Modularization of XHTML™ 2.0

W3C Editor's Draft 23 January 2009

This version:
http://www.w3.org/MarkUp/2009/ED-xhtml-modularization2-20090123
Latest public version:
http://www.w3.org/TR/xhtml-modularization2
Previous Recommendation:
http://www.w3.org/TR/2008/REC-xhtml-modularization-20081008
Editors:
Mark Birbeck, x-port.net
Markus Gylling, DAISY Consortium
Shane McCarron, Applied Testing and Technology
Steven Pemberton, CWI (XHTML 2 Working Group Co-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 © 2001-2009 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark and document use rules apply.

Abstract

This Working Draft specifies an abstract modularization of XHTML and an implementation of the abstraction using XML Document Type Definitions (DTDs), XML Schema, and RelaxNG. This modularization provides a means for subsetting and extending XHTML, a feature needed to permit the definition of custom markup languages that are targeted at specific classes of devices or specific types of documents.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.

This document is the development version of a *very early* form and is riddled with errors. It should in no way be considered stable, and should not be referenced for any purposes whatsoever.

This document has been produced by the W3C XHTML 2 Working Group as part of the HTML Activity. The goals of the XHTML 2 Working Group are discussed in the XHTML 2 Working Group charter.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; 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) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Please report errors in this specification 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).

Quick Table of Contents

1. Introduction 1.1
2. Terms and Definitions 1.5
3. Conformance Definition 1.9
4. Module Definition Conventions 25
5. XHTML Attribute Collections 31
6. XHTML Document Module 33
7. XHTML Structural Module 37
8. XHTML Text Module 47
9. XHTML Hypertext Module 55
10. XHTML List Module 57
11. XHTML Core Attributes Module 63
12. XHTML Hypertext Attributes Module 67
13. XHTML I18N Attribute Module 7.3
14. XHTML Bi-directional Text Attribute Module 7.5
15. XHTML Caption Module 7.9
16. XHTML Edit Attributes Module 81
17. XHTML Embedding Attributes Module 83
18. XHTML Image Module 89
19. XHTML Image Map Attributes Module 91
20. XHTML Media Attribute Module 95
21. XHTML Metainformation Module 97
22. XHTML Object Module 10.3
23. XHTML Style Sheet Module 11.3
24. XHTML Style Attribute Module 11.7
25. XHTML Tables Module 11.9
A. Building XHTML Relax NG Modules 14.1
B. Developing XHTML Relax NG with defined and extended modules 14.9
C. XHTML RELAX NG Module Implementations 16.3

D. Building Schema Modules 213
E. Developing Schema with defined and extended modules 217
F. XHTML Schema Module Implementations 223
G. Building DTD Modules 141
H. Developing DTDs with defined and extended modules 269
I. XHTML DTD Module Implementations 283
J. List of Elements 285
K. List of Attributes 289
L. Cross-reference Index 293
M. References 295
N. Acknowledgements 299

Full Table of Contents

1. Introduction 1.1
1.1. What is XHTML? 1.1
1.2. What is XHTML Modularization? 1.1
1.3. Why Modularize XHTML? 1.1
1.3.1. Abstract modules 1.2
1.3.2. Module implementations 1.2
1.3.3. Hybrid document types 1.2
1.3.4. Validation 1.3
1.3.5. Formatting Model 1.3
1.4. Issues 1.3
2. Terms and Definitions 1.5
3. Conformance Definition 1.9
3.1. XHTML Host Language Document Type Conformance 1.9
3.2. XHTML Integration Set Document Type Conformance 2.0
3.3. XHTML Family Module Conformance 2.0
3.4. XHTML Family Document Conformance 2.1
3.5. XHTML Family User Agent Conformance 2.1
3.6. Naming Rules 2.2
3.7. XHTML Module Evolution 2.2
4. Module Definition Conventions 2.5
4.1. Module Structure 2.5
4.2. Abstract Module Definitions 2.5
4.3. Syntactic Conventions 2.5
4.4. Content Models 2.6
4.5. Attribute Types 2.7
4.6. Issues 2.9
5. XHTML Attribute Collections 3.1
5.1. Issues 3.2
6. XHTML Document Module 3.3

6.1. The html element 33
6.2. The head element 34
6.3. The title element 35
6.4. The body element 35
7. XHTML Structural Module 37
7.1. The address element 39
7.2. The blockcode element 39
7.3. The blockquote element 40
7.4. The div element 40
7.5. The heading elements 41
7.6. The p element 42
7.7. The pre element 43
7.8. The section element 43
7.9. The separator element 44
7.10. Issues 45
8. XHTML Text Module 47
8.1. The abbr element 48
8.2. The cite element 48
8.3. The code element 49
8.4. The dfn element 49
8.5. The em element 49
8.6. The kbd element 50
8.7. The l element 50
8.8. The q element 51
8.9. The samp element 51
8.10. The span element 52
8.11. The strong element 52
8.12. The sub element 52
8.13. The sup element 53
8.14. The var element 53
8.15. Issues 54
9. XHTML Hypertext Module 55
9.1. The a element 55
10. XHTML List Module 57
10.1. Definition lists: the dl, di, dt, and dd elements 58
10.2. The nl element 59
10.3. The ol and ul elements 60
10.4. The li element 60
10.5. Issues 61
11. XHTML Core Attributes Module 63
11.1. Core Attribute Collection 63
12. XHTML Hypertext Attributes Module 67
12.1. Hypertext Attribute Collection 67

12.2. Issues 71
13. XHTML I18N Attribute Module 73
13.1. I18N Attribute Collection 73
13.2. Issues 74
14. XHTML Bi-directional Text Attribute Module 75
14.1. Bi-directional Text Collection 75
14.1.1. Inheritance of text direction information 75
14.1.2. The effect of style sheets on bidirectionality 76
14.2. Issues 77
15. XHTML Caption Module 79
15.1. The caption element 79
16. XHTML Edit Attributes Module 81
16.1. Edit Collection 81
17. XHTML Embedding Attributes Module 83
17.1. Embedding Attribute Collection 83
17.2. Issues 84
18. XHTML Image Module 89
18.1. The img element 89
19. XHTML Image Map Attributes Module 91
19.1. Image Map Attribute Collection 92
20. XHTML Media Attribute Module 95
20.1. Media Attribute Collection 95
21. XHTML Metainformation Module 97
21.1. The link element 97
21.1.1. Forward and reverse links 98
21.1.2. Links and search engines 98
21.2. The meta element 99
21.2.1. meta and search engines 100
21.3. Issues 101
22. XHTML Object Module 103
22.1. The object element 103
22.1.1. Defining terminology 104
22.1.2. Basic Information for Object Handlers 104
22.1.3. Rules for processing objects 105
22.2. The param element 108
22.2.1. Referencing object data 110
22.2.2. Object element declarations and instantiations 111
22.3. The standby element 112
23. XHTML Style Sheet Module 113
23.1. The style element 113
23.1.1. External style sheets 114
23.1.2. Preferred and alternate style sheets 114
23.1.3. Specifying external style sheets 115

24. XHTML Style Attribute Module 117
24.1. Style Attribute Collection 117
24.2. Issues 117
25. XHTML Tables Module 119
25.1. The col and colgroup elements 120
25.1.1. Calculating the number of columns in a table 122
25.2. The summary element 122
25.3. The table element 123
25.3.1. Visual Rendering 123
25.3.2. Table directionality 124
25.3.3. Table rendering by non-visual user agents 124
25.4. The tbody element 133
25.5. The td and th elements 133
25.5.1. Cells that span several rows or columns 135
25.6. The thead and tfoot elements 138
25.7. The tr element 139
25.8. Issues 139
A. Building XHTML Relax NG Modules 141
A.1. Defining the Namespace of a Module 142
A.1.1. Qualified Names sub-module 142
A.1.2. Declaration sub-module(s) 144
A.1.3. Using the module as a stand-alone DTD 145
A.1.4. Namespace Idiosyncrasies 147
B. Developing XHTML Relax NG with defined and extended modules 149
B.1. Defining additional attributes 150
B.2. Defining additional elements 150
B.3. Defining the content model for a collection of modules 151
B.3.1. Integrating a stand-alone module into XHTML 151
B.3.2. Mixing a new module throughout the modules in XHTML 152
B.4. Creating a new DTD 152
B.4.1. Creating a simple DTD 152
B.4.2. Creating a DTD by extending XHTML 154
B.4.3. Creating a DTD by removing and replacing XHTML modules . . . 155
B.4.4. Creating a new DTD 155
B.5. Using the new DTD 160
C. XHTML RELAX NG Module Implementations 163
C.1. XHTML Module Implementations 163
C.1.1. Attribute Collections 163
C.1.2. Document 164
C.1.3. Structural 166
C.1.4. Text 170
C.1.5. Hypertext 174
C.1.6. List 174

C.1.7. Core Attributes 177
C.1.8. Hypertext Attributes 178
C.1.9. I18N Attribute 180
C.1.10. Access 180
C.1.11. Bi-directional Text Attribute 181
C.1.12. Edit Attributes 182
C.1.13. Embedding Attributes 183
C.1.14. Image 183
C.1.15. Image Map Attributes 184
C.1.16. Media Attribute 185
C.1.17. Metainformation Attributes 185
C.1.18. Metainformation 187
C.1.19. Object 188
C.1.20. Role Access 189
C.1.21. Style Attribute 190
C.1.22. Style Sheet 190
C.1.23. Tables 191
C.2. XHTML RELAX NG Support Modules 195
C.2.1. Datatypes 195
C.2.2. Events 198
C.2.3. Param 201
C.2.4. Caption 201
C.2.5. Role Attribute 202
C.3. RELAX NG External Modules 202
C.3.1. Ruby 202
C.3.2. Ruby Driver for Full Ruby Markup 205
C.3.3. XML Events 205
C.3.4. XML Handlers 207
C.3.5. XML Script 209
C.3.6. XML Schema instance 211
D. Building Schema Modules 213
D.1. Named Content Models 213
D.2. Defining the Namespace of a Module 142
D.2.1. Global and Local Element Declarations 214
D.2.2. Global and Local Attribute Declarations 214
D.3. Importing External Namespace Schema Components 215
D.4. Datatype Definitions and Namespaces 215
D.5. Content Model Redefinitions 215
E. Developing Schema with defined and extended modules 217
E.1. Defining additional attributes 150
E.2. Defining additional elements 150
E.3. Defining the content model for a collection of modules 151
E.3.1. Integrating a stand-alone module into XHTML 219

E.3.2. Mixing a new module throughout the modules in XHTML 220
E.4. Creating a new Document Type 220
E.4.1. Creating a simple Document Type 221
E.4.2. Creating a Language by extending XHTML 221
E.4.3. Creating a Language by removing and replacing XHTML modules . . 221
E.4.4. Creating a the new Document Type 222
F. XHTML Schema Module Implementations 223
F.1. Required Modules 223
F.1.1. Datatypes Module 223
F.1.2. Document Module 225
F.1.3. Structural Module 227
F.1.4. Text Module 231
F.1.5. Hypertext Module 235
F.1.6. List Module 235
F.1.7. Core Attributes Module 238
F.1.8. Hypertext Attributes Module 239
F.1.9. I18N Attribute Module 240
F.2. Optional Modules 240
F.2.1. Bi-directional Text Module 240
F.2.2. Caption Module 241
F.2.3. Edit Attributes Module 241
F.2.4. Embedding Attributes Module 242
F.2.5. Image Module 243
F.2.6. Image Map Attributes Module 243
F.2.7. Media Attribute Module 244
F.2.8. Metainformation Module 244
F.2.9. Metainformation Attributes Module 245
F.2.10. Object Module 246
F.2.11. Style Sheet Module 247
F.2.12. Style Attribute Module 248
F.2.13. Tables Module 248
F.3. Modules from Other Specifications 251
F.3.1. Access Module 251
F.3.2. Role Attribute Module 252
F.3.3. Ruby Module 252
F.3.4. XForms Modules 255
F.3.5. XML Events Module 255
F.3.6. XML Handlers Module 256
F.3.7. XML Scripting Module 258
G. Building DTD Modules 141
G.1. Defining the Namespace of a Module 142
G.1.1. Qualified Names sub-module 142
G.1.2. Declaration sub-module(s) 144

G.1.3. Using the module as a stand-alone DTD 145
G.1.4. Namespace Idiosyncrasies 267
H. Developing DTDs with defined and extended modules 269
H.1. Defining additional attributes 150
H.2. Defining additional elements 150
H.3. Defining the content model for a collection of modules 151
H.3.1. Integrating a stand-alone module into XHTML 27.1
H.3.2. Mixing a new module throughout the modules in XHTML 27.2
H.4. Creating a new DTD 27.2
H.4.1. Creating a simple DTD 27.2
H.4.2. Creating a DTD by extending XHTML 27.4
H.4.3. Creating a DTD by removing and replacing XHTML modules . . . 27.5
H.4.4. Creating a new DTD 27.5
H.5. Using the new DTD 280
I. XHTML DTD Module Implementations 283
I.1. XHTML Modular Framework 283
I.2. XHTML Module Implementations 283
I.3. XHTML DTD Support Modules 283
J. List of Elements 285
K. List of Attributes 289
L. Cross-reference Index 293
M. References 295
M.1. Normative References 295
M.2. Informative References 298
N. Acknowledgements 299

1. Introduction

This section is *informative*.

1.1. What is XHTML?

XHTML is the reformulation of HTML 4 as an application of XML. XHTML 1.0 [XHTML1] [p.??] specifies three XML document types that correspond to the three HTML 4 DTDs: Strict, Transitional, and Frameset. XHTML 1.0 is the basis for a family of document types that subset and extend HTML.

1.2. What is XHTML Modularization?

XHTML Modularization is a decomposition of XHTML 1.0, and by reference HTML 4, into a collection of abstract modules that provide specific types of functionality. These abstract modules are implemented in this specification using the XML Document Type Definition language, but an implementation using XML Schemas is expected. The rules for defining the abstract modules, and for implementing them using XML DTDs, are also defined in this document.

These modules may be combined with each other and with other modules to create XHTML subset and extension document types that qualify as members of the XHTML-family of document types.

1.3. Why Modularize XHTML?

The modularization of XHTML refers to the task of specifying well-defined sets of XHTML elements that can be combined and extended by document authors, document type architects, other XML standards specifications, and application and product designers to make it economically feasible for content developers to deliver content on a greater number and diversity of platforms.

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

Modularizing XHTML provides a means for product designers to specify which elements are supported by a device using standard building blocks and standard methods for specifying which building blocks are used. These modules serve as "points of conformance" for the content community. The content community can now target the installed base that supports a certain collection of modules, rather than worry about the installed base that supports this or that permutation of XHTML elements. The use of standards is critical for modularized XHTML to be

successful on a large scale. It is not economically feasible for content developers to tailor content to each and every permutation of XHTML elements. By specifying a standard, either software processes can autonomously tailor content to a device, or the device can automatically load the software required to process a module.

Modularization also allows for the extension of XHTML's layout and presentation capabilities, using the extensibility of XML, without breaking the XHTML standard. This development path provides a stable, useful, and implementable framework for content developers and publishers to manage the rapid pace of technological change on the Web.

1.3.1. Abstract modules

An XHTML document type is defined as a set of abstract modules. A abstract module defines one kind of data that is semantically different from all others. Abstract modules can be combined into document types without a deep understanding of the underlying schemas that define the modules.

1.3.2. Module implementations

A module implementation consists of a set of element types, a set of attribute-list declarations, and a set of content model declarations, where any of these three sets may be empty. An attribute-list declaration in a module may modify an element type outside the element types defined in the module, and a content model declaration may modify an element type outside the element type set of the module.

One implementation mechanism is XML DTDs. An XML DTD is a means of describing the structure of a class of XML documents, collectively known as an XML document type. XML DTDs are described in the XML 1.0 Recommendation [XML] [p.??] . Another implementation mechanism is XML Schema [XMLSCHEMA] [p.??] .

1.3.3. Hybrid document types

A hybrid document type is an document type composed from a collection of XML DTDs or DTD Modules. The primary purpose of the modularization framework described in this document is to allow a DTD author to combine elements from multiple abstract modules into a hybrid document type, develop documents against that hybrid document type, and to validate that document against the associated hybrid document type definition.

One of the most valuable benefits of XML over SGML is that XML reduces the barrier to entry for standardization of element sets that allow communities to exchange data in an interoperable format. However, the relatively static nature of XHTML as the content language for the Web has meant that any one of these communities have previously held out little hope that their XML document types would be able to see widespread adoption as part of Web standards. The modularization framework allows for the dynamic incorporation of these diverse document types within the XHTML-family of document types, further reducing the barriers to the incorporation of these domain-specific vocabularies in XHTML documents.

1.3.4. Validation

The use of well-formed, but not valid, documents is an important benefit of XML. In the process of developing a document type, however, the additional leverage provided by a validating parser for error checking is important. The same statement applies to XHTML document types with elements from multiple abstract modules.

A document is an instance of one particular document type defined by the DTD identified in the document's prologue. Validating the document is the process of checking that the document complies with the rules in the document type definition.

One document can consist of multiple document fragments. Validating only fragments of a document, where each fragment is of a different document type than the other fragments in the document, is beyond the scope of this framework - since it would require technology that is not yet defined.

However, the modularization framework allows multiple document type definitions to be integrated and form a new document type (e.g. SVG integrated into XHTML). The new document type definition can be used for normal XML 1.0 validation.

1.3.5. Formatting Model

Earlier versions of HTML attempted to define parts of the model that user agents are required to use when formatting a document. With the advent of HTML 4, the W3C started the process of divorcing presentation from structure. XHTML 1.0 maintained this separation, and this document continues moving XHTML and its descendants down this path. Consequently, this document makes no requirements on the formatting model associated with the presentation of documents marked up with XHTML Family document types.

Instead, this document recommends that content authors rely upon style mechanisms such as CSS to define the formatting model for their content. When user agents support the style mechanisms, documents will format as expected. When user agents do not support the style mechanisms, documents will format as appropriate for that user agent. This permits XHTML Family user agents to support rich formatting models on devices where that is appropriate, and lean formatting models on devices where *that* is appropriate.

1.4. Issues

[XHTML2] Spirit of "1.1.3. XHTML 2 and Presentation" PR #7759
State: Suspended
Resolution: None
User: None

Notes:
Suspended until last call

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.297]] 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 documents 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.296]] 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.

focusable
: Elements are considered "focusable" if they are *visible* (e.g., have the equivalent of the [CSS2 [p.295]] property of "display" with a value other than `none`) not disabled (see [XFORMS [p.297]]), and either 1) have an @href [p.67] attribute or 2) are considered a form control as defined in [XFORMS [p.297]].

fragment identifier
: A portion of a [URI [p.296]] as defined in RFC 3986.

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.

unavailable resource
: any resource that is referenced as a URI in an attribute, but that cannot be accessed for any reason, is considered unavailable. Example reasons include, but are not limited to: network unavailable, no resource available at the URI given, inability of the user agent to process the type of resource, etc.

user agent
: any software that retrieves and renders Strictly Conforming Documents [p.??] for users. This may include browsers, media players, plug-ins, and other programs — including assistive technologies — that help in retrieving and rendering such documents. See also Conforming User Agent [p.21] .

3. Conformance Definition

This section is *normative*.

In order to ensure that XHTML-family documents are maximally portable among XHTML-family user agents, this specification rigidly defines conformance requirements for both of these and for XHTML-family document types. While the conformance definitions can be found in this section, they necessarily reference normative text within this document and within other related specifications. It is only possible to fully comprehend the conformance requirements of XHTML through a complete reading of all normative references.

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.??] .

3.1. XHTML Host Language Document Type Conformance

It is possible to modify existing document types and define wholly new document types using both modules defined in this specification and other modules. Such a document type is "XHTML Host Language Conforming" when it meets the following criteria:

1. The document type MUST be defined using one of the implementation methods defined by the W3C. Currently this is limited to Relad NG, XML DTDs and XML Schema.
2. The schema which defines the document type MUST have a unique identifier as defined in Naming Rules [p.22] that begins with the character sequence "`XHTML`".
3. The schema which defines the document type must include, at a minimum, the Document, Structural, Text, Hypertext, Text, List, Core Attribute, Hypertext Attributes, and I18N Attribute modules defined in this specification.
4. For each of the W3C-defined modules that are included, all of the elements, attributes, types of attributes (including any required enumerated value lists), and any required minimal content models must be included (and optionally extended) in the document type's content model. When content models are extended, all of the elements and attributes (along with their types or any required enumerated value lists) required in the original content model must continue to be required.
5. The schema that defines the document type may define additional elements and attributes. However, these MUST be in their own XML namespace [XMLNAMES] [p.??] . If additional elements are defined by a module, the attributes defined in included XHTML modules are available for use on those elements, but SHOULD be referenced using their namespace-qualified identifier (e.g., xhtml:class). The semantics of the attributes remain the same as when used on an XHTML-namespace element.

3.2. XHTML Integration Set Document Type Conformance

It is also possible to define document types that are based upon XHTML, but do not adhere to its structure. Such a document type is "XHTML Integration Set Conforming" when it meets the following criteria:

1. The document type MUST be defined using one of the implementation methods defined by the W3C. Currently this is limited to Relax NG, XML DTDs and XML Schemas.
2. The schema that defines the document type MUST have a unique identifier as defined in Naming Rules [p.22] . This identifier MUST contain the character sequence "XHTML", but MUST NOT start with that character sequence.
3. The schema which defines the document type MUST include, at a minimum, the Structural, Hypertext, Text, and List modules defined in this specification.
4. For each of the W3C-defined modules that are included, all of the elements, attributes, types of attributes (including any required enumerated lists), and any required minimal content models MUST be included (and optionally extended) in the document type's content model. When content models are extended, all of the elements and attributes (along with their types or any required enumerated value lists) required in the original content model MUST continue to be required.
5. The schema that defines the document type MAY define additional elements and attributes. However, these MUST be in their own XML namespace [XMLNAMES] [p.??] . If additional elements are defined by a module, the attributes defined in included XHTML modules are available for use on those elements, but SHOULD be referenced using their namespace-qualified identifier (e.g., xhtml:class). The semantics of the attributes remain the same as when used on an XHTML-namespace element.

3.3. XHTML Family Module Conformance

This specification defines a method for defining XHTML-conforming modules. A module conforms to this specification when it meets all of the following criteria:

1. The document type MUST be defined using one of the implementation methods defined by the W3C. Currently this is limited to Relax NG, XML DTDs and XML Schemas.
2. The schema that defines the module MUST have a unique identifier as defined in Naming Rules [p.22] .
3. When the module is defined using an XML DTD, the module MUST isolate its parameter entity names through the use of unique prefixes or other, similar methods.
4. The module definition MUST have a prose definition that describes the syntactic and semantic requirements of the elements, attributes, and/or content models that it declares. For the avoidance of doubt, if there is any discrepancy between the prose definition of a module and its schema implementation(s), the prose definition MUST take precedence.
5. The module definition MUST NOT reuse any element names that are defined in other W3C-defined modules, except when the content model and semantics of those elements are either identical to the original or an extension of the original, or when the reused element names are within their own XML namespace (see below).

6. The module definition's elements and attributes MUST be part of an XML namespace [XMLNAMES] [p.??] . If the module is defined by an organization other than the W3C, this namespace MUST NOT be the same as the namespace in which other W3C modules are defined.

3.4. XHTML Family Document Conformance

A conforming XHTML family document is a valid instance of an XHTML Host Language Conforming Document Type. For the avoidance of doubt, the behavior of User Agents in the presence of invalid documents is undefined.

3.5. XHTML Family User Agent Conformance

A conforming user agent must meet all of the following criteria (as defined in [XHTML1] [p.??]):

1. In order to be consistent with the XML 1.0 Recommendation [XML] [p.??] , the user agent MUST parse and evaluate an XHTML document for well-formedness. If the user agent claims to be a validating user agent, it MUST also validate documents against their referenced schemas.
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 document as generic [XML] [p.??] , it MUST recognize only attributes of type `ID` (e.g., the `id` attribute on most XHTML elements) as fragment identifiers.
4. If a user agent encounters an element it does not recognize, it MUST continue to process the children 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 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.
7. 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.
8. Whitespace is defined as in [XML] [p.??] . On input all whitespace is preserved - this is exactly as if the value of `xml:space`, as defined in [XML] [p.??] , is set to "preserve". If the value of that attribute is set to "default", that is the same as if it were set to "preserve". On rendering, whitespace is processed according to the rules of [CSS2] [p.??] .

3.6. Naming Rules

XHTML Host Language document types must adhere to strict naming conventions so that it is possible for software and users to readily determine the relationship of document types to XHTML. The names for document types implemented as XML Document Type Definitions are defined through Formal Public Identifiers (FPIs). Within FPIs, fields are separated by double slash character sequences (`//`). The various fields must be composed as follows:

1. The leading field must be "-" to indicate a privately defined resource.
2. The second field must contain the name of the organization responsible for maintaining the named item. There is no formal registry for these organization names. Each organization should define a name that is unique. The name used by the W3C is, for example, `W3C`.
3. The third field contains two constructs: the public text class followed by the public text description. The first token in the third field is the public text class which should adhere to ISO 8879 Clause 10.2.2.1 Public Text Class. Only XHTML Host Language conforming documents should begin the public text description with the token XHTML. The public text description should contain the string XHTML if the document type is Integration Set conforming. The field must also contain an organization-defined unique identifier (e.g., MyML 1.0). This identifier should be composed of a unique name and a version identifier that can be updated as the document type evolves.
4. The fourth field defines the language in which the item is developed (e.g., `EN`).

Using these rules, the name for an XHTML Host Language conforming document type might be `-//MyCompany//DTD XHTML MyML 1.0//EN`. The name for an XHTML family conforming module might be `-//MyCompany//ELEMENTS XHTML MyElements 1.0//EN`. The name for an XHTML Integration Set conforming document type might be `-//MyCompany//DTD Special Markup with XHTML//EN`.

3.7. XHTML Module Evolution

Each module defined in this specification is given a unique identifier that adheres to the naming rules in the previous section. Over time, a module may evolve. A logical ramification of such evolution may be that some aspects of the module are no longer compatible with its previous definition. To help ensure that document types defined against modules defined in this specification continue to operate, the identifiers associated with a module that changes will be updated. Specifically, the Formal Public Identifier and System Identifier of the module will be changed by modifying the version identifier included in each. Document types that wish to incorporate the updated functionality will need to be similarly updated.

In addition, the earlier version(s) of the module will continue to be available via its earlier, unique identifier(s). In this way, document types developed using XHTML modules will continue to function seamlessly using their original definitions even as the collection expands and evolves. Similarly, document instances written against such document types will continue to validate using the earlier module definitions.

Other XHTML Family Module and Document Type authors are encouraged to adopt a similar strategy to ensure the continued functioning of document types based upon those modules and document instances based upon those document types.

4. 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.

4.1. 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.31] 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.

4.2. 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.

4.3. 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. Content sets names always begin with an uppercase letter.

`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.

4.4. Content Models

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 (parsed characted data). 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.

4.5. 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.297]].
NMTOKENS	One or more white space separated NMTOKEN values
NUMBER	Sequence of one or more digits ([0-9])

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

Data type	Description
Character	A single character, as per section 2.2 of [XML [p.297]].
Charset	A character encoding, as per [RFC2045] [p.??] .
Encodings	A comma-separated list of ’charset’s with optional q parameters, as defined in section 14.2 of [RFC2616 [p.296]] as the field value of the Accept-Charset request header.
ContentType	A media type, as per [RFC2045 [p.296]].

ContentTypes	Attributes of this type identify the allowable content type(s) of an associated URI [p.29] (usually a value of another attribute on the same element). 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.296]] 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 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 must imitate similar behavior when using other methods than HTTP. For instance, when accessing files in a local filestore, `<p src="logo" srctype="image/png, image/jpeg">` might cause the user agent first to look for a file `logo.png`, and then for `logo.jpg`. If a value for the content type is not given, "*/*" must be used for its value. For the current list of registered content types, please consult [MIMETYPES [p.295]].
Coordinates	Comma separated list of Length [p.29] s used in defining areas.
CURIE	A Compact URI [CURIE [p.295]].
CURIEs	One or more white space separated CURIE [p.28] values
Datetime	Date and time information, as defined by the type `dateTime` in [XMLSCHEMA [p.297]] except that the timezone part is required.
HrefTarget	Name used as destination for results of certain actions, with legal values as defined by NMTOKEN [p.27] .
LanguageCode	A language code. The values should conform to [RFC3066 [p.296]] or its successors.
LanguageCodes	A comma-separated list of language ranges with optional q parameters, as defined in section 14.4 of [RFC2616 [p.296]] as the field value of the Accept-Language request header. Individual language codes should conform to [RFC3066 [p.296]] or its successors.

Length	Either a number, representing a number of pixels, or a percentage, representing 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.298]].
MediaDesc	A comma-separated list of media descriptors as described by [CSS2 [p.295]]. The default is `all`.
Number	One or more digits
QName	An [XMLNS [p.297]]-qualified name. See QName for a formal definition.
QNames	One or more white space separated QName [p.29] values
Text	A character string.
URI	An Internationalized Resource Identifier Reference, as defined by [IRI [p.295]].
URIorSafeCURIE	A URI (as defined above) or Safe Compact URI [CURIE [p.295]].
URIs	A space-separated list of URIs as defined above.

Implementation: RELAX NG [p.??] , XML Schema [p.223]

4.6. Issues

Fw: [XHTML 2] Section 5.5 quality values. PR #7799
State: Open
Resolution: None
User: None

Notes:
Discussed this a bit. What we mean is that it is an intersection of media types, and that quality values in the document should take precedence. Will continue to work on this and craft wording. Upon further review, Steven Pemberton has agreed to think about the right solution.

Fw: [XHTML 2] Section 5.5 intersection of mime-types PR #7800
State: Open
Resolution: None
User: None

Notes:
This is related to issue 7799 - the answer may be that asterisks are not relevant. We will clarify this once Steven's action item is complete.

5. 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 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.

Note that while these Collections are referenced throughout this document, these expressions should in no way be considered normative or mandatory. They are an editorial convenience. When used in this document, it is the expansion of the term that is normative, not the term itself.

For the avoidance of doubt, if an attribute collection is referenced by an element's definition, and an attribute in that collection is also explicitly referenced by that element's definition, this does NOT cause a conflict. It is up to the schema implementation to update the content models accordingly.

Collection	Module	Description
Core [p.??]	Core Attributes Module [p.63]	Basic attributes used to identify and classify elements and their content.
I18N [p.73]	Internationalization Attribute Module [p.73]	Attribute to identify the language of an element and its contents.
Bi-directional [p.75]	Bi-directional Text Collection [p.75]	Attributes used to manage bi-directional text.
Edit [p.81]	Edit Attributes Module [p.81]	Attributes used to annotate when and how an element's content was edited.
Embedding [p.83]	Embedding Attributes Module [p.83]	Attributes used to embed content from other resources within the current element.
Events	XML Events Module	Attributes that allow associating of events and event processing with an element and its contents.
Forms	XForms Module	Attributes that designate provide a mechanism of repeating table rows within a form.
Hypertext [p.67]	Hyptertext Attributes Module [p.67]	Attributes that designate characteristics of links within and among documents.
I18N [p.73]	I18N Attribute Module [p.73]	Attributes for defining the language of the contents of an element.

Collection	Module	Description
Map [p.92]	Image Map Attributes Module [p.91]	Attributes for defining and referencing client-side image maps.
Media [p.95]	Media Attribute Module [p.95]	Attribute for performing element selection based upon media type as defined in MediaDesc [p.29]
Metainformation	Metainformation Attributes	Attributes that allow associating of elements with metainformation about those elements
Role	Role Attribute Module	Attribute for the specification of the "role" of an element.
Style [p.117]	Style Attribute Module [p.117]	Attribute for associating style information with an element and its contents.
Common	Attribute Collections Module	A meta-collection of all the other collections, including the Core [p.??] , Bi-directional [p.75] , Events, Edit [p.81] , Embedding [p.83] , Hypertext [p.67] , I18N [p.73] , Map [p.92] , Media [p.95] , Metainformation, Role, and Style [p.117] attribute collections.

Implementations: RELAX NG [p.163] , XML Schema

If a module associated with an attribute collection is not included in an XHTML Host Language or XHTML Integration Set, then that collection's attributes are not included in the definition of the Common attribute collection.

Each of the attributes defined in an XHTML attribute collection is available for use when their corresponding module is included in an XHTML Host Language or an XHTML Integration Set. In such a situation, the attributes are available for use in the definition of elements that are NOT in the XHTML namespace when they are referenced using their namespace-qualified identifier (e.g., `xhtml:class`). The semantics of the attributes remain the same regardless of whether they are referenced using their qualified identifier or not. **It is an error to use an XHTML namespace-qualified attribute on elements from the XHTML Namespace.**

5.1. Issues

[XHTML2] Constraining attribute relationship PR #7661
State: Suspended
Resolution: Defer
User: None

Notes:
We think this is a good thing for M12N 2.0, but is not necessary for XHTML 2 right now.

6. XHTML Document Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

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.33]	Common [p.32] , @version [p.34] (CDATA [p.27]), xmlns (URI [p.29] = "http://www.w3.org/1999/xhtml"), @xsi:schemaLocation [p.34] (URIs [p.29] = "http://www.w3.org/1999/xhtml http://www.w3.org/MarkUp/SCHEMA/xhtml2.xsd")	head [p.34] , body [p.35]
head [p.34]	Common [p.32] , @profile [p.34] (URIs)	title [p.35]
title [p.35]	Common [p.32]	PCDATA*
body [p.35]	Common [p.32]	(Heading [p.39] | Structural [p.39] | List [p.57])*

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/1999/xhtml". Also note that because the xmlns declaration is treated specially by XML namespace-aware parsers [XMLNS [p.297]], 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 must be `http://www.w3.org/1999/xhtml`.

Implementations: RELAX NG [p.164] , XML Schema [p.225]

6.1. The html element

The html [p.33] element is the root element for all XHTML Family Document Types. The @xml:lang [p.73] attribute is required on this element.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

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. However, all values beginning with the character sequence `xhtml` are reserved for use by XHTML Family Document Types.

xsi:schemaLocation = URIs [p.29]
: This attribute allows the specification of a location where an XML Schema [XMLSCHEMA [p.297]] 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 [p.??] .

6.2. The head element

The head [p.34] 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 style sheet for special purpose use. User agents may however make information in the head [p.34] available to users through other mechanisms.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

profile = URIs [p.29]
: This attribute specifies the location of one or more metadata profiles, separated by white space. For future extensions, user agents should consider the value to be a list even though this specification only considers the first URI to be significant. Profiles are discussed in the section on meta data [p.97] .

