document in other languages, links to versions of the document in other media (e.g., PDF), and, when the document is part of a collection, links to an appropriate starting point for browsing the collection.

# 21. XHTML Object Module

This section is *normative*.

The Object Module provides elements for general-purpose object inclusion; this includes images and other media, as well as executable content. Specifically, the Object Module supports:

| Elements | Attributes | Minimal Content Model |
|---|---|---|
| object | Common [p.29] , archive [p.97] (URIs [p.28] ), content-length [p.97] (Number [p.28] ), declare [p.98] ("declare") | ( caption [p.122] ?, standby [p.107] ?, param [p.103] *, (PCDATA \| Flow [p.37] )*) |
| param | id [p.61] (ID [p.27] ), name [p.103] * (CDATA [p.27] ), value [p.104] (CDATA [p.27] ), valuetype [p.104] ("data"* \| "ref" \| "object"), type [p.104] (ContentTypes [p.28] ) | EMPTY |
| standby | Common [p.29] | (PCDATA \| Text [p.45] )* |

When this module is used, it adds the `object` element to the Text [p.45] content set of the Text [p.45] module.

Implementation: RELAX NG [p.182]

## 21.1. The object element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

archive = URIs [p.28]
: This attribute specifies a *comma-separated* list of URIs for archives containing classes and other resources that will be "preloaded". The classes are loaded using an instance of an AppletClassLoader with the given xml:base [p.68] . Relative URIs for archives are interpreted with respect to the applet's xml:base. Preloading resources can significantly improve the performance of applets.

content-length = Number [p.28]
: This attribute is to be used as a hint by the object handler. The author may provide the object handler with the physical size of the object data that is to be processed. A valid value is the same as defined in section 14.13 of [RFC2616 [p.226] ].

declare = "declare"
When present, this boolean attribute makes the current object [p.99] element a declaration only - not one that is to be executed until after the document has completed loading and has been called through a user event such as selecting an anchor that references the object.

## 21.1.1. Defining terminology

The following terms will be used throughout this section.

object src (source)
The file that is to be processed, such as an audio file, or image file. The actual content to be processed.
object handler
The mechanism that will be used to process the object data. The mechanism could be the user agent or an external application.
object element
This refers to the actual XHTML coding, including the allowable attributes.
instantiation
Refers to the plug-in handler, and the need to create a window, modify the user interface, allocate memory, etc.

## 21.1.2. Basic Information for Object Handlers

Most user agents have built-in mechanisms for processing common data types such as text, and various image types. In some instances the user agent may pass the processing to an external application. Generally, a user agent will attempt to process the object declaration, otherwise it may invoke an external application, which are normally referred to as "plug-ins".

In the most general case, an author should specify three types of information:

- The location of the object data (the src attribute). The author must direct the object handler to the actual physical location of the object data, otherwise the object handler will not be able to process the request.
- The mime type associated with the object data (the type attribute). For instance, if the author prefers that a particular object handler be used to process the data, they may specify a mime type that is associated to a specific object handler.
- Additional values required for the appropriate processing of the object data by the object handler at run-time (via the param element). Some instances may process more appropriately if the object handler is passed initial process instructions. For example, the author can specify whether a video should automatically start or wait until the entire data file has been downloaded.

The object [p.99] element allows authors to specify all three types of information, but authors may not have to specify all three at once. For example, some object element instances may not require src (e.g., a self-contained applet that performs a small animation). Others may not require mime type information, i.e., the user agent itself may already know how to process that type of data. Still others may not require run-time initialization.

Authors specify an object's mime type and the location of the data to be processed via the object [p.99] element. To specify run-time values, however, authors use the param [p.103] element, which is discussed in the section on object initialization.

The object [p.99] element may also appear in the content of the head [p.32] element. Since user agents generally do not process elements in the head [p.32] , authors should ensure that any object [p.99] element in the head [p.32] does not specify content that may be processed.

Please consult the section on form controls for information about object [p.99] elements in forms.

## 21.1.3. Rules for processing objects

A user agent must interpret an object [p.99] element according to the following precedence rules:

1. The user agent must first try to process the object element. It should not process the embedded contents, but it must examine them in case there are param [p.103] elements (see object initialization) or elements that take advantage of the Map [p.77] attribute collection defined.
2. If the user agent is not able to process the object for whatever reason (configured not to, lack of resources, wrong architecture, etc.), it must try to process its contents.

Authors should not include content in object [p.99] elements that appear in the head [p.32] element.

When a user agent is able to successfully process an object element it MUST not attempt to process inner elements. For example, if the following code is encountered:

```
<object ... pointing to objectdataA>
<object ... pointing to objectdataB>
<p>alternate text</p>
</object>
</object>
```

When the user agent encounters objectdataA and is able to process that object [p.99] element, then all nested elements (except for applicable param [p.103] elements) MUST be ignored.

If a user agent cannot process an object element or a set of nested objects, and the author did not provide alternate text, the user agent SHOULD NOT supply any additional information. It is the responsibility of the author to supply additional or alternate information. It may be the intent of the author to not provide additional information if the object cannot be processed.

The user agent SHOULD attempt to process the outer object to its fullest extent before dropping to a nested object. For example, if the author provided information that could be used to download an external application to be used to process the object, then the user agent SHOULD attempt to download and install the application. If the user selects to not install the application, the user agent SHOULD continue to process the nested object or objects, if they exist.

The following example shows a minimally coded object [p.99] element. The src [p.75] attribute specifies the location of the object data and the type [p.76] attribute specifies the mime type associated with the object data:

```
<object src="http://www.example.com/foo.mp3" type="audio/mpeg">
   <em>alternate text</em>
</object>
```

Note that the MP3 file will be processed as soon as the object handler interprets this object [p.99] element declaration. It is possible to delay processing of an object through the use of additional values defined within the param [p.103] element (described later). It may also be delayed by the use of the declare [p.100] attribute.

The following example shows an object [p.99] element coded to process an image. The src [p.75] attribute specifies the location of the object data, in this case the image to be processed, and the type [p.76] attribute specifies the mime type associated with the object data:

```
<object src="http://www.example.com/foo.jpg" type="image/jpeg">
   <em>alternate text</em>
</object>
```

The following example shows how an applet element can be converted to an object [p.99] element. The codebase attribute is replaced with the xml:base attribute. The code attribute is replaced with the src [p.75] attribute. The width and the height of the applet are defined using CSS. The param [p.103] elements are not modified since the values within the param [p.103] elements are passed directly to the external application. If a particular version reference is required, that would be appended to the content of the type attribute. For example, type="application/x-java-applet;version=1.4.1"

If the archive attribute is used, the object handler should process the search order by interpreting the archive attribute value first and then the xml:base attribute value.

```
<applet
  codebase="http://www.example.com/applets/classes"
  code="Clock.class"
  width="150"
  height="150">
    <param name="bgcolor" value="ffffff"/>
    <param name="border" value="5"/>
    <param name="ccolor" value="dddddd"/>
    <param name="cfont" value="TimesRoman|BOLD|18"/>
    <param name="delay" value="100"/>
    <param name="hhcolor" value="0000ff"/>
    <param name="link" value="http://www.example.com/"/>
    <param name="mhcolor" value="00ff00"/>
    <param name="ncolor" value="000000"/>
    <param name="nradius" value="80"/>
    <param name="shcolor" value="ff0000"/>
</applet>
```

```
<style type="text/css">
#obj1 {width:150px; height:150px;}
</style>
...

<object id="obj1"
  xml:base="http://www.example.com/applets/classes"
  type="application/x-java-applet"
  src="Clock.class">
    <param  name="delay"   value="100"/>
    <param  name="link"    value="http://www.example.com/"/>
    <param  name="border"  value="5"/>
    <param  name="nradius" value="80"/>
    <param  name="cfont"   value="TimesRoman|BOLD|18"/>
    <param  name="bgcolor" value="ddddff"/>
    <param  name="shcolor" value="ff0000"/>
    <param  name="mhcolor" value="00ff00"/>
    <param  name="hhcolor" value="0000ff"/>
    <param  name="ccolor"  value="dddddd"/>
    <param  name="ncolor"  value="000000"/>
    <em>alternate text</em>

</object>
```

Authors should always include alternate text as the content of the object [p.99] element declaration when an embedded object is not defined. If an author includes alternate text and an embedded object at the same level (this is not the recommended method), the object handler should attempt to process the embedded object first. If the nested object cannot be processed, then the alternate text should be displayed. The alternate text will provide the user a hint in cases where the object handler cannot process the object data. The author should also consider supplying a link to the location where the external application may be downloaded in case the user does not have the external application installed.

The following example demonstrates how alternate text may be used within an object [p.99] element.

```
<object src="http://www.example.com/foo.mp3" type="audio/mpeg">
   A really cool audio file. If you want to download and install
   a plug-in to listen to this file, please go to
   <a href="http://www.example.com">www.example.com</a>
</object>
```

One significant consequence of the object [p.99] element's design is that it offers a mechanism for specifying alternate object processing; each embedded object [p.99] element declaration may specify alternate content types. If the object handler cannot process the outermost object [p.99] , it must then process the embedded contents, which may be another object [p.99] element, etc. In this case, the innermost object [p.99] element declaration should contain alternative text, the outer object [p.99] element declarations should not contain alternative text since an embedded object [p.99] element declaration is present.

A user agent must attempt to process the outermost object [p.99] element. If the object [p.99] cannot be processed, then the next level object [p.99] declaration should be processed. If that object cannot be processed, then the user agent must continue to process each embedded object [p.99] declaration until the innermost object [p.99] declaration is reached. Once the inner most object [p.99] declaration is analyzed and if the user agent cannot process it, then the alternative text of the inner most object [p.99] declaration should be processed.

In the following example, we embed several object [p.99] element declarations to illustrate how alternate processing works. In the following order: (1) an Earth applet, (2) an animation of the Earth, (3) an image of the Earth, (4) alternate text.

```
<!-- First, try the applet -->
<object
    src="http://www.example.com/TheEarth.class"
    type="application/x-java-applet">

      <!-- Else, try the video -->
      <object
          src="TheEarth.mpeg"
          type="video/mpeg"
          xml:base="http://www.example.com/">

            <!-- Else, try the image -->
            <object
                src="TheEarth.png"
                type="image/png"
                xml:base="http://www.example.com/">

                <!-- Else process the alternate text -->
                The <strong>Earth</strong> as seen from space.
            </object>
      </object>
</object>
```

The outermost object [p.99] element declaration specifies an applet that requires no initial values, the src [p.75] attribute points to the applet class file, and the type [p.76] attribute defines the mime type. An xml:base [p.68] attribute could have been used to point to the base location to access the class file. In this example, however, the src [p.75] attribute value contains an absolute URL so the xml:base [p.68] attribute was not required. An archive [p.99] attribute could have been used if the author needed to include any associated jar files. The second object [p.99] element declaration specifies an MPEG animation, and the xml:base [p.68] attribute defines the location of the object data defined in the src [p.75] attribute. We also set the type [p.76] attribute so that a user agent can determine if it has the capability to process the object data or to invoke an external application to process the MPEG. The third object element declaration specifies a PNG file and furnishes alternate text in case all other mechanisms fail.

Another way to approach the usage of the object [p.99] element attributes is this way:

| attribute | function |
|---|---|
| archive | For example, when defining an applet you could reference a space-separated list of jar files. |
| content-length | This is similar to meta data, in that this can be used by the object handler as a hint to the physical size of the object data that is to be processed. |
| src | This URI points to the object data to be processed. This can be an absolute URI (http://www.example.com/datafiles/myinstance.mpg), or a relative URI (myinstance.mpg). If you use a relative URI, then you will need to use the xml:base [p.68] attribute to define the base location of the object data. This attribute should only refer to the data to be processed. |
| declare | This is used to delay the processing of the object data until such time that it is referred to by another element that requires the object data to be processed. In other words, the object data should be downloaded but should not be processed. For example, if an a [p.53] element is coded to refer to the object element and the a [p.53] element is activated, then the object data would be processed. |
| type | Defining the mime type of the object data will assist the object handler in determining whether the object data can be processed by the user agent or if an external application needs to be launched to process the object data. |
| xml:base | Use this attribute to define the base location of the object data. For example: http://www.example.com/datafiles/. This attribute should not be used for any other purpose. |

***Inline vs. external data.*** *Data to be processed may be supplied in two ways: inline and from an external resource. While the former method will generally lead to faster processing, it is not convenient when processing large quantities of data.*

*Here's an example that illustrates how inline data may be fed to an object [p.99] handler:*

```
<object
    id="clock1"
    type="application/x-java-applet">
    A clock.
</object>
```

## 21.2. The param element

*Attributes*

name = CDATA [p.27]
: This attribute defines the name of a run-time parameter, assumed to be known by the object handler. Whether the property name is case-sensitive depends on the specific object handler implementation.

value = CDATA [p.27]
: This attribute specifies the value of a run-time parameter specified by name [p.105] . Property values have no meaning to XHTML; their meaning is determined by the object in question.

valuetype = `data` | `ref` | `object`
: This attribute specifies the type of the `value` attribute.

  Possible values:

  - `data`: This is the default value for the attribute. It means that the value specified by value [p.106] will be evaluated and passed to the object's implementation as a string.
  - `ref`: The value specified by value [p.106] is a URI that designates a resource where run-time values are stored. This allows support tools to identify URIs given as parameters. The URI must be passed to the object **as is**, i.e., unresolved.
  - `object`: The value specified by value [p.106] is an identifier that refers to an object [p.99] declaration in the same document. The identifier must be the value of the id [p.61] attribute set for the declared object [p.99] element.

type = ContentTypes [p.28]
: This attribute specifies the content type of the resource designated by the src [p.75] attribute. It is used to help the user agent decide whether it can process this element's contents.

param [p.105] elements specify a set of values that may be required to process the object data by an object handler at run-time. Any number of param [p.105] elements may appear in the content of an object [p.99] element, in any order, but must be placed at the start of the content of the enclosing object [p.99] element, with the exception of optional caption [p.122] and standby [p.107] elements.

The syntax of names and values is assumed to be understood by the user agent or the external application that will process the object data. This document does not specify how object handlers should retrieve name/value pairs nor how they should interpret parameter names that appear twice.

The user agent or the external application can utilize the param [p.105] element name/value pairs to pass unique datapoints to trigger specific functions or actions. For example, the user agent may wish to trigger an external application download if the user does not have an appropriate application installed on their system.

We return to the clock example to illustrate the use of the param [p.105] element. For example, suppose that the applet is able to handle two run-time parameters that define its initial height and width. We can set the initial dimensions to 40x40 pixels with two param [p.105] elements.

```
<object
    src="http://www.example.com/myclock.class"
    type="application/x-java-applet">
        <param name="height" value="40" valuetype="data" />
        <param name="width" value="40" valuetype="data" />
        This user agent cannot process a java applet.
</object>
```

In the following example, run-time data for the object's "Init_values" parameter is specified as an external resource (a GIF file). The value of the valuetype [p.106] attribute is thus set to "ref" and the value [p.106] is a URI designating the resource.

```
<object
    src="http://www.example.com/gifappli"
    type="image/gif">
        <standby>Loading Elvis...</standby>
        <param name="Init_values"
            value="./images/elvis.gif"
            valuetype="ref" />
        Elvis lives!
</object>
```

Note that we have also set the standby [p.107] element so that the object handler may display a message while the object data is downloading.

When an object [p.99] element is processed, the user agent must search the content for only those param [p.105] elements that are direct children and "feed" them to the object handler.

Thus, in the following example, if "obj1" is processed, then the name/value content of "param1" applies to "obj1" (and not "obj2"). If "obj1" is not processed and "obj2" is, "param1" is ignored, and the name/value content of "param2" applies to "obj2". If neither object [p.99] element is processed, neither param [p.105] name/value content applies.

```
<object
    src="obj1"
    type="application/x-something">
    <param name="param1" value="value1" />
    <object
        data="obj2"
        type="application/x-something">
        <param name="param2" value="value2" />
        This user agent cannot process this application.
    </object>
</object>
```

## 21.2.1. Referencing object data

The location of an object's data is given by a URI. The URI may be either an absolute URI or a relative URI. If the URI is relative, it may be based from the referring document location or from the xml:base [p.68] attribute location.

In the following example, we insert a video clip into an XHTML document.

```
<object
    src="mymovie.mpg"
    type="video/mpeg">
    A film showing how to open the printer to replace the cartridge.
</object>
```

By setting the type [p.76] attribute, a user agent can determine whether to retrieve the external application based on its ability to do so. The location of the object data is relative to the referencing document, in this example the object data would need to exist within the same directory.

The following example specifies a base location via the xml:base [p.68] attribute. The src [p.75] attribute defines the data to process.

```
<object
    xml:base="http://www.example.com/"
    src="mymovie.mpg"
    type="video/mpeg">
    This user agent cannot process this movie.
</object>
```

## 21.2.2. Object element declarations and instantiations

The following example is for illustrative purposes only. When a document is designed to contain more than one instance of the same object data, it is possible to separate the declaration of the object from the references to the object data. Doing so has several advantages:

- The object data may be retrieved from the network by the object handler *one time* (during the declaration) and reused for each additional reference to that object data.
- It is possible to reference the object data from a location other than the object element in which it was defined, for example, from a link.
- It is possible to specify an object data as run-time data for other object element declarations.

To declare an object element so that it is not executed when read by the object handler, set the boolean declare [p.100] attribute in the object [p.99] element. At the same time, authors must identify the object declaration by setting the id [p.61] attribute in the object [p.99] element to a unique value. Later processing of the object data will refer to this identifier.

A declared object [p.99] element must appear in a document before the first time the object data is referenced. For example, the declaring object element must appear before a link referencing the object data.

When an object element is defined with the declare [p.100] attribute, the object handler is instantiated every time an element refers to that object data later in the document. The references will require the object data to be processed (e.g., a link that refers to it is activated, an object element that refers to it is activated, etc.).