Example

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

6.3. The title element

Every XHTML document must have a title [p.35] element in the head [p.34] section.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

The title [p.35] 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.35] 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>
```

The title of a document is metadata about the document, and so a title like `<title>About W3C</title>` is equivalent to `<meta about="" property="title">About W3C</meta>`.

6.4. 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 properties of the text.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

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

7. XHTML Structural Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

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.39]	Common [p.32]	(PCDATA | Text [p.47])*
blockcode [p.39]	Common [p.32]	(PCDATA | Text [p.47] | Heading [p.39] | Structural [p.39] | List [p.57])*
blockquote [p.40]	Common [p.32]	(PCDATA | Text [p.47] | Heading [p.39] | Structural [p.39] | List [p.57])*
div [p.40]	Common [p.32]	(PCDATA | Flow [p.39])*
h [p.41]	Common [p.32]	(PCDATA | Text [p.47])*
h1 [p.41]	Common [p.32]	(PCDATA | Text [p.47])*
h2 [p.41]	Common [p.32]	(PCDATA | Text [p.47])*
h3 [p.41]	Common [p.32]	(PCDATA | Text [p.47])*
h4 [p.41]	Common [p.32]	(PCDATA | Text [p.47])*
h5 [p.41]	Common [p.32]	(PCDATA | Text [p.47])*
h6 [p.41]	Common [p.32]	(PCDATA | Text [p.47])*
p [p.42]	Common [p.32]	(PCDATA | Text [p.47] | List [p.57] | blockcode [p.39] | blockquote [p.40] | pre [p.43])*
pre [p.43]	Common [p.32]	(PCDATA | Text [p.47])*
section [p.43]	Common [p.32]	(PCDATA | Flow [p.39])*
separator [p.44]	Common [p.32]	EMPTY

The content model for this module defines some content sets:

Heading
: h [p.41] | h1 [p.41] | h2 [p.41] | h3 [p.41] | h4 [p.41] | h5 [p.41] | h6 [p.41]

Structural
: address [p.39] | blockcode [p.39] | blockquote [p.40] | div [p.40] | p [p.42] | pre [p.43] | section [p.43] | separator [p.44]

Flow
: Heading [p.39] | Structural [p.39] | Text [p.47]

Implementations: RELAX NG [p.166] , XML Schema [p.227]

7.1. The address element

The address [p.39] element may be used by authors to supply contact information for a document or a major part of a document such as a form.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

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

7.2. The blockcode element

This element indicates that its contents are a block of "code" (see the code [p.49] element). This element is similar to the pre [p.43] element, in that whitespace in the enclosed text has semantic relevance. As a result, the default value of the @layout [p.64] attribute is `relevant`.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example of 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 ;
}
```

7.3. The blockquote element

This element designates a block of quoted text.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

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>
```

7.4. The div element

The div [p.40] element, in conjunction with the @id [p.64] , @class [p.63] and @role 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.113] , the @xml:lang [p.73] attribute, etc., to tailor XHTML to their own needs and tastes.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

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.40] element, with a @class [p.63] of `slide`:

div with a class 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>
```

7.5. 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.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

There are two styles of headings in XHTML: the numbered versions h1 [p.41] , h2 [p.41] etc., and the structured version h [p.41] , which is used in combination with the section [p.43] element.

There are six levels of numbered headings in XHTML with h1 [p.41] as the most important and h6 [p.41] as the least.

Structured headings use the single h element, in combination with the section [p.43] 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.

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>
    </body>
```

Sample style sheet for section levels

```
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 */
```

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.

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.*

7.6. The p element

The p [p.42] element represents a paragraph.

In comparison with earlier versions of HTML, where a paragraph could only contain inline text, XHTML2's paragraphs represent the concept of a paragraph, and so may contain lists, blockquotes, pre's and tables as well as inline text. Note however that they may not contain directly nested p [p.42] elements.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

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

7.7. The pre element

The pre [p.43] element indicates that whitespace in the enclosed text has semantic relevance. As such, the default value of the @layout [p.64] attribute is relevant.

Note that *all* elements in the XHTML family preserve their whitespace in the document, which is only removed on rendering, via a style sheet, according to the rules of CSS [CSS3-TEXT [p.295]]. This means that in principle any elements may preserve or collapse whitespace on rendering, under control of a style sheet.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

A bad poem where whitespace matters

```
<pre>
                If
             I   had
          any      talent
             I   would
              be a
              poet
</pre>
```

Here is how this might be rendered:

```
                If
             I   had
          any      talent
             I   would
              be a
              poet
```

Note that while historically one use of the pre [p.43] element has been as a container for source code, the new blockcode [p.39] element more appropriate for that.

7.8. The section element

The section [p.43] element, in conjunction with the h [p.41] 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.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

By representing the structure of documents explicitly using the section [p.43] and h [p.41] 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.

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>
</body>
```

7.9. The separator element

The separator [p.44] element separates parts of the document from each other.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

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

Example

```
<nl>
<label>Navigation</label>
<li href="/">Home</li>
<li><separator/></li>
<li href="prev">Previous</li>
<li href="..">Up</li>
<li href="next">Next</li>
</nl>
```

7.10. Issues

[XHTML2] How are UAs to interpret <h> and <hx> elements? PR #7820
State: Open
Resolution: None
User: None

Notes:

[XHTML2] How are UAs to interpret <h> and <hx> elements? PR #7830
State: Open
Resolution: None
User: None

Notes:

block@kind vs elt@structure PR #7874
State: Open
Resolution: None
User: None

Notes:

redundant content model PR #7875
State: Open
Resolution: None
User: None

Notes:
Agreed - should change these to be Flow.

headings -- numbered vs bare PR #7877
State: Open
Resolution: None
User: None

Notes:
You are correct - there is nothing but good taste to prevent people doing silly things. However, we need better wording to clarify that the content model does indeed support bad taste.

What is the scope of a header? PR #7878
State: Approved
Resolution: Accepted
User: None

Notes:
Group agrees that the scope needs a better explanation.

8. XHTML Text Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

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.32] , @full [p.48]	(PCDATA | Text [p.47])*
cite	Common [p.32]	(PCDATA | Text [p.47])*
code	Common [p.32]	(PCDATA | Text [p.47])*
dfn	Common [p.32]	(PCDATA | Text [p.47])*
em	Common [p.32]	(PCDATA | Text [p.47])*
kbd	Common [p.32]	(PCDATA | Text [p.47])*
l	Common [p.32]	(PCDATA | Text [p.47])*
q	Common [p.32]	(PCDATA | Text [p.47])*
samp	Common [p.32]	(PCDATA | Text [p.47])*
span	Common [p.32]	(PCDATA | Text [p.47])*
strong	Common [p.32]	(PCDATA | Text [p.47])*
sub	Common [p.32]	(PCDATA | Text [p.47])*
sup	Common [p.32]	(PCDATA | Text [p.47])*
var	Common [p.32]	(PCDATA | Text [p.47])*

l element content model

Content model of the l element should not allow nested lines

The content model for this module defines a content set:

Text
abbr [p.48] | cite [p.48] | code [p.49] | dfn [p.49] | em [p.49] | kbd [p.50] | q [p.51] | samp [p.51] | span [p.52] | strong [p.52] | var [p.53]

Implementations: RELAX NG [p.170] , XML Schema [p.231]

8.1. The abbr element

The abbr [p.48] element indicates that a text fragment is an abbreviation (e.g., W3C, XML, Inc., Ltd., Mass., etc.); this includes acronyms.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

full = IDREF [p.27]
: This attribute locates an element that defines the full expansion of an abbreviation. The referenced element must be in the same document as the abbreviation.

The content of the abbr [p.48] element specifies the abbreviated expression itself, as it would normally appear in running text. The @title [p.64] or @full [p.48] attributes may be used to provide the full or expanded form of the expression. Such an attribute should be repeated each time the abbreviation is used in the document.

Examples

```
<abbr title="Limited">Ltd.</abbr>
<abbr title="Abbreviation">abbr.</abbr>
<p>The <span id="w3c">World Wide Web Consortium</span> (<abbr full="#w3c">W3C</abbr>)
   develops interoperable technologies (specifications, guidelines, software, and tools)
   to lead the Web to its full potential. <abbr full="#w3c">W3C</abbr> is a forum for
   information, commerce, communication, and collective understanding.</p>
```

8.2. The cite element

The cite [p.48] element contains a citation or a reference to other sources.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

In the following example, the cite [p.48] element is used to reference the book from which the quotation is taken:

cite as book reference

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

cite to reference another specification

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

8.3. The code element

The code [p.49] element contains a fragment of computer code.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

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

8.4. The dfn element

The dfn [p.49] element contains the defining instance of the enclosed term.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

```
<p role="definition">
    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.
</p>
```

8.5. The em element

The em [p.49] element indicates emphasis for its contents.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

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

8.6. The kbd element

The kbd [p.50] element indicates input to be entered by the user.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

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

8.7. The l element

The l [p.50] element represents a semantic line of text (e.g., a line of verse or a line of computer code).

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

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 style sheet if needed.

Sample program listing

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

CSS Style sheet to number each line

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

8.8. The q element

This element designates an inline text fragment of quoted text.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

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

Nested quotations using q

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

q with a cite attribute

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

8.9. The samp element

The samp [p.51] element designates sample output from programs, scripts, etc.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

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

8.10. The span element

The span [p.52] element, in conjunction with the @id [p.64] , @class [p.63] and @role 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.113] , the @xml:lang [p.73] attribute, the @dir [p.75] attribute etc., to tailor XHTML to their own needs and tastes.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example of span for cross references

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

8.11. The strong element

The strong [p.52] element indicates higher importance for its contents than that of the surrounding content.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

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

8.12. The sub element

The sub [p.52] element indicates that its contents should regarded as a subscript.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

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

8.13. The sup element

The sup [p.53] element indicates that its contents should be regarded as a super-script.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Many scripts (e.g., French) require superscripts or subscripts for proper rendering. The sub [p.52] and sup [p.53] elements should be used to markup text in these cases.

Example

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

8.14. The var element

The var [p.53] element indicates an instance of a variable or program argument.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

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

8.15. Issues

PCData not in Text PR #7876
State: Approved
Resolution: Accepted
User: None

Notes:
The group agrees that we should change the abstract module definitions such that PCDATA is just included in Text.

XHTML 2.0: Text Module/<l> vs.
 element PR #7882
State: Approved
Resolution: Accepted
User: None

Notes:
We will put br back into XHTML 2.

RE: [off list] XHTML 2.0 - dfn : Content model and usability PR #7885
State: Approved
Resolution: Accepted
User: None

Notes:
The specification has always had the dfn element.

Re: [XHTML 2.0] emphesis PR #7899
State: Approved
Resolution: Accepted
User: None

Notes:
The working group has agreed that nested "em" elements should indicate higher emphasis. This will be reflected in the text in a future draft.

9. XHTML Hypertext Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

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.67] attribute) or hyperlink anchor (using the @id [p.64] attribute), this element remains for explicit markup of links, though is otherwise identical in semantics to the span [p.52] element.

This module supports the following element:

Element	Attributes	Minimal Content Model
a [p.55]	Common [p.32]	(PCDATA | Text [p.47])*

This module adds the a [p.55] element to the Text [p.47] content set of the Text [p.47] Module and the access [p.??] element to the head [p.34] element of the Document [p.33] module.

Implementations: RELAX NG [p.174] , XML Schema [p.235]

9.1. The a element

An a [p.55] element defines an anchor. Since hypertext attributes such as @href [p.67] may be applied to any element, this element is not strictly necessary, being equivalent to a span [p.52] , but has been retained to allow the expression of explicit links.

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Example

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

10. XHTML List Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

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.58]	Common [p.32]	caption [p.79] ?, ((dt [p.58] | dd [p.58])+ | di [p.58] +)
di [p.58]	Common [p.32]	(dt [p.58] +, dd [p.58] *)
dt [p.58]	Common [p.32]	(PCDATA | Text [p.47])*
dd [p.58]	Common [p.32]	(PCDATA | Flow [p.39])*
caption [p.79]	Common [p.32]	(PCDATA | Text [p.47])*
nl [p.59]	Common [p.32]	caption [p.79] , li [p.60] +
ol [p.60]	Common [p.32]	caption [p.79] ?, li [p.60] +
ul [p.60]	Common [p.32]	caption [p.79] ?, li [p.60] +
li [p.60]	Common [p.32] , @value [p.61]	(PCDATA | Flow [p.39])*

This module also defines the content set List with the minimal content model (dl | nl | ol | ul)+ and adds this set to the Structural [p.39] content set of the Structural [p.37] Module.

Implementations: RELAX NG [p.174] , XML Schema [p.235]

XHTML offers authors several mechanisms for specifying lists of information. Lists may contain:

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

The previous list, for example, is an unordered list, created with the ul [p.60] element:

Example

```
<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.60] element, contains information where order is important, 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.58] 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:

Example

```
<dl>
<dt>Lower cost</dt>
<dd>The new version of this product costs significantly less than the
previous one!</dd>
<dt>Easier to use</dt>
<dd>We've changed the product so that it's much easier to
use!</dd>
<dt>Safe for kids</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>
```

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

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and

Metainformation

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 [p.58] element. The description is given with a dd [p.58] element. The term and its definition can be grouped within a di [p.58] element to help clarify the relationship between a term and its definition(s).

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:

Example

```
<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>
```

10.2. The nl element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Navigation lists are intended to be used to define lists of selectable items for ordered presentation. These may be presented in a number of ways, for instance as a navigation bar, or as a menu. Note that a navigation list always starts with a label [p.??] element that defines the label for the list.

Basic navigation list structure

```
<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>
```

10.3. The ol and ul elements

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Both types of lists are made up of sequences of list items defined by the li [p.60] element. The difference is that ol [p.60] lists represent lists of items that are essentially ordered (such as the steps of a recipe), while ul [p.60] lists represent lists of items that are essentially unordered (such as shopping lists).

Basic list structure

```
<ol>
   <li>Spring</li>
   <li>Summer</li>
   <li>Autumn</li>
   <li>Winter</li>
</ol>
```

10.4. The li element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

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.60] .

The li [p.60] element defines a list item within an ordered, unordered, or navigation list.

Within a list, each `li` element has an associated number, which is used for numbering list items in ordered lists:

- If the `li` element has a `value` attribute, the associated number is the value of that attribute;
- otherwise, if the `li` element is the first in the list, then the number has the value 1;
- otherwise the number is one higher than the number of the preceding `li` in the same list.

10.5. Issues

[XHTML2] 11.3. The ol , and ul elements PR #7663
State: Open
Resolution: None
User: None

Notes:
The working group is not in favor or the definition of a "continueFrom" attribute that would allow continuation of list numbering, simply because there is no way to describe the behavior in current styling languages. However, there is a usecase for being able to define groups of list items and label them.... The working group is continuing to discuss this issue. To be *really* fair to the required structure in his use case ... you really want something like this: <ol> <group> .. .. </group> <group> <label>... ... ... </group> The use case has two different structures imposed on top of each other A bit like <label for=""> in HTML4

Re: WD-xhtml2-20040722: Some navigation list requirements (IMHO) PR #7867
State: Open
Resolution: None
User: None

Notes:

11. XHTML Core Attributes Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

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

Attribute	Notes
@class [p.63] (NMTOKENS [p.27])	
@id [p.64] (ID [p.27])	
@xml:id [p.64] (ID [p.27])	
@layout [p.64] (`irrelevant*|relevant`)	
@title [p.64] (Text [p.29])	

11.1. 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.63] attribute can be used for different purposes in XHTML, for instance as a style sheet [p.113] 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.42] element is used in conjunction with the @class [p.63] attribute to identify a particular type of paragraph.

Example

```
<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 color, or where necessary by not displaying it at all.

It is good style to use names that represent the purpose of the element rather than the visual presentation to be used. For instance don't use `class="red"`, but rather `class="urgent"`, or similar.

id = ID [p.27]
The @id [p.64] attribute assigns an identifier to an element. The value of this attribute must be unique within a document. This attribute MUST NOT be specified on an element in conjunction with the @xml:id [p.64] attribute.

The @id [p.64] attribute has several roles in XHTML:

- As a style sheet [p.113] selector.
- As a target anchor [p.55] for hypertext links.
- As the name of a declared object [p.103] 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.64] values:

Example

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

xml:id = ID [p.27]
The @xml:id [p.64] attribute assigns an identifier to an element. The value of this attribute must be unique within a document. This attribute MUST NOT be specified on an element in conjunction with the @id [p.64] attribute.

layout = `irrelevant*|relevant`
This attribute allows authors to indicate whether the whitespace within an element is relevant to the meaning of the content or not; for instance, visual user agents could display the whitespace. The default is that it is *irrelevant*. Some elements, notably pre [p.43] override this default. See whitespace handling [p.??] in the section on XHTML Family User Agent Conformance [p.21] for more information.

Example

```
<p class="poem" layout="relevant">
(with wee ears and see?
tail frisks)
              (gonE)
</p>
```

title = Text [p.29]
This attribute defines meta-information about the element on which it is set.

Example

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

Implementation: RELAX NG [p.177] , XML Schema [p.??]

12. XHTML Hypertext Attributes Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

The Hypertext Attributes Module defines the Hypertext [p.67] attribute collection. This collection allows an element to be the start point of a hypertext link to a remote resource.

12.1. Hypertext Attribute Collection

cite = URI [p.29]
: The value of this attribute is a URI [p.29] 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 MUST provide a means for the user to actuate the link.

  Example

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

href = URI [p.29]
: This attribute specifies a URI that is actuated when the element is activated.

  Actuation occurs as the default action of a [DOM [p.295]] DOMActivate event for the element on which the attribute occurs (for instance as the result of the user clicking on the associated element). If elements contained within an element using an @href [p.67] also use an @href [p.67] attribute, the User Agent must provide a mechanism for actuating any of these "nested" URIs.

  Example

```
<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>
```

hreflang = LanguageCodes [p.28]

  This attribute specifies the primary language of the resource designated by @href [p.67] . At its most general, it is a comma-separated list of language ranges with optional accept parameters, as defined in section 14.4 of [RFC2616 [p.296]] 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>
```

hrefmedia = MediaDesc [p.29]

This attribute indicates the type(s) of media to which to make available the content referenced by the associated @href [p.67] URI.

Example

```
<p href="http://www.example.com/forPrinters.html"
   hrefmedia="print">
   A printable version of this page.
</p>
```

hreftype = ContentTypes [p.28]

This attribute specifies the allowable content types of the relevant @href [p.67] URI. See ContentTypes [p.28] for details of how it is used.

Example

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

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 the user agent navigate to the next element that can receive focus.

The sequence of focusable [p.16] 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 reference to a fragment identifier (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 reference to a fragment identifier when the document is loaded:
   1. If the root element of the document has a @nextfocus [p.68] attribute, and the element referred to by the attribute is focusable, the element must receive focus. The user agent must ensure the beginning of the element is visible on the display.
   2. If the root element has no @nextfocus [p.68] 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, then no element receives focus.

Once a focusable element in the document has focus, upon requesting that focus change to the next focusable element, the user agent MUST follow these rules when determining where focus is next set:

1. The next focus of an element *without* a @nextfocus [p.68] attribute is the next focusable [p.16] element in document order. If there are no remaining focusable [p.16] elements in document order, the next focus must be on the first focusable [p.16] element in document order.
2. The next focus of an element *with* a @nextfocus [p.68] attribute is the element referenced by that attribute if it is focusable [p.16] , otherwise the next focus of that element.

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):

Example

```
<table>
<tr><td id="a" href="nw" nextfocus="b">NW</td>
    <td id="c" href="ne" nextfocus="d">NE</td></tr>
<tr><td id="b" href="sw" nextfocus="c">SW</td>
    <td id="d" href="se">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.

In the event no element in the document has focus, when the user requests the previous focusable [p.16] 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.70] 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.

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.298]]. However, values of this attribute that begin with the character '_' are reserved.

Example

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

xml:base = URI [p.29]

This attribute specifies the base URI from which to resolve relative URIs. It is normatively defined in [XMLBASE [p.297]]. 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):

1. The @xml:base [p.70] attribute set for the element itself.
2. The closest parent element that has the @xml:base [p.70] attribute set (i.e., the @xml:base [p.70] attribute is inherited).
3. The HTTP "Content-Location" header (which may be configured in a server).
4. 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>
```

When this module is selected, the Hypertext Attributes Collection is included in the Common [p.32] attribute collection.

Implementations: RELAX NG [p.178] , XML Schema [p.??]

12.2. Issues

Fw: [XHTML 2] 13.1 Hypertext Attributes Module - nextfocus PR #7792
State: Approved
Resolution: None
User: None

Notes:
It is up to the document author to ensure that anchors in a document are appropriate for the document's audience. It would be inappropriate for a user agent to second guess the author's intent by adjusting focus to some parent element of the target. With regard to form fields and incrementally loading user agents, the working group feels that it is unlikely a document's fields should be active before the document is done loading. Loading is complete when the DOMload event fires - before that the user should not be able to interact with the document's content - the content may not have been properly prepared / initialized until after that.

13. XHTML I18N Attribute Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

This module defines the I18N [p.73] attribute collection.

13.1. I18N Attribute Collection

xml:lang = LanguageCode [p.28]

This attribute indicates the language of an element's attribute values and text content, and of all elements it contains, unless overridden. It is defined normatively in [XML [p.297]] 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):

1. The @xml:lang [p.73] attribute set for the element itself.
2. The closest parent element that has the @xml:lang [p.73] attribute set (i.e., the @xml:lang [p.73] attribute is inherited).

In this example, the default text-processing 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.

Example

```
<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 had signs up saying <em xml:lang="fr">PAIN</em>,
but despite that the bakers seemed quite happy.
</p>
</body>
</html>
```

When this module is selected, the I18N Attribute Collection is included in the Common [p.32] attribute collection.

Implementations: RELAX NG [p.180] , XML Schema [p.240]

13.2. Issues

Internationalization: translate attribute PR #7883
State: Approved
Resolution: Reject
User: None

Notes:
This is what ITS is for. We plan to incorporate ITS into XHTML 2.

14. XHTML Bi-directional Text Attribute Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

The Bi-directional Text module defines the Bi-directional [p.75] attribute collection.

14.1. Bi-directional Text Collection

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

This attribute allows the author to specify the direction of the element's text content. This direction affects the display of characters as defined in Unicode Standard Annex #9: The Bidirectional Algorithm [UAX9 [p.296]], and defines directional properties of text as defined by CSS2 [CSS2 [p.295]]. The default value of this attribute is `ltr`. 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>
The Hebrew word for "Hebrew" is
<span xml:lang="he">&#x5e2;&#x5d1;&#x5e8;&#x5d9;&#x5ea;</span>,
but since Hebrew letters have intrinsic right-to-left directionality,
I had to type the word starting from the letter "&#x5e2;",
i.e. <span xml:lang="he" dir="lro">&#x5e2;&#x5d1;&#x5e8;&#x5d9;&#x5ea;</span>.
</p>
```

This might display as

The Hebrew word for "Hebrew" is עברית, but since Hebrew letters have intrinsic right-to-left directionality, I had to type the word starting from the letter "ע", i.e. תירבע.

14.1.1. 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.75] attribute. The default value of the @dir [p.75] attribute is `ltr` (left-to-right text).

When the @dir [p.75] 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.75] attribute on a nested element overrides the inherited value.

To set the base text direction for an entire document, set the @dir [p.75] attribute on the html [p.33] element.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 2.0//EN"
   "http://www.w3.org/MarkUp/DTD/xhtml2.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" dir="rtl">
<head>
<title><em>...a right-to-left title...</em></title>
</head>
<body>
<em>...right-to-left text...</em>
<p dir="ltr"><em>...left-to-right text...</em></p>
<p><em>...right-to-left text again...</em></p>
</body>
</html>
```

Inline-level elements, on the other hand, do not inherit the @dir [p.75] attribute. This means that an inline element without a @dir [p.75] attribute does **not** open an additional level of embedding with respect to the bidirectional algorithm.

An element is considered to be block-level if its presentation, when expressed in [CSS2 [p.295]], is `display: block` and inline-level if its presentation, when expressed in [CSS2 [p.295]], is `display: inline`.

14.1.2. The effect of style sheets on bidirectionality

In general, using style sheets (such as [CSS2 [p.295]] to change an element's visual rendering from the equivalent of `display: block` to `display: 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-level element that does not have a @dir [p.75] attribute is transformed to a block-level element by a style sheet, it inherits the @dir [p.75] attribute from its closest parent block-level element to define the base direction of the block.

When a block-level element that does not have a @dir [p.75] attribute is transformed to an inline-level 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.75] attribute (assigned the inherited value) to the transformed element.

When this module is selected, the Bi-directional Text Collection is included in the Common [p.32] attribute collection.

Implementations: RELAX NG [p.181] , XML Schema [p.240]

14.2. Issues

[XHTML 2] 15 Bi-directional text collection and embedded attributes? PR #7783
State: Approved
Resolution: Modify and Accept
User: None

Notes:
The dir attribute does not apply to embedded content. The src attribute is not equivalent to an xml "include" - it is a reference to a (potentially) external resource that is rendered in the way appropriate to that resources type. But that rendering is done in the context of a separate renderer; ala the object element. Similary, the styling from stylesheets that apply to the parent document does not apply to any embedded content. The fact that your example is "text" does not really matter - text is no more special than any other embedded content. It is handled by however the user agent processes text content, but in a different context than the parent element. We will add text to the src attribute description to clarify this.

15. XHTML Caption Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

The Caption Module defines an element to be used when annotating a description for certain elements. The element and attributes defined by this module are:

Elements	Attributes	Minimal Content Model
caption [p.79]	Common [p.32]	(PCDATA | Text [p.47])*

Implementations: RELAX NG [p.201] , XML Schema [p.241]

15.1. The caption element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

When present, the caption [p.79] element's text should describe the nature of the table, list or object. The caption [p.79] element is only permitted immediately after the table [p.??] start tag, the list [p.??] start tag or the object [p.103] start tag. A table [p.??] , list [p.??] or object [p.103] element may only contain one caption [p.79] 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.

16. XHTML Edit Attributes Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

This module defines the Edit [p.81] 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.

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"
```

When this module is selected, the Edit Attribute Collection is included in the Common [p.32] attribute collection.

Implementations: RELAX NG [p.182] , XML Schema [p.241]

17. XHTML Embedding Attributes Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

The Embedding Attributes module defines the Embedding [p.83] 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 (e.g., network unavailable, no resource available at the URI given, inability of the user agent to process the type of resource) or an associated @ismap [p.92] attribute fails, 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.

Example

```
<p src="holiday.png" srctype="image/png">
        <span src="holiday.gif" srctype="image/gif">
                An image of us on holiday.
        </span>
</p>
<table src="temperature-graph.png" srctype="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

encoding = Encodings [p.27]

This attribute specifies the allowable encoding of the external resource referenced by the @src [p.84] attribute. At its most general, it is a comma-separated list of encodings, such as "utf-8", "utf8, utf-16", or "utf-8, utf-16, *".

The user agent must use this list as the field value of the `accept-charset` 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-charset request header.

User agents should use a similar technique when using other protocols that allow encoding negotation

When using protocols that do not allow encoding negotiation to retrieve resources whose encodings are not self-identifying, the user agent should use the first encoding in the attribute's value as the indication of the resource.

Example

```
<style type="text/css" src="style/home" encoding="utf-8" />
```

src = URI [p.29]
This attribute specifies the location of an external source for the contents of the element. Actuation occurs as the default action of a [DOM [p.295]] load event for the element that the attribute occurs on.

srctype = ContentTypes [p.28]

This attribute specifies the allowable content types of the resource referenced by the relevant @src [p.84] URI.

Example

```
<handler src="pop" srctype="application/x-javascript, text/x-newspeak" />
<style src="midnight" srctype="text/css, text/x-mystyle" />
<p src="w3c-logo" srctype="image/png, image/jpeg;q=0.2">W3C logo</p>
<span src="logo.png">Our logo</span>
<span src="theme.mp3" srctype="audio/x-mpeg">Our theme jingle</span>
```

When this module is selected, the Embedding Attribute Collection is included in the Common [p.32] attribute collection.

Implementations: RELAX NG [p.183] , XML Schema [p.242]

17.2. Issues

Re: Formal Response to My issue on styling embedding attributes. PR #7724
State: Open
Resolution: None
User: None

Notes:
Original message at: http://lists.w3.org/Archives/Public/www-html-editor/2005AprJun/0064 This is a reply to issue 7655, where SP also replied

17 Embedding Attributes User aborted downloads. PR #7730
State: Approved
Resolution: Accepted
User: None

Notes:
Any failure to completely process the reference results in falling back. User abort is also such a failure. We will ensure that the definition of the embedding attributes is clear about this.

17 Embedding Attributes - Invalid XHTML 2 documents PR #7731
State: Approved
Resolution: Modify and Accept
User: None

Notes:
The working group believes that if a resource is successfully accessed, the fallback is NOT accessed even if the resource cannot be appropriately rendered by the user agent. We will ensure the specification makes this clear. Moreover, things brought in via the src attribute are in an independent context - just as in the object element.

[XHTML 2] 17.1 Encoding attribute PR #7732
State: Approved
Resolution: Reject
User: None

Notes:
The encoding applies to retrieving a specific version of a resource to hand off to a resource processor for an embedded attribute. It has nothing to do, per se, with the capabilities of the user agent. We will make it clear in the text that this is the case.

[XHTML 2] Embedding attributes and nextFocus PR #7733
State: Approved
Resolution: Modify and Accept
User: None

Notes:
Embedded content is not in the document flow, so it is not availabnle for use in nextfocus. We will clarify this.

17 Embedding Attributes Success/Failure status codes. PR #7734
State: Approved
Resolution: Modify and Accept
User: None

Notes:
This specification should not specify anything about how the underlying protocols associated with a URI scheme report success or failure. We will clarify in the spec that this is the responsibility of the associated protocol.

[XHTML 2] 17 Embedding Attributes - srcType PR #7735
State: Approved
Resolution: None
User: None

Notes:
Add note to section that effect is defined in the datatype definition.

[XHTML 2] Embedding XHTML Resources Linking. PR #7736
State: Approved
Resolution: Modify and Accept
User: None

Notes:
Content brought in via the src attribute is placed *within* the surrounding element, so any annotation on the surrounding element would apply to that element. In this case, the href attribute would make the surrounding element (the a) linkable. We will clarify this in the specification.

[XHTML 2] Embedded Resources containing links PR #7737
State: Approved
Resolution: Modify and Accept
User: None

Notes:
Embedded content is in an independent context, so any links within that content would replace the embedded content, and not effect the surrounding document. We will examine providing functionality to embed content inline in a specification separately.

[XHTML 2] Embedding Attributes Examples PR #7738
State: Approved
Resolution: Reject
User: None

Notes:
The purpose of this example is to demonstrate that it is possible to use src on tables. While it might be more sensible to embed the table within an img element, it would not achieve the same result in the example.

[XHTML 2] 17 Clipping of embedded documents to viewport. PR #7739
State: Approved
Resolution: Accepted
User: None

Notes:
The embedded resource is in its own context and has its own environment (e.g., object or iframe) and any styling would be relative to that environment.

[XHTML 2] 17 Embedding Attributes PR #7774
State: Approved
Resolution: Accepted
User: None

Notes:
We will provide additional use cases and examples.

xhtml2 attributes type, srctype and hreftype PR #7892
State: Approved
Resolution: Accepted
User: None

Notes:
We will add srclang. All the embedding attributes should be equivalent to the linking attributes. We will also ensure that @type is used consistently.

18. XHTML Image Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

The Image Module provides basic image embedding, and may be used in some implementations independently of client side image maps. The Image Module supports the following element and attributes:

Elements	Attributes	Minimal Content Model
img	Common [p.32]	(PCDATA | caption [p.79] | Text [p.47])*

When this module is used, it adds the `img` element to the Text [p.47] content set of the Text [p.47] Module.

Implementations: RELAX NG [p.183] , XML Schema [p.243]

18.1. The img element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

The img [p.89] element is a holder for embedding attributes such as @src [p.84] . Since these attributes may be applied to any element, the img [p.89] element is not strictly necessary, but is included to ease the transition to XHTML2. Like the object [p.103] element, this element's content is only presented if the referenced resource is unavailable.

In the following example, the W3C logo would be presented if it were available. If it were unavailable, then the enclosed text would be presented.

Example

```
<img src="W3C.png">W3C</img>
```

19. XHTML Image Map Attributes Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

This module defines a collection of 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.84] attribute mean that the `nl` element will be displayed instead of the image, thus making the page still useful:

Example

```
<html xmlns="http://www.w3.org/2002/06/xhtml2">
   <head>
      <title>The cool site!</title>
   </head>
   <body>
     <p src="navbar1.png" srctype="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.60] in an nl [p.59] is not required to have an @href [p.67] attribute. In that case, the relevant region of the image is inactive.

Example

```
<p src="image.png" srctype="image/png" usemap="#map1">
   <nl id="map1">
   <label>Navigation that has an inactive ring</label>
   <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>
```

Note that W3C is working on profiles of XHTML that include versions of SVG [SVG [p.298]], which include more structured ways of creating imagemap-like behavior.

The Image Map Attributes Module enables the Map [p.92] attribute collection. If this module is included, the embedding [p.83] module must also be included.

19.1. Image Map Attribute Collection

usemap = URI [p.29]

This attribute associates an image map with an nl [p.59] element. The value of `usemap` should match the value of the @id [p.64] attribute of an nl [p.59] element that contains one or more li [p.60] elements with @shape [p.93] and @coords [p.93] attributes.

If accessing the URI fails or the referenced element is not an nl [p.59] element, then the associated @src [p.84] attribute is considered to have failed as well, so that the nested content will be processed instead.

ismap = "ismap"

This attribute indicates that the associated embedded resource 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. Coordinates are expressed as pixel values relative to the embedded resource, and start at (0,0) at the top left corner.

When an ismap attribute is specified, click events are not delivered to the embedded resource, regardless of its type.

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.67] 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.) must 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.93] 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.29] . All values are separated by commas. The coordinates of the top, left corner of an area are `0, 0`.

Implementation: RELAX NG [p.184] , XML Schema [p.243]

20. XHTML Media Attribute Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

The Media Attribute Module defines the `media` attribute. When this module is selected, it activates the Media [p.95] attribute collection.

20.1. Media Attribute Collection

media = MediaDesc [p.29]

The value of this attribute is a comma-separated list of media descriptors for which this `access` element is intended. When the value of this attribute matches the current processing media, the associated `access` element is considered *active* and processed normally; otherwise it is *inactive* and ignored. The default value for this attribute is `all`.

Example

```
<style src="style.css" type="text/css" media="screen" />
<span src="photo.jpg" media="screen">Me at work</span>
<span src="photo-hires.jpg" media="print">Me at work</span>
```

When this module is selected, the Media Attribute Collection is included in the Common [p.32] attribute collection.

Implementations: RELAX NG [p.185] , XML Schema [p.244]

21. XHTML Metainformation Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

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 module. The interpretation of those attributes in conjunction with any elements, including the ones defined in this module, are spelled out in [RDFASYNTAX [p.296]].

Elements and attributes in this module are:

Elements	Attributes	Minimal Content Model
link	Common [p.32]	(link | meta)*
meta	Common [p.32]	(PCDATA | Text [p.47])*

When this module is selected, the link [p.97] and meta [p.99] elements are added to the Structural [p.39] and Text [p.47] content sets of the Structural [p.37] and Text [p.47] Modules. In addition, the elements are added to the content model of the head [p.34] element defined in the Document [p.33] Module. Finally, when this module is selected, the associated Metainformation Attributes module must also be selected.

Implementations: RELAX NG [p.187] , XML Schema [p.244]

21.1. The link element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

This element defines a link. Link [p.97] 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). User agents should enable activation of links and the retrieval of link targets. Since link [p.97] elements may have no content, information from the @rel and @title [p.64] attributes should be used when labelling links.

This example illustrates how several link [p.97] definitions may appear in the head [p.34] section of a document. The current document is "Chapter2.html". The @rel 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.

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

21.1.1. Forward and reverse links

While the @rel attribute specifies a relationship *from* this document *to* another resource, the @rev attribute specifies the reverse relationship.

Consider two documents A and B.

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

Has exactly the same meaning as:

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

namely that document B is the index for document A.

Both the @rel and @rev attributes may be specified simultaneously.

21.1.2. Links and search engines

Authors may use the link [p.97] 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 @hreflang [p.67] attribute to indicate to a search engine where to find other language versions of a document. Note that for the sake of the example the @xml:lang [p.73] attribute has been used to indicate that the value of the @title [p.64] attribute for the link [p.97] element designating the French manual is in French.

```
<html ... xml:lang="en">
<head>
<title>The manual in English</title>
<link title="The manual in Dutch"
      rel="alternate"
      hreflang="nl"
      href="http://example.com/manual/dutch.html"/>
<link title="La documentation en Français"
      rel="alternate"
      hreflang="fr" xml:lang="fr"
      href="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"
      hreftype="application/postscript"
      rel="alternate"
      href="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"
      hreftype="application/xhtml+xml"
      href="http://example.com/manual/start.html"/>
</head>
```

21.2. The meta element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

The meta [p.99] 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 defines a small normative set of properties, but users may extend this set as described for the @property attribute.

Each meta [p.99] element specifies a property/value pair. The @property attribute identifies the property and the content of the element or the value of the @content attribute specifies the property's value.