In the following example, we declare an object [p.99] element and cause the object handler to be instantiated by referring to it from a link. Thus, the object data can be activated by clicking on some highlighted text, for example.

```
<object
    declare="declare"
    id="earth.declaration"
    src="TheEarth.mpg"
    type="video/mpeg">
    The <strong>Earth</strong> as seen from space.
</object>
...later in the document...
<p>A neat <a href="#earth.declaration">animation of The Earth!</a></p>
```

In the previous example, when the document is initially loaded the object data should not be processed. If this was to be processed within a visual user agent, the object data would not be displayed. When the user selects the anchor data, the object data would then be initialized and displayed. This would also be the case for an audio file, where the file would be instantiated but would not be processed. Selecting the anchor data would then trigger the audio file to be processed.

User agents that do not support the declare [p.100] attribute must process the contents of the object [p.99] element.

## 21.3. The standby element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

The standby [p.109] element specifies a message that a user agent may render while loading the object's implementation and data.

## 21.4. Issues

Efficient param's PR #7642
State: Open
Resolution: Defer
User: None

**Notes:**
The working group has agreed to review the param element for the next draft. BAE: f2f - it makes no difference to the actual processing. Values interpreted by handler - see discussion dated lists.w3.org/Archives/Member/w3c-html-wg/2003AprJun/0071.html item 16

# 22. Ruby Module

This section is *informative*.

The Ruby Module defines elements for ruby annotation markup as defined in Ruby Annotation [RUBY [p.226] ].

This module adds the ruby element to the Text [p.45] content set of the Text [p.45] Module. XHTML 2.0 supports the maximal content model for the `ruby` element, defined as follows:

```
((rb, (rt | (rp, rt, rp))) | (rbc, rtc, rtc?))
```

As defined in [RUBY [p.226] ], the `ruby` element is not allowed to nest.

Implementation: RELAX NG [p.200] , DTD

# 23. XHTML Scripting Module

This section is *normative*.

The Scripting Module defines elements that are used to contain information pertaining to executable scripts or the lack of support for executable scripts. Elements and attributes included in this module are:

| Elements | Attributes | Minimal Content Model |
|---|---|---|
| script [p.111] | Common [p.29] , charset (Charset [p.28] ), declare ("declare") xml:space="preserve" | PCDATA \| script [p.111] |

When this module is used, the script [p.111] element is added to the Structural [p.37] and Text [p.45] content sets of the Structural [p.35] and Text [p.45] Modules. In addition, the script [p.111] element is added to the content model of the head [p.32] element defined in the Document [p.31] Module.

Implementation: RELAX NG [p.184]

## 23.1. The script element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

declare = "`declare`"
: When present, this boolean attribute makes the current object [p.99] element a declaration only - not one that is to be executed until after the document has completed loading and has been called through a user event such as selecting an anchor that references the object.

charset = Charset [p.28]
: This attribute specifies the character encoding of the resource designated by the link. Please consult the section on character encodings for more details.

The script [p.113] element places a script within a document. This element may appear any number of times in the head [p.32] or body [p.33] of an XHTML document.

The script may be defined within the contents of the script [p.113] element or in an external file. If the src [p.75] attribute is not set, user agents must interpret the contents of the element as the script. If the src [p.75] has a URI value, user agents must ignore the element's contents and retrieve the script via the URI. Note that the charset [p.113] attribute refers to the character

encoding [p.28] of the script designated by the src [p.75] attribute; it does not concern the content of the script [p.113] element.

## 23.1.1. Rules for processing scripts

Scripts are evaluated by *script engines* that must be known to a user agent.

A user agent must interpret a script [p.113] element according to the following precedence rules:

1. The user agent must first try to process the script element, but not the embedded content.
2. If the user agent is not able to process the script for any reason (configured not to, no support of the scripting language, no access to an external resource, etc.), it must try to process its contents.
3. If the content is inline text it must be evaluated as script data of the scripting language of the containing script element.
4. If the content is a script [p.113] element, its content should be processed according to these rules.

The syntax of script data depends on the scripting language.

## 23.1.2. Specifying the scripting language

As XHTML does not rely on a specific scripting language, document authors must explicitly tell user agents the language of each script. This may be done either through a default declaration or a local declaration.

## 23.1.3. Declaration of a scripting language

The type [p.76] attribute must be specified for each script [p.113] element instance in a document.

In this example, we include one script [p.113] in the header, whose script is located in an external file and is in the scripting language "text/vbscript". The JavaScript code contained in the inner script [p.113] will be evaluated if and only if the user agent isn't evaluating the outer script [p.113] . We also include one script [p.113] in the body, which contains its own script written in "text/x-perl".

```
<html xmlns="http://www.w3.org/2002/06/xhtml2">
 <head>
  <title>A document with script</title>
  <script type="text/x-vbscript" src="http://example.org/progs/vbcalc">
   <script type="text/javascript">
    ...some inline JavaScript...
   </script>
  </script>
 </head>
```

```
 <body>
    <script type="text/x-perl"/>
 </body>
</html>
```

## 23.1.4. Dynamic modification of documents

The processing model of XML means that the EcmaScript method `document.write` cannot be used in XHTML2. To dynamically generate content in XHTML you have to add elements to the DOM tree using DOM calls [DOM [p.225] ] rather than using `document.write` to generate text that then gets parsed.

# 24. XHTML Style Attribute Module

This section is *normative*.

The Style Attribute Module defines the `style` attribute. When this module is selected, it activates the Style [p.115] attribute collection.

Note: use of the style [p.115] attribute is strongly discouraged in favor of the style [p.117] element and external style sheets. In addition, content developers are advised to avoid use of the style [p.115] attribute on content intended for use on small devices, since those devices may not support the use of in-line styles.

## 24.1. Style Attribute Collection

style = CDATA [p.27]

This attribute specifies style information for the current element.

The syntax of the value of the style [p.117] attribute is determined by the default style sheet language.

This CSS example sets color and font size information for the text in a specific paragraph.

```
<p style="font-size: 12pt; color: fuchsia">Aren't style sheets wonderful?</p>
```

In CSS, property declarations have the form "name : value" and are separated by a semi-colon.

To specify style information for more than one element, authors should use the style [p.117] element. For optimal flexibility, authors should define styles in external style sheets.

Implementation: RELAX NG [p.185]

# 25. XHTML Style Sheet Module

This section is *normative*.

The Style Sheet Module defines an element to be used when declaring internal style sheets. The element and attributes defined by this module are:

| Elements | Attributes | Minimal Content Model |
|---|---|---|
| style [p.117] | Common [p.29] , disabled [p.117] ("disabled"), media [p.117] (MediaDesc [p.28] ) | PCDATA |

When this module is used, it adds the style [p.117] element to the content model of the head [p.32] element of the Document [p.31] Module.

Implementation: RELAX NG [p.185]

## 25.1. The style element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

disabled = "`disabled`"
: When present, this boolean attribute makes the current element inactive (e.g., a "disabled" style element would have its styles removed from the active style collection).

media = MediaDesc [p.28]
: The value of this attribute specifies the type of media for which the element is intended.

The style [p.119] element allows an author to put style sheet rules in the head of the document. XHTML permits any number of style [p.119] elements in the head [p.32] section of a document.

User agents that don't support style sheets, or don't support the specific style sheet language used by a style [p.119] element, must hide the contents of the style [p.119] element. It is an error to render the content as part of the document's text.

The syntax of style data depends on the style sheet language.

Rules for style rule precedences and inheritance depend on the style sheet language.

Example:

```
<style type="text/css">
  h1 {border-width: thin; border-style: solid; text-align: center}
</style>
```

## 25.1.1. External style sheets

Authors may separate style sheets from XHTML documents. This offers several benefits:

- Authors and web site managers may share style sheets across a number of documents (and sites).
- Authors may change the style sheet without requiring modifications to the document.
- User agents may load style sheets selectively (based on media descriptors).

## 25.1.2. Preferred and alternate style sheets

XHTML allows authors to associate any number of external style sheets with a document. The style sheet language defines how multiple external style sheets interact (for example, the CSS "cascade" rules).

Authors may specify a number of mutually exclusive style sheets called *alternate* style sheets. Users may select their favorite among these depending on their preferences. For instance, an author may specify one style sheet designed for small screens and another for users with weak vision (e.g., large fonts). User agents should allow users to select from alternate style sheets.

The author may specify that one of the alternates is a *preferred* style sheet. User agents should apply the author's preferred style sheet unless the user has selected a different alternate.

Authors may group several alternate style sheets (including the author's preferred style sheets) under a single *style name*. When a user selects a named style, the user agent must apply all style sheets with that name. User agents must not apply alternate style sheets with a different style name. The section on specifying external style sheets explains how to name a group of style sheets.

Authors may also specify *persistent* style sheets that user agents must apply in addition to any alternate style sheet.

User agents must respect media descriptors [p.28] when applying any style sheet.

User agents should also allow users to disable the author's style sheets entirely, in which case the user agent must not apply any persistent or alternate style sheets.

## 25.1.3. Specifying external style sheets

Authors specify external style sheets using the `xml-stylesheet` processing instruction [XMLSTYLE [p.227] ], or, for CSS, by using the `@import` facility within a style [p.119] element.

User agents should provide a means for users to view and pick from the list of alternate styles, if specified.

In this example, we first specify a persistent style sheet located in the file `mystyle.css`:

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

Setting the `title` pseudo-attribute makes this the author's preferred style sheet:

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

Adding the `alternate` pseudo-attribute makes it an alternate style sheet:

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

# 26. XHTML Tables Module

This section is *normative*.

The Tables Module provides elements for marking up tabular information in a document.

The module supports the following elements, attributes, and content model:

| Elements | Attributes | Minimal Content Model |
|---|---|---|
| table | Common [p.29] | caption [p.122] ?, summary [p.125] ?, ( col [p.123] * \| colgroup [p.123] * ), (( thead [p.142] ?, tfoot [p.142] ?, tbody [p.136] + ) \| ( tr [p.143] + )) |
| caption | Common [p.29] | (PCDATA \| Text [p.45] )* |
| summary | Common [p.29] | (PCDATA \| Flow [p.37] )* |
| col | Common [p.29] , span [p.123] (Number [p.28] ) | EMPTY |
| colgroup | Common [p.29] , span [p.123] (Number [p.28] ) | col [p.123] * |
| thead | Common [p.29] | tr [p.143] + |
| tfoot | Common [p.29] | tr [p.143] + |
| tbody | Common [p.29] | tr [p.143] + |
| tr | Common [p.29] | ( td [p.137] \| th [p.137] )+ |
| td | Common [p.29] , abbr [p.137] (Text [p.28] ), axis [p.137] (CDATA [p.27] ), colspan [p.137] (Number [p.28] ), headers [p.137] (IDREFS [p.27] ), rowspan [p.137] (Number [p.28] ), scope [p.138] ("row", "col", "rowgroup", "colgroup") | (PCDATA \| Flow [p.37] )* |
| th | Common [p.29] , abbr [p.137] (Text [p.28] ), axis [p.137] (CDATA [p.27] ), colspan [p.137] (Number [p.28] ), headers [p.137] (IDREFS [p.27] ), rowspan [p.137] (Number [p.28] ), scope [p.138] ("row", "col", "rowgroup", "colgroup") | (PCDATA \| Flow [p.37] )* |

When this module is used, it adds the table [p.125] element to the Structural [p.37] content set of the Structural Module.

Implementation: RELAX NG [p.186]

# 26.1. The caption element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

When present, the caption [p.124] element's text should describe the nature of the table or the object. The caption [p.124] element is only permitted immediately after the table [p.125] start tag or the object [p.99] start tag. A table [p.125] element or an object [p.99] element may only contain one caption [p.124] element.

For tables, visual user agents allow sighted people to quickly grasp the structure of the table from the headings as well as the caption. A consequence of this is that captions will often be inadequate as a summary of the purpose and structure of the table from the perspective of people relying on non-visual user agents.

Authors should therefore take care to provide additional information summarizing the purpose and structure of the table using the summary [p.125] element. This is especially important for tables without captions. The example below illustrates the use of the summary [p.125] element.

User agents must provide access to the content of the summary [p.125] element. As an example, access could be provided through a menu option, a mouse-over function, or through a dialog.

The following example demonstrates the difference between a table caption and a table summary.

```
<table>
  <caption>Student Class Roster</caption>
  <summary>The table defines the class roster.
    The columns contain the following data:
    students name, permanent address, permanent phone,
    local address, local phone,
    declared major, assigned academic advisor,
    student standing</summary>
  ...the rest of the table...
</table>
```

## 26.1.1. Caption Clipping

Visual user agents must avoid clipping any part of the table including the caption, unless a means is provided to access all parts, e.g., by horizontal or vertical scrolling. We recommend that the caption text be wrapped to the same width as the table.

# 26.2. The col and colgroup elements

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

span = Number [p.28]
: This attribute must be an integer > 0; the default value is 1. This specifies the number of columns in a colgroup [p.125] , or specifies the number of columns "spanned" by the col [p.125] element.

  Values mean the following:

  - In the absence of a span [p.125] attribute, each colgroup [p.125] defines a column group containing one column.
  - If the span [p.125] attribute is used with the colgroup [p.125] element and the value is set to N > 0, that defines a column group containing N columns.
  - If the span [p.125] attribute is used with the col [p.125] element and the value is set to N > 1, the current col [p.125] element shares its attributes with the next N-1 columns.

  User agents must ignore this attribute if the colgroup [p.125] element contains one or more col [p.125] elements.

The colgroup [p.125] element allows authors to create structural divisions within a table. Authors may highlight this structure through style sheets. For example, the author may wish to divide the columns into logical groups such as the student’s permanent address, phone number and emergency contact information. And group the student’s local address, phone and email address into another logical group.

A table [p.125] may either contain a single implicit column group (no colgroup [p.125] element delimits the columns) or any number of explicit column groups (each delimited by an instance of the colgroup [p.125] element).

The col [p.125] element allows authors to share attributes among several columns without implying any structural grouping. The "span" of the col [p.125] element is the number of columns that will share the element’s attributes. For example, the author may wish to apply a specific style to the student’s permanent data and apply a different style to the student’s local data.

The colgroup [p.125] element creates an explicit column group. The number of columns in the column group may be specified in two, mutually exclusive ways:

1. The colgroup span [p.125] attribute (default value 1) specifies the number of columns in the group.
2. Each embedded col [p.125] element in the colgroup [p.125] represents one or more columns in the group.

The advantage of using the colgroup [p.125] element is that authors may logically group multiple columns. By grouping columns, the author can apply rules across the entire group. The author can also apply column width balancing across the group of columns. For example, if the author has a table with five columns and the author divides the table into two column groups, one with three columns and the other with two columns. The author could define the first column group to consume 300 pixels and the second column group to consume 100 pixels. Each column within the first column group would be 100 pixels wide and the remaining two columns would be 50 pixels wide. If the author added embedded col [p.125] elements, she could force one or more columns to be a specific width and the remaining columns within the group would be evenly divided within the remaining allotted width.

For example, the following table defines a column group and embedded columns with differing widths.

```
<style type="text/css">
#colgrp1 { width: 300px }
#col1 { width: 100px }
#col2 { width: 50px }
</style>

...

<table>
  <colgroup id="colgrp1">
    <col id="col1" span="3"/>
    <col id="col2" span="2"/>
  </colgroup>
...the rest of the table...
</table>
```

In this example, the defined width for the colgroup [p.125] constrains all of the columns to fit within that value regardless of the of the defined values within the col [p.125] elements. In this example, the width of the columns within the column group must be constrained to fit the defined width of the column group.

When it is necessary to single out a column (e.g., for style information, to specify width information, etc.) within a group, authors must identify that column with a col [p.125] element.

The col [p.125] element allows authors to group together attribute specifications for table columns. The col [p.125] does **not** group columns together structurally -- that is the role of the colgroup [p.125] element. col [p.125] elements are empty and serve only as a support for attributes. They may appear inside or outside an explicit column group (i.e., colgroup [p.125] element).

## 26.2.1. Calculating the number of columns in a table

There are two ways to determine the number of columns in a table (in order of precedence):

1. If the table [p.125] element contains any colgroup [p.125] or col [p.125] elements, user agents should calculate the number of columns by summing the following:
   - For each col [p.125] element, take the value of its span [p.125] attribute (default value 1).
   - For each colgroup [p.125] element containing at least one col [p.125] element, ignore the span [p.125] attribute for the colgroup [p.125] element. For each col [p.125] element, perform the calculation of step 1.
   - For each empty colgroup [p.125] element, take the value of its span [p.125] attribute (default value 1).
2. Otherwise, if the table [p.125] element contains no colgroup [p.125] or col [p.125] elements, user agents should base the number of columns on what is required by the rows. The number of columns is equal to the number of columns required by the row with the most columns, including cells that span multiple columns. For any row that has fewer than this number of columns, the end of that row should be padded with empty cells. The "end" of a row depends on the directionality of a table.

It is an error if a table contains colgroup [p.125] or col [p.125] elements and the two calculations do not result in the same number of columns.

Once the user agent has calculated the number of columns in the table, it may group them into a colgroup [p.125] .

## 26.3. The summary element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

This element provides a summary of the table's purpose and structure for user agents rendering to non-visual media such as speech and Braille.

In the following example, we define a summary for the table.

```
<table>
<summary>This table shows the number of times we needed to edit
         this part of XHTML each week.</summary>
Other table elements...
</table>
```

## 26.4. The table element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

The table [p.127] element contains all other elements that specify the caption, column groups, columns, rows, and content for a table.

## 26.4.1. Visual Rendering

All style associated with table rendering MUST use proper CSS2 properties.

Although CSS2 is not required, the equivalent effect MUST BE followed and integrated into the rendering model.

The following informative list describes what operations visual user agents may carry out when rendering a table:

- Provide access to the content of the summary [p.127] element. As an example, access could be provided through a menu option, a mouse-over function, or through a dialog. Authors should provide a summary of a table's content and structure so that people using non-visual user agents may better understand it.
- Render the caption, if one is defined. The caption may be rendered, for example, either on the top or the bottom of the table.
- Render the table header, if one is specified. Render the table footer, if one is specified. User agents must know where to render the header and footer. For instance, if the output medium is paged, user agents may put the header at the top of each page and the footer at the bottom. Similarly, if the user agent provides a mechanism to scroll the rows, the header may appear at the top of the scrolled area and the footer at the bottom.
- Calculate the number of columns [p.126] in the table. Note that the number of rows in a table is equal to the number of tr [p.143] elements contained by the table [p.127] element.
- Group the columns according to any column groups [p.125] specifications.
- Render the cells, row by row and grouped in appropriate columns, between the header and footer. Visual user agents should format the table according to XHTML attributes and style sheet specification.

The XHTML table model has been designed so that, with author assistance, user agents may render tables *incrementally* (i.e., as table rows arrive) rather than having to wait for all the data before beginning to render.

In order for a user agent to format a table in one pass, authors must tell the user agent:

- The number of columns in the table.
- The widths of these columns.

More precisely, a user agent may render a table in a single pass when the column widths are specified using a combination of colgroup [p.125] and col [p.125] elements. If any of the columns are specified in relative or percentage terms, authors must also specify the width of the table itself.

## 26.4.2. Table directionality

The directionality of a table is either the inherited directionality (the default is left-to-right) or that specified by the dir [p.71] attribute for the table [p.127] element.

For a left-to-right table, column zero is on the left side and row zero is at the top. For a right-to-left table, column zero is on the right side and row zero is at the top.

When a user agent allots extra cells to a row, extra row cells are added to the right of the table for left-to-right tables and to the left side for right-to-left tables.

Note that table [p.127] is the only element on which dir [p.71] reverses the visual order of the columns; a single table row ( tr [p.143] ) or a group of columns ( colgroup [p.125] ) cannot be independently reversed.

When set for the table [p.127] element, the dir [p.71] attribute also affects the direction of text within table cells (since the dir [p.71] attribute is inherited by block-level elements).

To specify a right-to-left table, set the dir [p.71] attribute as follows:

```
<table dir="rtl">
...the rest of the table...
</table>
```

The direction of text in individual cells can be changed by setting the dir [p.71] attribute in an element that defines the cell.

## 26.4.3. Table rendering by non-visual user agents

This section provides more detailed discussion on cell header data and how non-visual agents may utilize that information.

### 26.4.3.1. Associating header information with data cells

Non-visual user agents such as speech synthesizers and Braille-based devices may use the following td [p.137] and th [p.137] element attributes to render table cells more intuitively:

- For a given data cell, the headers [p.137] attribute lists which cells provide pertinent header information. For this purpose, each header cell must be named using the id [p.61] attribute. Note that it's not always possible to make a clean division of cells into headers or data. You should use the td [p.137] element for such cells together with the id [p.61] or scope [p.138] attributes as appropriate.
- For a given header cell, the scope [p.138] attribute tells the user agent the data cells for

which this header provides information. Authors may choose to use this attribute instead of headers [p.137] according to which is more convenient; the two attributes fulfill the same function. The headers [p.137] attribute is generally needed when headers are placed in irregular positions with respect to the data they apply to.

- The abbr [p.137] attribute specifies an abbreviated header for header cells so that user agents may render header information more rapidly.

In the following example, we assign header information to cells by setting the headers [p.137] attribute. Each cell in the same column refers to the same header cell (via the id [p.61] attribute).

```
<table>
<caption>Cups of coffee consumed by each senator</caption>
<summary>This table charts the number of cups
          of coffee consumed by each senator, the type
          of coffee (decaf or regular), and whether
          taken with sugar.</summary>
<tbody>
   <tr>
      <th id="t1">Name</th>
      <th id="t2">Cups</th>
      <th id="t3" abbr="Type">Type of Coffee</th>
      <th id="t4">Sugar?</th>
   </tr>
   <tr>
      <td headers="t1">T. Sexton</td>
      <td headers="t2">10</td>
      <td headers="t3">Espresso</td>
      <td headers="t4">No</td>
   </tr>
   <tr>
      <td headers="t1">J. Dinnen</td>
      <td headers="t2">5</td>
      <td headers="t3">Decaf</td>
      <td headers="t4">Yes</td>
   </tr>
</tbody>
</table>
```

A speech synthesizer might render this table as follows:

```
Caption: Cups of coffee consumed by each senator
Summary: This table charts the number of cups
         of coffee consumed by each senator, the type
         of coffee (decaf or regular), and whether
         taken with sugar.
Name: T. Sexton,   Cups: 10,   Type: Espresso,   Sugar: No
Name: J. Dinnen,   Cups: 5,    Type: Decaf,      Sugar: Yes
```

Note how the header "Type of Coffee" is abbreviated to "Type" using the abbr [p.137] attribute.

Here is the same example substituting the scope [p.138] attribute for the headers [p.137] attribute. Note the value "col" for the scope [p.138] attribute, meaning "all cells in the current column":

```
<table>
<caption>Cups of coffee consumed by each senator</caption>
<summary>
  This table charts the number of cups
  of coffee consumed by each senator, the type
  of coffee (decaf or regular), and whether
  taken with sugar.
</summary>
<tbody>
   <tr>
      <th scope="col">Name</th>
      <th scope="col">Cups</th>
      <th scope="col" abbr="Type">Type of Coffee</th>
      <th scope="col">Sugar?</th>
   </tr>
   <tr>
      <td>T. Sexton</td>
      <td>10</td>
      <td>Espresso</td>
      <td>No</td>
   </tr>
   <tr>
      <td>J. Dinnen</td>
      <td>5</td>
      <td>Decaf</td>
      <td>Yes</td>
</tbody>
</table>
```

Here's a somewhat more complex example illustrating other values for the scope [p.138] attribute:

```
<table>
<summary>
  History courses offered in the community of
  Bath arranged by course name, tutor, summary,
  code, and fee
</summary>
<thead>
  <tr>
    <th colspan="5" scope="colgroup">Community Courses -- Bath Autumn 1997</th>
  </tr>
</thead>
<tbody>
  <tr>
    <th scope="col" abbr="Name">Course Name</th>
    <th scope="col" abbr="Tutor">Course Tutor</th>
    <th scope="col">Summary</th>
    <th scope="col">Code</th>
    <th scope="col">Fee</th>
  </tr>
  <tr>
    <td scope="row">After the Civil War</td>
    <td>Dr. John Wroughton</td>
    <td>
       The course will examine the turbulent years in England
```

```
        after 1646. <em>6 weekly meetings starting Monday 13th
        October.</em>
      </td>
      <td>H27</td>
      <td>&pound;32</td>
    </tr>
    <tr>
      <td scope="row">An Introduction to Anglo-Saxon England</td>
      <td>Mark Cottle</td>
      <td>
        One day course introducing the early medieval
        period reconstruction the Anglo-Saxons and
        their society. <em>Saturday 18th October.</em>
      </td>
      <td>H28</td>
      <td>&pound;18</td>
    </tr>
    <tr>
      <td scope="row">The Glory that was Greece</td>
      <td>Valerie Lorenz</td>
      <td>
       Birthplace of democracy, philosophy, heartland of theater, home of
       argument. The Romans may have done it but the Greeks did it
       first. <em>Saturday day school 25th October 1997</em>
      </td>
      <td>H30</td>
      <td>&pound;18</td>
    </tr>
  </tbody>
  </table>
```

A graphical user agent might render this as:

| Community Courses -- Bath Autumn 1997 | | | | |
|---|---|---|---|---|
| **Course Name** | **Course Tutor** | **Summary** | **Code** | **Fee** |
| After the Civil War | Dr. John Wroughton | The course will examine the turbulent years in England after 1646. *6 weekly meetings starting Monday 13th October.* | H27 | £32 |
| An Introduction to Anglo-Saxon England | Mark Cottle | One day course introducing the early medieval period reconstruction the Anglo-Saxons and their society. *Saturday 18th October.* | H28 | £18 |
| The Glory that was Greece | Valerie Lorenz | Birthplace of democracy, philosophy, heartland of theater, home of argument. The Romans may have done it but the Greeks did it first. *Saturday day school 25th October 1997* | H30 | £18 |

Note the use of the scope [p.138] attribute with the "row" value. Although the first cell in each row contains data, not header information, the scope [p.138] attribute makes the data cell behave like a row header cell. This allows speech synthesizers to provide the relevant course name upon request or to state it immediately before each cell's content.

### 26.4.3.2. Categorizing cells

Users browsing a table with a speech-based user agent may wish to hear an explanation of a cell's contents in addition to the contents themselves. One way the user might provide an explanation is by speaking associated header information before speaking the data cell's contents (see the section on associating header information with data cells [p.129] ).

Users may also want information about more than one cell, in which case header information provided at the cell level (by headers [p.137] , scope [p.138] , and abbr [p.137] ) may not provide adequate context. Consider the following table, which classifies expenses for meals, hotels, and transport in two locations (San Jose and Seattle) over several days:

**Travel Expense Report**

| | Meals | Hotels | Transport | subtotals |
|---|---|---|---|---|
| **San Jose** | | | | |
| 25-Aug-97 | 37.74 | 112.00 | 45.00 | |
| 26-Aug-97 | 27.28 | 112.00 | 45.00 | |
| subtotals | 65.02 | 224.00 | 90.00 | 379.02 |
| **Seattle** | | | | |
| 27-Aug-97 | 96.25 | 109.00 | 36.00 | |
| 28-Aug-97 | 35.00 | 109.00 | 36.00 | |
| subtotals | 131.25 | 218.00 | 72.00 | 421.25 |
| **Totals** | 196.27 | 442.00 | 162.00 | **800.27** |

Users might want to extract information from the table in the form of queries:

- "What did I spend for all my meals?"
- "What did I spend for meals on 25 August?"
- "What did I spend for all expenses in San Jose?"

Each query involves a computation by the user agent that may involve zero or more cells. In order to determine, for example, the costs of meals on 25 August, the user agent must know which table cells refer to "Meals" (all of them) and which refer to "Dates" (specifically, 25 August), and find the intersection of the two sets.

To accommodate this type of query, the table model allows authors to place cell headers and data into categories. For example, for the travel expense table, an author could group the header cells "San Jose" and "Seattle" into the category "Location", the headers "Meals", "Hotels", and "Transport" in the category "Expenses", and the four days into the category "Date". The previous three questions would then have the following meanings:

- "What did I spend for all my meals?" means "What are all the data cells in the "Expenses=Meals" category?
- "What did I spend for meals on 25 August?" means "What are all the data cells in the "Expenses=Meals" and "Date=Aug-25-1997" categories?
- "What did I spend for all expenses in San Jose?" means "What are all the data cells in the "Expenses=Meals, Hotels, Transport" and "Location=San Jose" categories?

Authors categorize a header or data cell by setting the axis [p.137] attribute for the cell. For instance, in the travel expense table, the cell containing the information "San Jose" could be placed in the "Location" category as follows:

```
<th id="a6" axis="location">San Jose</th>
```

Any cell containing information related to "San Jose" should refer to this header cell via either the headers [p.137] or the scope [p.138] attribute. Thus, meal expenses for 25-Aug-1997 should be marked up to refer to id [p.61] attribute (whose value here is "a6") of the "San Jose" header cell:

```
<td headers="a6">37.74</td>
```

Each headers [p.137] attribute provides a list of id [p.61] references. Authors may thus categorize a given cell in any number of ways (or, along any number of "headers", hence the name).

Below we mark up the travel expense table with category information:

```
<table>
<caption>Travel Expense Report</caption>
<summary>
  This table summarizes travel expenses
  incurred during August trips to
  San Jose and Seattle
</summary>
<tbody>
   <tr>
      <th></th>
      <th id="a2" axis="expenses">Meals</th>
      <th id="a3" axis="expenses">Hotels</th>
      <th id="a4" axis="expenses">Transport</th>
      <td>subtotals</td>
   </tr>
   <tr>
      <th id="a6" axis="location">San Jose</th>
      <th></th>
      <th></th>
      <th></th>
      <td></td>
   </tr>
   <tr>
      <td id="a7" axis="date">25-Aug-97</td>
      <td headers="a6 a7 a2">37.74</td>
      <td headers="a6 a7 a3">112.00</td>
```

```
      <td headers="a6 a7 a4">45.00</td>
      <td></td>
    </tr>
    <tr>
      <td id="a8" axis="date">26-Aug-97</td>
      <td headers="a6 a8 a2">27.28</td>
      <td headers="a6 a8 a3">112.00</td>
      <td headers="a6 a8 a4">45.00</td>
      <td></td>
    </tr>
    <tr>
      <td>subtotals</td>
      <td>65.02</td>
      <td>224.00</td>
      <td>90.00</td>
      <td>379.02</td>
    </tr>
    <tr>
      <th id="a10" axis="location">Seattle</th>
      <th></th>
      <th></th>
      <th></th>
      <td></td>
    </tr>
    <tr>
      <td id="a11" axis="date">27-Aug-97</td>
      <td headers="a10 a11 a2">96.25</td>
      <td headers="a10 a11 a3">109.00</td>
      <td headers="a10 a11 a4">36.00</td>
      <td></td>
    </tr>
    <tr>
      <td id="a12" axis="date">28-Aug-97</td>
      <td headers="a10 a12 a2">35.00</td>
      <td headers="a10 a12 a3">109.00</td>
      <td headers="a10 a12 a4">36.00</td>
      <td></td>
    </tr>
    <tr>
      <td>subtotals</td>
      <td>131.25</td>
      <td>218.00</td>
      <td>72.00</td>
      <td>421.25</td>
    </tr>
    <tr>
      <th>Totals</th>
      <td>196.27</td>
      <td>442.00</td>
      <td>162.00</td>
      <td>800.27</td>
    </tr>
  </tbody>
  </table>
```

Note that marking up the table this way also allows user agents to avoid confusing the user with unwanted information. For instance, if a speech synthesizer were to speak all of the figures in the "Meals" column of this table in response to the query "What were all my meal expenses?", a user would not be able to distinguish a day's expenses from subtotals or totals. By carefully categorizing cell data, authors allow user agents to make important semantic distinctions when rendering.

Of course, there is no limit to how authors may categorize information in a table. In the travel expense table, for example, we could add the additional categories "subtotals" and "totals".

This specification does not require user agents to handle information provided by the axis [p.137] attribute, nor does it make any recommendations about how user agents may present axis [p.137] information to users or how users may query the user agent about this information.

However, user agents, particularly speech synthesizers, may want to factor out information common to several cells that are the result of a query. For instance, if the user asks "What did I spend for meals in San Jose?", the user agent would first determine the cells in question (25-Aug-1997: 37.74, 26-Aug-1997:27.28), then render this information. A user agent speaking this information might read it:

```
Location: San Jose. Date: 25-Aug-1997. Expenses, Meals: 37.74
Location: San Jose. Date: 26-Aug-1997. Expenses, Meals: 27.28
```

or, more compactly:

```
San Jose, 25-Aug-1997, Meals: 37.74
San Jose, 26-Aug-1997, Meals: 27.28
```

An even more economical rendering would factor the common information and reorder it:

```
San Jose, Meals, 25-Aug-1997: 37.74
                 26-Aug-1997: 27.28
```

User agents that support this type of rendering should allow authors a means to customize rendering (e.g., through style sheets).

## 26.4.3.3. Algorithm to find heading information

In the absence of header information from either the scope [p.138] or headers [p.137] attribute, user agents may construct header information according to the following algorithm. The goal of the algorithm is to find an ordered list of headers. (In the following description of the algorithm the table directionality [p.129] is assumed to be left-to-right.)

- First, search left from the cell's position to find row header cells. Then search upwards to find column header cells. The search in a given direction stops when the edge of the table is reached or when a data cell is found after a header cell.
- Row headers are inserted into the list in the order they appear in the table. For left-to-right tables, headers are inserted from left to right.
- Column headers are inserted after row headers, in the order they appear in the table, from

top to bottom.

- If a header cell has the headers [p.137] attribute set, then the headers referenced by this attribute are inserted into the list and the search stops for the current direction.
- td [p.137] cells that set the axis [p.137] attribute are also treated as header cells.

## 26.4.4. Sample table

This sample illustrates grouped rows and columns.

```
<table>
<caption>Calendar for 2002</caption>
<summary>
  Calendar for 2002. @@need better summary
</summary>
<colgroup span="7"/>
<colgroup span="7"/>
<colgroup span="7"/>
<thead>
   <tr>
      <th colspan="7">January</th>
      <th colspan="7">February</th>
      <th colspan="7">March</th>
   </tr>
</thead>
<tbody>
   <tr>
      <td>S</td><td>M</td><td>T</td><td>W</td><td>T</td><td>F</td><td>S</td>
      <td>S</td><td>M</td><td>T</td><td>W</td><td>T</td><td>F</td><td>S</td>
      <td>S</td><td>M</td><td>T</td><td>W</td><td>T</td><td>F</td><td>S</td>
   </tr>
</tbody>
<tbody>
   <tr>
      <td></td><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td>
      <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td>2</td>
      <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td>2</td>

   </tr>
   <tr>
      <td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td>
      <td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td>
      <td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td>

   </tr>
   <tr>
      <td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td>
      <td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td>
      <td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td>
   </tr>
   <tr>
      <td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td>
      <td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td>
      <td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td>
   </tr>
   <tr>
```

```
      <td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td></td><td></td>
      <td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td></td><td></td>
      <td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td>
   </tr>
   <tr>
      <td></td><td></td><td></td><td></td><td></td><td></td><td></td>
      <td></td><td></td><td></td><td></td><td></td><td></td><td></td>
      <td>31</td><td></td><td></td><td></td><td></td><td></td><td></td>
   </tr>
</tbody>
</table>
```

would be rendered something like this:

```
                    Calendar for 2002
=====================================================================
      January          |       February       |        March
 S  M  T  W  T  F  S   | S  M  T  W  T  F  S  | S  M  T  W  T  F  S
---------------------------------------------------------------------
       1  2  3  4  5   |                1  2  |                1  2
 6  7  8  9  10 11 12  | 3  4  5  6  7  8  9  | 3  4  5  6  7  8  9
 13 14 15 16 17 18 19  | 10 11 12 13 14 15 16 | 10 11 12 13 14 15 16
 20 21 22 23 24 25 26  | 17 18 19 20 21 22 23 | 17 18 19 20 21 22 23
 27 28 29 30 31        | 24 25 26 27 28       | 24 25 26 27 28 29 30
                       |                      |                   31
=====================================================================
```

A graphical user agent might render this as:

**Calendar for 2002**

| January | | | | | | | February | | | | | | | March | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | M | T | W | T | F | S | S | M | T | W | T | F | S | S | M | T | W | T | F | S |
| | | 1 | 2 | 3 | 4 | 5 | | | | | | 1 | 2 | | | | | | 1 | 2 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 27 | 28 | 29 | 30 | 31 | | | 24 | 25 | 26 | 27 | 28 | | | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| | | | | | | | | | | | | | | 31 | | | | | | |

This example illustrates how colgroup [p.125] can be used to group columns and set the default column alignment. Similarly, tbody [p.136] is used to group rows.