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

Example

```
<meta property="dc:creator">Steven Pemberton</meta>
```

Note. *The meta [p.99] element is a generic mechanism for specifying metadata. However, some XHTML elements and attributes already handle certain pieces of metadata and may be used by authors instead of meta [p.99] to specify those pieces: the title [p.35] element, the address [p.39] element, the @edit [p.81] and related attributes, the @title [p.64] attribute, and the @cite [p.67] attribute.*

Note. *When a property specified by a meta [p.99] element takes a value that is a URI [p.29] , some authors prefer to specify the metadata via the link [p.97] element. Thus, the following metadata declaration:*

Example

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

might also be written:

Example

```
<link rel="dc:identifier"
      href="http://www.rfc-editor.org/rfc/rfc3236.txt" />
```

21.2.1. meta and search engines

A common use for meta [p.99] is to specify keywords that a search engine may use to improve the quality of search results. When several meta [p.99] elements provide language-dependent information about a document, search engines may filter on the @xml:lang [p.73] attribute to display search results using the language preferences of the user. For example,

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, Grèce, soleil</meta>
```

The effectiveness of search engines can also be increased by using the link [p.97] 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.

21.3. Issues

rebuild link element: chapter, section / subsection PR #7869
State: Open
Resolution: None
User: None

Notes:

22. XHTML Object Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

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.32] , @archive [p.103] (URIs [p.29]), @content-length [p.103] (Number [p.29]), @declare [p.103] ("declare")	(caption [p.79] ?, standby [p.112] ?, param [p.108] *, (PCDATA | Flow [p.39])*)
param	@id [p.64] (ID [p.27]), @name [p.108] * (CDATA [p.27]), @value [p.108] (CDATA [p.27]), @valuetype [p.108] ("data"* | "ref" | "object"), @paramtype [p.108] (ContentType [p.27])	EMPTY
standby	Common [p.32]	(PCDATA | Text [p.47])*

When this module is used, it adds the `object` element to the Text [p.47] content set of the Text [p.47] module.

Implementations: RELAX NG [p.188] , XML Schema [p.246]

22.1. The object element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

archive = URIs [p.29]
: This attribute may be used to specify a space-separated list of URIs for archives containing resources relevant to the object, which may include the resources specified by the @src [p.84] attribute.

content-length = Number [p.29]
: 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.296]].

declare = "`declare`"
: When present, this boolean attribute makes the current element a declaration only - one that is to be executed only after the document has completed loading and has been called

through a user event.

22.1.1. Defining terminology

The following terms are 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.

22.1.2. 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 media 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 media 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.103] 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 media 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.

The object [p.103] element may also appear in the content of the head [p.34] element. Since user agents generally do not render elements in the head [p.34] , authors should ensure that any object [p.103] element in the head [p.34] does not specify content that is expected to be made available to the user.

22.1.3. Rules for processing objects

A user agent must interpret an object [p.103] 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 for definitions of param [p.108] elements (see object initialization) or elements that take advantage of the Map [p.92] attribute collection.
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.

When a user agent is able to successfully process an object element it MUST not attempt to process inner elements.

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 cascading 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.103] element. The @src [p.84] attribute specifies the location of the object data:

Example

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

The following example shows an object [p.103] element coded to process an image. The @src [p.84] attribute specifies the location of the object data, in this case the image to be processed, and the @srctype [p.84] attribute specifies the media type associated with the object data:

Example

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

The following example shows how an applet element can be converted to an object [p.103] element. The codebase attribute is replaced with the xml:base attribute. The code attribute is replaced with the @src [p.84] attribute. The width and the height of the applet are defined using CSS. The param [p.108] elements are not modified since the values within the param [p.108] 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.

Example

```
<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>
```

Example

```
<style type="text/css">
#obj1 {width:150px; height:150px;}
</style>
...
<object id="obj1"
  xml:base="http://www.example.com/applets/classes"
  srctype="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.103] element declaration when an embedded object is not defined.

The following example demonstrates how alternate text may be used within an object [p.103] element.

Example

```
<object src="http://www.example.com/foo.mp3" srctype="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>
```

In the following example, we embed several object [p.103] 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.

Example

```
<!-- First, try the applet -->
<object
    src="http://www.example.com/applets/classes/TheEarth.class"
    srctype="application/x-java-applet">
      <!-- Else, try the video -->
      <object
          src="TheEarth.mpeg"
          srctype="video/mpeg"
          xml:base="http://www.example.com/videos/">
            <!-- Else, try the image -->
            <object
                src="TheEarth.png"
                srctype="image/png"
                xml:base="http://www.example.com/images/">
                <!-- Else process the alternate text -->
                The <strong>Earth</strong> as seen from space.
            </object>
      </object>
</object>
```

The outermost object [p.103] element declaration specifies an applet that requires no initial values, the @src [p.84] attribute points to the applet class file, and the @srctype [p.84] attribute defines the media type. An @xml:base [p.70] attribute could have been used to point to the base location to access the class file. In this example, however, the @src [p.84] attribute value contains an absolute URL so the @xml:base [p.70] attribute was not required. An @archive [p.103] attribute could have been used if the author needed to include any associated files. The second object [p.103] element declaration specifies an MPEG animation, and the @xml:base [p.70] attribute defines the location of the object data defined in the @src [p.84] attribute. We

also set the @srctype [p.84] 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.

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.*

22.2. 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.108] . 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.108] will be evaluated and passed to the object's implementation as a string.
  - `ref`: The value specified by @value [p.108] 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.108] is an identifier that refers to an object [p.103] declaration in the same document. The identifier must be the value of the @id [p.64] attribute set for the declared object [p.103] element.

type = ContentType [p.27]
: This attribute specifies the content type of the resource designated by the @value [p.108] attribute only in the case where @valuetype [p.108] is set to "ref". This attribute thus specifies for the user agent, the type of values that will be found at the URI designated by value.

param [p.108] 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.108] elements may appear in the content of an object [p.103] element, in any order, but must be placed at the start of the content of the enclosing object [p.103] element, with the exception of optional caption [p.??] and standby

[p.112] 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.108] 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.108] 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.108] elements.

Example

```
<object
    src="http://www.example.com/myclock.class"
    srctype="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.108] attribute is thus set to "ref" and the @value [p.108] is a URI designating the resource.

Example

```
<object
    src="http://www.example.com/gifappli"
    srctype="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.112] element so that the object handler may display a message while the object data is downloading.

When an object [p.103] element is processed, the user agent must search the content for only those param [p.108] 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.103] element is

processed, neither param [p.108] name/value content applies.

Example

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

22.2.1. 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.70] attribute location.

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

Example

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

By setting the @srctype [p.84] 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.70] attribute. The @src [p.84] attribute defines the data to process.

Example

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

22.2.2. 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.103] attribute in the object [p.103] element. At the same time, authors must identify the object declaration by setting the @id [p.64] attribute in the object [p.103] element to a unique value. Later processing of the object data will refer to this identifier.

A declared object [p.103] 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.103] 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.103] 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.

Example

```
<object
    declare="declare"
    id="earth.declaration"
    src="TheEarth.mpg"
    srctype="video/mpeg">
    The <strong>Earth</strong> as seen from space.
</object>
<em>...later in the document...</em>
<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.103] attribute must process the contents of the object [p.103] element.

22.3. The standby element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

The standby [p.112] element specifies a message that a user agent may render while loading the object's implementation and data.

23. XHTML Style Sheet Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

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.113]	Common [p.32] , @disabled [p.113] ("disabled"), @media [p.113] (MediaDesc [p.29])	PCDATA

When this module is used, it adds the style [p.113] element to the content model of the head [p.34] element of the Document [p.33] Module, and also to the Structural content set as defined in the Structural [p.37] . module

Implementations: RELAX NG [p.190] , XML Schema [p.247]

23.1. The style element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

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.29]
: The value of this attribute is a comma-separated list of media descriptors for which this `access` element is intended. When the value of this attribute matches the current processing media, the associated `access` element is considered *active* and processed normally; otherwise it is *inactive* and ignored. The default value for this attribute is `all`.

The style [p.113] element allows an author to put style sheet rules in the head of the document. XHTML permits any number of style [p.113] elements in the head [p.34] section of a document.

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>
```

23.1.1. 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).

23.1.2. 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.29] 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.

23.1.3. Specifying external style sheets

Authors specify external style sheets using the `xml-stylesheet` processing instruction [XMLSTYLE [p.298]], or, for CSS, by using the `@import` facility within a style [p.113] 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`:

Example

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

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

Example

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

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

Example

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

24. XHTML Style Attribute Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

The Style Attribute Module defines the `style` attribute. When this module is selected, the Style Attribute Collection is included in the Common [p.32] attribute collection.

Note: use of the @style [p.117] attribute is strongly discouraged in favor of the style [p.113] element and external style sheets. In addition, content developers are advised to avoid use of the @style [p.117] 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.

Example

```
<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.113] element. For optimal flexibility, authors should define styles in external style sheets.

When this module is selected, the Style Attribute Collection is included in the Common [p.32] attribute collection.

Implementations: RELAX NG [p.190] , XML Schema [p.248]

24.2. Issues

RE: [BULK] - Re: [XHTML2] Spirit of "1.1.3. XHTML 2 and Presentation" PR #7870
State: Approved
Resolution: Accepted
User: None

Notes:
The working group resolved at the f2f meeting in June 08 to remove this attribute and to permit the style element throughout the content model.

Re: [BULK] - Re: [XHTML2] Spirit of "1.1.3. XHTML 2 and Presentation" PR #7871
State: Approved
Resolution: Reject
User: None

Notes:
We are removing the style attribute, but we are ensuring that it is possible to embed style definitions via the style element or the link element anywhere in your document.

25. XHTML Tables Module

This section is *informative*. For the normative version, see [XHTML2 [p.??]].

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.32]	caption [p.79] ?, summary [p.122] ?, (col [p.120] * | colgroup [p.120] *), ((thead [p.138] ?, tfoot [p.138] ?, tbody [p.133] +) | (tr [p.139] +))
summary	Common [p.32]	(PCDATA | Flow [p.39])*
col	Common [p.32] , @span [p.120] (Number [p.29])	EMPTY
colgroup	Common [p.32] , @span [p.120] (Number [p.29])	col [p.120] *
thead	Common [p.32]	tr [p.139] +
tfoot	Common [p.32]	tr [p.139] +
tbody	Common [p.32]	tr [p.139] +
tr	Common [p.32] , Forms	(td [p.133] | th [p.133])+
td	Common [p.32] , @abbr [p.133] (Text [p.29]), @axis [p.133] (CDATA [p.27]), @colspan [p.133] (Number [p.29]), @headers [p.133] (IDREFS [p.27]), @rowspan [p.134] (Number [p.29]), @scope [p.134] ("row", "col", "rowgroup", "colgroup")	(PCDATA | Flow [p.39])*
th	Common [p.32] , @abbr [p.133] (Text [p.29]), @axis [p.133] (CDATA [p.27]), @colspan [p.133] (Number [p.29]), @headers [p.133] (IDREFS [p.27]), @rowspan [p.134] (Number [p.29]), @scope [p.134] ("row", "col", "rowgroup", "colgroup")	(PCDATA | Flow [p.39])*

When this module is used, it adds the table [p.123] element to the Structural [p.39] content set of the Structural Module.

Implementations: RELAX NG [p.191] , XML Schema [p.248]

25.1. The col and colgroup elements

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

span = Number [p.29]
: This attribute must be an integer > 0; the default value is 1. This specifies the number of columns in a colgroup [p.120] , or specifies the number of columns "spanned" by the col [p.120] element.

  Values mean the following:

  - In the absence of a @span [p.120] attribute, each colgroup [p.120] defines a column group containing one column.
  - If the @span [p.120] attribute is used with the colgroup [p.120] element and the value is set to N > 0, that defines a column group containing N columns.
  - If the @span [p.120] attribute is used with the col [p.120] element and the value is set to N > 1, the current col [p.120] element shares its attributes with the next N-1 columns.

  User agents must ignore this attribute if the colgroup [p.120] element contains one or more col [p.120] elements. Instead, the value must be computed by summing the span attributes of the enclosed col [p.120] elements.

The colgroup [p.120] 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.123] may either contain a single implicit column group (no colgroup [p.120] element delimits the columns) or any number of explicit column groups (each delimited by an instance of the colgroup [p.120] element).

The col [p.120] element allows authors to share attributes among several columns without implying any structural grouping. The "span" of the col [p.120] 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.120] 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.120] attribute (default value 1) specifies the number of columns in the group.
2. Each embedded col [p.120] element in the colgroup [p.120] represents one or more columns in the group.

The advantage of using the colgroup [p.120] 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.120] 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.

Example

```
<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>
<em>...the rest of the table...</em>
</table>
```

In this example, the defined width for the colgroup [p.120] constrains all of the columns to fit within that value regardless of the of the defined values within the col [p.120] 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.120] element.

The col [p.120] element allows authors to group together attribute specifications for table columns. The col [p.120] does **not** group columns together structurally -- that is the role of the colgroup [p.120] element. col [p.120] 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.120] element).

25.1.1. 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.123] element contains any colgroup [p.120] or col [p.120] elements, user agents should calculate the number of columns by summing the following:
   - For each col [p.120] element, take the value of its @span [p.120] attribute (default value 1).
   - For each colgroup [p.120] element containing at least one col [p.120] element, ignore the @span [p.120] attribute for the colgroup [p.120] element. For each col [p.120] element, perform the calculation of step 1.
   - For each empty colgroup [p.120] element, take the value of its @span [p.120] attribute (default value 1).
2. Otherwise, if the table [p.123] element contains no colgroup [p.120] or col [p.120] 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.120] or col [p.120] 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.120] .

25.2. The summary element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

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.

User agents MUST provide access to the content of the summary [p.122] 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.

Example

```
<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>
  <em>...the rest of the table...</em>
</table>
```

25.3. The table element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

The table [p.123] element contains all other elements that specify the caption, column groups, columns, rows, and content for a table.

25.3.1. 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.122] 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.122] in the table. Note that the number of rows in a table is equal to the number of tr [p.139] elements contained by the table [p.123] element.

- Group the columns according to any column groups [p.120] 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.

25.3.2. 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.75] attribute for the table [p.123] 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.123] is the only element on which @dir [p.75] reverses the visual order of the columns; a single table row (tr [p.139]) or a group of columns (colgroup [p.120]) cannot be independently reversed.

When set for or inherited by the table [p.123] element, the @dir [p.75] attribute also affects the direction of text within table cells (since the @dir [p.75] attribute is inherited by block-level elements).

To specify a right-to-left table, set the @dir [p.75] attribute as follows:

Example

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

The direction of text in individual cells can be changed by setting the @dir [p.75] attribute in an element that defines the cell.

25.3.3. 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.

25.3.3.1. Associating header information with data cells

Non-visual user agents such as speech synthesizers and Braille-based devices may use the following td [p.133] and th [p.133] element attributes to render table cells more intuitively:

- For a given data cell, the @headers [p.133] attribute lists which cells provide pertinent header information. For this purpose, each header cell must be named using the @id [p.64] 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.133] element for such cells together with the @id [p.64] or @scope [p.134] attributes as appropriate.

- For a given header cell, the @scope [p.134] 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.133] according to which is more convenient; the two attributes fulfill the same function. The @headers [p.133] attribute is generally needed when headers are placed in irregular positions with respect to the data they apply to.
- The @abbr [p.133] 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.133] attribute. Each cell in the same column refers to the same header cell (via the @id [p.64] attribute).

Example

```
<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:

Example

```
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.133] attribute.

Here is the same example substituting the @scope [p.134] attribute for the @headers [p.133] attribute. Note the value "col" for the @scope [p.134] attribute, meaning "all cells in the current column":

Example

```
<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>
   </tr>
</tbody>
</table>
```

Here’s a somewhat more complex example illustrating other values for the @scope [p.134] attribute:

Example

```
<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>&amp;amp;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>&amp;amp;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>&amp;amp;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.134] attribute with the "row" value. Although the first cell in each row contains data, not header information, the @scope [p.134] 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.

25.3.3.2. 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.124]).

Users may also want information about more than one cell, in which case header information provided at the cell level (by @headers [p.133] , @scope [p.134] , and @abbr [p.133]) 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.133] 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:

Example

```
<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.133] or the @scope [p.134] attribute. Thus, meal expenses for 25-Aug-1997 should be marked up to refer to @id [p.64] attribute (whose value here is "a6") of the "San Jose" header cell:

Example

```
<td headers="a6">37.74</td>
```

Each @headers [p.133] attribute provides a list of @id [p.64] 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:

Example

```
<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.133] attribute, nor does it make any recommendations about how user agents may present @axis [p.133] 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:

Example

```
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:

Example

```
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:

Example

```
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).

25.3.3.3. Algorithm to find heading information

In the absence of header information from either the @scope [p.134] or @headers [p.133] 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.124] 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.133] attribute set, then the headers referenced by this attribute are inserted into the list and the search stops for the current direction.

- td [p.133] cells that set the @axis [p.133] attribute are also treated as header cells.

25.4. The tbody element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

The tbody [p.133] element contains rows of table data. In tables that also contain thead [p.138] or tfoot [p.138] elements, all of these sections must contain the same number of columns.

25.5. The td and th elements

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

abbr = Text [p.29]
: 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.128] for more information. The value of this attribute is a comma-separated list of category names.

colspan = Number [p.29]
: 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.120]) 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.64] attribute. Authors generally use the @headers [p.134] 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.134] attribute.

rowspan = Number [p.29]
: 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 current table section (rowgroup) in which the cell is defined. thead [p.138] , tbody [p.133] , and tfoot [p.138] elements are rowgroups.

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.134] 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.124]).
  - **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.120] 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.133] element defines a cell that contains header information. User agents have two pieces of header information available: the contents of the th [p.133] element and the value of the @abbr [p.133] attribute. User agents must render either the contents of the cell or the value of the @abbr [p.133] 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.133] 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.134] and @scope [p.134] 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.124] for information and examples.

The td [p.133] element defines a cell that contains data.

Cells may be empty (i.e., contain no data).

25.5.1. 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.134] and @colspan [p.133] attributes for the th [p.133] and td [p.133] 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</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>
   </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              |
 ---------------------------------------
```

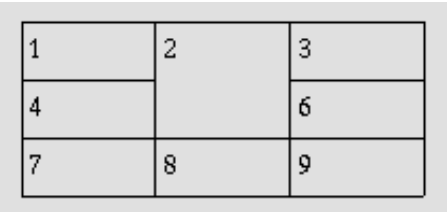
The next example illustrates (with the help of table borders) how cell definitions that span more than one row or column affect the definition of later cells. Consider the following table definition:

```
<table>
<tbody>
   <tr>
      <td>1</td>
      <td rowspan="2">2</td>
      <td>3</td>
   </tr>
   <tr>
      <td>4</td>
      <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.133] 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.133] 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:

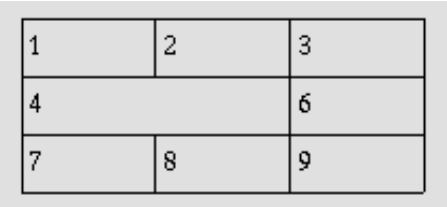
```
<table>
<tbody>
   <tr>
      <td>1</td>
```

```
      <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.133] 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>
```

25.6. The thead and tfoot elements

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

Table rows may be grouped into a table head, table foot, and one or more table body sections, using the thead [p.138] , tfoot [p.138] and tbody [p.133] 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 must contain rows of table data.

When present, each thead [p.138] , tfoot [p.138] , and tbody [p.133] creates a *row group*. Each row group must contain at least one row, defined by the tr [p.139] element.

If the thead [p.138] , tfoot [p.138] , and tbody [p.133] 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.

Example

```
<table>
<thead>
     <tr> <em>...header information...</em></tr>
</thead>
<tfoot>
     <tr> <em>...footer information...</em></tr>
</tfoot>
<tbody>
     <tr> <em>...first row of block one data...</em></tr>
     <tr> <em>...second row of block one data...</em></tr>
</tbody>
<tbody>
     <tr> <em>...first row of block two data...</em></tr>
```

```
      <tr> <em>...second row of block two data...</em></tr>
      <tr> <em>...third row of block two data...</em></tr>
</tbody>
</table>
```

tfoot [p.138] must appear before tbody [p.133] within a table [p.123] definition so that user agents can render the foot before receiving all of the (potentially numerous) rows of data.

25.7. The tr element

Attributes

The Common [p.32] collection
: A collection of other attribute collections, including: Bi-directional [p.75] , Core [p.??] , Edit [p.81] , Embedding [p.83] , Events, Hypertext [p.67] , I18N [p.73] , Map [p.92] , and Metainformation

The tr [p.139] elements acts as a container for a row of table cells.

This sample table contains three rows, each begun by the tr [p.139] 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>
</tbody>
</table>
```

25.8. Issues

Why no nested colgroup or rowgroup? PR #7828
State: Feedback
Resolution: None
User: None

Notes:
We have reviewed this request, but don’t understand how the additional markup supports the use case. What problem are you trying to solve exactly?

td/th scope attribute - rowgroup == (tbody, thead, tfoot) PR #7879
State: Open
Resolution: None
User: None

Notes:
Agree that we need to tighten up the terminology.

nesting colgroup and rowgroups PR #7881
State: Open
Resolution: None
User: None

Notes:
While there are notional "rowgroups" there are no explicit arbitrary rowgroups in HTML nor in XHTML because rows are explicit in tables. colgroups, on the other hand, exist because HTML / XHTML needs a way of referring to columns. Asked the submitter for more information and an example including a description of what the advantages would be.

A. Building XHTML Relax NG Modules

This section is *normative.*

XHTML modules are implemented as RelaxNG fragments. When these fragments are assembled in a specific manner (described in Developing RelaxNG with defined and extended modules [p.149]) is a representation of a complete document type. This representation can then be used for validation of instances of the document type.

The key to combining these fragments into a meaningful DTD is the rules used to define the fragments. This section defines those rules. When these rules are followed, DTD authors can be confident that their modules will interface cleanly with other XHTML-compatible modules.

This specification classifies parameter entities into seven categories and names them consistently using the following suffixes:

.mod
: parameter entities use the suffix `.mod` when they are used to represent a DTD module (a collection of elements, attributes, parameter entities, etc). In this specification, each module is an atomic unit and may be represented as a separate file entity.

.module
: parameter entities use the suffix `.module` when they are used to control the inclusion of a DTD module by containing either of the conditional section keywords `INCLUDE` or `IGNORE`.

.qname
: parameter entities use the suffix `.qname` when they are used to represent the qualified name of an element. See Defining the Namespace of a Module [p.142] for more information on qualified names.

.content
: parameter entities use the suffix `.content` when they are used to represent the content model of an element type.

.class
: parameter entities use the suffix `.class` when they are used to represent elements of the same class.

.mix
: parameter entities use the suffix `.mix` when they are used to represent a collection of element types from different classes.

.attrib
: parameter entities use the suffix `.attrib` when they are used to represent a group of tokens representing one or more complete attribute specifications within an ATTLIST declaration.

For example, in HTML 4, the `%block;` parameter entity is defined to represent the heterogeneous collection of element types that are block-level elements. In this specification, the corollary parameter entity is `%Block.mix;`.

When defining parameter entities in the classes defined here, modules should scope the names of the entities by using unique prefixes. For example, the content model for the element `myelement` in the module mymodule could be named `MYMODULE.myelement.content`. Other schemes are possible. Regardless of the scheme used, module authors should strive to ensure that parameter entities they define are named uniquely so that they do not collide with other parameter entities and so that the interface methods for the module are obvious to its users.

A.1. Defining the Namespace of a Module

XHTML requires that the elements and attributes declared in a module be within a defined XML namespace [XMLNS [p.297]]. The identification of this namespace is an arbitrary URI. XHTML requires that when a module is implemented using an XML DTD, the module declares the namespace in a special manner. The purpose of this is to permit the selection, at document parse/validation time, of the use of namespace prefixes and of the *prefix* that is used to identify elements and attributes from the module.

Content developers who wish to develop documents based upon hybrid document types may choose to use XML namespace prefixes on elements from the XHTML namespace, on elements from other namespaces, or on both. In order to ensure that such documents are XHTML conforming and backward compatible with non-namespace aware tools, the W3C recommends that content developers *do not* use XML namespace prefixes on elements from the XHTML namespace. When content developers are interested in having their content processed by namespace-aware processors, the W3C further recommends that elements in non-XHTML namespaces be specified using an XML namespace prefix rather than relying upon XML namespace defaulting mechanisms.

Each XHTML-conforming module implemented as an XML DTD is required to define a default XML namespace prefix, a method for changing this prefix within a document instance, and a marked section that turns on the processing of the prefix.

Note that it is legal and expected for multiple modules to be part of the same namespace when they are related. All of the XHTML modules, for example, are part of the same namespace.

A.1.1. Qualified Names sub-module

First, you need to define a qualified names sub-module (a sub-module is just a *file entity* that is separated so that it can be incorporated into the ultimate DTD at the appropriate point). The qualified names sub-module is built using the following steps (where the string MODULE is replaced with an appropriate string for the new module):

1. Define a parameter entity MODULE.prefixed that announces whether the elements in the module are being used with XML namespace prefixed names or not. This parameter entity's default value should be "%NS.prefixed;". The NS.prefixed parameter entity is defined by the XHTML framework to be IGNORE by default, and can be used in a document instance to switch on prefixing for all included namespaces (including that of the XHTML modules).
2. Define a parameter entity MODULE.xmlns that contains the namespace identifier for this

module.

3. Define a parameter entity MODULE.prefix that contains the default prefix string to use when prefixing is enabled.
4. Define a parameter entity MODULE.pfx that is "%MODULE.prefix;:" when prefixing is enabled, and "" when it is not.
5. Define a parameter entity MODULE.xmlns.extra.attrib that contains the declaration of any XML namespace attributes for namespaces referenced by this module (e.g., xmlns:xlink). When %MODULE.prefixed is set to INCLUDE, this attribute should include the xmlns:%MODULE.prefix; declaration as well.
6. Define a parameter entity XHTML.xmlns.extra.attrib as MODULE.xmlns.extra.attrib. This is usually overridden by the document type's driver file, but if not this definition will take over as the default.
7. For each of the elements defined by the module, create a parameter entity of the form "MODULE.NAME.qname" to hold its qualified name. The value for this parameter entity must be "%MODULE.pfx;NAME". In this way, the parsed value will be "PREFIX:NAME" when prefixes are enabled, and "NAME" otherwise.

   If the module adds attributes to elements defined in modules that do not share the namespace of this module, declare those attributes so that they use the %MODULE.pfx prefix. For example:

```
<ENTITY % MODULE.img.myattr.qname "%MODULE.pfx;myattr" >
```

An example of a qname sub-module for a hypothetical Inventory Module is included below:

```
<!-- ....................................................................... -->
<!-- Inventory Qname Module ................................................. -->
<!-- file: inventory-qname-1.mod
         PUBLIC "-//MY COMPANY//ELEMENTS XHTML Inventory Qnames 1.0//EN"
         SYSTEM "http://www.example.com/DTDs/inventory-qname-1.mod"
         xmlns:inventory="http://www.example.com/xmlns/inventory"
     ....................................................................... -->
<!-- Declare the default value for prefixing of this module's elements -->
<!-- Note that the NS.prefixed will get overridden by the XHTML Framework or
     by a document instance. -->
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % Inventory.prefixed "%NS.prefixed;" >
<!-- Declare the actual namespace of this module -->
<!ENTITY % Inventory.xmlns "http://www.example.com/xmlns/inventory" >
<!-- Declare the default prefix for this module -->
<!ENTITY % Inventory.prefix "inventory" >
<!-- Declare the prefix for this module -->
<![%Inventory.prefixed;[
<!ENTITY % Inventory.pfx "%Inventory.prefix;:" >
]]>
<!ENTITY % Inventory.pfx "" >
<!-- Declare the xml namespace attribute for this module -->
<![%Inventory.prefixed;[
<!ENTITY % Inventory.xmlns.extra.attrib
        "xmlns:%Inventory.prefix;       %URI.datatype;  #FIXED  '%Inventory.xmlns;'" >
]]>
<!ENTITY % Inventory.xmlns.extra.attrib "" >
<!-- Declare the extra namespace that should be included in the XHTML
```

```
     elements -->
<!ENTITY % XHTML.xmlns.extra.attrib
        "%Inventory.xmlns.extra.attrib;" >
<!-- Now declare the qualified names for all of the elements in the
     module -->
<!ENTITY % Inventory.shelf.qname "%Inventory.pfx;shelf" >
<!ENTITY % Inventory.item.qname "%Inventory.pfx;item" >
<!ENTITY % Inventory.desc.qname "%Inventory.pfx;desc" >
<!ENTITY % Inventory.sku.qname "%Inventory.pfx;sku" >
<!ENTITY % Inventory.price.qname "%Inventory.pfx;price" >
```

A.1.2. Declaration sub-module(s)

Next, you need to define one or more "declaration sub-modules". The purpose of these *file entities* is to declare the XML DTD elements and attribute lists. An XHTML declaration module should be constructed using the following process:

1. Define a parameter entity to use within the ATTLIST of each declared element. This parameter entity should contain %NS.decl.attrib; when %MODULE.prefixed; is set to INCLUDE, and %NS.decl.attrib; plus "xmlns %URI.datatype; #FIXED '%MODULE.xmlns;'" when %MODULE.prefixed; is set to IGNORE.
2. Declare all of the elements and attributes for the module. Within each ATTLIST for an element, include the parameter entity defined above so that all of the required xmlns attributes are available on each element in the module.

3. If the module adds attributes to elements defined in modules that do not share the namespace of this module, declare those attributes so that they use the %MODULE.pfx prefix. For example:

   ```
   <ENTITY % MODULE.img.myattr.qname "%MODULE.pfx;myattr" >
   <!ATTLIST %img.qname;
         %MODULE.img.myattr.qname;    CDATA          #IMPLIED
   >
   ```

   This would add an attribute to the `img` element of the Image Module, but the attribute's name will be the qualified name, including prefix, when prefixes are selected for a document instance. It also adds the xmlns:MODULE_PREFIX attribute to the img element's attribute list so that an XML Namespace-aware parser would know how to resolve the namespace based upon its prefix.

The following example shows a declaration sub-module for a hypothetical Inventory module.

```
<!-- ....................................................................... -->
<!-- Inventory Elements Module ............................................. -->
<!-- file: inventory-1.mod
        PUBLIC "-//MY COMPANY//ELEMENTS XHTML Inventory Elements 1.0//EN"
        SYSTEM "http://www.example.com/DTDs/inventory-1.mod"
        xmlns:inventory="http://www.example.com/xmlns/inventory"
     ....................................................................... -->
<!-- Inventory Module
     shelf
        item
```

```
            sku
            desc
            price
     This module defines a simple inventory item structure
-->
<!-- Define the global namespace attributes -->
<![%Inventory.prefixed;[
<!ENTITY % Inventory.xmlns.attrib
    "%NS.decl.attrib;"
>
]]>
<!ENTITY % Inventory.xmlns.attrib
         "xmlns %URI.datatype;  #FIXED '%Inventory.xmlns;'"
>
<!-- Define a common set of attributes for all module elements -->
<!ENTITY % Inventory.Common.attrib
         "%Inventory.xmlns.attrib;
          id              ID                  #IMPLIED"
>
<!-- Define the elements and attributes of the module -->
<!ELEMENT %Inventory.shelf.qname;
     ( %Inventory.item.qname; )* >
<!ATTLIST %Inventory.shelf.qname;
     location   CDATA   #IMPLIED
     %Inventory.Common.attrib;
>
<!ELEMENT %Inventory.item.qname;
     ( %Inventory.desc.qname;, %Inventory.sku.qname;, %Inventory.price.qname;) >
<!ATTLIST %Inventory.item.qname;
     location   CDATA   #IMPLIED
     %Inventory.Common.attrib;
>
<!ELEMENT %Inventory.desc.qname; ( #PCDATA ) >
<!ATTLIST %Inventory.desc.qname;
     %Inventory.Common.attrib;
>
<!ELEMENT %Inventory.sku.qname; ( #PCDATA ) >
<!ATTLIST %Inventory.sku.qname;
     %Inventory.Common.attrib;
>
<!ELEMENT %Inventory.price.qname; ( #PCDATA ) >
<!ATTLIST %Inventory.price.qname;
     %Inventory.Common.attrib;
>
<!-- end of inventory-1.mod -->
```

A.1.3. Using the module as a stand-alone DTD

It is sometimes desirable to have an XHTML module also usable as a stand alone DTD. A good example of this is our Inventory module above. These items need to be embeddable in an XHTML document, and also need to be available as free-standing documents extracted from a database (for example). The easiest way to accomplish this is to define a DTD file that instantiates the components of your module. Such a DTD would have this structure:

1. Include the XHTML Datatypes Module (your qnames module likely uses some of these datatypes - it certainly uses the URI datatype for the xmlns attribute).
2. Include the Qnames Module for your module.
3. Define the parameter entity NS.decl.attrib to be %MODULE.xmlns.extra.attrib;.
4. Include the Declaration Module(s) for your module.

An example of this for our Inventory module is included below:

```
<!-- .................................................................... -->
<!-- Inventory Elements DTD ..............................................-->
<!-- file: inventory-1.dtd
         PUBLIC "-//MY COMPANY//DTD XHTML Inventory 1.0//EN"
         SYSTEM "http://www.example.com/DTDs/inventory-1.dtd"
         xmlns:inventory="http://www.example.com/xmlns/inventory"
     .................................................................... -->
<!-- Inventory Module
     shelf
        item
           sku
           desc
           price
     This module defines a simple inventory item structure
-->
<!-- Bring in the datatypes -->
<!ENTITY % xhtml-datatypes.mod
         PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
                "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-datatypes-1.mod" >
%xhtml-datatypes.mod;
<!-- Bring in the qualified names -->
<!ENTITY % Inventory-qname.mod SYSTEM "inventory-qname-1.mod" >
%Inventory-qname.mod;
<!ENTITY % NS.decl.attrib "%Inventory.xmlns.extra.attrib;">
<!ENTITY % Inventory.mod SYSTEM "inventory-1.mod" >
%Inventory.mod;
<!-- end of inventory-1.dtd -->
```

This DTD can then be referenced by documents that use only the elements from your module:

```
<!DOCTYPE shelf SYSTEM "inventory-1.dtd">
<shelf xmlns="http://www.example.com/xmlns/inventory">
    <item>
        <desc>
          this is a description.
        </desc>
        <sku>
          this is the price.
        </sku>
        <price>
          this is the price.
        </price>
    </item>
</shelf>
```

This method permits the definition of elements and attributes that are scoped within their own namespace. It also permits content developers to use the default prefix for the elements and attributes:

```
<!DOCTYPE inventory:shelf SYSTEM "inventory-1.dtd" [
        <!ENTITY % Inventory.prefixed "INCLUDE">
]>
<inventory:shelf xmlns:inventory="http://www.example.com/xmlns/inventory">
    <inventory:item>
        <inventory:desc>
          this is a description.
        </inventory:desc>
        <inventory:sku>
          this is the sku.
        </inventory:sku>
        <inventory:price>
          this is the price.
        </inventory:price>
    </inventory:item>
</inventory:shelf>
```

Finally, a document instance can use a different XML namespace prefix by redeclaring it in the DOCTYPE header and its internal subset:

```
<!DOCTYPE i:shelf SYSTEM "inventory-1.dtd" [
        <!ENTITY % Inventory.prefixed "INCLUDE">
        <!ENTITY % Inventory.prefix "i">
]>
<i:shelf xmlns:i="http://www.example.com/xmlns/inventory">
    <i:item>
        <i:desc>
          this is a description.
        </i:desc>
        <i:sku>
          this is the price.
        </i:sku>
        <i:price>
          this is the price.
        </i:price>
    </i:item>
</i:shelf>
```

A.1.4. Namespace Idiosyncrasies

While the approach defined here permits the definition of markup languages that are XML and XML namespaces conforming, some behaviors defined by the XML namespaces specification are not supported:

1. XML namespaces permit the redeclaration of the xmlns attribute for a namespace at any point in the tree. It further permits this redeclaration to switch between namespace defaulting and prefixed usage, and permits the changing of the prefix. The method defined in this document does not permit this. Throughout a document instance a given namespace must continue to use the same namespace prefix (when prefixing is used), or must continue

to be used in the default scope.

2. When using XML namespace defaulting, it is legal to rely upon the DTD of the document to inform parsers of the namespace of elements. However, since namespace aware processors are not required to include the DTD when evaluating a document, content developers should declare the XML namespace of an element whenever the namespace changes:

```
...
<p>
   <myelement xmlns="..." />
</p>
```

B. Developing XHTML Relax NG with defined and extended modules

This section is **informative**.

The primary purpose of defining XHTML modules and a general modularization methodology is to ease the development of document types that are based upon XHTML. These document types may extend XHTML by integrating additional capabilities (e.g., [SMIL] [p.??]), or they may define a subset of XHTML for use in a specialized device. This section describes the techniques that document type designers must use in order to take advantage of the XML DTD implementation of this modularization architecture. It does this by applying the XHTML Modularization techniques in progressively more complex ways, culminating in the creation of a complete document type from disparate modules.

Note that in no case do these examples require the modification of the XHTML-provided module *file entities* themselves. The XHTML module file entities are completely parameterized, so that it is possible through separate module definitions and *driver files* to customize the definition and the content model of each element and each element's hierarchy.

Finally, remember that most users of XHTML are *not* expected to be DTD authors. DTD authors are generally people who are defining specialized markup that will improve the readability, simplify the rendering of a document, or ease machine-processing of documents, or they are client designers that need to define the specialized DTD for their specific client. Consider these cases:

- An organization is providing subscriber's information via a Web interface. The organization stores its subscriber information in an XML-based database. One way to report that information out from the database to the Web is to embed the XML records from the database directly in the XHTML document. While it is possible to merely embed the records, the organization could define a DTD module that describes the records, attach that module to an XHTML DTD, and thereby create a complete DTD for the pages. The organization can then access the data within the new elements via the Document Object Model [DOM] [p.??] , validate the documents, provide style definitions for the elements that cascade using Cascading Style Sheets [CSS2] [p.??] , etc. By taking the time to define the structure of their data and create a DTD using the processes defined in this section, the organization can realize the full benefits of XML.

- An Internet client developer is designing a specialized device. That device will only support a subset of XHTML, and the devices will always access the Internet via a proxy server that validates content before passing it on to the client (to minimize error handling on the client). In order to ensure that the content is valid, the developer creates a DTD that is a subset of XHTML using the processes defined in this section. They then use the new DTD in their proxy server and in their devices, and also make the DTD available to content developers so that developers can validate their content before making it available. By performing a few simple steps, the client developer can use the architecture defined in this document to greatly ease their DTD development cost *and* ensure that they are fully supporting the

subset of XHTML that they choose to include.

B.1. Defining additional attributes

In some cases, an extension to XHTML can be as simple as additional attributes. Attributes can be added to an element just by specifying an additional ATTLIST for the element, for example:

Example

```
<!ATTLIST %a.qname;
          %MyModule.pfx;myattr   CDATA        #IMPLIED
          %MyModule.xmlns.extras.attrib;
>
```

would add the "myattr" attribute, with an optional prefix defined by "%MyModule.pfx", with a value type of CDATA, to the "a" element. This works because XML permits the definition or extension of the attribute list for an element at any point in a DTD. *For a discussion of qualified names and namespace prefixes, see Defining the Namespace of a Module [p.142] .*

Naturally, adding an attribute to a DTD does not mean that any new behavior is defined for arbitrary clients. However, a content developer could use an extra attribute to store information that is accessed by associated scripts via the Document Object Model (for example).

B.2. Defining additional elements

Defining additional elements is only slightly more complicated than defining additional attributes. Basically, DTD authors should write the element declaration for each element:

Example

```
<!-- In the qname sub-module -->
<!ENTITY % MyModule.myelement.qname  "%MyModule.pfx;myelement" >
<!ENTITY % MyModule.myotherelement.qname  "%MyModule.pfx;myotherelement" >
<!-- In the declaration sub-module -->
<!ELEMENT %MyModule.myelement.qname;
           ( #PCDATA | %MyModule.myotherelement.qname; )* >
<!ATTLIST %MyModule.myelement.qname;
          myattribute    CDATA    #IMPLIED
>
<!ELEMENT %MyModule.myotherelement.qname; EMPTY >
```

After the elements are defined, they need to be integrated into the content model. Strategies for integrating new elements or sets of elements into the content model are addressed in the next section.

B.3. Defining the content model for a collection of modules

Since the content model of XHTML modules is fully parameterized, DTD authors may modify the content model for every element in every module. The details of the DTD module interface are defined in Building DTD Modules [p.141] . Basically there are two ways to approach this modification:

1. Re-define the ".content" parameter entity for each element.
2. Re-define one or more of the global content model entities (normally via the ".extras" parameter entity).

The strategy taken will depend upon the nature of the modules being combined and the nature of the elements being integrated. The remainder of this section describes techniques for integrating two different classes of modules.

B.3.1. Integrating a stand-alone module into XHTML

When a module (and remember, a module can be a collection of other modules) contains elements that only reference each other in their content model, it is said to be "internally complete". As such, the module can be used on its own; (for example, you could define a DTD that was just that module, and use one of its elements as the root element). Integrating such a module into XHTML is a three step process:

1. Decide what element(s) can be thought of as the root(s) of the new module.
2. Decide where these elements need to attach in the XHTML content tree.
3. Then, for each attachment point in the content tree, add the root element(s) to the content definition for the XHTML elements.

Consider attaching the elements defined above [p.150] . In that example, the element `myelement` is the root. To attach this element under the `img` element, and only the `img` element, of XHTML, the following would work:

Example

```
<!ENTITY % img.content "( %MyModule.myelement.qname; )*">
```

A DTD defined with this content model would allow a document like the following fragment:

Example

```
<img src="...">
<myml:myelement >This is content of a locally defined element</myml:myelement>
</img>
```

It is important to note that normally the `img` element has a content model of `EMPTY`. By adding myelement to that content model, we are really just replacing `EMPTY` with `myelement`. In the case of other elements that already have content models defined, the addition of an element would require the restating of the existing content model in addition to `myelement`.

B.3.2. Mixing a new module throughout the modules in XHTML

Extending the example above, to attach this module everywhere that the `%Flow.mix` content model group is permitted, would require something like the following:

Example

```
<!ENTITY % Misc.extra
     "| %MyModule.myelement.qname;" >
```

Since the %Misc.extra content model class is used in the %Misc.class parameter entity, and that parameter entity is used throughout the XHTML modules, the new module would become available throughout an extended XHTML document type.

B.4. Creating a new DTD

So far the examples in this section have described the methods of extending XHTML and XHTML's content model. Once this is done, the next step is to collect the modules that comprise the DTD into a single DTD driver, incorporating the new definitions so that they override and augment the basic XHTML definitions as appropriate.

B.4.1. Creating a simple DTD

Using the trivial example above, it is possible to define a new DTD that uses and extends the XHTML modules pretty easily. First, define the new elements and their content model in a module:

```
<!-- File: simpleml-model-1.mod -->
<!-- Declare a Parameter Entity (PE) that defines any external namespaces
     that are used by this module -->
<!-- Set the PE that is used in every ATTLIST in this module
     NS.prefixed.attrib is initialized in the xhtml-qname module, and
         SimpleML.ns.noprefix.attrib is initialized in the SimpleML DTD driver
         file.-->
<!ENTITY % SimpleML.xmlns.attrib
  "%NS.decl.attrib;"
>
<!ENTITY % SimpleML.Common.attrib
  "%SimpleML.xmlns.attrib;
   id           ID           #IMPLIED"
>
<!ENTITY % SimpleML.element.qname "%SimpleML.pfx;element" >
<!ENTITY % SimpleML.otherelement.qname "%SimpleML.pfx;otherelement" >
<!ELEMENT %SimpleML.element.qname;
          ( #PCDATA | %SimpleML.otherelement.qname; )* >
<!ATTLIST %SimpleML.element.qname;
          myattribute   CDATA  #IMPLIED
          %SimpleML.Common.attrib;
>
<!ELEMENT %SimpleML.otherelement.qname; EMPTY >
<!ATTLIST %SimpleML.otherelement.qname;
```

```
          %SimpleML.Common.attrib;
>
<!ENTITY % SimpleML.img.myattr.qname "%SimpleML.pfx;myattr" >
<!ATTLIST %img.qname;
          %SimpleML.img.myattr.qname;  CDATA  #IMPLIED
>
<!-- Add our elements to the XHTML content model -->
<!ENTITY % Misc.class
     "| %SimpleML.element.qname;" >
<!-- Now bring in the XHTML Basic content model -->
<!ENTITY % xhtml-basic-model.mod
     PUBLIC "-//W3C//ENTITIES XHTML Basic 1.0 Document Model 1.0//EN"
                "http://www.w3.org/TR/xhtml-basic/xhtml-basic10-model-1.mod" >
%xhtml-basic-model.mod;
```

Next, define the DTD driver for the new language:

```
<!-- file: simpleml-1_0.dtd -->
<!-- Bring in the XHTML datatypes -->
<!ENTITY % xhtml-datatypes.mod
     PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-datatypes-1.mod" >
%xhtml-datatypes.mod;
<!-- Declare the actual namespace of this module -->
<!ENTITY % SimpleML.xmlns "http://www.example.com/xmlns/simpleml1" >
<!-- By default, disable prefixing of new module -->
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % SimpleML.prefixed "%NS.prefixed;" >
<!-- Default prefix for module elements and attributes -->
<!ENTITY % SimpleML.prefix "simpleml" >
<!-- If this module's namespace is prefixed -->
<![%SimpleML.prefixed;[
  <!ENTITY % SimpleML.pfx  "%SimpleML.prefix;:" >
]]>
<!ENTITY % SimpleML.pfx  "" >
<![%SimpleML.prefixed;[
  <!ENTITY % SimpleML.xmlns.extra.attrib
       "xmlns:%SimpleML.prefix; %URI.datatype; #FIXED '%SimpleML.xmlns;'" >
]]>
<!ENTITY % SimpleML.xmlns.extra.attrib "" >
<!ENTITY % XHTML.xmlns.extra.attrib
       "%SimpleML.xmlns.extra.attrib;"
>
<!-- Set the content model for our language -->
<!ENTITY % xhtml-model.mod
     SYSTEM "simpleml-model-1.mod" >
<!-- Instantiate xhtml basic's DTD to do all the work -->
<!ENTITY % xhtml-basic.dtd
     PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
            "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd" >
%xhtml-basic.dtd;
```

When using this DTD, it is possible to enable the use of XML namespace prefixes. When so doing, the start of a document using this new DTD might look like:

```
<!DOCTYPE html SYSTEM "simpleml-1_0.dtd" [
  <!ENTITY % SimpleML.prefixed "INCLUDE">
]>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:simpleml="http://www.example.com/xmlns/simpleml1" >
<head>
<title>An example using defaults</title>
</head>
<body>
<p>This is content in the XHTML namespace</p>
<simpleml:element>
  This is content in the SimpleML namespace.
  <simpleml:otherelement />
</simpleml:element>
<p><img src="missing" alt="Missing image" simpleml:myattr="value"/></p>
</body>
</html>
```

B.4.2. Creating a DTD by extending XHTML

Next, there is the situation where a complete, additional, and complex module is added to XHTML (or to a subset of XHTML). In essence, this is the same as in the trivial example above, the only difference being that the module being added is incorporated in the DTD by reference rather than explicitly including the new definitions in the DTD.

One such complex module is the DTD for [MATHML] [p.??] . In order to combine MathML and XHTML into a single DTD, an author would just decide where MathML content should be legal in the document, and add the MathML root element to the content model at that point. First, define a content model module that instantiates the MathML DTD and connects it to the content model:

Example

```
<!-- File: mathml-model.mod -->
<!ENTITY % XHTML1-math
     PUBLIC "-//W3C//DTD MathML 2.0//EN"
            "http://www.w3.org/TR/MathML2/dtd/mathml2.dtd" >
%XHTML1-math;
<!ENTITY % Inlspecial.extra
     "%a.qname; | %img.qname; | %object.qname; | %map.qname;
          | %Mathml.Math.qname;" >
```

Next, define a DTD driver that identifies our new content model module as the content model for the DTD, and hands off processing to the XHTML 1.1 driver (for example):

Example

```
<!-- File: xhtml-mathml.dtd -->
<!ENTITY % xhtml-model.mod
      SYSTEM "mathml-model.mod" >
<!ENTITY % xhtml11.dtd
     PUBLIC "-//W3C//DTD XHTML 1.1//EN"
            "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd" >
%xhtml11.dtd;
```

B.4.3. Creating a DTD by removing and replacing XHTML modules

Another way in which DTD authors may use XHTML modules is to define a DTD that is a subset of an XHTML family document type (because, for example, they are building devices or software that only supports a subset of XHTML). Doing this is only slightly more complex than the previous example. The basic steps to follow are:

1. Take an XHTML family DTD as the basis of the new document type (we will use XHTML 1.1).
2. Select the modules to remove from that DTD.
3. Define a new DTD that "IGNOREs" the modules.
4. Introduce some new modules.

For example, consider a device that uses XHTML modules, but without forms or tables. The DTD for such a device would look like this:

Example

```
<!-- File: xhtml-simple.dtd -->
<!ENTITY % xhtml-form.module "IGNORE" >
<!ENTITY % xhtml-table.module "IGNORE" >
<!ENTITY % xhtml-table.module "IGNORE" >
<!-- Bring in the basic tables module -->
<!ENTITY % xhtml-basic-table.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Basic Tables 1.0//EN"
           "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-basic-table-1.mod"
>
%xhtml-basic-table.mod;
<!ENTITY % xhtml11.mod
     PUBLIC "-//W3C//DTD XHTML 1.1//EN"
            "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd" >
%xhtml11.mod;
```

Note that this does not actually modify the content model for the XHTML 1.1 DTD. However, since XML ignores elements in content models that are not defined, the form and table elements are dropped from the model automatically.

B.4.4. Creating a new DTD

Finally, some DTD authors may wish to start from scratch, using the XHTML Modularization framework as a toolkit for building a new markup language. This language must be made up of the minimal, required modules from XHTML. It may also contain other XHTML-defined modules or any other module that the author wishes to employ. In this example, we will take the XHTML required modules, add some XHTML-defined modules, and also add in the module we defined above.

The first step is to use the XHTML-provided template for a new qualified names module, modified to define the qualified names and namespace for our new elements.

```
<!-- file: myml-qname-1.mod -->
<!-- Bring in the datatypes - we use the URI.datatype PE for declaring the
     xmlns attributes. -->
<!ENTITY % MyML-datatypes.mod
         PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
                "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-datatypes-1.mod" >
%MyML-datatypes.mod;
<!-- By default, disable prefixing of this module -->
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % MyML.prefixed "%NS.prefixed;" >
<!-- Declare the actual namespace of this module -->
<!ENTITY % MyML.xmlns "http://www.example.com/xmlns/myml" >
<!-- Declare the default prefix for this module -->
<!ENTITY % MyML.prefix "myml" >
<!-- If this module's namespace is prefixed -->
<![%MyML.prefixed;[
  <!ENTITY % MyML.pfx  "%MyML.prefix;:" >
]]>
<!ENTITY % MyML.pfx  "" >
<!-- Declare a Parameter Entity (PE) that defines any external namespaces
     that are used by this module -->
<!ENTITY % MyML.xmlns.extra.attrib "" >
<!-- Declare a PE that defines the xmlns attributes for use by MyML. -->
<![%MyML.prefixed;[
<!ENTITY % MyML.xmlns.attrib
   "xmlns:%MyML.prefix;  %URI.datatype;  #FIXED '%MyML.xmlns;'
   %MyML.xmlns.extra.attrib;"
>
]]>
<!ENTITY % MyML.xmlns.attrib
   "xmlns       %URI.datatype;  #FIXED '%MyML.xmlns;'
        %MyML.xmlns.extra.attrib;"
>
<!-- Make sure that the MyML namespace attributes are included on the XHTML
     attribute set -->
<![%NS.prefixed;[
<!ENTITY % XHTML.xmlns.extra.attrib
        "%MyML.xmlns.attrib;" >
]]>
<!ENTITY % XHTML.xmlns.extra.attrib
        ""
>
<!-- Now declare the element names -->
<!ENTITY % MyML.myelement.qname "%MyML.pfx;myelement" >
<!ENTITY % MyML.myotherelement.qname "%MyML.pfx;myotherelement" >
```

Next, define a module that defines the elements and attributes using the XHTML provided template.

```
<!-- ....................................................................... -->
<!-- My Elements Module ..................................................... -->
<!-- file: myml-elements-1_0.mod
         PUBLIC "-//MY COMPANY//ELEMENTS XHTML MyML Elements 1.0//EN"
         SYSTEM "http://example.com/DTDs/myml-elements-1_0.mod"
         xmlns:myml="http://example.com/DTDs/myml-1_0.dtd"
     ....................................................................... -->
<!-- My Elements Module
     myelement
     myotherelement
```

```
     This module has no purpose other than to provide structure for some
         PCDATA content.
-->
<!ELEMENT %MyML.myelement.qname;
     ( #PCDATA | %MyML.myotherelement.qname; )* >
<!ATTLIST %MyML.myelement.qname;
     myattribute        CDATA   #IMPLIED
     %MyML.xmlns.attrib;
>
<!ELEMENT %MyML.myotherelement.qname; EMPTY >
<!ATTLIST %MyML.myotherelement.qname;
      %MyML.xmlns.attrib;
>
<!ENTITY % MyML.img.myattr.qname "%MyML.pfx;myattr" >
<!ATTLIST %img.qname;
      %MyML.img.myattr.qname;   CDATA  #IMPLIED
      %MyML.xmlns.attrib;
>
<!-- end of myml-elements-1_0.mod -->
```

Now, build a content model description that hooks the new elements and attributes into the other XHTML elements. The following example is patterned after the XHTML Basic content model, but is a complete, free-standing content model module:

```
<!-- ....................................................................... -->
<!-- MyML Model Module  ..................................................... -->
<!-- file: myml-model-1.mod
         PUBLIC "-//MY COMPANY//ELEMENTS XHTML MyML Model 1.0//EN"
         SYSTEM "http://example.com/DTDs/myml-model-1_0.mod"
         xmlns:myml="http://www.example.com/xmlns/myml"
     ....................................................................... -->
<!-- Define the content model for Misc.extra -->
<!ENTITY % Misc.class
     "| %MyML.myelement.qname; ">
<!-- ...................  Inline Elements  ..................... -->
<!ENTITY % HeadOpts.mix
     "( %meta.qname; )*" >
<!ENTITY % I18n.class "" >
<!ENTITY % InlStruct.class "%br.qname; | %span.qname;" >
<!ENTITY % InlPhras.class
     "| %em.qname; | %strong.qname; | %dfn.qname; | %code.qname;
      | %samp.qname; | %kbd.qname; | %var.qname; | %cite.qname;
      | %abbr.qname; | %acronym.qname; | %q.qname;" >
<!ENTITY % InlPres.class
     "" >
<!ENTITY % Anchor.class "| %a.qname;" >
<!ENTITY % InlSpecial.class "| %img.qname; " >
<!ENTITY % Inline.extra "" >
<!-- %Inline.class; includes all inline elements,
     used as a component in mixes
-->
<!ENTITY % Inline.class
     "%InlStruct.class;
      %InlPhras.class;
      %InlPres.class;
      %Anchor.class;
```

```
      %InlSpecial.class;"
>
<!-- %InlNoAnchor.class; includes all non-anchor inlines,
     used as a component in mixes
-->
<!ENTITY % InlNoAnchor.class
     "%InlStruct.class;
      %InlPhras.class;
      %InlPres.class;
      %InlSpecial.class;"
>
<!-- %InlNoAnchor.mix; includes all non-anchor inlines
-->
<!ENTITY % InlNoAnchor.mix
     "%InlNoAnchor.class;
      %Misc.class;"
>
<!-- %Inline.mix; includes all inline elements, including %Misc.class;
-->
<!ENTITY % Inline.mix
     "%Inline.class;
      %Misc.class;"
>
<!-- .....................  Block Elements  ..................... -->
<!ENTITY % Heading.class
     "%h1.qname; | %h2.qname; | %h3.qname;
      | %h4.qname; | %h5.qname; | %h6.qname;" >
<!ENTITY % List.class "%ul.qname; | %ol.qname; | %dl.qname;" >
<!ENTITY % BlkStruct.class "%p.qname; | %div.qname;" >
<!ENTITY % BlkPhras.class
     "| %pre.qname; | %blockquote.qname; | %address.qname;" >
<!ENTITY % BlkPres.class "" >
<!ENTITY % Block.extra "" >
<!-- %Block.class; includes all block elements,
     used as an component in mixes
-->
<!ENTITY % Block.class
     "%BlkStruct.class;
      %BlkPhras.class;
      %BlkPres.class;
      %Block.extra;"
>
<!-- %Block.mix; includes all block elements plus %Misc.class;
-->
<!ENTITY % Block.mix
     "%Heading.class;
      | %List.class;
      | %Block.class;
      %Misc.class;"
>
<!-- .................  All Content Elements  .................. -->
<!-- %Flow.mix; includes all text content, block and inline
-->
<!ENTITY % Flow.mix
     "%Heading.class;
      | %List.class;
      | %Block.class;
```

```
      | %Inline.class;
      %Misc.class;"
>
<!-- special content model for pre element -->
<!ENTITY % pre.content
       "( #PCDATA
        | %Inline.class; )*"
>
<!-- end of myml-model-1.mod -->
```

Finally, use the XHTML-provided template for a new DTD, modified as appropriate for our new markup language:

```
<!-- .................................................................... -->
<!-- MYML DTD  ........................................................... -->
<!-- file: myml-1_0.dtd -->
<!-- This is the DTD driver for myml 1.0.
     Please use this formal public identifier to identify it:
         "-//MY COMPANY//DTD XHTML MYML 1.0//EN"
     And this namespace for myml-unique elements:
         xmlns:myml="http://www.example.com/xmlns/myml"
-->
<!ENTITY % XHTML.version  "-//MY COMPANY//DTD XHTML MYML 1.0//EN" >
<!-- reserved for use with document profiles -->
<!ENTITY % XHTML.profile  "" >
<!-- Tell the framework to use our qualified names module as an extra qname
driver -->
<!ENTITY % xhtml-qname-extra.mod
     SYSTEM "myml-qname-1.mod" >
<!-- Define the Content Model for the framework to use -->
<!ENTITY % xhtml-model.mod
         SYSTEM "myml-model-1.mod" >
<!-- Disable bidirectional text support -->
<!ENTITY % XHTML.bidi  "IGNORE" >
<!-- Bring in the XHTML Framework -->
<!ENTITY % xhtml-framework.mod
     PUBLIC "-//W3C//ENTITIES XHTML Modular Framework 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-framework-1.mod" >
%xhtml-framework.mod;
<!-- Basic Text Module (Required)  ............................... -->
<!ENTITY % xhtml-text.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Basic Text 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-text-1.mod" >
%xhtml-text.mod;
<!-- Hypertext Module (required) ................................. -->
<!ENTITY % xhtml-hypertext.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Hypertext 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-hypertext-1.mod" >
%xhtml-hypertext.mod;
<!-- Lists Module (required)  .................................... -->
<!ENTITY % xhtml-list.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Lists 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-list-1.mod" >
%xhtml-list.mod;
<!-- My Elements Module   ....................................... -->
<!ENTITY % MyML-elements.mod
```

```
     SYSTEM "myml-elements-1.mod" >
%MyML-elements.mod;
<!-- XHTML Images module  ..................................... -->
<!ENTITY % xhtml-image.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Images 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-image-1.mod" >
%xhtml-image.mod;
<!-- Document Metainformation Module  ......................... -->
<!ENTITY % xhtml-meta.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Metainformation 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-meta-1.mod" >
%xhtml-meta.mod;
<!-- Document Structure Module (required)  ..................... -->
<!ENTITY % xhtml-struct.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Document Structure 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-struct-1.mod" >
%xhtml-struct.mod;
```

B.5. Using the new DTD

Once a new DTD has been developed, it can be used in any document. Using the DTD is as simple as just referencing it in the DOCTYPE declaration of a document:

```
<!DOCTYPE html SYSTEM "myml-1_0.dtd" >
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>An example using defaults</title>
</head>
<body>
<p>This is content in the XHTML namespace</p>
<myelement>
  This is content in the SimpleML namespace.
  <myotherelement />
</myelement>
<p><img src="missing" alt="Missing image" myattr="value"/></p>
</body>
</html>
```

The document can also use the elements outside of the XHTML namespace by prefixing them:

```
<!DOCTYPE html SYSTEM "myml-1_0.dtd" [
  <!ENTITY % MyML.prefixed "INCLUDE" >
]>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>An example using defaults</title>
</head>
<body>
<p>This is content in the XHTML namespace</p>
<myml:myelement>
  This is content in the myml namespace.
  <myml:myotherelement />
```

```
</myml:myelement>
<p><img src="missing" alt="Missing image" myml:myattr="value" /></p>
</body>
</html>
```

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 Attributes Module</x:h2>
    <include href="xhtml-core-attrib-2.rng"/>
  </div>
  <div>
    <x:h2>Internationalization Attribute Module</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 Attributes Module</x:h2>
    <include href="xhtml-edit-attrib-2.rng"/>
  </div>
  <div>
    <x:h2>Embedding Attributes Module</x:h2>
    <include href="xhtml-embed-attrib-2.rng"/>
  </div>
  <!--
  <div>
    <x:h2>XForms Repeat Attribute Collection</x:h2>
    <include href="xforms-repeat-attrib.rng"/>
    <define name="Common.attrib" combine="interleave">
      <ref name="XFORMS.Repeat.attrib"/>
    </define>
  </div>
  -->
  <div>
    <x:h2>Hypertext Attributes Module</x:h2>
    <include href="xhtml-hypertext-attrib-2.rng"/>
  </div>
  <div>
    <x:h2>Image Map Attributes Module</x:h2>
```

```
    <include href="xhtml-imagemap-attrib-2.rng"/>
  </div>
  <div>
    <x:h2>Media Attribute Module</x:h2>
    <include href="xhtml-media-attrib-2.rng"/>
  </div>
  <div>
    <x:h2>Metainformation Attributes Module</x:h2>
    <include href="xhtml-meta-attrib-2.rng"/>
  </div>
  <div>
    <x:h2>Role Attribute Module</x:h2>
    <include href="xhtml-role-attrib-2.rng"/>
  </div>
  <div>
    <x:h2>Style Attribute Module</x:h2>
    <include href="xhtml-inlstyle-2.rng"/>
  </div>
  <define name="Common.extra.attrib">
    <empty/>
  </define>
  <define name="Common.attrib">
    <ref name="Common.extra.attrib"/>
  </define>
  <define name="CommonNoEvents.attrib" combine="interleave">
    <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"/>
    </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="List.class"/-->
      <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"/>
      <optional>
        <attribute name="full">
          <ref name="URI.datatype"/>
        </attribute>
      </optional>
    </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 Structural or Text 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="caption"/>
        </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 nl element</x:h2>
  <define name="nl">
    <element name="nl">
      <ref name="nl.attlist"/>
      <ref name="caption"/>
      <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="caption"/>
      </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="caption"/>
      </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="dl"/>
        <ref name="nl"/>
        <ref name="ol"/>
        <ref name="ul"/>
      </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:a="http://relaxng.org/ns/compatibility/annotations/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>
        <choice>
          <attribute name="xml:id">
            <ref name="ID.datatype"/>
          </attribute>
```

```
            <attribute name="id">
              <ref name="ID.datatype"/>
            </attribute>
          </choice>
        </optional>
      </define>
      <define name="layout.attrib">
        <optional>
          <attribute name="layout" a:defaultValue="irrelevant">
            <choice>
              <value>irrelevant</value>
              <value>relevant</value>
            </choice>
          </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="layout.attrib"/>
        <ref name="title.attrib"/>
      </define>
    </div>
    <define name="Common.attrib" combine="interleave">
      <ref name="Core.attrib"/>
    </define>
    <define name="CommonNoEvents.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="cite.attrib">
      <optional>
        <attribute name="cite">
          <ref name="URI.datatype"/>
        </attribute>
      </optional>
    </define>
    <define name="href.attrib">
      <optional>
        <attribute name="href">
```

```
      <ref name="URI.datatype"/>
    </attribute>
  </optional>
</define>
<define name="hreflang.attrib">
  <optional>
    <attribute name="hreflang">
      <ref name="LanguageCodes.datatype"/>
    </attribute>
  </optional>
</define>
<define name="hrefmedia.attrib">
  <optional>
    <attribute name="hrefmedia">
      <ref name="MediaDesc.datatype"/>
    </attribute>
  </optional>
</define>
<define name="hreftype.attrib">
  <optional>
    <attribute name="hreftype">
      <ref name="ContentTypes.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="target.attrib">
  <optional>
    <attribute name="target">
      <ref name="HrefTarget.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="cite.attrib"/>
  <ref name="href.attrib"/>
  <ref name="hreflang.attrib"/>
```

```
      <ref name="hrefmedia.attrib"/>
      <ref name="hreftype.attrib"/>
      <ref name="nextfocus.attrib"/>
      <ref name="prevfocus.attrib"/>
      <ref name="target.attrib"/>
      <ref name="base.attrib"/>
    </define>
  </div>
  <define name="Common.attrib" combine="interleave">
    <ref name="Hypertext.attrib"/>
  </define>
  <define name="CommonNoEvents.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="CommonNoEvents.attrib" combine="interleave">
    <ref name="I18n.attrib"/>
  </define>
</grammar>
```

C.1.10. Access

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml"
         xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">
  <x:h1>Access Module</x:h1>
  <div>
    <x:h2>The access element</x:h2>
    <define name="access">
      <element name="access">
        <ref name="access.attlist"/>
      </element>
```

```
    </define>
    <define name="access.attlist">
      <ref name="Common.attrib"/>
      <optional>
        <attribute name="activate" a:defaultValue="false">
          <choice>
            <value>true</value>
            <value>false</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="key">
          <ref name="Characters.datatype"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="order" a:defaultValue="document">
          <choice>
            <value>document</value>
            <value>list</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="targetid">
          <ref name="IDREF.datatype"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="targetrole">
          <ref name="CURIEs.datatype"/>
        </attribute>
      </optional>
    </define>
  </div>
  <define name="head.misc" combine="choice">
    <ref name="access"/>
  </define>
</grammar>
```

C.1.11. Bi-directional Text Attribute

```
<?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>Bi-directional Text Attribute Module</x:h1>
  <div>
    <x:h2>Bi-directional Text Collection</x:h2>
    <define name="dir.attrib">
      <optional>
        <attribute name="dir" a:defaultValue="ltr">
          <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="CommonNoEvents.attrib" combine="interleave">
    <ref name="Bidi.attrib"/>
  </define>
</grammar>
```

C.1.12. 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="CommonNoEvents.attrib" combine="interleave">
    <ref name="Edit.attrib"/>
  </define>
</grammar>
```

C.1.13. 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="srcencoding.attrib">
      <optional>
        <attribute name="srcencoding">
          <ref name="Encodings.datatype"/>
        </attribute>
      </optional>
    </define>
    <define name="srctype.attrib">
      <optional>
        <attribute name="srctype">
          <ref name="ContentTypes.datatype"/>
        </attribute>
      </optional>
    </define>
    <define name="Embedding.attrib">
      <ref name="src.attrib"/>
      <ref name="srcencoding.attrib"/>
      <ref name="srctype.attrib"/>
    </define>
  </div>
  <define name="Common.attrib" combine="interleave">
    <ref name="Embedding.attrib"/>
  </define>
  <define name="CommonNoEvents.attrib" combine="interleave">
    <ref name="Embedding.attrib"/>
  </define>
</grammar>
```

C.1.14. Image

```
<?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 Module</x:h1>
  <div>
    <x:h2>The img element</x:h2>
```

```
    <define name="img">
      <element name="img">
        <ref name="img.attlist"/>
        <ref name="Text.model"/>
      </element>
    </define>
    <define name="img.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>
  <define name="Text.class" combine="choice">
    <ref name="img"/>
  </define>
</grammar>
```

C.1.15. 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">
          <ref name="Shape.datatype"/>
        </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="CommonNoEvents.attrib" combine="interleave">
    <ref name="Map.attrib"/>
  </define>
</grammar>
```

C.1.16. Media Attribute

```
<?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>Media Attribute Module</x:h1>
  <define name="media.attrib">
    <optional>
      <attribute name="media" a:defaultValue="all">
        <ref name="MediaDesc.datatype"/>
      </attribute>
    </optional>
  </define>
  <define name="Common.attrib" combine="interleave">
    <ref name="media.attrib"/>
  </define>
  <define name="CommonNoEvents.attrib" combine="interleave">
    <ref name="media.attrib"/>
  </define>
</grammar>
```

C.1.17. Metainformation Attributes

```
<?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>Metainformation Attributes Module</x:h1>
  <div>
    <x:h2>Metadata Attribute Collection</x:h2>
    <define name="about.attrib">
      <optional>
        <attribute name="about">
          <ref name="URIorSafeCURIE.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="CURIE.datatype"/>
      </attribute>
    </optional>
  </define>
  <define name="typeof.attrib">
    <optional>
      <attribute name="typeof">
        <ref name="CURIEs.datatype"/>
      </attribute>
    </optional>
  </define>
  <define name="property.attrib">
    <optional>
      <attribute name="property" a:defaultValue="reference">
        <ref name="CURIEs.datatype"/>
      </attribute>
    </optional>
  </define>
  <define name="rel.attrib">
    <optional>
      <attribute name="rel">
        <ref name="CURIEs.datatype"/>
      </attribute>
    </optional>
  </define>
  <define name="resource.attrib">
    <optional>
      <attribute name="resource">
        <ref name="URIorSafeCURIE.datatype"/>
      </attribute>
    </optional>
  </define>
  <define name="rev.attrib">
    <optional>
      <attribute name="rev">
        <ref name="CURIEs.datatype"/>
      </attribute>
    </optional>
  </define>
  <define name="Metadata.attrib">
    <ref name="about.attrib"/>
    <ref name="content.attrib"/>
    <ref name="datatype.attrib"/>
    <ref name="typeof.attrib"/>
    <ref name="property.attrib"/>
    <ref name="rel.attrib"/>
    <ref name="resource.attrib"/>
    <ref name="rev.attrib"/>
  </define>
</div>
<define name="Common.attrib" combine="interleave">
  <ref name="Metadata.attrib"/>
</define>
```

```
  <define name="CommonNoEvents.attrib" combine="interleave">
    <ref name="Metadata.attrib"/>
  </define>
</grammar>
```

C.1.18. 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="Misc.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"/>
        <ref name="Text.model"/>
      </element>
    </define>
    <define name="meta.attlist">
      <ref name="Common.attrib"/>
    </define>
  </div>
  <define name="head.misc" combine="choice">
    <ref name="meta"/>
  </define>
  <define name="Misc.class" combine="choice">
    <ref name="meta"/>
  </define>
</grammar>
```

C.1.19. 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>
  <x:p>Note. Also include the Caption Module when this module is used.</x:p>
  <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.1.20. Role Access

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:x="http://www.w3.org/1999/xhtml"
         xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">
  <x:h1>Access Module</x:h1>
  <div>
    <x:h2>The access element</x:h2>
    <define name="access">
      <element name="access">
        <ref name="access.attlist"/>
      </element>
    </define>
    <define name="access.attlist">
      <ref name="Common.attrib"/>
      <optional>
        <attribute name="activate" a:defaultValue="false">
          <choice>
            <value>true</value>
            <value>false</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="key">
          <ref name="Characters.datatype"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="order" a:defaultValue="document">
          <choice>
            <value>document</value>
            <value>list</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="targetid">
          <ref name="IDREF.datatype"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="targetrole">
          <ref name="CURIEs.datatype"/>
        </attribute>
      </optional>
    </define>
  </div>
```

```
  <define name="head.misc" combine="choice">
    <ref name="access"/>
  </define>
</grammar>
```

C.1.21. 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.attrib" combine="interleave">
    <ref name="style.attrib"/>
  </define>
  <define name="CommonNoEvents.attrib" combine="interleave">
    <ref name="style.attrib"/>
  </define>
</grammar>
```