## 26.5. The tbody element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

The tbody [p.139] element contains rows of table data. In tables that also contain thead [p.142] or tfoot [p.142] elements, all of these sections must contain the same number of columns.

## 26.6. The td and th elements

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

abbr = Text [p.28]
: This attribute should be used to provide an abbreviated form of the cell's content, and may be rendered by user agents when appropriate in place of the cell's content. Abbreviated names should be short since user agents may render them repeatedly. For instance, speech synthesizers may render the abbreviated headers relating to a particular cell before rendering that cell's content.

axis = CDATA [p.27]
: This attribute may be used to place a cell into conceptual categories that can be considered to form axes in an n-dimensional space. User agents may give users access to these categories (e.g., the user may query the user agent for all cells that belong to certain categories, the user agent may present a table in the form of a table of contents, etc.). Please consult the section on categorizing cells [p.133] for more information. The value of this attribute is a comma-separated list of category names.

colspan = Number [p.28]
: This attribute specifies the number of columns spanned by the current cell. The default value of this attribute is one ("1"). The value zero ("0") means that the cell spans all columns from the current column to the last column of the column group ( colgroup [p.125] ) in which the cell is defined.

headers = IDREFS [p.27]
: This attribute specifies the list of header cells that provide header information for the current data cell. The value of this attribute is a space-separated list of cell names; those cells must be named by setting their id [p.61] attribute. Authors generally use the headers [p.139] attribute to help non-visual user agents render header information about data cells (e.g.,

header information is spoken prior to the cell data), but the attribute may also be used in conjunction with style sheets. See also the scope [p.138] attribute.

rowspan = Number [p.28]
: This attribute specifies the number of rows spanned by the current cell. The default value of this attribute is one ("1"). The value zero ("0") means that the cell spans all rows from the current row to the last row of the table section ( thead [p.142] , tbody [p.139] , or tfoot [p.142] ) in which the cell is defined.

scope = `row`|`col`|`rowgroup`|`colgroup`
: This attribute specifies the set of data cells for which the current header cell provides header information. This attribute may be used in place of the headers [p.139] attribute, particularly for simple tables. When specified, this attribute must have one of the following values:
  - **row:** The current cell provides header information for the rest of the row that contains it (see also the section on table directionality [p.129] ).
  - **col:** The current cell provides header information for the rest of the column that contains it.
  - **rowgroup:** The header cell provides header information for the rest of the row group that contains it.
  - **colgroup:** The header cell provides header information for the rest of the column group [p.125] that contains it.

Table cells may contain two types of information: header information and data. This distinction enables user agents to render header and data cells distinctly, even in the absence of style sheets. For example, visual user agents may present header cell text with a bold font. Speech synthesizers may render header information with a distinct voice inflection.

The th [p.139] element defines a cell that contains header information. User agents have two pieces of header information available: the contents of the th [p.139] element and the value of the abbr [p.139] attribute. User agents must render either the contents of the cell or the value of the abbr [p.139] attribute. For visual media, the latter may be appropriate when there is insufficient space to render the full contents of the cell. For non-visual media abbr [p.139] may be used as an abbreviation for table headers when these are rendered along with the contents of the cells to which they apply.

The headers [p.139] and scope [p.140] attributes also allow authors to help non-visual user agents process header information. Please consult the section on labeling cells for non-visual user agents [p.129] for information and examples.

The td [p.139] element defines a cell that contains data.

Cells may be empty (i.e., contain no data).

For example, the following table contains four columns of data, each headed by a column description.

```
<table>
<caption>Cups of coffee consumed by each senator</caption>
<summary>This table charts the number of cups
  of coffee consumed by each senator, the type
  of coffee (decaf or regular), and whether
  taken with sugar.</summary>
<tbody>
   <tr>
      <th>Name</th>
      <th>Cups</th>
      <th>Type of Coffee</th>
      <th>Sugar?</th>
   </tr>
   <tr>
      <td>T. Sexton</td>
      <td>10</td>
      <td>Espresso</td>
      <td>No</td>
   </tr>
   <tr>
      <td>J. Dinnen</td>
      <td>5</td>
      <td>Decaf</td>
      <td>Yes</td>
</tbody>
</table>
```

A user agent rendering to a tty device might display this as follows:

| **Name** | **Cups** | **Type of Coffee** | **Sugar?** |
|---|---|---|---|
| T. Sexton | 10 | Espresso | No |
| J. Dinnen | 5 | Decaf | Yes |

## 26.6.1. Cells that span several rows or columns

Cells may span several rows or columns. The number of rows or columns spanned by a cell is set by the rowspan [p.140] and colspan [p.139] attributes for the th [p.139] and td [p.139] elements.

In this table definition, we specify that the cell in row four, column two should span a total of three columns, including the current column.

```
<table>
<caption>Cups of coffee consumed by each senator</caption>
<tbody>
   <tr>
      <th>Name</td>
      <th>Cups</td>
      <th>Type of Coffee</td>
      <th>Sugar?</td>
   </tr>
   <tr>
      <td>T. Sexton</td>
      <td>10</td>
      <td>Espresso</td>
```

```
      <td>No</td>
   </tr>
   <tr>
      <td>J. Dinnen</td>
      <td>5</td>
      <td>Decaf</td>
      <td>Yes</td>
   </tr>
   <tr>
      <td>A. Soria</td>
      <td colspan="3"><em>Not available</em></td>
   </tr>
</tbody>
</table>
```

This table might be rendered on a tty device by a visual user agent as follows:

```
Cups of coffee consumed by each senator
 ---------------------------------------
 |   Name  |Cups|Type of Coffee|Sugar?|
 ---------------------------------------
 |T. Sexton|10  |Espresso      |No    |
 ---------------------------------------
 |J. Dinnen|5   |Decaf         |Yes   |
 ---------------------------------------
 |A. Soria |Not available              |
 ---------------------------------------
```

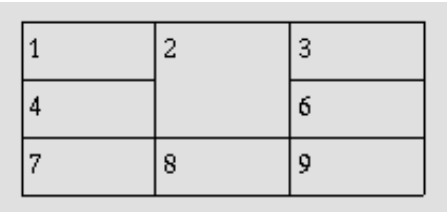
The next example illustrates (with the help of table borders) how cell definitions that span more than one row or column affect the definition of later cells. Consider the following table definition:

```
<table>
<tbody>
   <tr>
      <td>1</td>
      <td rowspan="2">2</td>
      <td>3</td>
   </tr>
   <tr>
      <td>4
      <td>6</td>
   </tr>
   <tr>
      <td>7</td>
      <td>8</td>
      <td>9</td>
      </td></td>
   </tr>
</tbody>
</table>
```

As cell "2" spans the first and second rows, the definition of the second row will take it into account. Thus, the second td [p.139] in row two actually defines the row's third cell. Visually, the table might be rendered to a tty device as:

```
-------------
| 1 | 2 | 3 |
----|   |----
| 4 |   | 6 |
----|---|----
| 7 | 8 | 9 |
-------------
```

while a graphical user agent might render this as:




Note that if the td [p.139] defining cell "6" had been omitted, an extra empty cell would have been added by the user agent to complete the row.

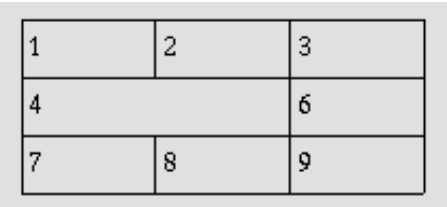
Similarly, in the following table definition:

```
<table>
<tbody>
   <tr>
      <td>1</td>
      <td>2</td>
      <td>3</td>
   </tr>
   <tr>
      <td colspan="2">4</td>
      <td>6</td>
   </tr>
   <tr>
      <td>7</td>
      <td>8</td>
      <td>9</td>
   </tr>
</tbody>
</table>
```

cell "4" spans two columns, so the second td [p.139] in the row actually defines the third cell ("6"):

```
-------------
| 1 | 2 | 3 |
--------|----
| 4     | 6 |
--------|----
| 7 | 8 | 9 |
-------------
```

A graphical user agent might render this as:




Defining overlapping cells is an error. User agents may vary in how they handle this error (e.g., rendering may vary).

The following illegal example illustrates how one might create overlapping cells. In this table, cell "5" spans two rows and cell "7" spans two columns, so there is overlap in the cell between "7" and "9":

```
<table">
<tbody>
   <tr>
      <td>1</td>
      <td>2</td>
      <td>3</td>
   </tr>
   <tr>
      <td>4</td>
      <td rowspan="2">5</td>
      <td>6</td>
   </tr>
   <tr>
      <td colspan="2">7</td>
      <td>9</td>
   </tr>
</tbody>
</table>
```

# 26.7. The thead and tfoot elements

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

Table rows may be grouped into a table head, table foot, and one or more table body sections, using the thead [p.144] , tfoot [p.144] and tbody [p.139] elements, respectively. This division enables user agents to support scrolling of table bodies independently of the table head and foot. When long tables are printed, the table head and foot information may be repeated on each page that contains table data.

The table head and table foot should contain information about the table's columns. The table body should contain rows of table data.

When present, each thead [p.144] , tfoot [p.144] , and tbody [p.139] contains a *row group*. Each row group must contain at least one row, defined by the tr [p.143] element.

If the thead [p.144] , tfoot [p.144] , and tbody [p.139] elements are used, and a rowspan attribute is used within a group, the rowspan must remain within the group boundaries of which it is defined.

This example illustrates the order and structure of the table head, foot, and bodies.

```
<table>
<thead>
     <tr> ...header information...</tr>
</thead>
<tfoot>
     <tr> ...footer information...</tr>
</tfoot>
<tbody>
     <tr> ...first row of block one data...</tr>
     <tr> ...second row of block one data...</tr>
</tbody>
<tbody>
     <tr> ...first row of block two data...</tr>
     <tr> ...second row of block two data...</tr>
     <tr> ...third row of block two data...</tr>
</tbody>
</table>
```

tfoot [p.144] must appear before tbody [p.139] within a table [p.127] definition so that user agents can render the foot before receiving all of the (potentially numerous) rows of data.

## 26.8. The tr element

*Attributes*

The Common [p.29] collection
: A collection of other attribute collections, including: Core [p.61] , Events [p.147] , I18N [p.69] , Bi-directional [p.71] , Edit [p.73] , Embedding [p.75] , Map [p.77] , Metainformation [p.81] , and Hypertext [p.63]

The tr [p.145] elements acts as a container for a row of table cells.

This sample table contains three rows, each begun by the tr [p.145] element:

```
<table>
<caption>Cups of coffee consumed by each senator</caption>
<summary>This table charts the number of cups
  of coffee consumed by each senator, the type
  of coffee (decaf or regular), and whether
  taken with sugar.</summary>
```

```
<tbody>
   <tr> ...A header row...</tr>
   <tr> ...First row of data...</tr>
   <tr> ...Second row of data...</tr>
   ...the rest of the table...</tr>
</tbody>
</table>
```

# 27. XForms Module

This section is *informative*.

The XForms Module provides a rich collection of forms features. It is defined in [XFORMS [p.226] ].

When the XFORMS module is included, the xforms:model element is added to the content model of the head [p.32] element in the Document [p.31] module.

The content model for this module defines a content set:

XForms
: xforms:input | xforms:secret | xforms:textarea | xforms:output | xforms:upload | xforms:range | xforms:trigger | xforms:submit | xforms:select | xforms:select1 | xforms:group | xforms:switch | xforms:repeat

Note that by using the prefix `xforms:` this document is only demonstrating that the elements are in a different namespace, not that the prefix `xforms:` is required.

## 27.1. Forms Attribute Collection

This module also defines the Forms Attribute Collection via the some attributes from [XFORMS [p.226] ]. The normative definition of those attributes and their semantics is included in that specification. They are described briefly below:

repeat-model = IDREF [p.27]
: This attribute defines the XForms model to be associated with the element.

repeat-bind = IDREF [p.27]
: This attribute provides a reference to an XForms `bind` element.

repeat-nodeset = LocationPath [p.28]
: This attribute defines the an XPath LocationPath that maps to an XForms `bind` element.

repeat-startindex = Number [p.28]
: XForms hint to the processor as to which item from the collection to display first.

repeat-number = Number [p.28]
: XForms hint to the processor as to how many items from the collection to display at a time.

Implementation: XML Schema

# 28. XML Events Module

This section is *informative*.

The XML Events Module defines a linkage between XHTML and the XML Document Object Model [DOM [p.225] ]. XML Events are defined in [XMLEVENTS [p.226] ], and all XML Event elements and attributes are in their own namespace.

When this module is selected, the ev:listener element is added to the content model of the head [p.32] element as defined in the Document Module.

## 28.1. Events

This module also defines the Events Attribute Collection via the global attributes from [XMLEVENTS [p.226] ]. The normative definition of those attributes and their semantics is included in that specification. They are described briefly below:

defaultAction = "`cancel`|`perform`"
: This attribute defines whether or not the default action associated with the event should be processed. The default value is `perform`

event = CDATA [p.27]
: This attribute defines the event type that is being listened for. The set of legal names for XHTML 2 is to be defined.

handler = IDREF [p.27]
: This attribute specifies the ID of a handler element that defines the action that should be performed if the event reaches the observer.

observer = IDREF [p.27]
: This attribute specifies an ID for an observer element for which the listener is to be registered.

phase = "`capture`|`default`"
: This attribute specifies the phase of event propagation in which to process the event. If not specified, the default value of this attribute is `default`.

propagate = "`stop`|`continue`"
: This attribute specifies whether an event should stop propagating after this observer processes it, or continue for possible further processing. The default value of this attribute is `continue`.

target = IDREF [p.27]
: This attribute specifies the id of the target element of the event (i.e., the node that caused the event). If not specified, the default value of this attribute is the element on which the event attribute is specified.

Note that these attributes are **not** in the XHTML namespace but in the XML Events namespace. The XHTML namespace is the default namespace for XHTML documents, so XHTML elements and attributes may be expressed without namespace prefixes (although they are permitted on elements). XML Events attributes MUST use a prefix, since they are not in the default namespace of the document.

Implementation: RELAX NG [p.200] , XML Schema, DTD

# A. Changes from XHTML 1.1

This appendix is *informative*.

This Appendix describes the differences between XHTML 2.0 and XHTML 1.1.

# B. XHTML 2.0 RELAX NG Definition

This appendix is *normative*.

This appendix contains the implementation of the XHTML 2.0 RELAX NG driver file.

## B.0.1. RELAX NG XHTML 2.0 Driver

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar ns="http://www.w3.org/2002/06/xhtml2"
         xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>RELAX NG schema for XHTML 2.0</x:h1>

  <x:pre>
    Copyright &#xA9;2003-2004 W3C&#xAE; (MIT, ERCIM, Keio), All Rights Reserved.

      Editor:   Masayasu Ishikawa &lt;mimasa@w3.org&gt;
      Revision: $Id: xhtml2.rng,v 1.34 2004/07/21 10:33:05 mimasa Exp $

    Permission to use, copy, modify and distribute this RELAX NG schema
    for XHTML 2.0 and its accompanying documentation for any purpose and
    without fee is hereby granted in perpetuity, provided that the above
    copyright notice and this paragraph appear in all copies. The copyright
    holders make no representation about the suitability of this RELAX NG
    schema for any purpose.