C.1.22. 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>
    </define>
  </div>
  <define name="head.misc" combine="choice">
    <ref name="style"/>
  </define>
</grammar>
```

C.1.23. 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>CURIE Datatypes</x:h2>
    <define name="CURIE.datatype">
      <x:p>A single curie</x:p>
      <data type="string">
        <param name="pattern">(([\i-[:]][\c-[:]]*)?:)?.+</param>
        <param name="minLength">1</param>
      </data>
    </define>
    <define name="CURIEs.datatype">
      <x:p>A whitespace separated list of CURIEs</x:p>
      <list>
        <oneOrMore>
          <ref name="CURIE.datatype"/>
        </oneOrMore>
      </list>
    </define>
    <define name="SafeCURIE.datatype">
```

```
      <x:p>A single safe_curie</x:p>
      <data type="string">
        <param name="pattern">\[(([\i-[:]][\c-[:]]*)?:)?.+\]</param>
        <param name="minLength">3</param>
      </data>
    </define>
    <define name="SafeCURIEs.datatype">
      <x:p>A whitespace separated list of SafeCURIEs</x:p>
      <list>
        <oneOrMore>
          <ref name="SafeCURIE.datatype"/>
        </oneOrMore>
      </list>
    </define>
    <define name="URIorSafeCURIE.datatype">
      <x:p>A URI or a SafeCURIE (since you need a SafeCURIE
        to disambiguate between a common URI and a CURIE)</x:p>
      <choice>
        <ref name="URI.datatype"/>
        <ref name="CURIE.datatype"/>
      </choice>
    </define>
    <define name="URIorSafeCURIEs.datatype">
      <x:p>A whitespace separated list of URIorSafeCURIEs</x:p>
      <list>
        <oneOrMore>
          <ref name="URIorSafeCURIE.datatype"/>
        </oneOrMore>
      </list>
    </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="Characters.datatype">
      <data type="string"/>
    </define>
    <define name="Charset.datatype">
      <x:p>A character encoding, as per [RFC2045]</x:p>
      <text/>
    </define>
    <define name="Encodings.datatype">
      <x:p>A comma-separated list of 'charset's with optional q parameters,
        as defined in section 14.2 of [RFC2616] as the field value of
        the Accept-Charset request header.</x:p>
      <text/>
    </define>
    <define name="ContentType.datatype">
      <x:p>Media type, as per [RFC2045]</x:p>
      <text/>
    </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="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="QNames.datatype">
      <x:p>One or more white space separated QName values.</x:p>
      <list>
        <oneOrMore>
          <data type="QName"/>
        </oneOrMore>
      </list>
    </define>
    <define name="prefixedQName.datatype">
      <x:p>An [XMLNS]-qualified name.</x:p>
      <data type="QName">
        <param name="pattern">[\i-[:]][\c-[:]]*:[\i-[:]][\c-[:]]*</param>
      </data>
    </define>
    <define name="Shape.datatype">
      <x:p>The shape of a region.</x:p>
      <choice>
        <value>default</value>
        <value>rect</value>
        <value>circle</value>
        <value>poly</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>
          <ref name="URI.datatype"/>
        </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>
<x:p>This module imports the XMLEvents 2 modules, and
contextualizes them for the XHTML2 document type.</x:p>
  <x:h2>XHTML Events</x:h2>
  <include href="xml-events-2.rng">
      <define name="xmlEvents.Common.attrib">
              <ref name="CommonNoEvents.attrib"/>
      </define>
  </include>
  <define name="head.misc" combine="choice">
      <ref name="xmlEvents.listener"/>
  </define>
  <define name="Structural.class" combine="choice">
      <ref name="xmlEvents.listener"/>
  </define>
  <x:h2>XML Handlers</x:h2>
  <include href="xml-handlers-2.rng">
    <define name="xmlHandlers.Common.attrib">
      <ref name="CommonNoEvents.attrib"/>
    </define>
  </include>
  <define name="head.misc" combine="choice">
      <ref name="xmlHandlers.action"/>
  </define>
  <define name="Structural.class" combine="choice">
      <ref name="xmlHandlers.action"/>
  </define>
  <x:h2>XML Scripting</x:h2>
  <include href="xml-script-2.rng">
    <define name="xmlScripting.script.attlist">
      <ref name="xmlScripting.implements.attrib"/>
      <ref name="xmlScripting.charset.attrib"/>
      <ref name="Common.attrib"/>
    </define>
    <!-- redefine to use xhtml2-datatypes, which the
    vanilla XML Events 2 modules know nothing about -->
    <define name="xmlScripting.encoding.attrib">
      <optional>
        <attribute name="encoding">
          <ref name="Encodings.datatype"/>
        </attribute>
      </optional>
    </define>
    <define name="xmlScripting.implements.attrib">
      <optional>
        <attribute name="implements">
          <ref name="URIorSafeCURIEs.datatype"/>
        </attribute>
      </optional>
    </define>
    <define name="xmlScripting.type.attrib">
      <attribute name="type">
        <ref name="ContentType.datatype"/>
      </attribute>
    </define>
    <define name="xmlScripting.charset.attrib">
```

```
        <optional>
          <attribute name="charset">
            <ref name="Charset.datatype"/>
          </attribute>
        </optional>
      </define>
    </include>
    <define name="xmlHandlers.action.content" combine="choice">
        <ref name="xmlScripting.script"/>
    </define>
    <define name="head.misc" combine="choice">
        <ref name="xmlScripting.script"/>
    </define>
    <define name="Structural.class" combine="choice">
        <ref name="xmlScripting.script"/>
    </define>
    <define name="Text.class" combine="choice">
        <ref name="xmlScripting.script"/>
    </define>
  <x:h2>Events Global Attributes Collection</x:h2>
  <define name="Events.attrib">
    <optional>
      <ref name="xmlEvents.event.attrib"/>
    </optional>
    <optional>
      <ref name="xmlEvents.observer.attrib"/>
    </optional>
    <optional>
      <ref name="xmlEvents.eventTarget.attrib"/>
    </optional>
    <optional>
      <ref name="xmlEvents.function.attrib"/>
    </optional>
    <optional>
      <ref name="xmlEvents.handler.attrib"/>
    </optional>
    <optional>
      <ref name="xmlEvents.phase.attrib"/>
    </optional>
    <optional>
      <ref name="xmlEvents.propagate.attrib"/>
    </optional>
    <optional>
      <ref name="xmlEvents.defaultAction.attrib"/>
    </optional>
  </define>
  <define name="Common.attrib" combine="interleave">
    <ref name="Events.attrib"/>
  </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: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="caption.content"/>
      </element>
    </define>
    <define name="caption.attlist">
      <ref name="Common.attrib"/>
```

```
    </define>
    <define name="caption.content">
      <ref name="Text.model"/>
    </define>
  </div>
</grammar>
```

C.2.5. Role 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>Role Attribute Module</x:h1>
  <define name="role.attrib">
    <optional>
      <attribute name="role">
        <ref name="CURIEs.datatype"/>
      </attribute>
    </optional>
  </define>
  <define name="Common.attrib" combine="interleave">
    <ref name="role.attrib"/>
  </define>
  <define name="CommonNoEvents.attrib" combine="interleave">
    <ref name="role.attrib"/>
  </define>
</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: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro 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: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro 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 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>XML Events Module in RELAX NG</x:h1>
  <x:pre>
    This is the RELAX NG Schema for the XML Events 2 XML Events module.
    Copyright &#xA9;2003-2009 W3C&#xAE; (MIT, ERCIM, Keio), All Rights Reserved.
    URI: http://www.w3.org/MarkUp/RELAXNG/xml-events-2.rng
    Editors:   Masayasu Ishikawa &lt;mimasa@w3.org&gt;
               Markus Gylling &lt;markus.gylling@tpb.se&gt;
    Revision: $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro 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="xmlEvents.listener">
  <element name="listener">
    <ref name="xmlEvents.listener.attlist"/>
  </element>
</define>
<define name="xmlEvents.listener.attlist">
      <ref name="xmlEvents.event.attrib"/>
  <optional>
    <ref name="xmlEvents.observer.attrib"/>
  </optional>
  <optional>
    <ref name="xmlEvents.eventTarget.attrib"/>
  </optional>
  <optional>
        <ref name="xmlEvents.function.attrib"/>
      </optional>
  <optional>
    <ref name="xmlEvents.handler.attrib"/>
  </optional>
  <optional>
    <ref name="xmlEvents.phase.attrib"/>
  </optional>
  <optional>
    <ref name="xmlEvents.propagate.attrib"/>
  </optional>
  <optional>
    <ref name="xmlEvents.defaultAction.attrib"/>
  </optional>
  <ref name="xmlEvents.Common.attrib"/>
</define>
<define name="xmlEvents.Common.attrib">
  <optional>
    <attribute name="xml:id">
      <data type="ID"/>
    </attribute>
  </optional>
</define>
<define name="xmlEvents.event.attrib">
  <attribute name="event">
    <list>
      <oneOrMore>
        <data type="QName"/>
      </oneOrMore>
    </list>
  </attribute>
</define>
<define name="xmlEvents.eventTarget.attrib">
  <attribute name="eventTarget">
    <data type="IDREFS"/>
  </attribute>
</define>
<define name="xmlEvents.phase.attrib">
  <attribute name="phase" a:defaultValue="default">
    <choice>
      <value>bubble</value>
```

```
        <value>capture</value>
        <value>target</value>
        <value>default</value>
      </choice>
    </attribute>
  </define>
  <define name="xmlEvents.handler.attrib">
    <attribute name="handler">
      <data type="anyURI"/>
     </attribute>
  </define>
  <define name="xmlEvents.observer.attrib">
    <attribute name="observer">
      <data type="IDREFS"/>
    </attribute>
  </define>
  <define name="xmlEvents.function.attrib">
      <attribute name="function">
        <text/>
      </attribute>
  </define>
  <define name="xmlEvents.propagate.attrib">
    <attribute name="propagate" a:defaultValue="continue">
      <choice>
        <value>stop</value>
        <value>continue</value>
      </choice>
    </attribute>
  </define>
  <define name="xmlEvents.defaultAction.attrib">
    <attribute name="defaultAction" a:defaultValue="perform">
      <choice>
        <value>cancel</value>
        <value>perform</value>
      </choice>
    </attribute>
  </define>
</grammar>
```

C.3.4. XML Handlers

```
<?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>XML Handlers Module in RELAX NG</x:h1>
  <x:pre>
    This is the RELAX NG Schema for the XML Events 2 XML Handlers module.
    As per the XMLEvents 2 specification, this modules depends
    on the XML Events Module (xml-events-2.rng).
    Copyright &#xA9;2009 W3C&#xAE; (MIT, ERCIM, Keio), All Rights Reserved.
    URI: http://www.w3.org/MarkUp/RELAXNG/xml-handlers-2.rng
    Editor:   Markus Gylling &lt;markus.gylling@tpb.se&gt;
    Revision: $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    Permission to use, copy, modify and distribute this RELAX NG schema
    for XML Handlers 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="xmlHandlers.action">
      <element name="action">
              <ref name="xmlHandlers.action.attlist"/>
              <ref name="xmlHandlers.action.content"/>
      </element>
</define>
<define name="xmlHandlers.action.attlist">
  <ref name="xmlEvents.event.attrib"/>
  <optional>
    <ref name="xmlEvents.eventTarget.attrib"/>
  </optional>
  <ref name="xmlHandlers.declare.attrib"/>
  <ref name="xmlHandlers.if.attrib"/>
  <ref name="xmlHandlers.while.attrib"/>
  <ref name="xmlHandlers.Common.attrib"/>
</define>
<define name="xmlHandlers.action.content">
      <choice>
        <ref name="xmlHandlers.action"/>
        <ref name="xmlHandlers.dispatchEvent"/>
        <ref name="xmlHandlers.addEventListener"/>
        <ref name="xmlHandlers.removeEventListener"/>
        <ref name="xmlHandlers.stopPropagation"/>
        <ref name="xmlHandlers.preventDefault"/>
      </choice>
</define>
<define name="xmlHandlers.dispatchEvent">
      <element name="dispatchEvent">
    <ref name="xmlHandlers.dispatchEvent.attlist"/>
        <empty/>
      </element>
</define>
<define name="xmlHandlers.dispatchEvent.attlist">
  <ref name="xmlHandlers.Common.attrib"/>
  <ref name="xmlHandlers.eventType.attrib"/>
  <ref name="xmlEvents.eventTarget.attrib"/>
  <ref name="xmlHandlers.bubbles.attrib"/>
  <ref name="xmlHandlers.cancelable.attrib"/>
</define>
<define name="xmlHandlers.addEventListener">
      <element name="addEventListener">
    <ref name="xmlHandlers.addEventListener.attlist"/>
        <empty/>
      </element>
</define>
<define name="xmlHandlers.addEventListener.attlist">
      <ref name="xmlHandlers.removeEventListener.attlist"/>
</define>
<define name="xmlHandlers.removeEventListener">
      <element name="removeEventListener">
    <ref name="xmlHandlers.removeEventListener.attlist"/>
        <empty/>
      </element>
</define>
<define name="xmlHandlers.removeEventListener.attlist">
  <ref name="xmlHandlers.event.attrib"/>
  <ref name="xmlEvents.handler.attrib"/>
  <optional>
    <ref name="xmlEvents.phase.attrib"/>
  </optional>
  <ref name="xmlHandlers.Common.attrib"/>
</define>
<define name="xmlHandlers.stopPropagation">
      <element name="stopPropagation">
        <ref name="xmlHandlers.stopPropagation.attlist"/>
        <empty/>
      </element>
</define>
<define name="xmlHandlers.stopPropagation.attlist">
  <ref name="xmlHandlers.Common.attrib"/>
  <ref name="xmlHandlers.event.attrib"/>
</define>
<define name="xmlHandlers.preventDefault">
```

```
      <element name="preventDefault">
    <ref name="xmlHandlers.preventDefault.attlist"/>
       <empty/>
      </element>
  </define>
  <define name="xmlHandlers.preventDefault.attlist">
    <ref name="xmlHandlers.Common.attrib"/>
    <ref name="xmlHandlers.event.attrib"/>
  </define>
  <define name="xmlHandlers.Common.attrib">
    <optional>
      <attribute name="xml:id">
        <data type="ID"/>
      </attribute>
    </optional>
  </define>
  <define name="xmlHandlers.event.attrib">
    <attribute name="event">
      <data type="QName"/>
    </attribute>
  </define>
  <define name="xmlHandlers.declare.attrib">
    <optional>
      <attribute name="declare">
        <value>declare</value>
      </attribute>
    </optional>
  </define>
  <define name="xmlHandlers.if.attrib">
    <optional>
      <attribute name="if">
        <data type="normalizedString"/>
      </attribute>
    </optional>
  </define>
  <define name="xmlHandlers.while.attrib">
    <optional>
      <attribute name="while">
        <data type="normalizedString"/>
      </attribute>
    </optional>
  </define>
  <define name="xmlHandlers.eventType.attrib">
    <attribute name="eventType">
       <data type="QName"/>
    </attribute>
  </define>
  <define name="xmlHandlers.bubbles.attrib">
    <optional>
      <attribute name="bubbles">
        <value>bubbles</value>
      </attribute>
    </optional>
  </define>
  <define name="xmlHandlers.cancelable.attrib">
    <optional>
      <attribute name="cancelable">
        <value>cancelable</value>
      </attribute>
    </optional>
  </define>
</grammar>
```

C.3.5. XML Script

```
<?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>XML Scripting Module in RELAX NG</x:h1>
  <x:pre>
    This is the RELAX NG Schema for the XML Events 2 Scripting module.
```

```
  Copyright &#xA9;2009 W3C&#xAE; (MIT, ERCIM, Keio), All Rights Reserved.
  URI: http://www.w3.org/MarkUp/RELAXNG/xml-script-2.rng
  Editor:   Markus Gylling &lt;markus.gylling@tpb.se&gt;
  Revision: $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
  Permission to use, copy, modify and distribute this RELAX NG schema
  for XML Scripting 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="xmlScripting.script">
  <element name="script">
    <ref name="xmlScripting.script.attlist"/>
    <ref name="xmlScripting.script.content"/>
  </element>
</define>
<define name="xmlScripting.script.attlist">
  <ref name="xmlScripting.encoding.attrib"/>
  <ref name="xmlScripting.charset.attrib"/>
  <ref name="xmlScripting.defer.attrib"/>
  <ref name="xmlScripting.implements.attrib"/>
  <ref name="xmlScripting.src.attrib"/>
  <ref name="xmlScripting.type.attrib"/>
  <ref name="xmlScripting.Common.attrib"/>
</define>
<define name="xmlScripting.script.content">
      <text/>
</define>
<define name="xmlScripting.encoding.attrib">
  <optional>
    <attribute name="encoding">
      <list>
        <oneOrMore>
              <data type="string"/>
        </oneOrMore>
      </list>
    </attribute>
  </optional>
</define>
<define name="xmlScripting.charset.attrib">
  <optional>
    <attribute name="charset">
      <data type="string"/>
    </attribute>
  </optional>
</define>
<define name="xmlScripting.defer.attrib">
  <optional>
    <attribute name="defer">
      <value>defer</value>
    </attribute>
  </optional>
```

```
  </define>
  <define name="xmlScripting.Common.attrib">
    <optional>
      <attribute name="xml:id">
        <data type="ID"/>
      </attribute>
    </optional>
  </define>
  <define name="xmlScripting.implements.attrib">
    <optional>
      <attribute name="implements">
        <list> <!-- URIorSafeCURIEs -->
          <oneOrMore>
            <choice>
              <data type="anyURI"/>
              <data type="string">
                <param name="pattern">(([\i-[:]][\c-[:]]*)?:)?.+</param>
                <param name="minLength">1</param>
              </data>
            </choice>
          </oneOrMore>
        </list>
      </attribute>
    </optional>
  </define>
  <define name="xmlScripting.src.attrib">
    <optional>
      <attribute name="src">
        <data type="anyURI"/>
      </attribute>
    </optional>
  </define>
  <define name="xmlScripting.type.attrib">
    <attribute name="type">
      <data type="string"/>
    </attribute>
  </define>
</grammar>
```

C.3.6. 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. Building Schema Modules

This appendix is *normative*.

XHTML modules are implemented as XML Schemas. When these XML Schemas are assembled in a specific manner (described in Developing Schemas with defined and extended modules [p.217]), the resulting Schema is a representation of a complete document type. This representation can then be used for validation of instances of the document type.

The key to combining these schema components into a meaningful, cohesive schema is the rules used to define the individual XML Schemas. This section defines those rules. When these rules are followed, markup language authors can be confident that their modules will interface cleanly with other XHTML-compatible modules.

Modules conforming to these rules also need to satisfy the conformance requirements defined in *XHTML Family Module Conformance* in order to be called *XHTML Family Modules*.

D.1. Named Content Models

This specification classifies named content model into categories and names them consistently using the following suffixes

.content
: model group definitions use the suffix `.content` when they are used to represent the content model of an element type.

.class
: model group definitions use the suffix `.class` when they are used to represent elements of the same class.

.mix
: model group definitions use the suffix `.mix` when they are used to represent a collection of element types from different classes.

.extra
: model group definitions use the suffix `.extra` when they are used to extend other groups above.

.export
: model group definitions add the suffix `.export` when they are to be used by a host language as the basis for extending the related content model (e.g., `xhtml.Flow.mix` could have an `xhtml.Flow.mix.export` that defines a collection of elements that must be included in a redefinition of the `xhtml.Flow.mix` by a host language.

.type
: named complex type definitions use the suffix `.type` when they are used to represent type of an element. Types usually include the `.attlist` and `.content` components.

.attlist
: attribute groups use the suffix `.attlist` when they are used to represent the attributes for a specific element.

.attrib
: attribute groups use the suffix `.attrib` when they are used to represent a group of tokens representing one or more complete attribute specifications within an .attlist declaration.

For example, in HTML 4, the `%block;` parameter entity is defined to represent the heterogeneous collection of element types that are block-level elements. In this specification, the corollary named content model is `xhtml.Block.mix`.

When defining named content models in the classes defined here, modules should scope the names of the model group definitions and attribute groups by using unique prefixes (this recommendation uses the prefix `xhtml.`). For example, the content model for the element `myelement` in the module mymodule could be named `mymodule.myelement.content`. Other schemes are possible. Regardless of the scheme used, module authors should strive to ensure that named content model they define are named uniquely so that they do not collide with other named content model and so that the interface methods for the module are obvious to its users.

D.2. Defining the Namespace of a Module

XHTML requires that the elements and attributes declared in a module be within a defined XML namespace [XMLNAMES] [p.??] . The identification of this namespace is an arbitrary URI. XHTML **does not require** that a module declare its target namespace using the `targetnamespace` attribute. XHTML Modularization using XML Schema has adopted a "late binding" approach to associating with a namespace. This permits the development of so-called "chameleon" modules, where the elements and attributes of a module can be incorporated into more than one namespace.

D.2.1. Global and Local Element Declarations

While XML Schema allows an the definition of global and local element declarations, to be compatible with DTD definitions of XHTML Modularization module implementations must not declare local elements.

D.2.2. Global and Local Attribute Declarations

While the approach defined here permits the definition of global as well as local attribute declarations, schema authors should consider the consequences of such definitions on an document instance. Global attributes must always be explicitly prefixed in a instance document by declaring a namespace prefix `xmlns:prefix`, while local attributes depending on the schema implementation may be explicitly prefixed.

D.3. Importing External Namespace Schema Components

An XML Schema provides definitions that belong to a given target namespace. A schema must use the `import` element to include components from an XML Schema that uses a different target namespace. The `import` element in XML Schema requires a namespace attribute and a optional schemaLocation attribute. Multiple modules (included in a document type) importing components from the same external namespace but providing different schema location URI values will result in invalid driver schema. To avoid such issues modularization requires that modules importing external schemas must not provide a schemaLocation attribute so that a document type's driver file may import these schemas with the schemaLocation attribute.

D.4. Datatype Definitions and Namespaces

While the elements and attributes of a module should NOT be in a namespace until they are used by a markup language, the datatypes that a module relies upon may need to be. This is especially important if the datatypes are to be shared with other markup languages. If your module has datatypes that you want to share with other modules, you should define a namespace for those datatypes, place the datatype definitions in a separate "module" and bind that module to the namespace. In XHTML Modularization, for example, we use the namespace `http://www.w3.org/1999/xhtml/datatypes/`.

D.5. Content Model Redefinitions

Quite often modules change the content model of elements defined by other modules. For example, the XHTML Events module adds event attributes to elements defined by the forms module. It is also possible that multiple modules may change the content model of a single element defined by a third module, for example both XHTML Events Module and XHTML Image Map module add attributes to elements in form module.

XML Schemas allows for changes to a declared content model using the `redefine` element. While XML Schema supports the use of a `redefine` element that redefines the named content model and type definition, XML Schema does not directly support redefinition of an element or attribute declaration.

To support element content model redefinitions, all content models are defined with a `.content` identifier. This identifier can be easily redefined when creating a driver module.

- Schema module implementations must define the content model of an element using named complex types schema component `.type`. Further the named schema types must be defined in terms of named content model `.content` and `.attlist`
- `redefine` in XML Schema by default includes the referenced schema. Since the instantiation of a module is decided by document type's driver file, the module implementations must not directly redefine the content model of other modules in its implementation.
- Modules that expect to have their content model defined or extended by the host language must define a special `.export` group for each element or content class that needs to have

its content model extended. Host languages will use this `.export` group as the basis for the content model of an element, extending it with whatever additional content is appropriate. elements that

E. Developing Schema with defined and extended modules

This appendix is *informative*.

The primary purpose of defining XHTML modules and a general modularization methodology is to ease the development of document types that are based upon XHTML using XML Schemas. These document types may extend XHTML by integrating additional capabilities (e.g., [SMIL] [p.??]), or they may define a subset of XHTML for use in a specialized device. This section describes the techniques that document type designers must use in order to take advantage of the XML Schema implementation of this modularization architecture. It does this by applying the XHTML Modularization techniques in progressively more complex ways, culminating in the creation of a complete document type from disparate modules.

Note that in no case do these examples require the modification of the XHTML-provided module *file entities* themselves. The XHTML module file entities are completely parameterized, so that it is possible through separate module definitions and *driver files* to customize the definition and the content model of each element and each element's hierarchy.

Finally, remember that most users of XHTML are *not* expected to be XML Schema authors. XML Schema authors are generally people who are defining specialized markup that will improve the readability, simplify the rendering of a document, or ease machine-processing of documents, or they are client designers that need to define the specialized markup language for their specific client. Consider these cases:

- An organization is providing subscriber's information via a Web interface. The organization stores its subscriber information in an XML-based database. One way to report that information out from the database to the Web is to embed the XML records from the database directly in the XHTML document. While it is possible to merely embed the records, the organization could define a module that describes the records, attach that module to an XHTML implementation, and thereby create a complete markup language for the pages. The organization can then access the data within the new elements via the Document Object Model [DOM] [p.??] , validate the documents, provide style definitions for the elements that cascade using Cascading Style Sheets [CSS2] [p.??] , etc. By taking the time to define the structure of their data and create a markup language using the processes defined in this section, the organization can realize the full benefits of XML.

- An Internet client developer is designing a specialized device. That device will only support a subset of XHTML, and the devices will always access the Internet via a proxy server that validates content before passing it on to the client (to minimize error handling on the client). In order to ensure that the content is valid, the developer creates a markup language that is a subset of XHTML using the processes defined in this section. They then use the new language definition in their proxy server and in their devices, and also make the language definition available to content developers so that developers can validate their content before making it available. By performing a few simple steps, the client developer can use the architecture defined in this document to greatly ease their language development cost

and ensure that they are fully supporting the subset of XHTML that they choose to include.

E.1. Defining additional attributes

In some cases, an extension to XHTML can be as simple as additional attributes. Schema authors should an provide the attribute definitions for each attribute, for example:

```
<xs:attributeGroup name="myattrs.attrib">
        <xs:attribute name="myattribute" type="xs:string"/>
</xs:attributeGroup>
```

would declare an attribute "myattr" and attribute group "myattrs.attrib" in the target namespace of the schema ("xs" is the prefix for XML Schema Namespace). Authors should note that the attribute is created as local attribute (as part attribute group). Alternatively, declaring an attribute by placing the attribute declaration as direct child of `schema` element would create a Global attribute (and document instances would have to use qualified attribute name such as xlink:show). *For a discussion of qualified names and Namespace prefixes, see Defining the Namespace of a Module [p.142] .*

To add this attribute to the content model of an element, the attribute group (that makes the content model of the element) would need to be redefined (by the document type's driver file) to include the new attribute. for example:

```
<xs:redefine schemaLocation="xhtml-basic10.xsd">
        <xs:attributeGroup name="a.attlist">
                <xs:attributeGroup ref="a.attlist"/>
                <xs:attributeGroup ref="myml:myattrs.attrib"/>
        </xs:attributeGroup>
</xs:redefine>
```

The target namespace of the attribute group definition is not XHTML namespace and must be contained in a separate XML schema.

Naturally, adding an attribute to a schema does not mean that any new behavior is defined for arbitrary clients. However, a content developer could use an extra attribute to store information that is accessed by associated scripts via the Document Object Model (for example).

E.2. Defining additional elements

Defining additional elements is similar to attributes, but a typical XHTML module would define the element as a global element (as a direct child of `schema` element). Schema authors should first provide the element declaration for each element:

```
<!-- In the myml-module-1.xsd -->
<xs:group name="myelement.content">
        <xs:choice>
                <xs:element name="otherelement"/>
        </xs:choice>
</xs:group>
<xs:attributeGroup name="myelement.attlist">
```

```
        <xs:attribute name="myattribute" type="xs:string"/>
</xs:attributeGroup>
<xs:complexType name="myelement.type" mixed="true">
        <xs:choice>
                <xs:group ref="myelement.content" minOccurs="0" maxOccurs="1"/>
        </xs:choice>
        <xs:attributeGroup ref="myelement.attlist"/>
</xs:complexType>
<xs:element name="myelement" type="myelement.type"/>
```

The target namespace of "myelement" declared is not XHTML namespace, hence must be contained in a separate XML Schema. "xs" is the prefix for XML Schema Namespace. After the elements are defined, they need to be integrated into the content model. Strategies for integrating new elements or sets of elements into the content model are addressed in the next section.

E.3. Defining the content model for a collection of modules

Since the content model of XHTML modules is fully parameterized using named content models, Schema authors may modify the content model for every element in every module. The details of the schema module interface are defined in Building Schema Modules [p.213] . Basically there are two ways to approach this modification:

1. Re-define the named content model, `.content`, for each element.
2. Define one or more of the global named content model entities to include the element in those named model definitions (normally via the named content model, `.extras`).

The strategy taken will depend upon the nature of the modules being combined and the nature of the elements being integrated. The remainder of this section describes techniques for integrating two different classes of modules.

E.3.1. Integrating a stand-alone module into XHTML

When a module (and remember, a module can be a collection of other modules) contains elements that only reference each other in their content model, it is said to be "internally complete". As such, the module can be used on its own; (for example, you could define a schema that was just that module, and use one of its elements as the root element). Integrating such a module into XHTML is a three step process:

1. Decide what element(s) can be thought of as the root(s) of the new module.
2. Decide where these elements need to attach in the XHTML content tree.
3. Then, for each attachment point in the content tree, add the root element(s) to the content definition for the XHTML elements.

Consider attaching the elements defined above [p.150] . In that example, the element `myelement` is the root. To attach this element under the `img` element, and only the `img` element, of XHTML, the following redefinition would work:

```
<xs:redefine schemaLocation="xhtml-basic10.xsd">
        <xs:group name="img.content">
                <xs:choice>
                    <xs:group ref="img.content"/>
                        <xs:element ref="myml:myelement"/>
                </xs:choice>
        </xs:group>
</xs:redefine>
```

Such redefinition must not be included in the module implementation, but instead provided as part of the document type's driver implementation. A schema defined with this content model would allow a document like the following fragment:

```
<img src="http://examples.com/image" alt="alt-text">
  <myml:myelement >This is content of a locally defined element</myml:myelement>
</img>
```

It is important to note that normally the `img` element has a content model of `EMPTY`. By adding myelement to that content model, we are really just replacing `EMPTY` with `myelement`. In the case of other elements that already have content models defined, the addition of an element would require the restating of the existing content model in addition to `myelement`.

E.3.2. Mixing a new module throughout the modules in XHTML

Extending the example above, to attach this module everywhere that the `%Flow.mix` content model group is permitted, would require something like the following in the schema that defines the document model of the document type:

```
<xs:redefine schemaLocation="xhtml11.xsd">
        <xs:group name="Misc.extra">
                <xs:choice>
                        <xs:group ref="Misc.extra"/>
                        <xs:element ref="myml:myelement"/>
                </xs:choice>
        </xs:group>
</xs:redefine>
```

Since the Misc.extra content model class is used in the content model the named model Misc.class, and that named model is used throughout the XHTML modules, the new module would become available throughout an extended XHTML document type.

E.4. Creating a new Document Type

So far the examples in this section have described the methods of extending XHTML and XHTML's content model. Once this is done, the next step is to collect the modules that comprise the Document Type into a schema driver and schema file that provides the content model redefinitions of included modules, incorporating the new definitions so that they override and augment the basic XHTML definitions as appropriate.

E.4.1. Creating a simple Document Type

Using the trivial example above, it is possible to define a new schema that uses and extends the XHTML modules pretty easily. First, define the new elements and their content model in a module:

```
Module schema_developing not found!
```

Now, define the schema driver for the new language:

```
Module schema_developing not found!
```

A schema defined with this content model would allow a document like the following:

```
Module schema_developing not found!
```

E.4.2. Creating a Language by extending XHTML

Next, there is the situation where a complete, additional, and complex module is added to XHTML (or to a subset of XHTML). In essence, this is the same as in the example above, the only difference being that the module being added is incorporated in the schema by creating an new document model schema.

One such complex module is the Schema for [MATHML] [p.??] . In order to combine MathML and XHTML into a single Schema, an author would just decide where MathML content should be legal in the document, and add the MathML root element to the content model at that point. First, define a new document model that instantiates the MathML Schema and connects it to the content XHTML content model by redefining the XHTML content model. Providing a redefinition of the XHTML content model by implication includes the XHTML content model in the new document content model :

```
Module schema_developing not found!
```

Next, define a Schema driver that includes our new document content model with XHTML1.1 modules and MathML module (for example):

```
Module schema_developing not found!
```

E.4.3. Creating a Language by removing and replacing XHTML modules

Another way in which Schema authors may use XHTML modules is to define a Schema that is a subset of an XHTML family document type (because, for example, they are building devices or software that only supports a subset of XHTML). To do this simple create a Schema driver that does not include the relevant modules. Schema author should note that `redefine` in schema by default includes all the content model of the referenced schema, authors should also not include any redefinitions of modules that they do not wish to include. The basic steps to follow are:

1. Take an XHTML family Schema as the basis of the new document type (e.g. XHTML 1.1).
2. Select the modules to remove from that Schema.
3. Physically, remove `include` and `redefine` schema elements that include any non relevant modules from the driver file. Also references to schema components from such modules used in redefinitions of other modules must be deleted.
4. Introduce some new modules

E.4.4. Creating a the new Document Type

Finally, some Schema authors may wish to start from scratch, using the XHTML Modularization framework as a toolkit for building a new markup language. This language must be made up of the minimal, required modules from XHTML. It may also contain other XHTML-defined modules or any other module that the author wishes to employ. In this example, we will take the XHTML required modules, add some XHTML-defined modules, and also add in the module we defined above.

The first step is to define a module that defines the elements and attributes using the provided template.

```
Module schema_developing not found!
```

Now, build a content model description that hooks the new elements and attributes into the other XHTML elements. The following example is patterned after the XHTML Basic content model, but is a complete, free-standing content model module:

```
Module schema_developing not found!
```

Finally, build a driver schema. For ease of extensibility this driver schema is split into two XML Schema files. The first file of driver schema collects (includes) all the modules needed for the new document type. This schema also provides the required redefinitions of schema components in included modules.(Note: in XML Schema `redefine` includes the schema referenced.