    It is provided "as is" without expressed or implied warranty.
    For details, please refer to the W3C software license at:

      <x:a href="http://www.w3.org/Consortium/Legal/copyright-software"
      >http://www.w3.org/Consortium/Legal/copyright-software</x:a>
  </x:pre>

  <div>
    <x:h2>XHTML 2.0 modules</x:h2>

    <x:h3>Attribute Collections Module</x:h3>
    <include href="xhtml-attribs-2.rng"/>

    <x:h3>Document Module</x:h3>
    <include href="xhtml-document-2.rng"/>

    <x:h3>Structural Module</x:h3>
    <include href="xhtml-structural-2.rng"/>

    <x:h3>Text Module</x:h3>
    <include href="xhtml-text-2.rng"/>

    <x:h3>Hypertext Module</x:h3>
    <include href="xhtml-hypertext-2.rng"/>

    <x:h3>List Module</x:h3>
    <include href="xhtml-list-2.rng"/>
```

```
  <x:h3>Metainformation Module</x:h3>
  <include href="xhtml-meta-2.rng"/>

  <x:h3>Object Module</x:h3>
  <include href="xhtml-object-2.rng"/>

  <x:h3>Scripting Module</x:h3>
  <include href="xhtml-script-2.rng"/>

  <x:h3>Style Attribute Module</x:h3>
  <include href="xhtml-inlstyle-2.rng"/>

  <x:h3>Style Sheet Module</x:h3>
  <include href="xhtml-style-2.rng"/>

  <x:h3>Tables Module</x:h3>
  <include href="xhtml-table-2.rng"/>

  <x:h3>Support Modules</x:h3>

  <x:h4>Datatypes Module</x:h4>
  <include href="xhtml-datatypes-2.rng"/>

  <x:h4>Events Module</x:h4>
  <include href="xhtml-events-2.rng"/>

  <x:h4>Param Module</x:h4>
  <include href="xhtml-param-2.rng"/>

  <x:h4>Caption Module</x:h4>
  <include href="xhtml-caption-2.rng"/>
</div>


<div>
  <x:h2>XML Events module</x:h2>
  <include href="xml-events-1.rng"/>
</div>

<div>
  <x:h2>Ruby module</x:h2>

  <include href="full-ruby-1.rng">

    <define name="Inline.class">
      <notAllowed/>
    </define>

    <define name="NoRuby.content">
      <ref name="Text.model"/>
    </define>

  </include>

  <define name="Inline.model">
    <notAllowed/>
```

```
    </define>

    <define name="Text.class" combine="choice">
      <ref name="ruby"/>
    </define>
  </div>

  <div>
    <x:h2>XForms module</x:h2>
    <x:p>To-Do: work out integration of XForms</x:p>
    <!--include href="xforms-11.rng"/-->
  </div>

  <div>
    <x:h2>XML Schema instance module</x:h2>
    <include href="XMLSchema-instance.rng"/>
  </div>

</grammar>
```

# C. XHTML RELAX NG Module Implementations

This appendix is *normative*.

This appendix contains implementations of the modules defined in this specification. These module implementations can be used in other XHTML Family Document Types.

## C.1. XHTML Module Implementations

This section contains the formal definition of each of the XHTML Abstract Modules as a RELAX NG module.

### C.1.1. Attribute Collections

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Attribute Collections Module</x:h1>

  <div>
    <x:h2>Core Attribute Collection</x:h2>
    <include href="xhtml-core-attrib-2.rng"/>
  </div>

  <div>
    <x:h2>I18N Attribute Collection</x:h2>
    <include href="xhtml-i18n-attrib-2.rng"/>
  </div>

  <div>
    <x:h2>Bi-directional Text Collection</x:h2>
    <include href="xhtml-bidi-attrib-2.rng"/>
  </div>

  <div>
    <x:h2>Edit Collection</x:h2>
    <include href="xhtml-edit-attrib-2.rng"/>
  </div>

  <div>
    <x:h2>Embedding Attributes Collection</x:h2>
    <include href="xhtml-embed-attrib-2.rng"/>
  </div>

  <div>
    <x:h2>Hypertext Attributes Collection</x:h2>
    <include href="xhtml-hypertext-attrib-2.rng"/>
  </div>

  <div>
    <x:h2>Metadata Attribute Collection</x:h2>
    <include href="xhtml-meta-attrib-2.rng"/>
```

```
  </div>

  <div>
    <x:h2>Image Map Attributes Collection</x:h2>
    <include href="xhtml-imagemap-attrib-2.rng"/>
  </div>

  <define name="Common.extra.attrib">
    <empty/>
  </define>

  <define name="Common.attrib">
    <ref name="Common.extra.attrib"/>
  </define>

  <define name="CommonIdRequired.attrib">
    <attribute name="id">
      <ref name="ID.datatype"/>
    </attribute>
    <ref name="class.attrib"/>
    <ref name="title.attrib"/>
    <ref name="Common.extra.attrib"/>
  </define>

  <define name="CommonTypeRequired.attrib">
    <ref name="src.attrib"/>
    <attribute name="type">
      <ref name="ContentTypes.datatype"/>
    </attribute>
    <ref name="Common.extra.attrib"/>
  </define>

</grammar>
```

## C.1.2. Document

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Document Module</x:h1>

  <start>
    <ref name="html"/>
  </start>

  <div>
    <x:h2>The html element</x:h2>

    <define name="html">
      <element name="html">
        <ref name="html.attlist"/>
        <ref name="head"/>
        <ref name="body"/>
      </element>
    </define>
```

```
  <define name="html.attlist">
    <ref name="Common.attrib"/>
    <ref name="version.attrib"/>
    <ref name="XSI.schemaLocation"/>
  </define>

  <define name="version.attrib">
    <optional>
      <attribute name="version">
        <ref name="CDATA.datatype"/>
      </attribute>
    </optional>
  </define>
</div>

<div>
  <x:h2>The head element</x:h2>

  <define name="head">
    <element name="head">
      <ref name="head.attlist"/>
      <ref name="head.content"/>
    </element>
  </define>

  <define name="head.attlist">
    <ref name="Common.attrib"/>
  </define>

  <define name="head.content">
    <ref name="title"/>
    <zeroOrMore>
      <choice>
        <ref name="head.misc"/>
      </choice>
    </zeroOrMore>
  </define>

  <define name="head.misc">
    <notAllowed/>
  </define>
</div>

<div>
  <x:h2>The title element</x:h2>

  <define name="title">
    <element name="title">
      <ref name="title.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="title.attlist">
    <ref name="Common.attrib"/>
  </define>
```

```
  </div>

  <div>
    <x:h2>The body element</x:h2>

    <define name="body">
      <element name="body">
        <ref name="body.attlist"/>
        <ref name="Structural.model"/>
      </element>
    </define>

    <define name="body.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>

</grammar>
```

## C.1.3. Structural

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Structural Module</x:h1>

  <div>
    <x:h2>The address element</x:h2>

    <define name="address">
      <element name="address">
        <ref name="address.attlist"/>
        <ref name="Text.model"/>
      </element>
    </define>

    <define name="address.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>

  <div>
    <x:h2>The blockcode element</x:h2>

    <define name="blockcode">
      <element name="blockcode">
        <ref name="blockcode.attlist"/>
        <ref name="blockcode.content"/>
      </element>
    </define>

    <define name="blockcode.attlist">
      <ref name="Common.attrib"/>
    </define>
```

```
  <define name="blockcode.content">
    <ref name="blockcode.model"/>
  </define>
</div>

<div>
  <x:h2>The blockquote element</x:h2>

  <define name="blockquote">
    <element name="blockquote">
      <ref name="blockquote.attlist"/>
      <ref name="blockquote.content"/>
    </element>
  </define>

  <define name="blockquote.attlist">
    <ref name="Common.attrib"/>
  </define>

  <define name="blockquote.content">
    <ref name="blockquote.model"/>
  </define>
</div>

<div>
  <x:h2>The div element</x:h2>

  <define name="div">
    <element name="div">
      <ref name="div.attlist"/>
      <ref name="Flow.model"/>
    </element>
  </define>

  <define name="div.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The heading elements</x:h2>

  <define name="h">
    <element name="h">
      <ref name="Heading.attlist"/>
      <ref name="Heading.content"/>
    </element>
  </define>

  <define name="h1">
    <element name="h1">
      <ref name="Heading.attlist"/>
      <ref name="Heading.content"/>
    </element>
  </define>

  <define name="h2">
```

```
      <element name="h2">
        <ref name="Heading.attlist"/>
        <ref name="Heading.content"/>
      </element>
    </define>

    <define name="h3">
      <element name="h3">
        <ref name="Heading.attlist"/>
        <ref name="Heading.content"/>
      </element>
    </define>

    <define name="h4">
      <element name="h4">
        <ref name="Heading.attlist"/>
        <ref name="Heading.content"/>
      </element>
    </define>

    <define name="h5">
      <element name="h5">
        <ref name="Heading.attlist"/>
        <ref name="Heading.content"/>
      </element>
    </define>

    <define name="h6">
      <element name="h6">
        <ref name="Heading.attlist"/>
        <ref name="Heading.content"/>
      </element>
    </define>

    <define name="Heading.attlist">
      <ref name="Common.attrib"/>
    </define>

    <define name="Heading.content">
      <ref name="Text.model"/>
    </define>
  </div>

  <div>
    <x:h2>The p element</x:h2>

    <define name="p">
      <element name="p">
        <ref name="p.attlist"/>
        <ref name="p.content"/>
      </element>
    </define>

    <define name="p.attlist">
      <ref name="Common.attrib"/>
    </define>
```

```
    <define name="p.content">
      <ref name="p.model"/>
    </define>
  </div>

  <div>
    <x:h2>The pre element</x:h2>

    <define name="pre">
      <element name="pre">
        <ref name="pre.attlist"/>
        <ref name="Text.model"/>
      </element>
    </define>

    <define name="pre.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>

  <div>
    <x:h2>The section element</x:h2>

    <define name="section">
      <element name="section">
        <ref name="section.attlist"/>
        <ref name="Flow.model"/>
      </element>
    </define>

    <define name="section.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>

  <div>
    <x:h2>The separator element</x:h2>

    <define name="separator">
      <element name="separator">
        <ref name="separator.attlist"/>
      </element>
    </define>

    <define name="separator.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>

  <div>
    <x:h2>Content Model</x:h2>

    <define name="Heading.class">
      <choice>
        <ref name="h"/>
        <ref name="h1"/>
        <ref name="h2"/>
```

```
    <ref name="h3"/>
    <ref name="h4"/>
    <ref name="h5"/>
    <ref name="h6"/>
  </choice>
</define>

<define name="Structural.class">
  <choice>
    <ref name="address"/>
    <ref name="blockcode"/>
    <ref name="blockquote"/>
    <ref name="div"/>
    <ref name="p"/>
    <ref name="pre"/>
    <ref name="section"/>
    <ref name="separator"/>
  </choice>
</define>

<define name="blockcode.model">
  <zeroOrMore>
    <choice>
      <text/>
      <ref name="Text.class"/>
      <ref name="Heading.class"/>
      <ref name="Structural.class"/>
      <ref name="List.class"/>
      <ref name="Misc.class"/>
    </choice>
  </zeroOrMore>
</define>

<define name="blockquote.model">
  <zeroOrMore>
    <choice>
      <text/>
      <ref name="Text.class"/>
      <ref name="Heading.class"/>
      <ref name="Structural.class"/>
      <ref name="List.class"/>
      <ref name="Misc.class"/>
    </choice>
  </zeroOrMore>
</define>

<define name="p.model">
  <zeroOrMore>
    <choice>
      <text/>
      <ref name="Text.class"/>
      <ref name="List.class"/>
      <ref name="blockcode"/>
      <ref name="blockquote"/>
      <ref name="pre"/>
      <ref name="table"/>
      <ref name="Misc.class"/>
```

```
        </choice>
      </zeroOrMore>
    </define>

    <define name="Structural.mix">
      <zeroOrMore>
        <choice>
          <ref name="Heading.class"/>
          <ref name="Structural.class"/>
          <ref name="List.class"/>
          <ref name="Misc.class"/>
        </choice>
      </zeroOrMore>
    </define>

    <define name="Structural.model">
      <oneOrMore>
        <ref name="Structural.mix"/>
      </oneOrMore>
    </define>

    <define name="Flow.model">
      <zeroOrMore>
        <choice>
          <text/>
          <ref name="Heading.class"/>
          <ref name="Structural.class"/>
          <ref name="List.class"/>
          <ref name="Text.class"/>
          <ref name="Misc.class"/>
        </choice>
      </zeroOrMore>
    </define>
  </div>

</grammar>
```

## C.1.4. Text

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Text Module</x:h1>

  <div>
    <x:h2>The abbr element</x:h2>

    <define name="abbr">
      <element name="abbr">
        <ref name="abbr.attlist"/>
        <ref name="Text.model"/>
      </element>
    </define>

    <define name="abbr.attlist">
```

```
      <ref name="Common.attrib"/>
    </define>
  </div>

  <div>
    <x:h2>The cite element</x:h2>

    <define name="cite">
      <element name="cite">
        <ref name="cite.attlist"/>
        <ref name="Text.model"/>
      </element>
    </define>

    <define name="cite.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>

  <div>
    <x:h2>The code element</x:h2>

    <define name="code">
      <element name="code">
        <ref name="code.attlist"/>
        <ref name="Text.model"/>
      </element>
    </define>

    <define name="code.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>

  <div>
    <x:h2>The dfn element</x:h2>

    <define name="dfn">
      <element name="dfn">
        <ref name="dfn.attlist"/>
        <ref name="Text.model"/>
      </element>
    </define>

    <define name="dfn.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>

  <div>
    <x:h2>The em element</x:h2>

    <define name="em">
      <element name="em">
        <ref name="em.attlist"/>
        <ref name="Text.model"/>
      </element>
```

```
  </define>

  <define name="em.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The kbd element</x:h2>

  <define name="kbd">
    <element name="kbd">
      <ref name="kbd.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="kbd.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The l element</x:h2>

  <define name="l">
    <element name="l">
      <ref name="l.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="l.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The quote element</x:h2>

  <define name="quote">
    <element name="quote">
      <ref name="quote.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="quote.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The samp element</x:h2>

  <define name="samp">
    <element name="samp">
```

```
      <ref name="samp.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="samp.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The span element</x:h2>

  <define name="span">
    <element name="span">
      <ref name="span.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="span.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The strong element</x:h2>

  <define name="strong">
    <element name="strong">
      <ref name="strong.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="strong.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The sub element</x:h2>

  <define name="sub">
    <element name="sub">
      <ref name="sub.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="sub.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The sup element</x:h2>
```

```
  <define name="sup">
    <element name="sup">
      <ref name="sup.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="sup.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The var element</x:h2>

  <define name="var">
    <element name="var">
      <ref name="var.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="var.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:p>these can occur at block or inline level</x:p>

  <define name="Misc.class">
    <empty/>
  </define>
</div>

<div>
  <x:h2>Content Model</x:h2>

  <define name="Text.class">
    <choice>
      <ref name="abbr"/>
      <ref name="cite"/>
      <ref name="code"/>
      <ref name="dfn"/>
      <ref name="em"/>
      <ref name="kbd"/>
      <ref name="l"/>
      <ref name="quote"/>
      <ref name="samp"/>
      <ref name="span"/>
      <ref name="strong"/>
      <ref name="sub"/>
      <ref name="sup"/>
      <ref name="var"/>
    </choice>
  </define>
```

```
    <define name="Text.model">
      <zeroOrMore>
        <choice>
          <text/>
          <ref name="Text.class"/>
          <ref name="Misc.class"/>
        </choice>
      </zeroOrMore>
    </define>
  </div>

</grammar>
```

## C.1.5. Hypertext

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Hypertext Module</x:h1>

  <div>
    <x:h2>The a element</x:h2>

    <define name="a">
      <element name="a">
        <ref name="a.attlist"/>
        <ref name="Text.model"/>
      </element>
    </define>

    <define name="a.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>

  <define name="Text.class" combine="choice">
    <ref name="a"/>
  </define>

</grammar>
```

## C.1.6. List

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>List Module</x:h1>

  <div>
    <x:h2>The dl element</x:h2>

    <define name="dl">
      <element name="dl">
```

```
      <ref name="dl.attlist"/>
      <optional>
        <ref name="label"/>
      </optional>
      <choice>
        <oneOrMore>
          <choice>
            <ref name="dt"/>
            <ref name="dd"/>
          </choice>
        </oneOrMore>
        <oneOrMore>
          <ref name="di"/>
        </oneOrMore>
      </choice>
    </element>
  </define>

  <define name="dl.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The di element</x:h2>

  <define name="di">
    <element name="di">
      <ref name="di.attlist"/>
      <oneOrMore>
        <ref name="dt"/>
      </oneOrMore>
      <zeroOrMore>
        <ref name="dd"/>
      </zeroOrMore>
    </element>
  </define>

  <define name="di.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The dt element</x:h2>

  <define name="dt">
    <element name="dt">
      <ref name="dt.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="dt.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>
```

```
<div>
  <x:h2>The dd element</x:h2>

  <define name="dd">
    <element name="dd">
      <ref name="dd.attlist"/>
      <ref name="Flow.model"/>
    </element>
  </define>

  <define name="dd.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The label element</x:h2>

  <define name="label">
    <element name="label">
      <ref name="label.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="label.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The nl element</x:h2>

  <define name="nl">
    <element name="nl">
      <ref name="nl.attlist"/>
      <ref name="label"/>
      <oneOrMore>
        <ref name="li"/>
      </oneOrMore>
    </element>
  </define>

  <define name="nl.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The ol element</x:h2>

  <define name="ol">
    <element name="ol">
      <ref name="ol.attlist"/>
      <optional>
        <ref name="label"/>
```

```
      </optional>
      <oneOrMore>
        <ref name="li-in-ol"/>
      </oneOrMore>
    </element>
  </define>

  <define name="ol.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The ul element</x:h2>

  <define name="ul">
    <element name="ul">
      <ref name="ul.attlist"/>
      <optional>
        <ref name="label"/>
      </optional>
      <oneOrMore>
        <ref name="li"/>
      </oneOrMore>
    </element>
  </define>

  <define name="ul.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The li element</x:h2>

  <define name="li">
    <element name="li">
      <ref name="li.attlist"/>
      <ref name="Flow.model"/>
    </element>
  </define>

  <define name="li.attlist">
    <ref name="Common.attrib"/>
  </define>

  <define name="li-in-ol">
    <element name="li">
      <ref name="li-in-ol.attlist"/>
      <ref name="Flow.model"/>
    </element>
  </define>

  <define name="li-in-ol.attlist">
    <ref name="Common.attrib"/>
    <ref name="value.attrib"/>
  </define>
```

```
    <define name="value.attrib">
      <optional>
        <attribute name="value">
          <ref name="Number.datatype"/>
        </attribute>
      </optional>
    </define>
  </div>

  <div>
    <x:h2>List Content Set</x:h2>

    <define name="List.class">
      <choice>
        <ref name="ul"/>
        <ref name="nl"/>
        <ref name="ol"/>
        <ref name="dl"/>
      </choice>
    </define>
  </div>