```
Module schema_developing not found!
```

The second file of the driver schema builds new document type based the content model and modules. Also this schema provides the schemaLocation for all imported namespaces (namespaces imported by the included modules)

```
Module schema_developing not found!
```

Once a new SCHEMA has been developed, it can be used in any document. Using the Schema is as simple as just referencing it in the schemaLocation attribute of a document root element:

```
Module schema_developing not found!
```

F. XHTML Schema Module Implementations

This appendix is *normative*.

This appendix contains implementations of the modules defined in this specification via XML Schema [XMLSCHEMA [p.297]].

F.1. Required Modules

F.1.1. Datatypes Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns="http://www.w3.org/1999/xhtml/datatypes/"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
 targetNamespace="http://www.w3.org/1999/xhtml/datatypes/"
 elementFormDefault="qualified"
>
    <xs:annotation>
        <xs:documentation>
          XHTML Datatypes
          This is the XML Schema datatypes module for XHTML
          Defines containers for the XHTML datatypes, many of
          these imported from other specifications and standards.
          $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
        </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstraction.html#s_common_attrtypes"/>
    </xs:annotation>
    <!-- nn for pixels or nn% for percentage length -->
    <xs:simpleType name="Length">
        <xs:union memberTypes="xs:nonNegativeInteger">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:pattern value="\d+[%]|\d*\.\d+[%]"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:union>
    </xs:simpleType>
    <!-- space-separated list of link types -->
    <xs:simpleType name="LinkTypes">
        <xs:list itemType="xs:NMTOKEN"/>
    </xs:simpleType>
    <!-- single or comma-separated list of media descriptors -->
    <xs:simpleType name="MediaDesc">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <!-- pixel, percentage, or relative -->
    <xs:simpleType name="MultiLength">
        <xs:union memberTypes="xh11d:Length">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:pattern value="\d*\*"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:union>
    </xs:simpleType>
    <!-- one or more digits (NUMBER) -->
    <xs:simpleType name="Number">
        <xs:restriction base="xs:nonNegativeInteger"/>
    </xs:simpleType>
    <!-- integer representing length in pixels -->
    <xs:simpleType name="Pixels">
        <xs:restriction base="xs:nonNegativeInteger"/>
    </xs:simpleType>
    <!-- script expression -->
    <xs:simpleType name="Script">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <!-- sixteen color names or RGB color expression-->
    <xs:simpleType name="Color">
        <xs:union memberTypes="xs:NMTOKEN">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:pattern value="#[0-9a-fA-F]{3}([0-9a-fA-F]{3})?"/>
                </xs:restriction>
            </xs:simpleType>
```

```
    </xs:union>
</xs:simpleType>
<!-- textual content -->
<xs:simpleType name="Text">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- Imported Datatypes  -->
<!-- a single character, as per section 2.2 of [XML] -->
<xs:simpleType name="Character">
    <xs:restriction base="xs:string">
        <xs:length value="1" fixed="true"/>
    </xs:restriction>
</xs:simpleType>
<!-- a character encoding, as per [RFC2045] -->
<xs:simpleType name="Charset">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- a space separated list of character encodings, as per [RFC2045] -->
<xs:simpleType name="Charsets">
    <xs:list itemType="Charset"/>
</xs:simpleType>
<!-- media type, as per [RFC2045] -->
<xs:simpleType name="ContentType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- comma-separated list of media types, as per [RFC2045] -->
<xs:simpleType name="ContentTypes">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- date and time information. ISO date format -->
<xs:simpleType name="Datetime">
    <xs:restriction base="xs:dateTime"/>
</xs:simpleType>
<!-- formal public identifier, as per [ISO8879] -->
<xs:simpleType name="FPI">
    <xs:restriction base="xs:normalizedString"/>
</xs:simpleType>
<!-- a window name as used in the target attribute -->
<xs:simpleType name="FrameTarget">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="_blank"/>
        <xs:enumeration value="_self"/>
        <xs:enumeration value="_parent"/>
        <xs:enumeration value="_top"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="[a-zA-Z].*"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
<!-- a language code, as per [RFC3066] -->
<xs:simpleType name="LanguageCode">
    <xs:restriction base="xs:language"/>
</xs:simpleType>
<!-- a comma separated list of language ranges -->
<xs:simpleType name="LanguageCodes">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- a Uniform Resource Identifier, see [URI] -->
<xs:simpleType name="URI">
    <xs:restriction base="xs:anyURI"/>
</xs:simpleType>
<!-- a space-separated list of Uniform Resource Identifiers, see [URI] -->
<xs:simpleType name="URIs">
    <xs:list itemType="xs:anyURI"/>
</xs:simpleType>
<!-- comma-separated list of MultiLength -->
<xs:simpleType name="MultiLengths">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- character Data -->
<xs:simpleType name="CDATA">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
<!-- CURIE placeholder datatypes -->
<xs:simpleType name="CURIE">
    <xs:restriction base="xs:string">
        <xs:pattern value="(([\i-[:]][\c-[:]]*)?:)?.+" />
        <xs:minLength value="1"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="CURIEs">
    <xs:list itemType="xhlld:CURIE"/>
</xs:simpleType>
<xs:simpleType name="SafeCURIE">
```

```
    <xs:restriction base="xs:string">
        <xs:pattern value="\[(([\i-[:]][\c-[:]]*)?:)?.+\]" />
        <xs:minLength value="3"/>
    </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="SafeCURIEs">
        <xs:list itemType="xh11d:SafeCURIE"/>
    </xs:simpleType>
    <xs:simpleType name="URIorSafeCURIE">
        <xs:union memberTypes="xs:anyURI xh11d:SafeCURIE" />
    </xs:simpleType>
    <xs:simpleType name="URIorSafeCURIEs">
        <xs:list itemType="xh11d:URIorSafeCURIE"/>
    </xs:simpleType>
    <xs:simpleType name="Encodings">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="HrefTarget">
        <xs:restriction base="xs:NMTOKEN"/>
    </xs:simpleType>
    <xs:simpleType name="LocationPath">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="QName">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="QNames">
        <xs:list itemType="xh11d:QName"/>
    </xs:simpleType>
</xs:schema>
```

F.1.2. Document Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/">
    <xs:import
        namespace="http://www.w3.org/1999/xhtml/datatypes/"
        schemaLocation="xhtml-datatypes-2.xsd"/>
    <xs:annotation>
        <xs:documentation>
        This is the XML Schema Document Module for XHTML
          * title, head, body, html
        The Document Module defines the major structural elements and
        their attributes.
        $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
      </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/xhtml2#s_documentmodule"/>
    </xs:annotation>
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd">
        <xs:annotation>
            <xs:documentation>
                This import brings in the XML namespace attributes
                The module itself does not provide the schemaLocation
                and expects the driver schema to provide the
                actual SchemaLocation.
            </xs:documentation>
        </xs:annotation>
    </xs:import>
    <xs:attributeGroup
        name="xhtml.title.attlist">
        <xs:attributeGroup
            ref="xhtml.Common.attrib"/>
```

```
</xs:attributeGroup>
<xs:group
    name="xhtml.title.content">
    <xs:sequence/>
</xs:group>
<xs:complexType
    name="xhtml.title.type"
    mixed="true">
    <xs:group
        ref="xhtml.title.content"/>
    <xs:attributeGroup
        ref="xhtml.title.attlist"/>
</xs:complexType>
<xs:attributeGroup
    name="xhtml.profile.attrib">
    <xs:attribute
        name="profile"
        type="xh11d:URIs"/>
</xs:attributeGroup>
<xs:attributeGroup
    name="xhtml.head.attlist">
    <xs:attributeGroup
        ref="xhtml.profile.attrib"/>
    <xs:attributeGroup
        ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:complexType
    name="xhtml.head.type">
    <xs:group
        ref="xhtml.head.content"/>
    <xs:attributeGroup
        ref="xhtml.head.attlist"/>
</xs:complexType>
<xs:attributeGroup
    name="xhtml.body.attlist">
    <xs:attributeGroup
        ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group
    name="xhtml.body.content">
    <xs:sequence>
        <xs:group
            ref="xhtml.Structural.mix"
            minOccurs="0"
            maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType
    name="xhtml.body.type">
    <xs:group
        ref="xhtml.body.content"/>
    <xs:attributeGroup
        ref="xhtml.body.attlist"/>
</xs:complexType>
<xs:attributeGroup
    name="xhtml.version.attrib">
    <xs:attribute
```

```
            name="version"
            type="xh11d:CDATA"/>
    </xs:attributeGroup>
    <xs:attributeGroup
        name="xhtml.html.attlist">
        <xs:attributeGroup
            ref="xhtml.version.attrib"/>
        <xs:attributeGroup
            ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group
        name="xhtml.html.content">
        <xs:sequence>
            <xs:element
                name="head"
                type="xhtml.head.type"/>
            <xs:element
                name="body"
                type="xhtml.body.type"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType
        name="xhtml.html.type">
        <xs:group
            ref="xhtml.html.content"/>
        <xs:attributeGroup
            ref="xhtml.html.attlist"/>
    </xs:complexType>
</xs:schema>
```

F.1.3. Structural Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
          This is the XML Schema Structural module for XHTML
          $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
       </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
    </xs:annotation>
    <xs:annotation>
        <xs:documentation>
           Structural
           This module declares the elements and their attributes used to
           support content structural markup.
           * address, blockcode, blockquote, div, h,
             h1, h2, h3, h4, h5, h6, p, pre, section, separator
      </xs:documentation>
      <xs:documentation source="http://www.w3.org/TR/xhtml2#s_structuralmodule"/>
```

```
  </xs:annotation>
<xs:import namespace="http://www.w3.org/XML/1998/namespace"
           schemaLocation="http://www.w3.org/2001/xml.xsd">
      <xs:annotation>
          <xs:documentation>
        This import brings in the XML namespace attributes
        The module itself does not provide the schemaLocation
        and expects the driver schema to provide the
        actual SchemaLocation.
      </xs:documentation>
      </xs:annotation>
  </xs:import>
  <!-- address -->
  <xs:attributeGroup name="xhtml.address.attlist">
      <xs:attributeGroup ref="xhtml.Common.attrib"/>
  </xs:attributeGroup>
  <xs:group name="xhtml.address.content">
      <xs:sequence>
          <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
  </xs:group>
  <xs:complexType name="xhtml.address.type" mixed="true">
      <xs:group ref="xhtml.address.content"/>
      <xs:attributeGroup ref="xhtml.address.attlist"/>
  </xs:complexType>
  <!-- blockquote -->
  <xs:attributeGroup name="xhtml.blockquote.attlist">
      <xs:attributeGroup ref="xhtml.Common.attrib"/>
  </xs:attributeGroup>
  <xs:group name="xhtml.blockquote.content">
      <xs:choice>
          <xs:group ref="xhtml.Flow.mix" minOccurs="0" maxOccurs="unbounded"/>
      </xs:choice>
  </xs:group>
  <xs:complexType name="xhtml.blockquote.type">
      <xs:group ref="xhtml.blockquote.content"/>
      <xs:attributeGroup ref="xhtml.blockquote.attlist"/>
  </xs:complexType>
  <!-- blockcode -->
  <xs:attributeGroup name="xhtml.blockcode.attlist">
      <xs:attributeGroup ref="xhtml.Common.attrib"/>
  </xs:attributeGroup>
  <xs:group name="xhtml.blockcode.content">
      <xs:choice>
          <xs:group ref="xhtml.Flow.mix" minOccurs="0" maxOccurs="unbounded"/>
      </xs:choice>
  </xs:group>
  <xs:complexType name="xhtml.blockcode.type">
      <xs:group ref="xhtml.blockcode.content"/>
      <xs:attributeGroup ref="xhtml.blockcode.attlist"/>
  </xs:complexType>
  <!-- div -->
  <xs:attributeGroup name="xhtml.div.attlist">
      <xs:attributeGroup ref="xhtml.Common.attrib"/>
  </xs:attributeGroup>
  <xs:group name="xhtml.div.content">
      <xs:choice>
```

```
        <xs:group ref="xhtml.Flow.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:choice>
</xs:group>
<xs:complexType name="xhtml.div.type">
    <xs:group ref="xhtml.div.content"/>
    <xs:attributeGroup ref="xhtml.div.attlist"/>
</xs:complexType>
<!-- h -->
<xs:attributeGroup name="xhtml.h.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.h.content">
    <xs:choice>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:choice>
</xs:group>
<xs:complexType name="xhtml.h.type">
    <xs:group ref="xhtml.h.content"/>
    <xs:attributeGroup ref="xhtml.h.attlist"/>
</xs:complexType>
<!-- Heading Elements  -->
<xs:attributeGroup name="xhtml.heading.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:complexType name="xhtml.heading.type" mixed="true">
    <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    <xs:attributeGroup ref="xhtml.heading.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.h1.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.h1.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.h1.type" mixed="true">
    <xs:group ref="xhtml.h1.content"/>
    <xs:attributeGroup ref="xhtml.h1.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.h2.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.h2.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.h2.type" mixed="true">
    <xs:group ref="xhtml.h2.content"/>
    <xs:attributeGroup ref="xhtml.h2.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.h3.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.h3.content">
    <xs:sequence>
```

```
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.h3.type" mixed="true">
    <xs:group ref="xhtml.h3.content"/>
    <xs:attributeGroup ref="xhtml.h3.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.h4.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.h4.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.h4.type" mixed="true">
    <xs:group ref="xhtml.h4.content"/>
    <xs:attributeGroup ref="xhtml.h4.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.h5.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.h5.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.h5.type" mixed="true">
    <xs:group ref="xhtml.h5.content"/>
    <xs:attributeGroup ref="xhtml.h5.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.h6.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.h6.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.h6.type" mixed="true">
    <xs:group ref="xhtml.h6.content"/>
    <xs:attributeGroup ref="xhtml.h6.attlist"/>
</xs:complexType>
<!-- p -->
<xs:attributeGroup name="xhtml.p.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.p.content">
    <xs:choice>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
        <xs:group ref="xhtml.List.mix" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="blockcode" type="xhtml.blockcode.type"/>
        <xs:element name="blockquote" type="xhtml.blockquote.type"/>
        <xs:element name="pre" type="xhtml.pre.type"/>
        <xs:element name="table" type="xhtml.table.type"/>
    </xs:choice>
</xs:group>
```

```
    <xs:complexType name="xhtml.p.type" mixed="true">
        <xs:group ref="xhtml.p.content"/>
        <xs:attributeGroup ref="xhtml.p.attlist"/>
    </xs:complexType>
    <!-- pre -->
    <xs:attributeGroup name="xhtml.pre.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.pre.content">
        <xs:choice>
            <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
            <xs:group ref="xhtml.List.mix" minOccurs="0" maxOccurs="unbounded"/>
        </xs:choice>
    </xs:group>
    <xs:complexType name="xhtml.pre.type" mixed="true">
        <xs:group ref="xhtml.pre.content"/>
        <xs:attributeGroup ref="xhtml.pre.attlist"/>
    </xs:complexType>
    <!-- section -->
    <xs:attributeGroup name="xhtml.section.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.section.content">
        <xs:sequence>
            <xs:group ref="xhtml.Flow.mix" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.section.type" mixed="true">
        <xs:group ref="xhtml.section.content"/>
        <xs:attributeGroup ref="xhtml.section.attlist"/>
    </xs:complexType>
    <!-- separator -->
    <xs:attributeGroup name="xhtml.separator.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.separator.content">
        <xs:sequence/>
    </xs:group>
    <xs:complexType name="xhtml.separator.type" mixed="true">
        <xs:group ref="xhtml.separator.content"/>
        <xs:attributeGroup ref="xhtml.separator.attlist"/>
    </xs:complexType>
</xs:schema>
```

F.1.4. Text Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
```

```
     This is the XML Schema Text module for XHTML
     $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
</xs:documentation>
    <xs:documentation source="xhtml-copyright-1.xsd"/>
</xs:annotation>
<xs:annotation>
    <xs:documentation>
  Text
  This module declares the elements and their attributes used to
  support textual markup.
  * abbr, cite, code, dfn, em, kbd, l, q, samp, span, strong,
    sub, sup, var
  $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
</xs:documentation>
<xs:documentation source="http://www.w3.org/TR/xhtml2#s_textmodule"/>
</xs:annotation>
<xs:attributeGroup name="xhtml.abbr.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
    <xs:attribute name="full" type="xs:IDREF"/>
</xs:attributeGroup>
<xs:group name="xhtml.abbr.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.abbr.type" mixed="true">
    <xs:group ref="xhtml.abbr.content"/>
    <xs:attributeGroup ref="xhtml.abbr.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.cite.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.cite.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.cite.type" mixed="true">
    <xs:group ref="xhtml.cite.content"/>
    <xs:attributeGroup ref="xhtml.cite.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.code.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.code.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.code.type" mixed="true">
    <xs:group ref="xhtml.code.content"/>
    <xs:attributeGroup ref="xhtml.code.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.dfn.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.dfn.content">
```

```
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.dfn.type" mixed="true">
    <xs:group ref="xhtml.dfn.content"/>
    <xs:attributeGroup ref="xhtml.dfn.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.em.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.em.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.em.type" mixed="true">
    <xs:group ref="xhtml.em.content"/>
    <xs:attributeGroup ref="xhtml.em.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.kbd.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.kbd.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.kbd.type" mixed="true">
    <xs:group ref="xhtml.kbd.content"/>
    <xs:attributeGroup ref="xhtml.kbd.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.l.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.l.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.l.type" mixed="true">
    <xs:group ref="xhtml.kbd.content"/>
    <xs:attributeGroup ref="xhtml.kbd.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.q.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.q.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.q.type" mixed="true">
    <xs:group ref="xhtml.q.content"/>
    <xs:attributeGroup ref="xhtml.q.attlist"/>
</xs:complexType>
<!-- samp -->
```

```
<xs:attributeGroup name="xhtml.samp.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.samp.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.samp.type" mixed="true">
    <xs:group ref="xhtml.samp.content"/>
    <xs:attributeGroup ref="xhtml.samp.attlist"/>
</xs:complexType>
<!-- span -->
<xs:attributeGroup name="xhtml.span.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.span.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.span.type" mixed="true">
    <xs:group ref="xhtml.span.content"/>
    <xs:attributeGroup ref="xhtml.span.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.strong.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.strong.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.strong.type" mixed="true">
    <xs:group ref="xhtml.strong.content"/>
    <xs:attributeGroup ref="xhtml.strong.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.sub.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.sub.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.sub.type" mixed="true">
    <xs:group ref="xhtml.sub.content"/>
    <xs:attributeGroup ref="xhtml.sub.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.sup.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.sup.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
```

```
    <xs:complexType name="xhtml.sup.type" mixed="true">
        <xs:group ref="xhtml.sup.content"/>
        <xs:attributeGroup ref="xhtml.sup.attlist"/>
    </xs:complexType>
    <xs:attributeGroup name="xhtml.var.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.var.content">
        <xs:sequence>
            <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.var.type" mixed="true">
        <xs:group ref="xhtml.var.content"/>
        <xs:attributeGroup ref="xhtml.var.attlist"/>
    </xs:complexType>
</xs:schema>
```

F.1.5. Hypertext Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      Hypertext Module
      This is the XML Schema Hypertext module for XHTML
        * a
      This module declares the anchor ('a') element type, which
      defines the source of a hypertext link. The destination
      (or link 'target') is identified via its 'id' attribute
      rather than the 'name' attribute as was used in HTML.
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_hypertextmodule"/>
    </xs:annotation>
    <xs:attributeGroup name="xhtml.a.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.a.content">
        <xs:sequence>
            <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.a.type" mixed="true">
        <xs:group ref="xhtml.a.content"/>
        <xs:attributeGroup ref="xhtml.a.attlist"/>
    </xs:complexType>
</xs:schema>
```

F.1.6. List Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      List Module
      This is the XML Schema Lists module for XHTML
      List Module Elements
```

```
    * dl, di, dt, dd, label, nl, ol, ul, li
  This module declares the list-oriented element types
  and their attributes.
  $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
</xs:documentation>
    <xs:documentation source="xhtml-copyright-1.xsd"/>
    <xs:documentation source="http://www.w3.org/TR/xhtml2#s_listmodule"/>
</xs:annotation>
<!-- label -->
<xs:attributeGroup name="xhtml.label.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.label.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.label.type" mixed="true">
    <xs:group ref="xhtml.label.content"/>
    <xs:attributeGroup ref="xhtml.label.attlist"/>
</xs:complexType>
<!-- dt -->
<xs:attributeGroup name="xhtml.dt.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.dt.content">
    <xs:sequence>
        <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.dt.type" mixed="true">
    <xs:group ref="xhtml.dt.content"/>
    <xs:attributeGroup ref="xhtml.dt.attlist"/>
</xs:complexType>
<!-- dd -->
<xs:attributeGroup name="xhtml.dd.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.dd.content">
    <xs:sequence>
        <xs:group ref="xhtml.Flow.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.dd.type" mixed="true">
    <xs:group ref="xhtml.dd.content"/>
    <xs:attributeGroup ref="xhtml.dd.attlist"/>
</xs:complexType>
<!-- di -->
<xs:attributeGroup name="xhtml.di.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.di.content">
    <xs:sequence>
        <xs:element name="dt" type="xhtml.dt.type" minOccurs="1" maxOccurs="unbounded"/>
        <xs:element name="dd" type="xhtml.dd.type" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.di.type" mixed="true">
    <xs:group ref="xhtml.di.content"/>
    <xs:attributeGroup ref="xhtml.di.attlist"/>
</xs:complexType>
<!-- dl -->
<xs:attributeGroup name="xhtml.dl.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.dl.content">
```

```
    <xs:sequence>
        <xs:element name="label" type="xhtml.label.type" minOccurs="0" maxOccurs="1"/>
        <xs:choice maxOccurs="unbounded">
            <xs:choice minOccurs="1" maxOccurs="unbounded">
                <xs:element name="dt" type="xhtml.dt.type"/>
                <xs:element name="dd" type="xhtml.dd.type"/>
            </xs:choice>
            <xs:element name="di" type="xhtml.di.type" minOccurs="1" maxOccurs="unbounded" />
        </xs:choice>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.dl.type">
    <xs:group ref="xhtml.dl.content"/>
    <xs:attributeGroup ref="xhtml.dl.attlist"/>
</xs:complexType>
<!-- li -->
<xs:attributeGroup name="xhtml.li.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.li.content">
    <xs:sequence>
        <xs:group ref="xhtml.Flow.mix" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.li.type" mixed="true">
    <xs:group ref="xhtml.li.content"/>
    <xs:attributeGroup ref="xhtml.li.attlist"/>
</xs:complexType>
<!-- nl -->
<xs:attributeGroup name="xhtml.nl.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.nl.content">
    <xs:sequence>
        <xs:element name="label" type="xhtml.label.type" minOccurs="1" maxOccurs="1"/>
        <xs:element name="li" type="xhtml.li.type" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.nl.type">
    <xs:group ref="xhtml.nl.content"/>
    <xs:attributeGroup ref="xhtml.nl.attlist"/>
</xs:complexType>
<!-- ol -->
<xs:attributeGroup name="xhtml.ol.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.ol.content">
    <xs:sequence>
        <xs:element name="label" type="xhtml.label.type" minOccurs="0" maxOccurs="1"/>
        <xs:element name="li" type="xhtml.li.type" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.ol.type">
    <xs:group ref="xhtml.ol.content"/>
    <xs:attributeGroup ref="xhtml.ol.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xhtml.ul.attlist">
    <xs:attributeGroup ref="xhtml.Common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.ul.content">
    <xs:sequence>
        <xs:element name="label" type="xhtml.label.type" minOccurs="0" maxOccurs="1"/>
        <xs:element name="li" type="xhtml.li.type" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.ul.type">
```

```
        <xs:group ref="xhtml.ul.content"/>
        <xs:attributeGroup ref="xhtml.ul.attlist"/>
    </xs:complexType>
</xs:schema>
```

F.1.7. Core Attributes Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      This is the XML Schema core attributes module for XHTML
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/xhtml2/mod-core.html"/>
    </xs:annotation>
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
             schemaLocation="http://www.w3.org/2001/xml.xsd">
        <xs:annotation>
            <xs:documentation>
        This import brings in the XML namespace attributes
        The module itself does not provide the schemaLocation
        and expects the driver schema to provide the
        actual SchemaLocation.
      </xs:documentation>
        </xs:annotation>
    </xs:import>
    <xs:attributeGroup name="xhtml.id">
        <xs:attribute name="id" type="xs:ID"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="xhtml.class">
        <xs:attribute name="class" type="xs:NMTOKENS"/>
    </xs:attributeGroup>
    <xs:simpleType name="xhtml.layout.Datatype">
        <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="relevant"/>
            <xs:enumeration value="irrelevant"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:attributeGroup name="xhtml.layout">
        <xs:attribute name="layout" type="xhtml.layout.Datatype" default="irrelevant"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="xhtml.title">
        <xs:attribute name="title" type="xs:string"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="xhtml.Core.attrib">
        <!-- <xs:attribute ref="xml:space" fixed="preserve"/> -->
        <xs:attributeGroup ref="xhtml.id"/>
        <xs:attributeGroup ref="xhtml.class"/>
        <xs:attributeGroup ref="xhtml.layout"/>
        <xs:attributeGroup ref="xhtml.title"/>
        <xs:attributeGroup ref="xhtml.Core.extra.attrib"/>
```

```
    </xs:attributeGroup>
    <!-- Global attributes -->
    <xs:attribute name="id" type="xs:ID"/>
    <xs:attribute name="class" type="xs:NMTOKENS"/>
    <xs:attribute name="title" type="xs:string"/>
    <xs:attribute name="layout" type="xhtml.layout.Datatype"/>
    <xs:attributeGroup name="xhtml.Global.Core.attrib">
        <xs:attribute ref="id"/>
        <xs:attribute ref="class"/>
        <xs:attribute ref="layout"/>
        <xs:attribute ref="title"/>
        <xs:attributeGroup ref="xhtml.Global.Core.extra.attrib" />
    </xs:attributeGroup>
</xs:schema>
```

F.1.8. Hypertext Attributes Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      Hypertext Attributes Module
      This is the XML Schema Hypertext Attributes module for XHTML
      * cite, href, hreflang, hrefmedia, hreftype, nextfocus, prevfocus,
        target, and xml:base
        This module declares attributes and the Hypertext attribute
        collection.
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_hypertextmodule"/>
    </xs:annotation>
    <xs:attributeGroup name="xhtml.Hypertext.attrib">
        <xs:attribute name="cite" type="xh11d:URI"/>
        <xs:attribute name="href" type="xh11d:URI"/>
        <xs:attribute name="hreflang" type="xh11d:LanguageCodes"/>
        <xs:attribute name="hrefmedia" type="xh11d:MediaDesc"/>
        <xs:attribute name="hreftype" type="xh11d:ContentTypes"/>
        <xs:attribute name="nextfocus" type="xs:IDREF"/>
        <xs:attribute name="prevfocus" type="xs:IDREF"/>
        <xs:attribute name="target" type="xh11d:HrefTarget"/>
        <xs:attributeGroup ref="xhtml.Hypertext.extra.attrib" />
    </xs:attributeGroup>
    <!-- global attributes -->
    <xs:attribute name="cite" type="xh11d:URI"/>
    <xs:attribute name="href" type="xh11d:URI"/>
    <xs:attribute name="hreflang" type="xh11d:LanguageCodes"/>
    <xs:attribute name="hrefmedia" type="xh11d:MediaDesc"/>
    <xs:attribute name="hreftype" type="xh11d:ContentTypes"/>
    <xs:attribute name="nextfocus" type="xs:IDREF"/>
    <xs:attribute name="prevfocus" type="xs:IDREF"/>
    <xs:attribute name="target" type="xh11d:HrefTarget"/>
    <xs:attributeGroup name="xhtml.Global.Hypertext.attrib">
        <xs:attribute ref="cite" />
        <xs:attribute ref="href" />
        <xs:attribute ref="hreflang" />
        <xs:attribute ref="hrefmedia" />
        <xs:attribute ref="hreftype" />
        <xs:attribute ref="nextfocus" />
        <xs:attribute ref="prevfocus" />
        <xs:attribute ref="target" />
        <xs:attributeGroup ref="xhtml.Global.Hypertext.extra.attrib" />
    </xs:attributeGroup>
</xs:schema>
```

F.1.9. I18N Attribute Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      This is the XML Schema I18N attributes module for XHTML
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/xhtml2/mod-i18n.html"/>
    </xs:annotation>
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
              schemaLocation="http://www.w3.org/2001/xml.xsd">
        <xs:annotation>
            <xs:documentation>
        This import brings in the XML namespace attributes
        The module itself does not provide the schemaLocation
        and expects the driver schema to provide the
        actual SchemaLocation.
      </xs:documentation>
        </xs:annotation>
    </xs:import>
    <xs:attributeGroup name="xhtml.I18n.attrib">
        <xs:attribute ref="xml:lang" />
        <xs:attributeGroup ref="xhtml.I18n.extra.attrib"/>
    </xs:attributeGroup>
</xs:schema>
```

F.2. Optional Modules

F.2.1. Bi-directional Text Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      This is the XML Schema Bidirectional Text attribute module for XHTML
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/xhtml2/mod-bidi.html"/>
    </xs:annotation>
```

```
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
             schemaLocation="http://www.w3.org/2001/xml.xsd">
        <xs:annotation>
            <xs:documentation>
        This import brings in the XML namespace attributes
        The module itself does not provide the schemaLocation
        and expects the driver schema to provide the
        actual SchemaLocation.
       </xs:documentation>
        </xs:annotation>
  </xs:import>
    <xs:simpleType name="xhtml.Direction.datatype">
        <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="ltr"/>
            <xs:enumeration value="rtl"/>
            <xs:enumeration value="lro"/>
            <xs:enumeration value="rlo"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:attributeGroup name="xhtml.Bidi.attrib">
        <xs:attribute name="dir" default="ltr" type="xhtml.Direction.datatype"/>
    </xs:attributeGroup>
</xs:schema>
```

F.2.2. Caption Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
          This is the XML Schema Caption module for XHTML
          $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
        </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
    </xs:annotation>
    <xs:annotation>
        <xs:documentation>
           * caption
           This module declares the caption element.
        </xs:documentation>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_tablemodule"/>
    </xs:annotation>
    <xs:attributeGroup name="xhtml.caption.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.caption.content">
        <xs:sequence>
            <xs:group ref="xhtml.Flow.mix" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.caption.type" mixed="true">
        <xs:group ref="xhtml.caption.content"/>
        <xs:attributeGroup ref="xhtml.caption.attlist"/>
    </xs:complexType>
</xs:schema>
```

F.2.3. Edit Attributes Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
```

```
            schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
          Editing Elements
          This is the XML Schema Editing Markup module for XHTML
           * edit, datetime
         This module declares attributes used to indicate
         inserted and deleted content while editing a document.
          $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
        </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_editmodule"/>
    </xs:annotation>
    <xs:simpleType name="xhtml.Edit.datatype">
        <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="inserted"/>
            <xs:enumeration value="deleted"/>
            <xs:enumeration value="changed"/>
            <xs:enumeration value="moved"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:attributeGroup name="xhtml.Edit.attrib">
        <xs:attribute name="edit" default="changed" type="xhtml.Edit.datatype"/>
        <xs:attribute name="datetime" type="xh11d:Datetime"/>
        <xs:attributeGroup ref="xhtml.Edit.extra.attrib"/>
    </xs:attributeGroup>
    <xs:attribute name="edit" default="changed" type="xhtml.Edit.datatype"/>
    <xs:attribute name="datetime" type="xh11d:Datetime"/>
    <xs:attributeGroup name="xhtml.Global.Edit.attrib">
        <xs:attribute ref="edit"/>
        <xs:attribute ref="datetime"/>
        <xs:attributeGroup ref="xhtml.Global.Edit.extra.attrib"/>
    </xs:attributeGroup>
</xs:schema>
```

F.2.4. Embedding Attributes Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      Embedding Attributes Module
      This is the XML Schema Embedding Attributes module for XHTML
      * encoding, src, srctype
        This module declares attributes and the Embedding attribute
        collection.
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/xhtml2/mod-embedding.html"/>
    </xs:annotation>
    <xs:attributeGroup name="xhtml.Embedding.attrib">
        <xs:attribute name="encoding" type="xh11d:Encodings"/>
        <xs:attribute name="src" type="xh11d:URI"/>
        <xs:attribute name="srctype" type="xh11d:ContentTypes"/>
        <xs:attributeGroup ref="xhtml.Embedding.extra.attrib" />
    </xs:attributeGroup>
    <!-- global attributes -->
    <xs:attribute name="encoding" type="xh11d:Encodings"/>
    <xs:attribute name="src" type="xh11d:URI"/>
    <xs:attribute name="srctype" type="xh11d:ContentTypes"/>
    <xs:attributeGroup name="xhtml.Global.Embedding.attrib">
        <xs:attribute ref="encoding"/>
```

```
        <xs:attribute ref="src"/>
        <xs:attribute ref="srctype"/>
        <xs:attributeGroup ref="xhtml.Global.Embedding.extra.attrib" />
    </xs:attributeGroup>
</xs:schema>
```

F.2.5. Image Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      Images
      This is the XML Schema Images module for XHTML
        * img
      This module provides markup to support basic image embedding.
      To avoid problems with text-only UAs as well as to make
      image content understandable and navigable to users of
      non-visual UAs, you need to provide a description with
      the 'alt' attribute, and avoid server-side image maps.
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_imagemodule"/>
    </xs:annotation>
    <xs:attributeGroup name="xhtml.img.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.img.content">
        <xs:sequence>
            <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.img.type">
        <xs:group ref="xhtml.img.content"/>
        <xs:attributeGroup ref="xhtml.img.attlist"/>
    </xs:complexType>
</xs:schema>
```

F.2.6. Image Map Attributes Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      Client-side Image Maps
      This is the XML Schema Client-side Image Maps module for XHTML
        * area, map
      This module declares elements and attributes to support client-side
      image maps.
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_imapmodule"/>
    </xs:annotation>
    <xs:simpleType name="xhtml.Shape.Datatype">
        <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="rect"/>
            <xs:enumeration value="circle"/>
            <xs:enumeration value="poly"/>
            <xs:enumeration value="default"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="xhtml.Coords.Datatype">
        <xs:restriction base="xh11d:Text"/>
    </xs:simpleType>
    <xs:attributeGroup name="xhtml.ImageMap.attrib">
        <xs:attribute name="usemap" type="xh11d:URI"/>
        <xs:attribute name="ismap" type="xs:NMTOKEN" fixed="ismap"/>
```

```
        <xs:attribute name="shape" type="xhtml.Shape.Datatype" default="rect"/>
        <xs:attribute name="coords" type="xhtml.Coords.Datatype"/>
    </xs:attributeGroup>
</xs:schema>
```

F.2.7. Media Attribute Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      This is the XML Schema core attributes module for XHTML
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/xhtml2/mod-core.html"/>
    </xs:annotation>
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
              schemaLocation="http://www.w3.org/2001/xml.xsd">
        <xs:annotation>
            <xs:documentation>
        This import brings in the XML namespace attributes
        The module itself does not provide the schemaLocation
        and expects the driver schema to provide the
        actual SchemaLocation.
      </xs:documentation>
        </xs:annotation>
    </xs:import>
    <xs:attributeGroup name="xhtml.Media.attrib">
        <xs:attribute name="media" type="xh11d:MediaDesc"/>
    </xs:attributeGroup>
    <!-- Global attributes -->
    <xs:attribute name="media" type="xh11d:MediaDesc"/>
    <xs:attributeGroup name="xhtml.Global.Media.attrib">
        <xs:attribute ref="media"/>
        <xs:attributeGroup ref="xhtml.Global.Media.extra.attrib" />
    </xs:attributeGroup>
</xs:schema>
```

F.2.8. Metainformation Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      This is the XML Schema Metainformation module for XHTML
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
    </xs:annotation>
    <xs:annotation>
```

```
      <xs:documentation>
     Meta Information
       * meta
     This module declares the meta element type and its attributes,
     used to provide declarative document metainformation.
    </xs:documentation>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_metamodule"/>
    </xs:annotation>
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd">
        <xs:annotation>
            <xs:documentation>
                This import brings in the XML namespace attributes
                The module itself does not provide the schemaLocation
                and expects the driver schema to provide the
                actual SchemaLocation.
            </xs:documentation>
        </xs:annotation>
    </xs:import>
    <xs:attributeGroup name="xhtml.meta.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.meta.content">
        <xs:sequence>
            <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.meta.type">
        <xs:group ref="xhtml.meta.content"/>
        <xs:attributeGroup ref="xhtml.meta.attlist"/>
    </xs:complexType>
    <xs:attributeGroup name="xhtml.link.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.link.content">
        <xs:choice>
            <xs:element name="link" type="xhtml.link.type"/>
            <xs:element name="meta" type="xhtml.meta.type"/>
        </xs:choice>
    </xs:group>
    <xs:complexType name="xhtml.link.type">
        <xs:group ref="xhtml.link.content"/>
        <xs:attributeGroup ref="xhtml.link.attlist"/>
    </xs:complexType>
</xs:schema>
```

F.2.9. Metainformation Attributes Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
    elementFormDefault="qualified"
>
  <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
             schemaLocation="xhtml-datatypes-2.xsd" />
  <xs:annotation>
    <xs:documentation>
      This is the XML Schema Metainformation Attributes module for XHTML
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
    <xs:documentation source="xhtml-rdfa-copyright-1.xsd"/>
  </xs:annotation>
  <xs:annotation>
    <xs:documentation>
      XHTML Metainformation Attributes
    </xs:documentation>
  </xs:annotation>
  <xs:attribute name="about" type="xh11d:URIorSafeCURIE"/>
  <xs:attribute name="content" type="xh11d:CDATA"/>
  <xs:attribute name="datatype" type="xh11d:CURIE"/>
  <xs:attribute name="typeof" type="xh11d:CURIEs"/>
  <xs:attribute name="property" type="xh11d:CURIEs"/>
```