</grammar>
```

## C.1.7. Core Attributes

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Core Attributes Module</x:h1>

  <div>
    <x:h2>Core Attribute Collection</x:h2>

    <define name="class.attrib">
      <optional>
        <attribute name="class">
          <ref name="NMTOKENS.datatype"/>
        </attribute>
      </optional>
    </define>

    <define name="id.attrib">
      <optional>
        <attribute name="id">
          <ref name="ID.datatype"/>
        </attribute>
      </optional>
    </define>

    <define name="title.attrib">
      <optional>
        <attribute name="title">
          <ref name="Text.datatype"/>
```

```
        </attribute>
      </optional>
    </define>

    <define name="Core.attrib">
      <ref name="id.attrib"/>
      <ref name="class.attrib"/>
      <ref name="title.attrib"/>
    </define>
  </div>

  <define name="Common.attrib" combine="interleave">
    <ref name="Core.attrib"/>
  </define>

  <define name="CommonTypeRequired.attrib" combine="interleave">
    <ref name="Core.attrib"/>
  </define>

</grammar>
```

## C.1.8. Hypertext Attributes

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Hypertext Attributes Module</x:h1>

  <div>
    <x:h2>Hypertext Attributes Collection</x:h2>

    <define name="href.attrib">
      <optional>
        <attribute name="href">
          <ref name="URI.datatype"/>
        </attribute>
      </optional>
    </define>

    <define name="hreftype.attrib">
      <optional>
        <attribute name="hreftype">
          <ref name="ContentTypes.datatype"/>
        </attribute>
      </optional>
    </define>

    <define name="hreflang.attrib">
      <optional>
        <attribute name="hreflang">
          <ref name="LanguageCodes.datatype"/>
        </attribute>
      </optional>
    </define>
```

```
<define name="cite.attrib">
  <optional>
    <attribute name="cite">
      <ref name="URI.datatype"/>
    </attribute>
  </optional>
</define>

<define name="target.attrib">
  <optional>
    <attribute name="target">
      <ref name="HrefTarget.datatype"/>
    </attribute>
  </optional>
</define>

<define name="access.attrib">
  <optional>
    <attribute name="access">
      <ref name="QName.datatype"/>
    </attribute>
  </optional>
</define>

<define name="nextfocus.attrib">
  <optional>
    <attribute name="nextfocus">
      <ref name="IDREF.datatype"/>
    </attribute>
  </optional>
</define>

<define name="prevfocus.attrib">
  <optional>
    <attribute name="prevfocus">
      <ref name="IDREF.datatype"/>
    </attribute>
  </optional>
</define>

<define name="base.attrib">
  <optional>
    <attribute name="xml:base">
      <ref name="URI.datatype"/>
    </attribute>
  </optional>
</define>

<define name="Hypertext.attrib">
  <ref name="href.attrib"/>
  <ref name="hreftype.attrib"/>
  <ref name="hreflang.attrib"/>
  <ref name="cite.attrib"/>
  <ref name="target.attrib"/>
  <ref name="access.attrib"/>
  <ref name="nextfocus.attrib"/>
  <ref name="prevfocus.attrib"/>
```

```
      <ref name="base.attrib"/>
    </define>
  </div>

  <define name="Common.attrib" combine="interleave">
    <ref name="Hypertext.attrib"/>
  </define>

  <define name="CommonIdRequired.attrib" combine="interleave">
    <ref name="Hypertext.attrib"/>
  </define>

  <define name="CommonTypeRequired.attrib" combine="interleave">
    <ref name="Hypertext.attrib"/>
  </define>

</grammar>
```

## C.1.9. I18N Attribute

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>I18N Attribute Module</x:h1>

  <div>
    <x:h2>I18N Attribute Collection</x:h2>

    <define name="lang.attrib">
      <optional>
        <attribute name="xml:lang">
          <ref name="LanguageCode.datatype"/>
        </attribute>
      </optional>
    </define>

    <define name="I18n.attrib">
      <ref name="lang.attrib"/>
    </define>
  </div>

  <define name="Common.attrib" combine="interleave">
    <ref name="I18n.attrib"/>
  </define>

  <define name="CommonIdRequired.attrib" combine="interleave">
    <ref name="I18n.attrib"/>
  </define>

  <define name="CommonTypeRequired.attrib" combine="interleave">
    <ref name="I18n.attrib"/>
  </define>

</grammar>
```

## C.1.10. Bi-directional Text Attribute

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Bi-directional Text Attribute Module</x:h1>

  <div>
    <x:h2>Bi-directional Text Collection</x:h2>

    <define name="dir.attrib">
      <optional>
        <attribute name="dir">
          <choice>
            <value>ltr</value>
            <value>rtl</value>
            <value>lro</value>
            <value>rlo</value>
          </choice>
        </attribute>
      </optional>
    </define>

    <define name="Bidi.attrib">
      <ref name="dir.attrib"/>
    </define>
  </div>

  <define name="Common.attrib" combine="interleave">
    <ref name="Bidi.attrib"/>
  </define>

  <define name="CommonIdRequired.attrib" combine="interleave">
    <ref name="Bidi.attrib"/>
  </define>

  <define name="CommonTypeRequired.attrib" combine="interleave">
    <ref name="Bidi.attrib"/>
  </define>

</grammar>
```

## C.1.11. Edit Attributes

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Edit Attribute Module</x:h1>

  <div>
    <x:h2>Edit Collection</x:h2>

    <define name="edit.attrib">
      <optional>
```

```
        <attribute name="edit">
          <choice>
            <value>inserted</value>
            <value>deleted</value>
            <value>changed</value>
            <value>moved</value>
          </choice>
        </attribute>
      </optional>
    </define>

    <define name="datetime.attrib">
      <optional>
        <attribute name="datetime">
          <ref name="Datetime.datatype"/>
        </attribute>
      </optional>
    </define>

    <define name="Edit.attrib">
      <ref name="edit.attrib"/>
      <ref name="datetime.attrib"/>
    </define>
  </div>

  <define name="Common.attrib" combine="interleave">
    <ref name="Edit.attrib"/>
  </define>

  <define name="CommonIdRequired.attrib" combine="interleave">
    <ref name="Edit.attrib"/>
  </define>

  <define name="CommonTypeRequired.attrib" combine="interleave">
    <ref name="Edit.attrib"/>
  </define>

</grammar>
```

## C.1.12. Embedding Attributes

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Embedding Attributes Module</x:h1>

  <div>
    <x:h2>Embedding Attributes Collection</x:h2>

    <define name="src.attrib">
      <optional>
        <attribute name="src">
          <ref name="URI.datatype"/>
        </attribute>
      </optional>
```

```
    </define>

    <define name="type.attrib">
      <optional>
        <attribute name="type">
          <ref name="ContentTypes.datatype"/>
        </attribute>
      </optional>
    </define>

    <define name="Embedding.attrib">
      <ref name="src.attrib"/>
      <ref name="type.attrib"/>
    </define>
  </div>

  <define name="Common.attrib" combine="interleave">
    <ref name="Embedding.attrib"/>
  </define>

  <define name="CommonIdRequired.attrib" combine="interleave">
    <ref name="Embedding.attrib"/>
  </define>

</grammar>
```

## C.1.13. Image Map Attributes

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Image Map Attributes Module</x:h1>

  <div>
    <x:h2>Image Map Attributes Collection</x:h2>

    <define name="usemap.attrib">
      <optional>
        <attribute name="usemap">
          <ref name="URI.datatype"/>
        </attribute>
      </optional>
    </define>

    <define name="ismap.attrib">
      <optional>
        <attribute name="ismap">
          <value>ismap</value>
        </attribute>
      </optional>
    </define>

    <define name="shape.attrib">
      <optional>
        <attribute name="shape">
```

```
          <choice>
            <value>default</value>
            <value>rect</value>
            <value>circle</value>
            <value>poly</value>
          </choice>
        </attribute>
      </optional>
    </define>

    <define name="coords.attrib">
      <optional>
        <attribute name="coords">
          <ref name="Coordinates.datatype"/>
        </attribute>
      </optional>
    </define>

    <define name="Map.attrib">
      <ref name="usemap.attrib"/>
      <ref name="ismap.attrib"/>
      <ref name="shape.attrib"/>
      <ref name="coords.attrib"/>
    </define>
  </div>

  <define name="Common.attrib" combine="interleave">
    <ref name="Map.attrib"/>
  </define>

  <define name="CommonIdRequired.attrib" combine="interleave">
    <ref name="Map.attrib"/>
  </define>

  <define name="CommonTypeRequired.attrib" combine="interleave">
    <ref name="Map.attrib"/>
  </define>

</grammar>
```

## C.1.14. Metainformation Attributes

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

<x:h1>MetaInformation Attributes Module</x:h1>

  <div>
    <x:h2>Metadata Attribute Collection</x:h2>

    <define name="about.attrib">
      <optional>
        <attribute name="about">
          <ref name="URI.datatype"/>
        </attribute>
```

```
  </optional>
</define>

<define name="content.attrib">
  <optional>
    <attribute name="content">
      <ref name="CDATA.datatype"/>
    </attribute>
  </optional>
</define>

<define name="datatype.attrib">
  <optional>
    <attribute name="datatype">
      <ref name="QName.datatype"/>
    </attribute>
  </optional>
</define>

<define name="property.attrib">
  <optional>
    <attribute name="property">
      <ref name="QName.datatype"/>
    </attribute>
  </optional>
</define>

<define name="rel.attrib">
  <optional>
    <attribute name="rel">
      <!--ref name="QName.datatype"/-->
      <ref name="LinkTypes.datatype"/>
    </attribute>
  </optional>
</define>

<define name="resource.attrib">
  <optional>
    <attribute name="resource">
      <ref name="URI.datatype"/>
    </attribute>
  </optional>
</define>

<define name="restype.attrib">
  <optional>
    <attribute name="restype">
      <ref name="ContentTypes.datatype"/>
    </attribute>
  </optional>
</define>

<define name="rev.attrib">
  <optional>
    <attribute name="rev">
      <!--ref name="QName.datatype"/-->
      <ref name="LinkTypes.datatype"/>
```

```
        </attribute>
      </optional>
    </define>

    <define name="Metadata.attrib">
      <ref name="about.attrib"/>
      <ref name="content.attrib"/>
      <ref name="datatype.attrib"/>
      <ref name="property.attrib"/>
      <ref name="rel.attrib"/>
      <ref name="resource.attrib"/>
      <ref name="restype.attrib"/>
      <ref name="rev.attrib"/>
    </define>
  </div>

  <define name="Common.attrib" combine="interleave">
    <ref name="Metadata.attrib"/>
  </define>

  <define name="CommonIdRequired.attrib" combine="interleave">
    <ref name="Metadata.attrib"/>
  </define>

  <define name="CommonTypeRequired.attrib" combine="interleave">
    <ref name="Metadata.attrib"/>
  </define>

</grammar>
```

## C.1.15. Metainformation

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Metainformation Module</x:h1>

  <div>
    <x:h2>The link element</x:h2>

    <define name="link">
      <element name="link">
        <ref name="link.attlist"/>
        <zeroOrMore>
          <choice>
            <ref name="link"/>
            <ref name="meta"/>
          </choice>
        </zeroOrMore>
      </element>
    </define>

    <define name="link.attlist">
      <ref name="Common.attrib"/>
    </define>
```

```
  </div>

  <define name="head.misc" combine="choice">
    <ref name="link"/>
  </define>

  <define name="Structural.class" combine="choice">
    <ref name="link"/>
  </define>

  <define name="Text.class" combine="choice">
    <ref name="link"/>
  </define>

  <div>
    <x:h2>The meta element</x:h2>

    <define name="meta">
      <element name="meta">
        <ref name="meta.attlist"/>
        <choice>
          <ref name="Text.model"/>
          <oneOrMore>
            <ref name="meta"/>
          </oneOrMore>
        </choice>
      </element>
    </define>

    <define name="meta.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>

  <define name="head.misc" combine="choice">
    <ref name="meta"/>
  </define>

  <define name="Structural.class" combine="choice">
    <ref name="meta"/>
  </define>

  <define name="Text.class" combine="choice">
    <ref name="meta"/>
  </define>

</grammar>
```

## C.1.16. Object

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Object Module</x:h1>
```

```
<div>
  <x:h2>The object element</x:h2>

  <define name="object">
    <element name="object">
      <ref name="object.attlist"/>
      <optional>
        <ref name="caption"/>
      </optional>
      <optional>
        <ref name="standby"/>
      </optional>
      <zeroOrMore>
        <ref name="param"/>
      </zeroOrMore>
      <ref name="Flow.model"/>
    </element>
  </define>

  <define name="object.attlist">
    <ref name="Common.attrib"/>
    <optional>
      <attribute name="archive">
        <ref name="URIs.datatype"/>
      </attribute>
    </optional>
    <optional>
      <attribute name="content-length">
        <ref name="Number.datatype"/>
      </attribute>
    </optional>
    <optional>
      <attribute name="declare">
        <value>declare</value>
      </attribute>
    </optional>
  </define>
</div>

<div>
  <x:h2>The standby element</x:h2>

  <define name="standby">
    <element name="standby">
      <ref name="standby.attlist"/>
      <ref name="Text.model"/>
    </element>
  </define>

  <define name="standby.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<define name="Text.class" combine="choice">
```

```
    <ref name="object"/>
  </define>

</grammar>
```

## C.1.17. Scripting

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Scripting Module</x:h1>

  <div>
    <x:h2>The script element</x:h2>

    <define name="script">
      <element name="script">
        <ref name="script.attlist"/>
        <choice>
          <text/>
          <ref name="script"/>
        </choice>
      </element>
    </define>

    <define name="script.attlist">
      <ref name="CommonTypeRequired.attrib"/>
      <optional>
        <attribute name="charset">
          <ref name="Charset.datatype"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="declare">
          <value>declare</value>
        </attribute>
      </optional>
      <optional>
        <attribute name="xml:space">
          <value>preserve</value>
        </attribute>
      </optional>
    </define>
  </div>

  <define name="head.misc" combine="choice">
    <ref name="script"/>
  </define>

  <define name="Script.class">
    <ref name="script"/>
  </define>

  <define name="Text.class" combine="choice">
    <ref name="Script.class"/>
```

```
  </define>

  <define name="Structural.class" combine="choice">
    <ref name="Script.class"/>
  </define>

</grammar>
```

## C.1.18. Style Attribute

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Style Attribute Module</x:h1>

  <define name="style.attrib">
    <optional>
      <attribute name="style"/>
    </optional>
  </define>

  <define name="Common.extra.attrib" combine="interleave">
    <ref name="style.attrib"/>
  </define>

</grammar>
```

## C.1.19. Style Sheet

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Style Module</x:h1>

  <div>
    <x:h2>The style element</x:h2>

    <define name="style">
      <element name="style">
        <ref name="style.attlist"/>
        <text/>
      </element>
    </define>

    <define name="style.attlist">
      <ref name="Common.attrib"/>
      <optional>
        <attribute name="disabled">
          <value>disabled</value>
        </attribute>
      </optional>
      <optional>
        <attribute name="media">
          <ref name="MediaDesc.datatype"/>
```

```
        </attribute>
      </optional>
    </define>
  </div>

  <define name="head.misc" combine="choice">
    <ref name="style"/>
  </define>

</grammar>
```

## C.1.20. Tables

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Tables Module</x:h1>

  <x:p>Note. Also include the Caption Module when this module is used.</x:p>

  <div>
    <x:h2>The table element</x:h2>

    <define name="table">
      <element name="table">
        <ref name="table.attlist"/>
        <optional>
          <ref name="caption"/>
        </optional>
        <optional>
          <ref name="summary"/>
        </optional>
        <choice>
          <zeroOrMore>
            <ref name="col"/>
          </zeroOrMore>
          <zeroOrMore>
            <ref name="colgroup"/>
          </zeroOrMore>
        </choice>
        <choice>
          <group>
            <optional>
              <ref name="thead"/>
            </optional>
            <optional>
              <ref name="tfoot"/>
            </optional>
            <oneOrMore>
              <ref name="tbody"/>
            </oneOrMore>
          </group>
          <oneOrMore>
            <ref name="tr"/>
```

```
        </oneOrMore>
      </choice>
    </element>
  </define>

  <define name="table.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The summary element</x:h2>

  <define name="summary">
    <element name="summary">
      <ref name="summary.attlist"/>
      <ref name="Flow.model"/>
    </element>
  </define>

  <define name="summary.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The col element</x:h2>

  <define name="col">
    <element name="col">
      <ref name="col.attlist"/>
    </element>
  </define>

  <define name="col.attlist">
    <ref name="Common.attrib"/>
    <ref name="span.attrib"/>
  </define>
</div>

<div>
  <x:h2>The colgroup element</x:h2>

  <define name="colgroup">
    <element name="colgroup">
      <ref name="colgroup.attlist"/>
      <zeroOrMore>
        <ref name="col"/>
      </zeroOrMore>
    </element>
  </define>

  <define name="colgroup.attlist">
    <ref name="Common.attrib"/>
    <ref name="span.attrib"/>
  </define>
</div>
```

```
<div>
  <x:h2>The thead element</x:h2>

  <define name="thead">
    <element name="thead">
      <ref name="thead.attlist"/>
      <oneOrMore>
        <ref name="tr"/>
      </oneOrMore>
    </element>
  </define>

  <define name="thead.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The tfoot element</x:h2>

  <define name="tfoot">
    <element name="tfoot">
      <ref name="tfoot.attlist"/>
      <oneOrMore>
        <ref name="tr"/>
      </oneOrMore>
    </element>
  </define>

  <define name="tfoot.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The tbody element</x:h2>

  <define name="tbody">
    <element name="tbody">
      <ref name="tbody.attlist"/>
      <oneOrMore>
        <ref name="tr"/>
      </oneOrMore>
    </element>
  </define>

  <define name="tbody.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The tr element</x:h2>

  <define name="tr">
    <element name="tr">
```

```
      <ref name="tr.attlist"/>
      <oneOrMore>
        <choice>
          <ref name="th"/>
          <ref name="td"/>
        </choice>
      </oneOrMore>
    </element>
  </define>

  <define name="tr.attlist">
    <ref name="Common.attrib"/>
  </define>
</div>

<div>
  <x:h2>The th element</x:h2>

  <define name="th">
    <element name="th">
      <ref name="th.attlist"/>
      <ref name="Flow.model"/>
    </element>
  </define>

  <define name="th.attlist">
    <ref name="Cell.attrib"/>
  </define>
</div>

<div>
  <x:h2>The td element</x:h2>

  <define name="td">
    <element name="td">
      <ref name="td.attlist"/>
      <ref name="Flow.model"/>
    </element>
  </define>

  <define name="td.attlist">
    <ref name="Cell.attrib"/>
  </define>
</div>

<div>
  <x:h2>Attribute definitions</x:h2>

  <define name="span.attrib">
    <optional>
      <attribute name="span" a:defaultValue="1">
        <ref name="spanNumber.datatype"/>
      </attribute>
    </optional>
  </define>

  <define name="Cell.attrib">
```

```
      <ref name="Common.attrib"/>
      <optional>
        <attribute name="abbr">
          <ref name="Text.datatype"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="axis"/>
      </optional>
      <optional>
        <attribute name="colspan" a:defaultValue="1">
          <ref name="Number.datatype"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="headers">
          <ref name="IDREFS.datatype"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="rowspan" a:defaultValue="1">
          <ref name="Number.datatype"/>
        </attribute>
      </optional>
      <ref name="scope.attrib"/>
    </define>

    <define name="scope.attrib">
      <optional>
        <attribute name="scope">
          <choice>
            <value>row</value>
            <value>col</value>
            <value>rowgroup</value>
            <value>colgroup</value>
          </choice>
        </attribute>
      </optional>
    </define>
  </div>

  <define name="Structural.class" combine="choice">
    <ref name="table"/>
  </define>

</grammar>
```

## C.2. XHTML RELAX NG Support Modules

The modules in this section are elements and attributes of the XHTML RELAX NG implementation that, while hidden from casual users, are important to understand when creating derivative markup languages using the Modularization architecture.

## C.2.1. Datatypes

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

  <x:h1>Datatypes Module</x:h1>

  <div>
    <x:h2>Datatypes defined in XML 1.0</x:h2>

    <define name="CDATA.datatype">
      <text/>
    </define>

    <define name="ID.datatype">
      <data type="ID"/>
    </define>

    <define name="IDREF.datatype">
      <data type="IDREF"/>
    </define>

    <define name="IDREFS.datatype">
      <data type="IDREFS"/>
    </define>

    <define name="NAME.datatype">
      <data type="Name"/>
    </define>

    <define name="NMTOKEN.datatype">
      <data type="NMTOKEN"/>
    </define>

    <define name="NMTOKENS.datatype">
      <data type="NMTOKENS"/>
    </define>
  </div>

  <div>
    <x:h2>Additional Datatypes</x:h2>

    <define name="Character.datatype">
      <x:p>A single character, as per section 2.2 of [XML].</x:p>
      <data type="string">
        <param name="length">1</param>
      </data>
    </define>

    <define name="Charset.datatype">
      <x:p>A character encoding, as per [RFC2045].</x:p>
      <data type="string">
        <param name="pattern">\S+</param>
      </data>
```

```
</define>

<define name="ContentTypes.datatype">
  <x:p>A list of media ranges with optional accept parameters,
    as defined in section 14.1 of [RFC2616] as the field value
    of the accept request header.</x:p>
  <text/>
</define>

<define name="Coordinates.datatype">
  <x:p>Comma separated list of Lengths used in defining areas.</x:p>
  <data type="string">
    <param name="pattern">(\d+|\d+(\.\d+)?%)(,\s*(\d+|\d+(\.\d+)?%))*</param>
  </data>
</define>

<define name="Datetime.datatype">
  <x:p>Date and time information, as defined by the type dateTime
    in [XMLSCHEMA].</x:p>
  <data type="dateTime"/>
</define>

<define name="HrefTarget.datatype">
  <x:p>Name used as destination for results of certain actions.</x:p>
  <ref name="NMTOKEN.datatype"/>
</define>

<define name="LanguageCode.datatype">
  <x:p>A language code, as per [RFC3066].</x:p>
  <data type="language"/>
</define>

<define name="LanguageCodes.datatype">
  <x:p>A comma-separated list of language ranges.</x:p>
  <text/>
</define>

<define name="Length.datatype">
  <x:p>The value may be either in pixels or a percentage of the available
    horizontal or vertical space. Thus, the value "50%" means half of
    the available space.</x:p>
  <data type="string">
    <param name="pattern">(\d+|\d+(\.\d+)?%)</param>
  </data>
</define>

<define name="LocationPath.datatype">
  <x:p>A location path as defined in [XPATH].</x:p>
  <text/>
</define>

<define name="LinkTypes.datatype">
  <x:p>The value is a QName.</x:p>
  <choice>
    <value>alternate</value>
    <value>start</value>
    <value>next</value>
```

```
    <value>prev</value>
    <value>up</value>
    <value>contents</value>
    <value>index</value>
    <value>glossary</value>
    <value>copyright</value>
    <value>chapter</value>
    <value>section</value>
    <value>subsection</value>
    <value>appendix</value>
    <value>help</value>
    <value>bookmark</value>
    <value>meta</value>
    <value>icon</value>
    <value>p3pv1</value>
    <value>profile</value>
    <ref name="QName.datatype"/>
  </choice>
</define>

<define name="MediaDesc.datatype">
  <x:p>A comma-separated list of media descriptors as described by [CSS].
    The default is all.</x:p>
  <data type="string">
    <param name="pattern">[^,]+(,\s*[^,]+)*</param>
  </data>
</define>

<define name="Number.datatype">
  <x:p>One or more digits (NUMBER).</x:p>
  <data type="nonNegativeInteger">
    <param name="pattern">[0-9]+</param>
  </data>
</define>

<define name="spanNumber.datatype">
  <x:p>span: this attribute value must be an integer > 0;
    the default value is 1.</x:p>
  <data type="positiveInteger">
    <param name="pattern">[0-9]+</param>
  </data>
</define>

<define name="QName.datatype">
  <x:p>An [XMLNS]-qualified name.</x:p>
  <data type="QName"/>
</define>

<define name="Shape.datatype">
  <x:p>The shape of a region.</x:p>
  <choice>
    <value>rect</value>
    <value>circle</value>
    <value>poly</value>
    <value>default</value>
  </choice>
</define>
```

```
    <define name="Text.datatype">
      <x:p>Arbitrary textual data, likely meant to be human-readable.</x:p>
      <text/>
    </define>

    <define name="URI.datatype">
      <x:p>A Uniform Resource Identifier Reference, as defined by the type
        anyURI in [XMLSCHEMA].</x:p>
      <data type="anyURI"/>
    </define>

    <define name="URIs.datatype">
      <x:p>A space-separated list of URIs as defined above.</x:p>
      <list>
        <oneOrMore>
          <data type="anyURI"/>
        </oneOrMore>
      </list>
    </define>
  </div>

</grammar>
```

## C.2.2. Events

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:ev="http://www.w3.org/2001/xml-events"
         xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Events Attribute Collection Module</x:h1>

  <define name="Events.attrib">
    <optional>
      <attribute name="ev:event">
        <ref name="NMTOKEN.datatype"/>
      </attribute>
    </optional>
    <optional>
      <attribute name="ev:observer">
        <ref name="IDREF.datatype"/>
      </attribute>
    </optional>
    <optional>
      <attribute name="ev:target">
        <ref name="IDREF.datatype"/>
      </attribute>
    </optional>
    <optional>
      <attribute name="ev:handler">
        <ref name="URI.datatype"/>
      </attribute>
    </optional>
    <optional>
```

```
      <attribute name="ev:phase" a:defaultValue="default">
        <choice>
          <value>capture</value>
          <value>default</value>
        </choice>
      </attribute>
    </optional>
    <optional>
      <attribute name="ev:propagate" a:defaultValue="continue">
        <choice>
          <value>stop</value>
          <value>continue</value>
        </choice>
      </attribute>
    </optional>
    <optional>
      <attribute name="ev:defaultAction" a:defaultValue="perform">
        <choice>
          <value>cancel</value>
          <value>perform</value>
        </choice>
      </attribute>
    </optional>
  </define>

  <define name="Common.extra.attrib" combine="interleave">
    <ref name="Events.attrib"/>
  </define>

  <define name="head.misc" combine="choice">
    <ref name="listener" ns="http://www.w3.org/2001/xml-events"/>
  </define>

  <define name="Script.class" combine="choice">
    <ref name="listener" ns="http://www.w3.org/2001/xml-events"/>
  </define>

</grammar>
```

## C.2.3. Param

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Param Module</x:h1>

  <div>
    <x:h2>The param element</x:h2>

    <define name="param">
      <element name="param">
        <ref name="param.attlist"/>
      </element>
    </define>
```

```
    <define name="param.attlist">
      <optional>
        <ref name="id.attrib"/>
      </optional>
      <attribute name="name"/>
      <optional>
        <attribute name="value"/>
      </optional>
      <optional>
        <attribute name="valuetype" a:defaultValue="data">
          <choice>
            <value>data</value>
            <value>ref</value>
            <value>object</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="type">
          <ref name="ContentTypes.datatype"/>
        </attribute>
      </optional>
    </define>
  </div>

</grammar>
```

## C.2.4. Caption

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Caption Module</x:h1>

  <div>
    <x:h2>The caption element</x:h2>

    <define name="caption">
      <element name="caption">
        <ref name="caption.attlist"/>
        <ref name="Text.model"/>
      </element>
    </define>

    <define name="caption.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>

</grammar>
```

# C.3. RELAX NG External Modules

These modules are not defined by XHTML, but these definitions are included here for completeness.

## C.3.1. Ruby

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

  <x:h1>Ruby Module in RELAX NG</x:h1>

  <x:pre>
    Ruby Elements

      ruby, rbc, rtc, rb, rt, rp

    This module defines grammars to support ruby annotation markup.
    This module is based on the W3C Ruby Annotation Specification:

      http://www.w3.org/TR/ruby

    Copyright &#xA9;2003 W3C&#xAE; (MIT, ERCIM, Keio), All Rights Reserved.

      Editor:   Masayasu Ishikawa &lt;mimasa@w3.org&gt;
      Revision: $Id: ruby-1.rng,v 1.9 2004/07/21 09:46:41 mimasa Exp $

    Permission to use, copy, modify and distribute this RELAX NG schema
    for Ruby Annotation and its accompanying documentation for any purpose
    and without fee is hereby granted in perpetuity, provided that the above
    copyright notice and this paragraph appear in all copies. The copyright
    holders make no representation about the suitability of this RELAX NG
    schema for any purpose.

    It is provided "as is" without expressed or implied warranty.
    For details, please refer to the W3C software license at:

      <x:a href="http://www.w3.org/Consortium/Legal/copyright-software"
      >http://www.w3.org/Consortium/Legal/copyright-software</x:a>
  </x:pre>

  <div>
    <x:h2>patterns for the content model of the ruby element</x:h2>

    <define name="Ruby.content.simple">
      <x:p>Content model of simple ruby</x:p>
      <group>
        <ref name="rb"/>
        <choice>
          <ref name="rt-simple"/>
          <group>
            <ref name="rp"/>
```

```
          <ref name="rt-simple"/>
          <ref name="rp"/>
        </group>
      </choice>
    </group>
  </define>

  <define name="Ruby.content.complex">
    <x:p>Content model of complex ruby</x:p>
    <group>
      <ref name="rbc"/>
      <ref name="rtc"/>
      <optional>
        <ref name="rtc"/>
      </optional>
    </group>
  </define>

  <define name="Ruby.content">
    <x:p>Simple ruby is used by default</x:p>
    <ref name="Ruby.content.simple"/>
  </define>
</div>

<div>
  <x:h2>Ruby Elements</x:h2>

  <x:h3>ruby element</x:h3>

  <define name="ruby">
    <element name="ruby">
      <ref name="Ruby.content"/>
      <ref name="Ruby.common.attrib"/>
    </element>
  </define>

  <x:h3>rbc (ruby base component) element</x:h3>

  <define name="rbc">
    <element name="rbc">
      <oneOrMore>
        <ref name="rb"/>
      </oneOrMore>
      <ref name="Ruby.common.attrib"/>
    </element>
  </define>

  <x:h3>rtc (ruby text component) element</x:h3>

  <define name="rtc">
    <element name="rtc">
      <oneOrMore>
        <ref name="rt-complex"/>
      </oneOrMore>
      <ref name="Ruby.common.attrib"/>
    </element>
  </define>
```

```
  <x:h3>rb (ruby base) element</x:h3>

  <define name="rb">
    <element name="rb">
      <ref name="NoRuby.content"/>
      <ref name="Ruby.common.attrib"/>
    </element>
  </define>

  <x:h3>rt (ruby text) element</x:h3>

  <define name="rt-simple">
    <x:p>grammar for simple ruby</x:p>
    <x:p>rbspan attribute is not allowed in simple ruby</x:p>
    <element name="rt">
      <ref name="NoRuby.content"/>
      <ref name="Ruby.common.attrib"/>
    </element>
  </define>

  <define name="rt-complex">
    <x:p>grammar for complex ruby</x:p>
    <element name="rt">
      <ref name="NoRuby.content"/>
      <ref name="Ruby.common.attrib"/>
      <optional>
        <attribute name="rbspan" a:defaultValue="1">
          <data type="positiveInteger">
            <param name="pattern">[1-9][0-9]*</param>
          </data>
        </attribute>
      </optional>
    </element>
  </define>

  <x:h3>rp (ruby parenthesis) element</x:h3>

  <define name="rp">
    <element name="rp">
      <text/>
      <ref name="Ruby.common.attrib"/>
    </element>
  </define>
</div>

<div>
  <x:h2>Ruby Common Attributes</x:h2>

  <x:p>Ruby elements are intended to have common attributes of its
    parent markup language. The pattern "Common.attrib" MUST be
    defined to integrate this module.</x:p>

  <define name="Ruby.common.attrib">
    <ref name="Common.attrib"/>
  </define>
</div>
```

```
  <div>
    <x:p>Content models of the rb and the rt elements are intended to
      allow other inline-level elements of its parent markup language,
      but it should not include ruby descendent elements. This RELAX NG
      module itself doesn't check nesting of ruby elements.
      The patterns "Inline.class" and "Inline.model" MUST be defined
      to integrate this module.</x:p>

    <define name="Inline.class" combine="choice">
      <ref name="ruby"/>
    </define>

    <define name="NoRuby.content">
      <ref name="Inline.model"/>
    </define>
  </div>

</grammar>
```

## C.3.2. Ruby Driver for Full Ruby Markup

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml">

  <x:h1>Ruby Module in RELAX NG for full ruby markup</x:h1>

  <x:pre>
    Copyright &#xA9;2003 W3C&#xAE; (MIT, ERCIM, Keio), All Rights Reserved.

      Editor:   Masayasu Ishikawa &lt;mimasa@w3.org&gt;
      Revision: $Id: full-ruby-1.rng,v 1.4 2003/04/30 06:50:03 mimasa Exp $
  </x:pre>

  <include href="ruby-1.rng"/>

  <define name="Ruby.content" combine="choice">
    <x:p>Allow complex ruby markup in addition to simple ruby markup</x:p>
    <ref name="Ruby.content.complex"/>
  </define>

</grammar>
```

## C.3.3. XML Events

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar ns="http://www.w3.org/2001/xml-events"
         xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:ev="http://www.w3.org/2001/xml-events"
         xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

  <x:h1>XML Events Module in RELAX NG</x:h1>
```

```
<x:pre>
  Copyright &#xA9;2003 W3C&#xAE; (MIT, ERCIM, Keio), All Rights Reserved.

    Editor:   Masayasu Ishikawa &lt;mimasa@w3.org&gt;
    Revision: $Id: xml-events-1.rng,v 1.6 2003/04/30 06:50:03 mimasa Exp $

  Permission to use, copy, modify and distribute this RELAX NG schema
  for XML Events and its accompanying documentation for any purpose and
  without fee is hereby granted in perpetuity, provided that the above
  copyright notice and this paragraph appear in all copies. The copyright
  holders make no representation about the suitability of this RELAX NG
  schema for any purpose.

  It is provided "as is" without expressed or implied warranty.
  For details, please refer to the W3C software license at:

    <x:a href="http://www.w3.org/Consortium/Legal/copyright-software"
    >http://www.w3.org/Consortium/Legal/copyright-software</x:a>
</x:pre>

<define name="listener">
  <element name="listener">
    <ref name="listener.attlist"/>
  </element>
</define>

<define name="listener.attlist">
  <optional>
    <attribute name="event">
      <data type="NMTOKEN"/>
    </attribute>
  </optional>
  <optional>
    <attribute name="observer">
      <data type="IDREF"/>
    </attribute>
  </optional>
  <optional>
    <attribute name="target">
      <data type="IDREF"/>
    </attribute>
  </optional>
  <optional>
    <attribute name="handler">
      <data type="anyURI"/>
    </attribute>
  </optional>
  <optional>
    <attribute name="phase" a:defaultValue="default">
      <choice>
        <value>capture</value>
        <value>default</value>
      </choice>
    </attribute>
  </optional>
  <optional>
    <attribute name="propagate" a:defaultValue="continue">
```

```
        <choice>
          <value>stop</value>
          <value>continue</value>
        </choice>
      </attribute>
    </optional>
    <optional>
      <attribute name="defaultAction" a:defaultValue="perform">
        <choice>
          <value>cancel</value>
          <value>perform</value>
        </choice>
      </attribute>
    </optional>
    <optional>
      <attribute name="id">
        <data type="ID"/>
      </attribute>
    </optional>
  </define>

</grammar>
```