```
  <xs:attribute name="rel" type="xh11d:CURIEs"/>
  <xs:attribute name="resource" type="xh11d:URIorSafeCURIE"/>
  <xs:attribute name="rev" type="xh11d:CURIEs"/>
  <xs:attributeGroup name="xhtml.MetaAttributes.attrib">
    <xs:attribute name="about"/>
    <xs:attribute name="content"/>
    <xs:attribute name="datatype"/>
    <xs:attribute name="typeof"/>
    <xs:attribute name="property"/>
    <xs:attribute name="rel"/>
    <xs:attribute name="resource"/>
    <xs:attribute name="rev"/>
  </xs:attributeGroup>
</xs:schema>
```

F.2.10. Object Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
        <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
        <xs:annotation>
                <xs:documentation>
      This is the XML Schema Object module for XHTML
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
                <xs:documentation source="xhtml-copyright-1.xsd"/>
        </xs:annotation>
        <xs:annotation>
                <xs:documentation>
      This module declares the object element type and its attributes,
      used to embed external objects as part of XHTML pages. In the
      document, place param elements prior to the object elements
      that require their content.
      Note that use of this module requires instantiation of the
      Param Element Module prior to this module.
      Elements defined here:
        * object (param)
    </xs:documentation>
                <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_objectmodule"/>
        </xs:annotation>
        <xs:include schemaLocation="xhtml-param-2.xsd">
                <xs:annotation>
                        <xs:documentation>
        Param module
        Elements defined here:
          * param
      </xs:documentation>
                </xs:annotation>
    </xs:include>
    <xs:include schemaLocation="xhtml-caption-2.xsd">
        <xs:annotation>
            <xs:documentation>
                Caption module - defines the caption element
            </xs:documentation>
        </xs:annotation>
    </xs:include>
    <!-- standby -->
    <xs:attributeGroup name="xhtml.standby.attlist">
                <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.standby.content">
        <xs:sequence>
            <xs:group ref="xhtml.Text.mix" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.standby.type" mixed="true">
        <xs:group ref="xhtml.standby.content"/>
        <xs:attributeGroup ref="xhtml.standby.attlist"/>
    </xs:complexType>
    <!-- object -->
        <xs:attributeGroup name="xhtml.object.attlist">
                <xs:attributeGroup ref="xhtml.Common.attrib"/>
                <xs:attribute name="declare">
                        <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="declare"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
```

```
                <xs:attribute name="archive" type="xh11d:URIs"/>
                <xs:attribute name="content-length" type="xh11d:Number"/>
        </xs:attributeGroup>
        <xs:group name="xhtml.object.content">
                <xs:sequence>
            <xs:element name="caption" type="xhtml.caption.type"
                minOccurs="0" maxOccurs="1"/>
            <xs:element name="standby" type="xhtml.standby.type"
                minOccurs="0" maxOccurs="1"/>
                        <xs:choice minOccurs="0" maxOccurs="unbounded">
                                <xs:element name="param" type="xhtml.param.type"/>
                                <xs:group ref="xhtml.Flow.mix"/>
                        </xs:choice>
                </xs:sequence>
        </xs:group>
        <xs:complexType name="xhtml.object.type" mixed="true">
                <xs:group ref="xhtml.object.content"/>
                <xs:attributeGroup ref="xhtml.object.attlist"/>
        </xs:complexType>
</xs:schema>
```

F.2.10.1. Param Attribute

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      This is the XML Schema Param Element module for XHTML
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
      </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
    </xs:annotation>
    <xs:annotation>
        <xs:documentation>
      Parameters for Java Applets and Embedded Objects
        * param
      This module provides declarations for the param element,
      used to provide named property values for the applet
      and object elements.
    </xs:documentation>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_objectmodule"/>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_appletmodule"/>
    </xs:annotation>
    <xs:attributeGroup name="xhtml.param.attlist">
        <xs:attributeGroup ref="xhtml.id"/>
        <xs:attribute name="name" type="xh11d:CDATA" use="required"/>
        <xs:attribute name="value" type="xh11d:CDATA"/>
        <xs:attribute name="valuetype" default="data">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="data"/>
                    <xs:enumeration value="ref"/>
                    <xs:enumeration value="object"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="type" type="xh11d:ContentType"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.param.content">
        <xs:sequence/>
    </xs:group>
    <xs:complexType name="xhtml.param.type">
        <xs:group ref="xhtml.param.content"/>
        <xs:attributeGroup ref="xhtml.param.attlist"/>
    </xs:complexType>
</xs:schema>
```

F.2.11. Style Sheet Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      This is the XML Schema Stylesheets module for XHTML
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
```

```
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
    </xs:annotation>
    <xs:annotation>
        <xs:documentation>
      Stylesheets
        * style
      This module declares the style element type and its attributes,
      used to embed stylesheet information in the document head element.
    </xs:documentation>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_stylemodule"/>
    </xs:annotation>
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
             schemaLocation="http://www.w3.org/2001/xml.xsd">
        <xs:annotation>
            <xs:documentation>
          This import brings in the XML namespace attributes
          The module itself does not provide the schemaLocation
          and expects the driver schema to provide the
          actual SchemaLocation.
        </xs:documentation>
        </xs:annotation>
    </xs:import>
    <xs:attributeGroup name="xhtml.style.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
        <xs:attribute name="disabled" type="xs:NMTOKEN" fixed="disabled"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.style.content">
        <xs:sequence/>
    </xs:group>
    <xs:complexType name="xhtml.style.type" mixed="true">
        <xs:group ref="xhtml.style.content"/>
        <xs:attributeGroup ref="xhtml.style.attlist"/>
    </xs:complexType>
</xs:schema>
```

F.2.12. Style Attribute Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      Inline Style module
      This is the XML Schema Inline Style module for XHTML
         * styloe attribute
      This module declares the 'style' attribute, used to support inline
      style markup.
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_styleattributemodule"/>
    </xs:annotation>
    <xs:attributeGroup name="xhtml.Style.attrib">
        <xs:attribute name="style" type="xh11d:CDATA"/>
    </xs:attributeGroup>
</xs:schema>
```

F.2.13. Tables Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
          This is the XML Schema Tables module for XHTML
          $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
        </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
    </xs:annotation>
    <xs:annotation>
        <xs:documentation>
          Tables
           * table, caption, thead, tfoot, tbody, colgroup, col, tr, th, td
          This module declares element types and attributes used to provide
          table markup similar to HTML 4.0, including features that enable
```

```
          better accessibility for non-visual user agents.
        </xs:documentation>
        <xs:documentation source="http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html#s_tablemodule"/>
    </xs:annotation>
    <xs:include schemaLocation="xhtml-caption-2.xsd"/>
    <xs:attributeGroup name="xhtml.scope.attrib">
        <xs:attribute name="scope">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="row"/>
                    <xs:enumeration value="col"/>
                    <xs:enumeration value="rowgroup"/>
                    <xs:enumeration value="colgroup"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:attributeGroup>
    <xs:attributeGroup name="xhtml.td.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
        <xs:attribute name="abbr" type="xh11d:Text"/>
        <xs:attribute name="axis" type="xh11d:CDATA"/>
        <xs:attribute name="colspan" type="xh11d:Number" default="1"/>
        <xs:attribute name="headers" type="xs:IDREFS"/>
        <xs:attribute name="rowspan" type="xh11d:Number" default="1"/>
        <xs:attributeGroup ref="xhtml.scope.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.td.content">
        <xs:sequence>
            <xs:group ref="xhtml.Flow.mix" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.td.type" mixed="true">
        <xs:group ref="xhtml.td.content"/>
        <xs:attributeGroup ref="xhtml.td.attlist"/>
    </xs:complexType>
    <xs:attributeGroup name="xhtml.th.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
        <xs:attribute name="abbr" type="xh11d:Text"/>
        <xs:attribute name="axis" type="xh11d:CDATA"/>
        <xs:attribute name="colspan" type="xh11d:Number" default="1"/>
        <xs:attribute name="headers" type="xs:IDREFS"/>
        <xs:attribute name="rowspan" type="xh11d:Number" default="1"/>
        <xs:attributeGroup ref="xhtml.scope.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.th.content">
        <xs:sequence>
            <xs:group ref="xhtml.Flow.mix" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.th.type" mixed="true">
        <xs:group ref="xhtml.th.content"/>
        <xs:attributeGroup ref="xhtml.th.attlist"/>
    </xs:complexType>
    <xs:attributeGroup name="xhtml.tr.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
<!--        <xs:attributeGroup ref="xhtml.XFormsRepeat.attrib"/> -->
    </xs:attributeGroup>
    <xs:group name="xhtml.tr.content">
        <xs:sequence>
            <xs:choice maxOccurs="unbounded">
                <xs:element name="th" type="xhtml.th.type"/>
                <xs:element name="td" type="xhtml.td.type"/>
            </xs:choice>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.tr.type">
        <xs:group ref="xhtml.tr.content"/>
        <xs:attributeGroup ref="xhtml.tr.attlist"/>
    </xs:complexType>
    <xs:attributeGroup name="xhtml.col.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
        <xs:attribute name="span" type="xh11d:Number" default="1"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.col.content">
        <xs:sequence/>
    </xs:group>
    <xs:complexType name="xhtml.col.type">
        <xs:group ref="xhtml.col.content"/>
        <xs:attributeGroup ref="xhtml.col.attlist"/>
    </xs:complexType>
    <xs:attributeGroup name="xhtml.colgroup.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
        <xs:attribute name="span" type="xh11d:Number" default="1"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.colgroup.content">
        <xs:sequence>
            <xs:element name="col" type="xhtml.col.type" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.colgroup.type">
```

```
        <xs:group ref="xhtml.colgroup.content"/>
        <xs:attributeGroup ref="xhtml.colgroup.attlist"/>
    </xs:complexType>
    <xs:attributeGroup name="xhtml.tbody.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.tbody.content">
        <xs:sequence>
            <xs:element name="tr" type="xhtml.tr.type" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.tbody.type">
        <xs:group ref="xhtml.tbody.content"/>
        <xs:attributeGroup ref="xhtml.tbody.attlist"/>
    </xs:complexType>
    <xs:attributeGroup name="xhtml.tfoot.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.tfoot.content">
        <xs:sequence>
            <xs:element name="tr" type="xhtml.tr.type" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.tfoot.type">
        <xs:group ref="xhtml.tfoot.content"/>
        <xs:attributeGroup ref="xhtml.tfoot.attlist"/>
    </xs:complexType>
    <xs:attributeGroup name="xhtml.thead.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.thead.content">
        <xs:sequence>
            <xs:element name="tr" type="xhtml.tr.type" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.thead.type">
        <xs:group ref="xhtml.thead.content"/>
        <xs:attributeGroup ref="xhtml.thead.attlist"/>
    </xs:complexType>
    <xs:attributeGroup name="xhtml.summary.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.summary.content">
        <xs:sequence>
            <xs:group ref="xhtml.Flow.mix" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.summary.type" mixed="true">
        <xs:group ref="xhtml.summary.content"/>
        <xs:attributeGroup ref="xhtml.summary.attlist"/>
    </xs:complexType>
    <xs:attributeGroup name="xhtml.table.attlist">
        <xs:attributeGroup ref="xhtml.Common.attrib"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.table.content">
        <xs:sequence>
            <xs:element name="caption" type="xhtml.caption.type" minOccurs="0"/>
            <xs:element name="summary" type="xhtml.summary.type" minOccurs="0"/>
            <xs:choice>
                <xs:element name="col" type="xhtml.col.type" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="colgroup" type="xhtml.colgroup.type" minOccurs="0" maxOccurs="unbounded"/>
            </xs:choice>
            <xs:choice>
                <xs:sequence>
                    <xs:element name="thead" type="xhtml.thead.type" minOccurs="0"/>
                    <xs:element name="tfoot" type="xhtml.tfoot.type" minOccurs="0"/>
                    <xs:element name="tbody" type="xhtml.tbody.type" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:choice>
                    <xs:element name="tr" type="xhtml.tr.type" maxOccurs="unbounded"/>
                </xs:choice>
            </xs:choice>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="xhtml.table.type">
        <xs:group ref="xhtml.table.content"/>
        <xs:attributeGroup ref="xhtml.table.attlist"/>
    </xs:complexType>
</xs:schema>
```

F.3. Modules from Other Specifications

F.3.1. Access Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      This is the XML Schema module for XHTML Access
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/xhtml-role#A_role"/>
    </xs:annotation>
    <xs:attributeGroup name="xhtmlAccess.Common.attrib">
        <xs:attribute name="id" type="xs:ID"/>
        <xs:attribute name="media" type="xh11d:MediaDesc"/>
        <xs:anyAttribute/>
    </xs:attributeGroup>
    <xs:attributeGroup name="xhtml.access.attlist">
        <xs:attribute name="activate" default="no">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="yes"/>
                    <xs:enumeration value="no"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="key" type="xh11d:Character"/>
        <xs:attribute name="order" default="document">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="document"/>
                    <xs:enumeration value="list"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="targetid">
            <xs:simpleType>
                <xs:list itemType="xs:IDREF"/>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="targetrole" type="xh11d:CURIEs"/>
    </xs:attributeGroup>
    <xs:group name="xhtml.access.content">
        <xs:sequence/>
    </xs:group>
    <xs:complexType name="xhtml.access.type">
```

```
        <xs:group ref="xhtml.access.content"/>
        <xs:attributeGroup ref="xhtml.access.attlist"/>
    </xs:complexType>
</xs:schema>
```

F.3.2. Role Attribute Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:annotation>
        <xs:documentation>
      This is the XML Schema attribute module for XHTML Role
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
        <xs:documentation source="xhtml-copyright-1.xsd"/>
        <xs:documentation source="http://www.w3.org/TR/xhtml-role#A_role"/>
    </xs:annotation>
    <xs:attributeGroup name="xhtml.Role.attrib">
        <xs:attribute name="role" type="xh11d:CURIEs"/>
    </xs:attributeGroup>
</xs:schema>
```

F.3.3. Ruby Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
>
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
  <xs:annotation>
    <xs:documentation>
      This is the Ruby module for XHTML
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
    </xs:documentation>
    <xs:documentation source="xhtml-copyright-1.xsd"/>
  </xs:annotation>
  <xs:annotation>
    <xs:documentation>
      "Ruby" are short runs of text alongside the base text, typically
      used in East Asian documents to indicate pronunciation or to
      provide a short annotation. The full specification for Ruby is here:
        http://www.w3.org/TR/2001/REC-ruby-20010531/
      This module defines "Ruby " or "complex Ruby" as described
      in the specification:
        http://www.w3.org/TR/2001/REC-ruby-20010531/#complex
      Simple or Basic Ruby are defined in a separate module.
      This module declares the elements and their attributes used to
```

```
    support complex ruby annotation markup. Elements defined here
          * ruby, rbc, rtc, rb, rt, rp
    This module expects the document model to define the
    following content models
      + xhtml.TextNoRuby.class
  </xs:documentation>
  <xs:documentation
       source="http://www.w3.org/TR/2001/REC-ruby-20010531/"/>
</xs:annotation>
<xs:group name="xhtml.ruby.content.simple">
    <xs:sequence>
        <xs:element name="rb" type="xhtml.rb.type"/>
        <xs:choice>
            <xs:element name="rt" type="xhtml.rt.type"/>
            <xs:sequence>
                <xs:element name="rp" type="xhtml.rp.type"/>
                <xs:element name="rt" type="xhtml.rt.type"/>
                <xs:element name="rp" type="xhtml.rp.type"/>
            </xs:sequence>
        </xs:choice>
    </xs:sequence>
</xs:group>
<xs:group name="xhtml.ruby.content.complex">
  <xs:sequence>
    <xs:element name="rbc" type="xhtml.rbc.type"/>
    <xs:element name="rtc" type="xhtml.rtc.type" maxOccurs="2"/>
  </xs:sequence>
</xs:group>
<!--
 add to this group any common attributes for all Ruby elements
-->
<xs:attributeGroup name="xhtml.ruby.common.attrib"/>
<xs:group name="xhtml.ruby.content">
  <xs:choice>
    <xs:group ref="xhtml.ruby.content.simple"/>
    <xs:group ref="xhtml.ruby.content.complex"/>
  </xs:choice>
</xs:group>
<xs:complexType name="xhtml.ruby.type">
  <xs:group ref="xhtml.ruby.content"/>
  <xs:attributeGroup ref="xhtml.ruby.common.attrib"/>
</xs:complexType>
<!--
 rbc (ruby base component) element
-->
<xs:attributeGroup name="xhtml.rbc.attlist">
  <xs:attributeGroup ref="xhtml.ruby.common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.rbc.content">
  <xs:sequence>
    <xs:element name="rb" type="xhtml.rb.type" />
  </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.rbc.type">
  <xs:group ref="xhtml.rbc.content"/>
  <xs:attributeGroup ref="xhtml.rbc.attlist"/>
</xs:complexType>
```

```
<!--
 rtc (ruby text component) element
-->
<xs:attributeGroup name="xhtml.rtc.attlist">
  <xs:attributeGroup ref="xhtml.ruby.common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.rtc.content">
  <xs:sequence>
    <xs:element name="rt" type="xhtml.rt.type" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.rtc.type">
  <xs:group ref="xhtml.rt.content"/>
  <xs:attributeGroup ref="xhtml.rtc.attlist"/>
</xs:complexType>
<!--
 rb (ruby base) element
-->
<xs:attributeGroup name="xhtml.rb.attlist">
  <xs:attributeGroup ref="xhtml.ruby.common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.rb.content">
  <xs:sequence>
     <xs:group ref="xhtml.TextNoRuby.class" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.rb.type" mixed="true">
  <xs:group ref="xhtml.rb.content"/>
  <xs:attributeGroup ref="xhtml.rb.attlist"/>
</xs:complexType>
<!--
 rt (ruby text) element
-->
<xs:attributeGroup name="xhtml.rt.attlist">
  <xs:attributeGroup ref="xhtml.ruby.common.attrib"/>
  <xs:attribute name="rbspan" type="xh11d:Number" default="1"/>
</xs:attributeGroup>
<xs:group name="xhtml.rt.content">
  <xs:sequence>
     <xs:group ref="xhtml.TextNoRuby.class" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:group>
<xs:complexType name="xhtml.rt.type" mixed="true">
  <xs:group ref="xhtml.rt.content"/>
  <xs:attributeGroup ref="xhtml.rt.attlist"/>
</xs:complexType>
<!-- rp (ruby parenthesis) element  -->
<xs:attributeGroup name="xhtml.rp.attlist">
  <xs:attributeGroup ref="xhtml.ruby.common.attrib"/>
</xs:attributeGroup>
<xs:group name="xhtml.rp.content">
  <xs:sequence/>
</xs:group>
<xs:complexType name="xhtml.rp.type" mixed="true">
```

```
    <xs:group ref="xhtml.rp.content"/>
    <xs:attributeGroup ref="xhtml.rp.attlist"/>
  </xs:complexType>
</xs:schema>
```

F.3.4. XForms Modules

```
Module xforms-2.xsd not found!
```

F.3.5. XML Events Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
    xsi:schemaLocation="http://www.w3.org/2001/XMLSchema
                        http://www.w3.org/2001/XMLSchema.xsd"
    elementFormDefault="unqualified"
    blockDefault="#all"
    finalDefault="#all"
    attributeFormDefault="unqualified">
  <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
    schemaLocation="xhtml-datatypes-2.xsd" />
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
      schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:annotation>
    <xs:documentation>
      This is the XML Schema for XML Events
      URI: http://www.w3.org/MarkUp/SCHEMA/xml-events-2.xsd
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
      Note that this module requires the definition of
      xmlEvents.Common.attrib to be a collection of attributes to
      add to each element.  This collection MUST contain an 'id'
      attribute.
    </xs:documentation>
    <xs:documentation source="xml-events-copyright-2.xsd"/>
  </xs:annotation>
  <xs:annotation>
    <xs:documentation>
      XML Events element listener
        This module defines the listener element for XML Events.
        This element can be used to define event listeners. This
        module relies upon the XmlEvents.attlist attribute group
        defined in xml-events-attribs-2.xsd.
    </xs:documentation>
  </xs:annotation>
  <xs:attributeGroup name="xmlEvents.Common.attrib">
    <xs:attribute name="id" type="xs:ID"/>
    <xs:anyAttribute/>
  </xs:attributeGroup>
  <xs:attributeGroup name="xmlEvents.listener.attlist">
    <xs:attributeGroup ref="xmlEvents.attrib" />
    <xs:attributeGroup ref="xmlEvents.Common.attrib"/>
  </xs:attributeGroup>
  <xs:complexType name="xmlEvents.listener.type">
```

```
    <xs:attributeGroup ref="xmlEvents.listener.attlist"/>
  </xs:complexType>
  <xs:element name="listener" type="xmlEvents.listener.type"/>
</xs:schema>
```

F.3.6. XML Handlers Module

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/2001/XMLSchema
                        http://www.w3.org/2001/XMLSchema.xsd"
    elementFormDefault="unqualified"
    blockDefault="#all"
    finalDefault="#all"
    attributeFormDefault="unqualified">
  <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
    schemaLocation="xhtml-datatypes-2.xsd" />
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
      schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:annotation>
    <xs:documentation>
      This is the XML Schema for XML Handlers
      URI: http://www.w3.org/MarkUp/SCHEMA/xml-handlers-1.xsd
      $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
      Note that this module requires the definition of
      xmlHandlers.Common.attrib to be a collection of attributes to
      add to each element.  This collection MUST contain an 'id'
      attribute.
    </xs:documentation>
    <xs:documentation source="xml-events-copyright-2.xsd"/>
  </xs:annotation>
  <xs:attributeGroup name="xmlHandlers.Common.attrib">
      <xs:attribute name="id" type="xs:ID"/>
      <xs:anyAttribute/>
  </xs:attributeGroup>
  <xs:attributeGroup name="xmlHandlers.action.attlist">
    <xs:attributeGroup ref="xmlHandlers.Common.attrib"/>
    <xs:attribute name="event" use="required" type="xs:QName"/>
    <xs:attribute name="eventTarget">
        <xs:simpleType>
            <xs:list itemType="xs:IDREF"/>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="declare">
        <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="declare"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="if" type="xs:normalizedString"/>
    <xs:attribute name="while" type="xs:normalizedString"/>
    <xs:anyAttribute />
  </xs:attributeGroup>
  <xs:group name="xmlHandlers.action.content">
      <xs:choice>
        <xs:element name="action" type="xmlHandlers.action.type"/>
        <xs:element name="dispatchEvent" type="xmlHandlers.dispatchEvent.type"/>
```

```
      <xs:element name="addEventListener" type="xmlHandlers.addEventListener.type"/>
      <xs:element name="removeEventListener" type="xmlHandlers.removeEventListener.type"/>
      <xs:element name="stopPropagation" type="xmlHandlers.stopPropagation.type"/>
      <xs:element name="preventDefault" type="xmlHandlers.preventDefault.type"/>
    </xs:choice>
</xs:group>
<xs:complexType name="xmlHandlers.action.type">
  <xs:group ref="xmlHandlers.action.content"/>
  <xs:attributeGroup ref="xmlHandlers.action.attlist"/>
</xs:complexType>
<xs:attributeGroup name="xmlHandlers.dispatchEvent.attlist">
  <xs:attributeGroup ref="xmlHandlers.Common.attrib"/>
  <xs:attribute name="eventType" type="xs:QName"/>
  <xs:attribute name="eventTarget">
      <xs:simpleType>
          <xs:list itemType="xs:IDREF"/>
      </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="bubbles">
      <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
              <xs:enumeration value="bubbles"/>
          </xs:restriction>
      </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="cancelable">
      <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
              <xs:enumeration value="cancelable"/>
          </xs:restriction>
      </xs:simpleType>
  </xs:attribute>
  <xs:anyAttribute />
</xs:attributeGroup>
<xs:complexType name="xmlHandlers.dispatchEvent.type">
  <xs:attributeGroup ref="xmlHandlers.dispatchEvent.attlist"/>
</xs:complexType>
<xs:element name="dispatchEvent" type="xmlHandlers.dispatchEvent.type"/>
<xs:attributeGroup name="xmlHandlers.addEventListener.attlist">
  <xs:attributeGroup ref="xmlHandlers.Common.attrib"/>
  <xs:attribute name="event" use="required" type="xs:QName"/>
  <xs:attribute name="handler" use="required" type="xs:IDREF"/>
  <xs:attribute name="phase" default="default">
      <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
              <xs:enumeration value="bubble"/>
              <xs:enumeration value="capture"/>
              <xs:enumeration value="default"/>
              <xs:enumeration value="target"/>
          </xs:restriction>
      </xs:simpleType>
  </xs:attribute>
  <xs:anyAttribute />
</xs:attributeGroup>
<xs:complexType name="xmlHandlers.addEventListener.type">
  <xs:attributeGroup ref="xmlHandlers.addEventListener.attlist"/>
</xs:complexType>
<xs:element name="addEventListener" type="xmlHandlers.addEventListener.type"/>
<xs:attributeGroup name="xmlHandlers.removeEventListener.attlist">
  <xs:attributeGroup ref="xmlHandlers.Common.attrib"/>
  <xs:attribute name="event" use="required" type="xs:QName"/>
  <xs:attribute name="handler" use="required" type="xs:IDREF"/>
  <xs:attribute name="phase" default="default">
```

```
        <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="bubble"/>
                <xs:enumeration value="capture"/>
                <xs:enumeration value="default"/>
                <xs:enumeration value="target"/>
            </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:anyAttribute />
    </xs:attributeGroup>
    <xs:complexType name="xmlHandlers.removeEventListener.type">
      <xs:attributeGroup ref="xmlHandlers.removeEventListener.attlist"/>
    </xs:complexType>
    <xs:element name="removeEventListener" type="xmlHandlers.removeEventListener.type"/>
    <xs:attributeGroup name="xmlHandlers.stopPropagation.attlist">
      <xs:attributeGroup ref="xmlHandlers.Common.attrib"/>
      <xs:attribute name="event" use="required" type="xs:QName"/>
      <xs:anyAttribute />
    </xs:attributeGroup>
    <xs:complexType name="xmlHandlers.stopPropagation.type">
      <xs:attributeGroup ref="xmlHandlers.stopPropagation.attlist"/>
    </xs:complexType>
    <xs:element name="stopPropagation" type="xmlHandlers.stopPropagation.type"/>
    <xs:attributeGroup name="xmlHandlers.preventDefault.attlist">
      <xs:attributeGroup ref="xmlHandlers.Common.attrib"/>
      <xs:attribute name="event" use="required" type="xs:QName"/>
      <xs:anyAttribute />
    </xs:attributeGroup>
    <xs:complexType name="xmlHandlers.preventDefault.type">
      <xs:attributeGroup ref="xmlHandlers.preventDefault.attlist"/>
    </xs:complexType>
    <xs:element name="preventDefault" type="xmlHandlers.preventDefault.type"/>
  </xs:schema>
```

F.3.7. XML Scripting Module

```
  <?xml version="1.0" encoding="UTF-8"?>
  <xs:schema
      xmlns:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/XMLSchema
                          http://www.w3.org/2001/XMLSchema.xsd"
      elementFormDefault="unqualified"
      blockDefault="#all"
      finalDefault="#all"
      attributeFormDefault="unqualified">
    <xs:import namespace="http://www.w3.org/1999/xhtml/datatypes/"
               schemaLocation="xhtml-datatypes-2.xsd" />
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd" />
    <xs:annotation>
      <xs:documentation>
        This is the XML Schema for XML Scripting
        URI: http://www.w3.org/MarkUp/SCHEMA/xml-script-1.xsd
        $Id: xhtml-modularization2.ps,v 1.2 2009/01/23 20:12:44 smccarro Exp $
        Note that this module requires the definition of
        xmlScripting.Common.attrib to be a collection of attributes to
```

```
        add to each element.  This collection MUST contain an 'id'
        attribute.
      </xs:documentation>
      <xs:documentation source="xml-events-copyright-2.xsd"/>
    </xs:annotation>
    <xs:attributeGroup name="xmlScripting.Common.attrib">
      <xs:attribute name="id" type="xs:ID"/>
      <xs:anyAttribute/>
    </xs:attributeGroup>
    <xs:attributeGroup name="xmlScripting.script.attlist">
      <xs:attributeGroup ref="xmlScripting.Common.attrib"/>
      <xs:attribute name="charset" type="xh11d:Charset"/>
      <xs:attribute name="defer" type="xs:NMTOKEN" fixed="defer"/>
      <xs:attribute name="implements" type="xh11d:URIorSafeCURIEs"/>
      <xs:attribute name="src" type="xs:anyURI"/>
      <xs:attribute name="type" type="xh11d:ContentTypes"/>
      <xs:anyAttribute />
    </xs:attributeGroup>
    <xs:group name="xmlScripting.script.content">
        <xs:sequence/>
    </xs:group>
    <xs:complexType name="xmlScripting.script.type" mixed="true">
        <xs:group ref="xmlScripting.script.content"/>
        <xs:attributeGroup ref="xmlScripting.script.attlist"/>
    </xs:complexType>
  </xs:schema>
```

G. Building DTD Modules

This section is *normative.*

XHTML modules are implemented as DTD fragments. When these fragments are assembled in a specific manner (described in Developing DTDs with defined [p.269] is a representation of a complete document type. This representation can then be used for validation of instances of the document type.

The key to combining these fragments into a meaningful DTD is the rules used to define the fragments. This section defines those rules. When these rules are followed, DTD authors can be confident that their modules will interface cleanly with other XHTML-compatible modules.

This specification classifies parameter entities into seven categories and names them consistently using the following suffixes:

.mod
: parameter entities use the suffix `.mod` when they are used to represent a DTD module (a collection of elements, attributes, parameter entities, etc). In this specification, each module is an atomic unit and may be represented as a separate file entity.

.module
: parameter entities use the suffix `.module` when they are used to control the inclusion of a DTD module by containing either of the conditional section keywords `INCLUDE` or `IGNORE`.

.qname
: parameter entities use the suffix `.qname` when they are used to represent the qualified name of an element. See Defining the Namespace of a Module [p.142] for more information on qualified names.

.content
: parameter entities use the suffix `.content` when they are used to represent the content model of an element type.

.class
: parameter entities use the suffix `.class` when they are used to represent elements of the same class.

.mix
: parameter entities use the suffix `.mix` when they are used to represent a collection of element types from different classes.

.attrib
: parameter entities use the suffix `.attrib` when they are used to represent a group of tokens representing one or more complete attribute specifications within an ATTLIST declaration.

For example, in HTML 4, the `%block;` parameter entity is defined to represent the heterogeneous collection of element types that are block-level elements. In this specification, the corollary parameter entity is `%Block.mix;`.

When defining parameter entities in the classes defined here, modules should scope the names of the entities by using unique prefixes. For example, the content model for the element `myelement` in the module mymodule could be named `MYMODULE.myelement.content`. Other schemes are possible. Regardless of the scheme used, module authors should strive to ensure that parameter entities they define are named uniquely so that they do not collide with other parameter entities and so that the interface methods for the module are obvious to its users.

G.1. Defining the Namespace of a Module

XHTML requires that the elements and attributes declared in a module be within a defined XML namespace [XMLNS [p.297]]. The identification of this namespace is an arbitrary URI. XHTML requires that when a module is implemented using an XML DTD, the module declares the namespace in a special manner. The purpose of this is to permit the selection, at document parse/validation time, of the use of namespace prefixes and of the *prefix* that is used to identify elements and attributes from the module.

Content developers who wish to develop documents based upon hybrid document types may choose to use XML namespace prefixes on elements from the XHTML namespace, on elements from other namespaces, or on both. In order to ensure that such documents are XHTML conforming and backward compatible with non-namespace aware tools, the W3C recommends that content developers *do not* use XML namespace prefixes on elements from the XHTML namespace. When content developers are interested in having their content processed by namespace-aware processors, the W3C further recommends that elements in non-XHTML namespaces be specified using an XML namespace prefix rather than relying upon XML namespace defaulting mechanisms.

Each XHTML-conforming module implemented as an XML DTD is required to define a default XML namespace prefix, a method for changing this prefix within a document instance, and a marked section that turns on the processing of the prefix.

Note that it is legal and expected for multiple modules to be part of the same namespace when they are related. All of the XHTML modules, for example, are part of the same namespace.

G.1.1. Qualified Names sub-module

First, you need to define a qualified names sub-module (a sub-module is just a *file entity* that is separated so that it can be incorporated into the ultimate DTD at the appropriate point). The qualified names sub-module is built using the following steps (where the string MODULE is replaced with an appropriate string for the new module):

1. Define a parameter entity MODULE.prefixed that announces whether the elements in the module are being used with XML namespace prefixed names or not. This parameter entity's default value should be "%NS.prefixed;". The NS.prefixed parameter entity is defined by the XHTML framework to be IGNORE by default, and can be used in a document instance to switch on prefixing for all included namespaces (including that of the XHTML modules).
2. Define a parameter entity MODULE.xmlns that contains the namespace identifier for this

module.

3. Define a parameter entity MODULE.prefix that contains the default prefix string to use when prefixing is enabled.
4. Define a parameter entity MODULE.pfx that is "%MODULE.prefix;:" when prefixing is enabled, and "" when it is not.
5. Define a parameter entity MODULE.xmlns.extra.attrib that contains the declaration of any XML namespace attributes for namespaces referenced by this module (e.g., xmlns:xlink). When %MODULE.prefixed is set to INCLUDE, this attribute should include the xmlns:%MODULE.prefix; declaration as well.
6. Define a parameter entity XHTML.xmlns.extra.attrib as MODULE.xmlns.extra.attrib. This is usually overridden by the document type's driver file, but if not this definition will take over as the default.
7. For each of the elements defined by the module, create a parameter entity of the form "MODULE.NAME.qname" to hold its qualified name. The value for this parameter entity must be "%MODULE.pfx;NAME". In this way, the parsed value will be "PREFIX:NAME" when prefixes are enabled, and "NAME" otherwise.

   If the module adds attributes to elements defined in modules that do not share the namespace of this module, declare those attributes so that they use the %MODULE.pfx prefix. For example:

   Example

```
<ENTITY % MODULE.img.myattr.qname "%MODULE.pfx;myattr" >
```

An example of a qname sub-module for a hypothetical Inventory Module is included below:

```
<!-- ....................................................................... -->
<!-- Inventory Qname Module ................................................ -->
<!-- file: inventory-qname-1.mod
        PUBLIC "-//MY COMPANY//ELEMENTS XHTML Inventory Qnames 1.0//EN"
        SYSTEM "http://www.example.com/DTDs/inventory-qname-1.mod"
        xmlns:inventory="http://www.example.com/xmlns/inventory"
     ....................................................................... -->
<!-- Declare the default value for prefixing of this module's elements -->
<!-- Note that the NS.prefixed will get overridden by the XHTML Framework or
     by a document instance. -->
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % Inventory.prefixed "%NS.prefixed;" >
<!-- Declare the actual namespace of this module -->
<!ENTITY % Inventory.xmlns "http://www.example.com/xmlns/inventory" >
<!-- Declare the default prefix for this module -->
<!ENTITY % Inventory.prefix "inventory" >
<!-- Declare the prefix for this module -->
<![%Inventory.prefixed;[
<!ENTITY % Inventory.pfx "%Inventory.prefix;:" >
]]>
<!ENTITY % Inventory.pfx "" >
<!-- Declare the xml namespace attribute for this module -->
<![%Inventory.prefixed;[
<!ENTITY % Inventory.xmlns.extra.attrib
       "xmlns:%Inventory.prefix;       %URI.datatype;  #FIXED  '%Inventory.xmlns;'" >
```

```
]]>
<!ENTITY % Inventory.xmlns.extra.attrib "" >
<!-- Declare the extra namespace that should be included in the XHTML
     elements -->
<!ENTITY % XHTML.xmlns.extra.attrib
        "%Inventory.xmlns.extra.attrib;" >
<!-- Now declare the qualified names for all of the elements in the
     module -->
<!ENTITY % Inventory.shelf.qname "%Inventory.pfx;shelf" >
<!ENTITY % Inventory.item.qname "%Inventory.pfx;item" >
<!ENTITY % Inventory.desc.qname "%Inventory.pfx;desc" >
<!ENTITY % Inventory.sku.qname "%Inventory.pfx;sku" >
<!ENTITY % Inventory.price.qname "%Inventory.pfx;price" >
```

G.1.2. Declaration sub-module(s)

Next, you need to define one or more "declaration sub-modules". The purpose of these *file entities* is to declare the XML DTD elements and attribute lists. An XHTML declaration module should be constructed using the following process:

1. Define a parameter entity to use within the ATTLIST of each declared element. This parameter entity should contain %NS.decl.attrib; when %MODULE.prefixed; is set to INCLUDE, and %NS.decl.attrib; plus "xmlns %URI.datatype; #FIXED '%MODULE.xmlns;'" when %MODULE.prefixed; is set to IGNORE.
2. Declare all of the elements and attributes for the module. Within each ATTLIST for an element, include the parameter entity defined above so that all of the required xmlns attributes are available on each element in the module.

3. If the module adds attributes to elements defined in modules that do not share the namespace of this module, declare those attributes so that they use the %MODULE.pfx prefix. For example:

   Example

   ```
   <ENTITY % MODULE.img.myattr.qname "%MODULE.pfx;myattr" >
   <!ATTLIST %img.qname;
         %MODULE.img.myattr.qname;    CDATA          #IMPLIED
   >
   ```

   This would add an attribute to the `img` element of the Image Module, but the attribute's name will be the qualified name, including prefix, when prefixes are selected for a document instance. It also adds the xmlns:MODULE_PREFIX attribute to the img element's attribute list so that an XML Namespace-aware parser would know how to resolve the namespace based upon its prefix.

The following example shows a declaration sub-module for a hypothetical Inventory module.