## C.3.4. XML Schema instance

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         ns="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

  <define name="XSI.type">
    <attribute name="xsi:type">
      <data type="QName"/>
    </attribute>
  </define>

  <define name="XSI.nil">
    <attribute name="xsi:nil">
      <data type="boolean"/>
    </attribute>
  </define>

  <define name="XSI.schemaLocation">
    <attribute name="xsi:schemaLocation">
      <list>
        <oneOrMore>
          <data type="anyURI"/>
          <data type="anyURI"/>
        </oneOrMore>
      </list>
    </attribute>
  </define>

  <define name="XSI.noNamespaceSchemaLocation">
    <attribute name="xsi:noNamespaceSchemaLocation">
```

```
        <data type="anyURI"/>
      </attribute>
    </define>

  </grammar>
```

# D. XHTML 2.0 Schema

This appendix is *normative*.

This appendix will contain the implementation of the XHTML 2.0 Schema driver file and content model file.

# E. XHTML Schema Module Implementations

This appendix is *normative*.

This appendix will contain implementations of the modules defined in this specification via XML Schema [XMLSCHEMA [p.226] ].

# F. XHTML 2.0 Document Type Definition

This appendix is *normative*.

This appendix will contain the implementation of the XHTML 2.0 DTD driver file and content model file.

## F.1. Issues

Fw: Character entities in HTML and MathML PR #634
State: Open
Resolution: Accepted
User: None

**Notes:**
In principle the WG is in favor of defining unified character entities for XML. See also http://www.w3.org/2003/entities/

Entity management: do we still need it? PR #670
State: Approved
Resolution: Accepted
User: Agree

**Notes:**
We will support both XML Catalogs and SGML Open Catalogs for XHTML 2.

Character entities: do we still need them? PR #671
State: Approved
Resolution: Accepted
User: None

**Notes:**
The WG believes that there's still a need for character entities. We need to find a solution. On 9 September 2003, the WG agrees to retain character entities; DTDs therefore still necessary; it might be possible to supply a DTD that only provides the entities.

# G. XHTML DTD Module Implementations

This appendix is *normative*.

This appendix will contain implementations of the modules defined in this specification. These module implementations can be used in other XHTML Family Document Types.

## G.1. XHTML Modular Framework

In order to take advantage of the XHTML DTD Modules, DTD authors need to define the content model for their DTD. XHTML provides a variety of tools to ease this effort. They are defined in a set of support modules, instantiated by a main Framework module:

Note that the module above references a content model module. This module is defined on a per-document type basis in addition to the document type driver file.

## G.2. XHTML Module Implementations

This section will contain the formal definition of each of the XHTML Abstract Modules as a DTD module.

## G.3. XHTML DTD Support Modules

The modules in this section are elements of the XHTML DTD implementation that, while hidden from casual users, are important to understand when creating derivative markup languages using the Modularization architecture.

# H. Stylesheet for XHTML 2

This appendix is *normative*.

This Appendix defines a normative [CSS2 [p.225] ] stylesheet for XHTML 2. While visual user agents implementing XHTML 2 are not required to support CSS2, they are required to behave as if the following CSS2 styles are in effect.

```
@namespace url("http://www.w3.org/2002/06/xhtml2");

/*  A sample style sheet for XHTML 2.0

    This style sheet describes a very incomplete, sample rendering of
    XHTML 2.0 elements.

    Editor: Masayasu Ishikawa <mimasa@w3.org>
    Revision: $Id: xhtml2.css,v 1.1.2.5 2004/07/21 14:06:53 mimasa Exp $
*/

/* new elements */

section, h, nl, label, l, blockcode, separator
                { display: block; }
section, h, nl, label, l, blockcode
                { unicode-bidi: embed }
nl              { margin: 1.33em 0 }
summary, standby
                { display: none }
blockcode       { font-family: monospace; white-space: pre }
separator       { border: 1px inset }
h               { display: block; font-weight: bolder; }
body h {
  font-size: 2em;
  margin: .67em 0;
}

section h {
  font-size: 1.5em;
  margin: .83em 0;
}

section section h {
  font-size: 1.17em;
  margin: 1em 0;
}

section section section h, p, blockquote, ul, ol, dl
                { margin: 1.33em 0; }

section section section section h {
  font-size: .83em;
  line-height: 1.17em;
  margin: 1.67em 0;
}
```

```
section section section section section h {
  font-size: .67em;
  margin: 2.33em 0;
}

*[edit="deleted"]  { display: none }
/* no special presentation by default
*[edit="inserted"] {  }
*[edit="changed"]  {  }
*[edit="moved"]    {  }
*/

/* experimental navigation list style */

nl {
  height: 1.5em;
  overflow: auto;
  margin: 0;
  line-height: normal !important;
  white-space: nowrap;
  text-align: start;
  cursor: default;
  border-width: 2px !important;
  border-style: inset !important;
  vertical-align: baseline;
  padding: 0;
}

nl:hover { height: auto; overflow: visible; }

nl > li, nl > label {
  display: block;
  min-height: 1em;
  line-height: normal !important;
}
nl > li, nl > label {
  padding: 0 5px 0 3px;
}
nl > li {
  margin-left: 1em;
}
nl > label {
  font-weight: bold;
}

nl > nl > label {
  display: block;
  line-height: normal !important;
  font-style: italic;
  font-weight: bold;
}

nl > nl > li {
  padding-left: 2em;
  font-style: normal;
  font-weight: normal;
}
```

```
@media print {
  h             { page-break-after: avoid; page-break-inside: avoid }
  blockcode
                { page-break-inside: avoid }
}

@media aural {
  h {
     voice-family: paul, male;
     stress: 20;
     richness: 90;
     pitch: x-low;
     pitch-range: 90;
  }

  section h {
     pitch: x-low;
     pitch-range: 80;
  }

  section section h {
     pitch: low;
     pitch-range: 70;
  }

  section section section h {
     pitch: medium;
     pitch-range: 60;
  }

  section section section section h {
     pitch: medium;
     pitch-range: 50;
  }

  section section section section section h {
     pitch: medium;
     pitch-range: 40;
  }

  blockcode {
     pitch: medium; pitch-range: 0; stress: 0; richness: 80
  }
}

/* inherited elements */

html, body, div, p, h1, h2, h3, h4, h5, h6,
address, blockquote, pre, ol, ul, dl, dt, dd
                { display: block }
li              { display: list-item }
head, script, style, link, meta
                { display: none }
table           { display: table }
tr              { display: table-row }
thead           { display: table-header-group }
```

```
tbody           { display: table-row-group }
tfoot           { display: table-footer-group }
col             { display: table-column }
colgroup        { display: table-column-group }
td, th          { display: table-cell }
caption         { display: table-caption }
th              { font-weight: bolder; text-align: center }
caption         { text-align: center }
body            { padding: 8px; line-height: 1.2 }
h1              { font-size: 200%; margin: .67em 0 }
h2              { font-size: 150%; margin: .83em 0 }
h3              { font-size: 117%; margin: 1em 0 }
h4, p, blockquote, ol, ul, dl
                { margin: 1.33em 0 }
h5              { font-size: 83%; line-height: 1.17em; margin: 1.67em 0 }
h6              { font-size: 67%; margin: 2.33em 0 }
h1, h2, h3, h4, h5, h6
                { font-family: sans-serif; font-weight: bolder }
strong          { font-weight: bolder }
blockquote      { margin-left: 4em; margin-right: 4em }
cite, em, var, address
                { font-style: italic }
pre code, kbd, samp
                { font-family: monospace }
pre             { white-space: pre }
sub, sup        { font-size: smaller }
sub             { vertical-align: sub }
sup             { vertical-align: super }
ol, ul, dd      { margin-left: 4em }
ol              { list-style-type: decimal }
ol ul, ul ol, ul ul, ol ol
                { margin-top: 0; margin-bottom: 0 }

abbr[title]     { border-bottom: dotted 1px }
*[href]         { text-decoration: underline; color: blue; }
:focus          { outline: thin dotted invert }

/* Hover effects should be default */

*[href]:visited { color: #b7f } /* A little lighter than traditionally */
*[href]:hover   { color: red }

/* begin bidirectionality settings (do not change) */

*[dir="ltr"]    { direction: ltr; unicode-bidi: embed }
*[dir="rtl"]    { direction: rtl; unicode-bidi: embed }
*[dir="lro"]    { direction: ltr; unicode-bidi: bidi-override }
*[dir="rlo"]    { direction: rtl; unicode-bidi: bidi-override }

/* block-level elements */
body, div, p, hr, h1, h2, h3, h4, h5, h6,
address, blockquote, pre, ol, ul, li, dl, dt, dd,
noscript, table, thead, tbody, tfoot, tr, td, th,
col, colgroup, caption, object, summary, standby, blockcode
                { unicode-bidi: embed }
/* end bidi settings */
```

```
@media print {
  h1, h2, h3, h4, h5, h6
                { page-break-after: avoid; page-break-inside: avoid }
  blockquote, pre
                { page-break-inside: avoid }
  ul, ol, dl    { page-break-before: avoid }
}

@media aural {
  h1, h2, h3, h4, h5, h6
                { voice-family: paul, male; stress: 20; richness: 90 }
  h1            { pitch: x-low; pitch-range: 90 }
  h2            { pitch: x-low; pitch-range: 80 }
  h3            { pitch: low; pitch-range: 70 }
  h4            { pitch: medium; pitch-range: 60 }
  h5            { pitch: medium; pitch-range: 50 }
  h6            { pitch: medium; pitch-range: 40 }
  li, dt, dd    { pitch: medium; richness: 60 }
  dt            { stress: 80 }
  pre, code     { pitch: medium; pitch-range: 0; stress: 0; richness: 80 }
  em            { pitch: medium; pitch-range: 60; stress: 60; richness: 50 }
  strong        { pitch: medium; pitch-range: 60; stress: 90; richness: 90 }
  dfn           { pitch: high; pitch-range: 60; stress: 60 }
  :link         { voice-family: harry, male }
  :visited      { voice-family: betty, female }
  :active       { voice-family: betty, female; pitch-range: 80; pitch: x-high }
}

/* end xhtml2.css */
```

# I. List of Elements

This appendix is *informative*.

This appendix will contain a list of elements defined in this specification, sorted in alphabetical order, with some other relevant information and links to the element definitions.

| Element Name | Module | Description |
|---|---|---|

# J. List of Attributes

This appendix is *informative*.

This appendix will contain a list of attributes defined in this specification, sorted in alphabetical order, with some other relevant information and links to the attribute definitions.

| Attribute Name | Module | Description |
|---|---|---|

# K. Cross-reference Index

This appendix is *informative*.

This appendix will contain a detailed index of this document, with links to the indexed terms.

# L. References

This appendix is *normative*.

## L.1. Normative References

[CSS2]
"*Cascading Style Sheets, level 2 (CSS2) Specification*", W3C Recommendation, B. Bos *et al.*, *eds.*, 12 May 1998.
Available at: http://www.w3.org/TR/1998/REC-CSS2-19980512

[CSS3-TEXT]
"*CSS3 Text Module*", W3C Candidate Recommendation, M. Suignard, *ed.*, 14 May 2003, *work in progress*.
Available at: http://www.w3.org/TR/2003/CR-css3-text-20030514

[DCORE]
"*DCMI Metadata Terms*", 19 November 2003.
Available at: http://dublincore.org/documents/dcmi-terms/

[DOM]
"*Document Object Model (DOM) Level 2 Core Specification*", W3C Recommendation, A. Le Hors *et al.*, *eds.*, 13 November 2000.
Available at: http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113
A list of DOM Level 2 specifications can be found at: http://www.w3.org/DOM/DOMTR#dom2

[MIMETYPES]
List of registered content types (MIME media types). Download a list of registered content types from http://www.iana.org/assignments/media-types/.

[P3P]
"*The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*", W3C Recommendation, L. Cranor *et al.*, 16 April 2002.
Available at: http://www.w3.org/TR/2002/REC-P3P-20020416/

[RELAXNG]
"*RELAX NG Specification*", OASIS Committee Specification, J. Clark, Murata M., *eds.*, 3 December 2001.
Available at: http://relaxng.org/spec-20011203.html
RELAX NG has been standardized as part of ISO/IEC 19757 - Document Schema Definition Languages (DSDL), as ISO/IEC 19757-2:2003 "Information technology -- Document Schema Definition Language (DSDL) -- Part 2: Regular-grammar-based validation -- RELAX NG". See home page for Document Schema Definition Languages at http://dsdl.org/ for details.

[RFC2045]
"*Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*", RFC 2045, N. Freed and N. Borenstein, November 1996.
Available at: http://www.rfc-editor.org/rfc/rfc2045.txt

[RFC2119]
"*Key words for use in RFCs to indicate requirement levels*", RFC 2119, S. Bradner, March 1997.

Available at: http://www.rfc-editor.org/rfc/rfc2119.txt

[RFC2616]
"*Hypertext Transfer Protocol -- HTTP/1.1*", RFC 2616, R. Fielding *et al.*, June 1999.
Available at: http://www.rfc-editor.org/rfc/rfc2616.txt

[RFC3066]
"*Tags for the Identification of Languages*", RFC 3066, H. Alvestrand, January 2001.
Available at: http://www.rfc-editor.org/rfc/rfc3066.txt

[RUBY]
"*Ruby Annotation*", W3C Recommendation, M. Sawicki *et al.*, *eds.*, 31 May 2001.
Available at: http://www.w3.org/TR/2001/REC-ruby-20010531

[SGML]
"*Information Processing -- Text and Office Systems -- Standard Generalized Markup Language (SGML)*", ISO 8879:1986.
Please consult the ISO Web site at http://www.iso.org/ for information about the standard, or http://www.oasis-open.org/cover/general.html#overview about SGML.

[UAX9]
"*Unicode Standard Annex #9: The Bidirectional Algorithm*", M. Davis, 26 March 2004.
Available at: http://www.unicode.org/reports/tr9/tr9-13.html
The latest version of UAX #9 is available at: http://www.unicode.org/reports/tr9/

[URI]
"*Uniform Resource Identifiers (URI): Generic Syntax*", RFC 2396, T. Berners-Lee *et al.*, August 1998.
Available at: http://www.rfc-editor.org/rfc/rfc2396.txt.

[XFORMS]
"*XForms 1.0*", W3C Recommendation, M. Dubinko *et al.*, *eds.*, 14 October 2003.
Available at: http://www.w3.org/TR/2003/REC-xforms-20031014/

[XHTMLMOD]
"*Modularization of XHTML*", W3C Recommendation, M. Altheim *et al.*, *eds.*, 10 April 2001
Available at: http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410

[XML]
"*Extensible Markup Language (XML) 1.0 (Third Edition)*", W3C Recommendation, T. Bray *et al.*, *eds.*, 4 February 2004.
Available at: http://www.w3.org/TR/2004/REC-xml-20040204

[XMLBASE]
"*XML Base*", W3C Recommendation, J. Marsh, *ed.*, 27 June 2001.
Available at: http://www.w3.org/TR/2001/REC-xmlbase-20010627/

[XMLEVENTS]
"*XML Events*", W3C Recommendation, S. McCarron *et al.*, *eds.*, 14 October 2003.
Available at: http://www.w3.org/TR/2003/REC-xml-events-20031014

[XMLNS]
"*Namespaces in XML*", W3C Recommendation, T. Bray *et al.*, *eds.*, 14 January 1999.
Available at: http://www.w3.org/TR/1999/REC-xml-names-19990114

[XMLSCHEMA]
"*XML Schema Part 1: Structures*", W3C Recommendation, H. S. Thompson *et al.*, *eds.*, 2 May 2001.
Available at: http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

See also "*XML Schema Part 2: Datatypes*", available at:
http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

## L.2. Informative References

[HLINK]
"*HLINK - Link Recognition for the XHTML Family*", W3C Working Draft, S. Pemberton, M. Ishikawa, *eds.*, 13 September 2002.
Available at: http://www.w3.org/TR/2002/WD-hlink-20020913

[HTML4]
"*HTML 4.01 Specification*", W3C Recommendation, D. Raggett *et al.*, *eds.*, 24 December 1999.
Available at: http://www.w3.org/TR/1999/REC-html401-19991224

[RDF]
"*Resource Description Framework (RDF): Concepts and Abstract Syntax* ", W3C Recommendation, G. Klyne, J. Carrol, *ed.*, 10 February 2004.
Available at: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[XFRAMES]
"*XFrames*", W3C Working Draft, S. Pemberton, *ed.*, 6 August 2002, *work in progress*.
Available at: http://www.w3.org/TR/2002/WD-xframes-20020806

[XLINK]
"*XML Linking Language (XLink) Version 1.0*", W3C Recommendation, S. DeRose *et al.*, *eds.*, 27 June 2001.
Available at: http://www.w3.org/TR/2001/REC-xlink-20010627/

[XMLSTYLE]
"*Associating Style Sheets with XML documents Version 1.0*", W3C Recommendation, J. Clark, *ed.*, 29 June 1999.
Available at: http://www.w3.org/1999/06/REC-xml-stylesheet-19990629

[XPATH]
"*XML Path Language (XPath) Version 1.0*", W3C Recommendation, J. Clark *et al.*, *eds.*, 16 November 1999.
Available at: http://www.w3.org/TR/1999/REC-xpath-19991116

# M. Acknowledgements

This appendix is *informative*.

This specification was prepared by the W3C HTML Working Group. The participants at the time of publication were:

*This section will be updated at publication time.*

The HTML Working Group would like to acknowledge the great many people outside of the HTML Working Group who help with the process of developing the XHTML 2.0 specification. These people are too numerous to list individually. They include but are not limited to people who have contributed on the www-html@w3.org mailing list, other Working Groups at the W3C, and the W3C Team. XHTML 2.0 is truly a cooperative effort between the HTML Working Group, the rest of the W3C, and the public.