```
<!-- ..................................................................... -->
<!-- Inventory Elements Module ............................................ -->
<!-- file: inventory-1.mod
        PUBLIC "-//MY COMPANY//ELEMENTS XHTML Inventory Elements 1.0//EN"
        SYSTEM "http://www.example.com/DTDs/inventory-1.mod"
```

```
         xmlns:inventory="http://www.example.com/xmlns/inventory"
     ..................................................................... -->
<!-- Inventory Module
     shelf
        item
           sku
           desc
           price
     This module defines a simple inventory item structure
-->
<!-- Define the global namespace attributes -->
<![%Inventory.prefixed;[
<!ENTITY % Inventory.xmlns.attrib
    "%NS.decl.attrib;"
>
]]>
<!ENTITY % Inventory.xmlns.attrib
         "xmlns %URI.datatype;  #FIXED '%Inventory.xmlns;'"
>
<!-- Define a common set of attributes for all module elements -->
<!ENTITY % Inventory.Common.attrib
         "%Inventory.xmlns.attrib;
          id              ID                  #IMPLIED"
>
<!-- Define the elements and attributes of the module -->
<!ELEMENT %Inventory.shelf.qname;
     ( %Inventory.item.qname; )* >
<!ATTLIST %Inventory.shelf.qname;
     location   CDATA   #IMPLIED
     %Inventory.Common.attrib;
>
<!ELEMENT %Inventory.item.qname;
     ( %Inventory.desc.qname;, %Inventory.sku.qname;, %Inventory.price.qname;) >
<!ATTLIST %Inventory.item.qname;
     location   CDATA   #IMPLIED
     %Inventory.Common.attrib;
>
<!ELEMENT %Inventory.desc.qname; ( #PCDATA ) >
<!ATTLIST %Inventory.desc.qname;
     %Inventory.Common.attrib;
>
<!ELEMENT %Inventory.sku.qname; ( #PCDATA ) >
<!ATTLIST %Inventory.sku.qname;
     %Inventory.Common.attrib;
>
<!ELEMENT %Inventory.price.qname; ( #PCDATA ) >
<!ATTLIST %Inventory.price.qname;
     %Inventory.Common.attrib;
>
<!-- end of inventory-1.mod -->
```

G.1.3. Using the module as a stand-alone DTD

It is sometimes desirable to have an XHTML module also usable as a stand alone DTD. A good example of this is our Inventory module above. These items need to be embeddable in an XHTML document, and also need to be available as free-standing documents extracted from a database (for example). The easiest way to accomplish this is to define a DTD file that

instantiates the components of your module. Such a DTD would have this structure:

1. Include the XHTML Datatypes Module (your qnames module likely uses some of these datatypes - it certainly uses the URI datatype for the xmlns attribute).
2. Include the Qnames Module for your module.
3. Define the parameter entity NS.decl.attrib to be %MODULE.xmlns.extra.attrib;.
4. Include the Declaration Module(s) for your module.

An example of this for our Inventory module is included below:

```
<!-- ..................................................................... -->
<!-- Inventory Elements DTD ............................................. -->
<!-- file: inventory-1.dtd
         PUBLIC "-//MY COMPANY//DTD XHTML Inventory 1.0//EN"
         SYSTEM "http://www.example.com/DTDs/inventory-1.dtd"
         xmlns:inventory="http://www.example.com/xmlns/inventory"
     ..................................................................... -->
<!-- Inventory Module
     shelf
        item
           sku
           desc
           price
     This module defines a simple inventory item structure
-->
<!-- Bring in the datatypes -->
<!ENTITY % xhtml-datatypes.mod
         PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
                "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-datatypes-1.mod" >
%xhtml-datatypes.mod;
<!-- Bring in the qualified names -->
<!ENTITY % Inventory-qname.mod SYSTEM "inventory-qname-1.mod" >
%Inventory-qname.mod;
<!ENTITY % NS.decl.attrib "%Inventory.xmlns.extra.attrib;">
<!ENTITY % Inventory.mod SYSTEM "inventory-1.mod" >
%Inventory.mod;
<!-- end of inventory-1.dtd -->
```

This DTD can then be referenced by documents that use only the elements from your module:

```
<!DOCTYPE shelf SYSTEM "inventory-1.dtd">
<shelf xmlns="http://www.example.com/xmlns/inventory">
    <item>
        <desc>
          this is a description.
        </desc>
        <sku>
          this is the price.
        </sku>
        <price>
          this is the price.
        </price>
    </item>
</shelf>
```

This method permits the definition of elements and attributes that are scoped within their own namespace. It also permits content developers to use the default prefix for the elements and attributes:

```
<!DOCTYPE inventory:shelf SYSTEM "inventory-1.dtd" [
        <!ENTITY % Inventory.prefixed "INCLUDE">
]>
<inventory:shelf xmlns:inventory="http://www.example.com/xmlns/inventory">
    <inventory:item>
        <inventory:desc>
          this is a description.
        </inventory:desc>
        <inventory:sku>
          this is the sku.
        </inventory:sku>
        <inventory:price>
          this is the price.
        </inventory:price>
    </inventory:item>
</inventory:shelf>
```

Finally, a document instance can use a different XML namespace prefix by redeclaring it in the DOCTYPE header and its internal subset:

```
<!DOCTYPE i:shelf SYSTEM "inventory-1.dtd" [
        <!ENTITY % Inventory.prefixed "INCLUDE">
        <!ENTITY % Inventory.prefix "i">
]>
<i:shelf xmlns:i="http://www.example.com/xmlns/inventory">
    <i:item>
        <i:desc>
          this is a description.
        </i:desc>
        <i:sku>
          this is the price.
        </i:sku>
        <i:price>
          this is the price.
        </i:price>
    </i:item>
</i:shelf>
```

G.1.4. Namespace Idiosyncrasies

While the approach defined here permits the definition of markup languages that are XML and XML namespaces conforming, some behaviors defined by the XML namespaces specification are not supported:

1. XML namespaces permit the redeclaration of the xmlns attribute for a namespace at any point in the tree. It further permits this redeclaration to switch between namespace defaulting and prefixed usage, and permits the changing of the prefix. The method defined in this document does not permit this. Throughout a document instance a given namespace must continue to use the same namespace prefix (when prefixing is used), or must continue

to be used in the default scope.

2. When using XML namespace defaulting, it is legal to rely upon the DTD of the document to inform parsers of the namespace of elements. However, since namespace aware processors are not required to include the DTD when evaluating a document, content developers should declare the XML namespace of an element whenever the namespace changes:

   Example

```
...
<p>
   <myelement xmlns="..." />
</p>
```

H. Developing DTDs with defined and extended modules

This section is **informative**.

The primary purpose of defining XHTML modules and a general modularization methodology is to ease the development of document types that are based upon XHTML. These document types may extend XHTML by integrating additional capabilities (e.g., [SMIL] [p.??]), or they may define a subset of XHTML for use in a specialized device. This section describes the techniques that document type designers must use in order to take advantage of the XML DTD implementation of this modularization architecture. It does this by applying the XHTML Modularization techniques in progressively more complex ways, culminating in the creation of a complete document type from disparate modules.

Note that in no case do these examples require the modification of the XHTML-provided module *file entities* themselves. The XHTML module file entities are completely parameterized, so that it is possible through separate module definitions and *driver files* to customize the definition and the content model of each element and each element's hierarchy.

Finally, remember that most users of XHTML are *not* expected to be DTD authors. DTD authors are generally people who are defining specialized markup that will improve the readability, simplify the rendering of a document, or ease machine-processing of documents, or they are client designers that need to define the specialized DTD for their specific client. Consider these cases:

- An organization is providing subscriber's information via a Web interface. The organization stores its subscriber information in an XML-based database. One way to report that information out from the database to the Web is to embed the XML records from the database directly in the XHTML document. While it is possible to merely embed the records, the organization could define a DTD module that describes the records, attach that module to an XHTML DTD, and thereby create a complete DTD for the pages. The organization can then access the data within the new elements via the Document Object Model [DOM] [p.??] , validate the documents, provide style definitions for the elements that cascade using Cascading Style Sheets [CSS2] [p.??] , etc. By taking the time to define the structure of their data and create a DTD using the processes defined in this section, the organization can realize the full benefits of XML.

- An Internet client developer is designing a specialized device. That device will only support a subset of XHTML, and the devices will always access the Internet via a proxy server that validates content before passing it on to the client (to minimize error handling on the client). In order to ensure that the content is valid, the developer creates a DTD that is a subset of XHTML using the processes defined in this section. They then use the new DTD in their proxy server and in their devices, and also make the DTD available to content developers so that developers can validate their content before making it available. By performing a few simple steps, the client developer can use the architecture defined in this document to greatly ease their DTD development cost *and* ensure that they are fully supporting the

subset of XHTML that they choose to include.

H.1. Defining additional attributes

In some cases, an extension to XHTML can be as simple as additional attributes. Attributes can be added to an element just by specifying an additional ATTLIST for the element, for example:

Example

```
<!ATTLIST %a.qname;
          %MyModule.pfx;myattr   CDATA        #IMPLIED
          %MyModule.xmlns.extras.attrib;
>
```

would add the "myattr" attribute, with an optional prefix defined by "%MyModule.pfx", with a value type of CDATA, to the "a" element. This works because XML permits the definition or extension of the attribute list for an element at any point in a DTD. *For a discussion of qualified names and namespace prefixes, see Defining the Namespace of a Module [p.142] .*

Naturally, adding an attribute to a DTD does not mean that any new behavior is defined for arbitrary clients. However, a content developer could use an extra attribute to store information that is accessed by associated scripts via the Document Object Model (for example).

H.2. Defining additional elements

Defining additional elements is only slightly more complicated than defining additional attributes. Basically, DTD authors should write the element declaration for each element:

Example

```
<!-- In the qname sub-module -->
<!ENTITY % MyModule.myelement.qname  "%MyModule.pfx;myelement" >
<!ENTITY % MyModule.myotherelement.qname  "%MyModule.pfx;myotherelement" >
<!-- In the declaration sub-module -->
<!ELEMENT %MyModule.myelement.qname;
           ( #PCDATA | %MyModule.myotherelement.qname; )* >
<!ATTLIST %MyModule.myelement.qname;
          myattribute    CDATA    #IMPLIED
>
<!ELEMENT %MyModule.myotherelement.qname; EMPTY >
```

After the elements are defined, they need to be integrated into the content model. Strategies for integrating new elements or sets of elements into the content model are addressed in the next section.

H.3. Defining the content model for a collection of modules

Since the content model of XHTML modules is fully parameterized, DTD authors may modify the content model for every element in every module. The details of the DTD module interface are defined in Building DTD Modules [p.141] . Basically there are two ways to approach this modification:

1. Re-define the ".content" parameter entity for each element.
2. Re-define one or more of the global content model entities (normally via the ".extras" parameter entity).

The strategy taken will depend upon the nature of the modules being combined and the nature of the elements being integrated. The remainder of this section describes techniques for integrating two different classes of modules.

H.3.1. Integrating a stand-alone module into XHTML

When a module (and remember, a module can be a collection of other modules) contains elements that only reference each other in their content model, it is said to be "internally complete". As such, the module can be used on its own; (for example, you could define a DTD that was just that module, and use one of its elements as the root element). Integrating such a module into XHTML is a three step process:

1. Decide what element(s) can be thought of as the root(s) of the new module.
2. Decide where these elements need to attach in the XHTML content tree.
3. Then, for each attachment point in the content tree, add the root element(s) to the content definition for the XHTML elements.

Consider attaching the elements defined above [p.150] . In that example, the element `myelement` is the root. To attach this element under the `img` element, and only the `img` element, of XHTML, the following would work:

Example

```
<!ENTITY % img.content "( %MyModule.myelement.qname; )*">
```

A DTD defined with this content model would allow a document like the following fragment:

Example

```
<img src="...">
<myml:myelement >This is content of a locally defined element</myml:myelement>
</img>
```

It is important to note that normally the `img` element has a content model of `EMPTY`. By adding myelement to that content model, we are really just replacing `EMPTY` with `myelement`. In the case of other elements that already have content models defined, the addition of an element would require the restating of the existing content model in addition to `myelement`.

H.3.2. Mixing a new module throughout the modules in XHTML

Extending the example above, to attach this module everywhere that the `%Flow.mix` content model group is permitted, would require something like the following:

Example

```
<!ENTITY % Misc.extra
     "| %MyModule.myelement.qname;" >
```

Since the %Misc.extra content model class is used in the %Misc.class parameter entity, and that parameter entity is used throughout the XHTML modules, the new module would become available throughout an extended XHTML document type.

H.4. Creating a new DTD

So far the examples in this section have described the methods of extending XHTML and XHTML's content model. Once this is done, the next step is to collect the modules that comprise the DTD into a single DTD driver, incorporating the new definitions so that they override and augment the basic XHTML definitions as appropriate.

H.4.1. Creating a simple DTD

Using the trivial example above, it is possible to define a new DTD that uses and extends the XHTML modules pretty easily. First, define the new elements and their content model in a module:

```
<!-- File: simpleml-model-1.mod -->
<!-- Declare a Parameter Entity (PE) that defines any external namespaces
     that are used by this module -->
<!-- Set the PE that is used in every ATTLIST in this module
     NS.prefixed.attrib is initialized in the xhtml-qname module, and
         SimpleML.ns.noprefix.attrib is initialized in the SimpleML DTD driver
         file.-->
<!ENTITY % SimpleML.xmlns.attrib
  "%NS.decl.attrib;"
>
<!ENTITY % SimpleML.Common.attrib
  "%SimpleML.xmlns.attrib;
   id           ID           #IMPLIED"
>
<!ENTITY % SimpleML.element.qname "%SimpleML.pfx;element" >
<!ENTITY % SimpleML.otherelement.qname "%SimpleML.pfx;otherelement" >
<!ELEMENT %SimpleML.element.qname;
          ( #PCDATA | %SimpleML.otherelement.qname; )* >
<!ATTLIST %SimpleML.element.qname;
          myattribute   CDATA  #IMPLIED
          %SimpleML.Common.attrib;
>
<!ELEMENT %SimpleML.otherelement.qname; EMPTY >
<!ATTLIST %SimpleML.otherelement.qname;
```

```
          %SimpleML.Common.attrib;
>
<!ENTITY % SimpleML.img.myattr.qname "%SimpleML.pfx;myattr" >
<!ATTLIST %img.qname;
          %SimpleML.img.myattr.qname;  CDATA  #IMPLIED
>
<!-- Add our elements to the XHTML content model -->
<!ENTITY % Misc.class
     "| %SimpleML.element.qname;" >
<!-- Now bring in the XHTML Basic content model -->
<!ENTITY % xhtml-basic-model.mod
     PUBLIC "-//W3C//ENTITIES XHTML Basic 1.0 Document Model 1.0//EN"
                "http://www.w3.org/TR/xhtml-basic/xhtml-basic10-model-1.mod" >
%xhtml-basic-model.mod;
```

Next, define the DTD driver for the new language:

```
<!-- file: simpleml-1_0.dtd -->
<!-- Bring in the XHTML datatypes -->
<!ENTITY % xhtml-datatypes.mod
     PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-datatypes-1.mod" >
%xhtml-datatypes.mod;
<!-- Declare the actual namespace of this module -->
<!ENTITY % SimpleML.xmlns "http://www.example.com/xmlns/simpleml1" >
<!-- By default, disable prefixing of new module -->
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % SimpleML.prefixed "%NS.prefixed;" >
<!-- Default prefix for module elements and attributes -->
<!ENTITY % SimpleML.prefix "simpleml" >
<!-- If this module's namespace is prefixed -->
<![%SimpleML.prefixed;[
  <!ENTITY % SimpleML.pfx  "%SimpleML.prefix;:" >
]]>
<!ENTITY % SimpleML.pfx  "" >
<![%SimpleML.prefixed;[
  <!ENTITY % SimpleML.xmlns.extra.attrib
       "xmlns:%SimpleML.prefix; %URI.datatype; #FIXED '%SimpleML.xmlns;'" >
]]>
<!ENTITY % SimpleML.xmlns.extra.attrib "" >
<!ENTITY % XHTML.xmlns.extra.attrib
       "%SimpleML.xmlns.extra.attrib;"
>
<!-- Set the content model for our language -->
<!ENTITY % xhtml-model.mod
     SYSTEM "simpleml-model-1.mod" >
<!-- Instantiate xhtml basic's DTD to do all the work -->
<!ENTITY % xhtml-basic.dtd
     PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
            "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd" >
%xhtml-basic.dtd;
```

When using this DTD, it is possible to enable the use of XML namespace prefixes. When so doing, the start of a document using this new DTD might look like:

```
<!DOCTYPE html SYSTEM "simpleml-1_0.dtd" [
  <!ENTITY % SimpleML.prefixed "INCLUDE">
]>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:simpleml="http://www.example.com/xmlns/simpleml1" >
<head>
<title>An example using defaults</title>
</head>
<body>
<p>This is content in the XHTML namespace</p>
<simpleml:element>
  This is content in the SimpleML namespace.
  <simpleml:otherelement />
</simpleml:element>
<p><img src="missing" alt="Missing image" simpleml:myattr="value"/></p>
</body>
</html>
```

H.4.2. Creating a DTD by extending XHTML

Next, there is the situation where a complete, additional, and complex module is added to XHTML (or to a subset of XHTML). In essence, this is the same as in the trivial example above, the only difference being that the module being added is incorporated in the DTD by reference rather than explicitly including the new definitions in the DTD.

One such complex module is the DTD for [MATHML] [p.??] . In order to combine MathML and XHTML into a single DTD, an author would just decide where MathML content should be legal in the document, and add the MathML root element to the content model at that point. First, define a content model module that instantiates the MathML DTD and connects it to the content model:

Example

```
<!-- File: mathml-model.mod -->
<!ENTITY % XHTML1-math
     PUBLIC "-//W3C//DTD MathML 2.0//EN"
            "http://www.w3.org/TR/MathML2/dtd/mathml2.dtd" >
%XHTML1-math;
<!ENTITY % Inlspecial.extra
     "%a.qname; | %img.qname; | %object.qname; | %map.qname;
          | %Mathml.Math.qname;" >
```

Next, define a DTD driver that identifies our new content model module as the content model for the DTD, and hands off processing to the XHTML 1.1 driver (for example):

Example

```
<!-- File: xhtml-mathml.dtd -->
<!ENTITY % xhtml-model.mod
      SYSTEM "mathml-model.mod" >
<!ENTITY % xhtml11.dtd
     PUBLIC "-//W3C//DTD XHTML 1.1//EN"
            "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd" >
%xhtml11.dtd;
```

H.4.3. Creating a DTD by removing and replacing XHTML modules

Another way in which DTD authors may use XHTML modules is to define a DTD that is a subset of an XHTML family document type (because, for example, they are building devices or software that only supports a subset of XHTML). Doing this is only slightly more complex than the previous example. The basic steps to follow are:

1. Take an XHTML family DTD as the basis of the new document type (we will use XHTML 1.1).
2. Select the modules to remove from that DTD.
3. Define a new DTD that "IGNOREs" the modules.
4. Introduce some new modules.

For example, consider a device that uses XHTML modules, but without forms or tables. The DTD for such a device would look like this:

Example

```
<!-- File: xhtml-simple.dtd -->
<!ENTITY % xhtml-form.module "IGNORE" >
<!ENTITY % xhtml-table.module "IGNORE" >
<!ENTITY % xhtml-table.module "IGNORE" >
<!-- Bring in the basic tables module -->
<!ENTITY % xhtml-basic-table.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Basic Tables 1.0//EN"
           "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-basic-table-1.mod"
>
%xhtml-basic-table.mod;
<!ENTITY % xhtml11.mod
     PUBLIC "-//W3C//DTD XHTML 1.1//EN"
            "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd" >
%xhtml11.mod;
```

Note that this does not actually modify the content model for the XHTML 1.1 DTD. However, since XML ignores elements in content models that are not defined, the form and table elements are dropped from the model automatically.

H.4.4. Creating a new DTD

Finally, some DTD authors may wish to start from scratch, using the XHTML Modularization framework as a toolkit for building a new markup language. This language must be made up of the minimal, required modules from XHTML. It may also contain other XHTML-defined modules or any other module that the author wishes to employ. In this example, we will take the XHTML required modules, add some XHTML-defined modules, and also add in the module we defined above.

The first step is to use the XHTML-provided template for a new qualified names module, modified to define the qualified names and namespace for our new elements.

```
<!-- file: myml-qname-1.mod -->
<!-- Bring in the datatypes - we use the URI.datatype PE for declaring the
     xmlns attributes. -->
<!ENTITY % MyML-datatypes.mod
         PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
                "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-datatypes-1.mod" >
%MyML-datatypes.mod;
<!-- By default, disable prefixing of this module -->
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % MyML.prefixed "%NS.prefixed;" >
<!-- Declare the actual namespace of this module -->
<!ENTITY % MyML.xmlns "http://www.example.com/xmlns/myml" >
<!-- Declare the default prefix for this module -->
<!ENTITY % MyML.prefix "myml" >
<!-- If this module's namespace is prefixed -->
<![%MyML.prefixed;[
  <!ENTITY % MyML.pfx  "%MyML.prefix;:" >
]]>
<!ENTITY % MyML.pfx  "" >
<!-- Declare a Parameter Entity (PE) that defines any external namespaces
     that are used by this module -->
<!ENTITY % MyML.xmlns.extra.attrib "" >
<!-- Declare a PE that defines the xmlns attributes for use by MyML. -->
<![%MyML.prefixed;[
<!ENTITY % MyML.xmlns.attrib
   "xmlns:%MyML.prefix;  %URI.datatype;  #FIXED '%MyML.xmlns;'
   %MyML.xmlns.extra.attrib;"
>
]]>
<!ENTITY % MyML.xmlns.attrib
   "xmlns       %URI.datatype;  #FIXED '%MyML.xmlns;'
        %MyML.xmlns.extra.attrib;"
>
<!-- Make sure that the MyML namespace attributes are included on the XHTML
     attribute set -->
<![%NS.prefixed;[
<!ENTITY % XHTML.xmlns.extra.attrib
        "%MyML.xmlns.attrib;" >
]]>
<!ENTITY % XHTML.xmlns.extra.attrib
        ""
>
<!-- Now declare the element names -->
<!ENTITY % MyML.myelement.qname "%MyML.pfx;myelement" >
<!ENTITY % MyML.myotherelement.qname "%MyML.pfx;myotherelement" >
```

Next, define a module that defines the elements and attributes using the XHTML provided template.

```
<!-- ....................................................................... -->
<!-- My Elements Module ..................................................... -->
<!-- file: myml-elements-1_0.mod
         PUBLIC "-//MY COMPANY//ELEMENTS XHTML MyML Elements 1.0//EN"
         SYSTEM "http://example.com/DTDs/myml-elements-1_0.mod"
         xmlns:myml="http://example.com/DTDs/myml-1_0.dtd"
     ....................................................................... -->
<!-- My Elements Module
     myelement
     myotherelement
```

```
     This module has no purpose other than to provide structure for some
         PCDATA content.
-->
<!ELEMENT %MyML.myelement.qname;
     ( #PCDATA | %MyML.myotherelement.qname; )* >
<!ATTLIST %MyML.myelement.qname;
     myattribute        CDATA   #IMPLIED
     %MyML.xmlns.attrib;
>
<!ELEMENT %MyML.myotherelement.qname; EMPTY >
<!ATTLIST %MyML.myotherelement.qname;
      %MyML.xmlns.attrib;
>
<!ENTITY % MyML.img.myattr.qname "%MyML.pfx;myattr" >
<!ATTLIST %img.qname;
      %MyML.img.myattr.qname;   CDATA  #IMPLIED
      %MyML.xmlns.attrib;
>
<!-- end of myml-elements-1_0.mod -->
```

Now, build a content model description that hooks the new elements and attributes into the other XHTML elements. The following example is patterned after the XHTML Basic content model, but is a complete, free-standing content model module:

```
<!-- ....................................................................... -->
<!-- MyML Model Module  ..................................................... -->
<!-- file: myml-model-1.mod
         PUBLIC "-//MY COMPANY//ELEMENTS XHTML MyML Model 1.0//EN"
         SYSTEM "http://example.com/DTDs/myml-model-1_0.mod"
         xmlns:myml="http://www.example.com/xmlns/myml"
     ....................................................................... -->
<!-- Define the content model for Misc.extra -->
<!ENTITY % Misc.class
     "| %MyML.myelement.qname; ">
<!-- ...................  Inline Elements  ..................... -->
<!ENTITY % HeadOpts.mix
     "( %meta.qname; )*" >
<!ENTITY % I18n.class "" >
<!ENTITY % InlStruct.class "%br.qname; | %span.qname;" >
<!ENTITY % InlPhras.class
     "| %em.qname; | %strong.qname; | %dfn.qname; | %code.qname;
      | %samp.qname; | %kbd.qname; | %var.qname; | %cite.qname;
      | %abbr.qname; | %acronym.qname; | %q.qname;" >
<!ENTITY % InlPres.class
     "" >
<!ENTITY % Anchor.class "| %a.qname;" >
<!ENTITY % InlSpecial.class "| %img.qname; " >
<!ENTITY % Inline.extra "" >
<!-- %Inline.class; includes all inline elements,
     used as a component in mixes
-->
<!ENTITY % Inline.class
     "%InlStruct.class;
      %InlPhras.class;
      %InlPres.class;
      %Anchor.class;
```

```
      %InlSpecial.class;"
>
<!-- %InlNoAnchor.class; includes all non-anchor inlines,
     used as a component in mixes
-->
<!ENTITY % InlNoAnchor.class
     "%InlStruct.class;
      %InlPhras.class;
      %InlPres.class;
      %InlSpecial.class;"
>
<!-- %InlNoAnchor.mix; includes all non-anchor inlines
-->
<!ENTITY % InlNoAnchor.mix
     "%InlNoAnchor.class;
      %Misc.class;"
>
<!-- %Inline.mix; includes all inline elements, including %Misc.class;
-->
<!ENTITY % Inline.mix
     "%Inline.class;
      %Misc.class;"
>
<!-- .....................  Block Elements  ..................... -->
<!ENTITY % Heading.class
     "%h1.qname; | %h2.qname; | %h3.qname;
      | %h4.qname; | %h5.qname; | %h6.qname;" >
<!ENTITY % List.class "%ul.qname; | %ol.qname; | %dl.qname;" >
<!ENTITY % BlkStruct.class "%p.qname; | %div.qname;" >
<!ENTITY % BlkPhras.class
     "| %pre.qname; | %blockquote.qname; | %address.qname;" >
<!ENTITY % BlkPres.class "" >
<!ENTITY % Block.extra "" >
<!-- %Block.class; includes all block elements,
     used as an component in mixes
-->
<!ENTITY % Block.class
     "%BlkStruct.class;
      %BlkPhras.class;
      %BlkPres.class;
      %Block.extra;"
>
<!-- %Block.mix; includes all block elements plus %Misc.class;
-->
<!ENTITY % Block.mix
     "%Heading.class;
      | %List.class;
      | %Block.class;
      %Misc.class;"
>
<!-- .................  All Content Elements  .................. -->
<!-- %Flow.mix; includes all text content, block and inline
-->
<!ENTITY % Flow.mix
     "%Heading.class;
      | %List.class;
      | %Block.class;
```

```
      | %Inline.class;
      %Misc.class;"
>
<!-- special content model for pre element -->
<!ENTITY % pre.content
       "( #PCDATA
        | %Inline.class; )*"
>
<!-- end of myml-model-1.mod -->
```

Finally, use the XHTML-provided template for a new DTD, modified as appropriate for our new markup language:

```
<!-- .................................................................... -->
<!-- MYML DTD  ........................................................... -->
<!-- file: myml-1_0.dtd -->
<!-- This is the DTD driver for myml 1.0.
     Please use this formal public identifier to identify it:
         "-//MY COMPANY//DTD XHTML MYML 1.0//EN"
     And this namespace for myml-unique elements:
         xmlns:myml="http://www.example.com/xmlns/myml"
-->
<!ENTITY % XHTML.version  "-//MY COMPANY//DTD XHTML MYML 1.0//EN" >
<!-- reserved for use with document profiles -->
<!ENTITY % XHTML.profile  "" >
<!-- Tell the framework to use our qualified names module as an extra qname
driver -->
<!ENTITY % xhtml-qname-extra.mod
     SYSTEM "myml-qname-1.mod" >
<!-- Define the Content Model for the framework to use -->
<!ENTITY % xhtml-model.mod
         SYSTEM "myml-model-1.mod" >
<!-- Disable bidirectional text support -->
<!ENTITY % XHTML.bidi  "IGNORE" >
<!-- Bring in the XHTML Framework -->
<!ENTITY % xhtml-framework.mod
     PUBLIC "-//W3C//ENTITIES XHTML Modular Framework 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-framework-1.mod" >
%xhtml-framework.mod;
<!-- Basic Text Module (Required)  ............................... -->
<!ENTITY % xhtml-text.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Basic Text 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-text-1.mod" >
%xhtml-text.mod;
<!-- Hypertext Module (required) ................................. -->
<!ENTITY % xhtml-hypertext.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Hypertext 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-hypertext-1.mod" >
%xhtml-hypertext.mod;
<!-- Lists Module (required)  .................................... -->
<!ENTITY % xhtml-list.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Lists 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-list-1.mod" >
%xhtml-list.mod;
<!-- My Elements Module   ....................................... -->
<!ENTITY % MyML-elements.mod
```

```
     SYSTEM "myml-elements-1.mod" >
%MyML-elements.mod;
<!-- XHTML Images module  ..................................... -->
<!ENTITY % xhtml-image.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Images 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-image-1.mod" >
%xhtml-image.mod;
<!-- Document Metainformation Module  ......................... -->
<!ENTITY % xhtml-meta.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Metainformation 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-meta-1.mod" >
%xhtml-meta.mod;
<!-- Document Structure Module (required)  ..................... -->
<!ENTITY % xhtml-struct.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Document Structure 1.0//EN"
            "http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-struct-1.mod" >
%xhtml-struct.mod;
```

H.5. Using the new DTD

Once a new DTD has been developed, it can be used in any document. Using the DTD is as simple as just referencing it in the DOCTYPE declaration of a document:

```
<!DOCTYPE html SYSTEM "myml-1_0.dtd" >
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>An example using defaults</title>
</head>
<body>
<p>This is content in the XHTML namespace</p>
<myelement>
  This is content in the SimpleML namespace.
  <myotherelement />
</myelement>
<p><img src="missing" alt="Missing image" myattr="value"/></p>
</body>
</html>
```

The document can also use the elements outside of the XHTML namespace by prefixing them:

```
<!DOCTYPE html SYSTEM "myml-1_0.dtd" [
  <!ENTITY % MyML.prefixed "INCLUDE" >
]>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>An example using defaults</title>
</head>
<body>
<p>This is content in the XHTML namespace</p>
<myml:myelement>
  This is content in the myml namespace.
  <myml:myotherelement />
```

```
</myml:myelement>
<p><img src="missing" alt="Missing image" myml:myattr="value" /></p>
</body>
</html>
```

I. 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.

I.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.

I.2. XHTML Module Implementations

This section will contain the formal definition of each of the XHTML Abstract Modules as a DTD module.

I.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.

J. 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
a [p.55]	XHTML Hypertext Module [p.55]	Define an anchor or a link
abbr [p.48]	XHTML Text Module [p.47]	An abbreviation
address [p.39]	XHTML Structural Module [p.37]	Define an address
applet	XHTML Applet Module	Embed a Java applet
blockcode [p.39]	XHTML Structural Module [p.37]	Define a block of computer code
blockquote [p.40]	XHTML Structural Module [p.37]	Define a large quotation
body [p.35]	XHTML Document Module [p.33]	Content of the document
caption [p.79]	XHTML Caption Module [p.79]	Set the caption for a table
cite [p.48]	XHTML Text Module [p.47]	A citation
code [p.49]	XHTML Text Module [p.47]	A code fragment
col [p.120]	XHTML Tables Module [p.119]	Define attributes for a column
colgroup [p.120]	XHTML Tables Module [p.119]	Define attributes for a group of columns
dd [p.58]	XHTML List Module [p.57]	Definition Data
dfn [p.49]	XHTML Text Module [p.47]	A definition
di [p.58]	XHTML List Module [p.57]	Definition Item
div [p.40]	XHTML Structural Module [p.37]	Define the characteristics of a block
dl [p.58]	XHTML List Module [p.57]	Definition List
dt [p.58]	XHTML List Module [p.57]	Definition Term
em [p.49]	XHTML Text Module [p.47]	Emphasis
frame	XHTML Frames Module	No title
frameset	XHTML Frames Module	No title
h [p.41]	XHTML Structural Module [p.37]	Structured heading
head [p.34]	XHTML Document Module [p.33]	Document metadata
headings [p.41]	XHTML Structural Module [p.37]	Set a heading
html [p.33]	XHTML Document Module [p.33]	Document root
iframe	XHTML Iframe Module	No title

img [p.89]	XHTML Image Module [p.89]	Incorporate an image
kbd [p.50]	XHTML Text Module [p.47]	User input
l [p.50]	XHTML Text Module [p.47]	A line of text
li [p.60]	XHTML List Module [p.57]	List item
link [p.97]	XHTML Metainformation Module [p.97]	A link to another resource
meta [p.99]	XHTML Metainformation Module [p.97]	Meta information
nl [p.59]	XHTML List Module [p.57]	Navigation list
noframes	XHTML Frames Module	No title
object [p.103]	XHTML Object Module [p.103]	External object
ol [p.60]	XHTML List Module [p.57]	Ordered list
p [p.42]	XHTML Structural Module [p.37]	Define a paragraph
param	XHTML Applet Module	Parameter for a Java applet
param [p.108]	XHTML Object Module [p.103]	Parameter for external object
pre [p.43]	XHTML Structural Module [p.37]	Define a preformatted block
q [p.51]	XHTML Text Module [p.47]	A quotation
samp [p.51]	XHTML Text Module [p.47]	Sample output
section [p.43]	XHTML Structural Module [p.37]	Define a section of a document
separator [p.44]	XHTML Structural Module [p.37]	Insert a break in a document
span [p.52]	XHTML Text Module [p.47]	Define characteristics of text
standby [p.112]	XHTML Object Module [p.103]	Message to render while loading object
strong [p.52]	XHTML Text Module [p.47]	Strong emphasis
style [p.113]	XHTML Style Sheet Module [p.113]	Definition of style rules
sub [p.52]	XHTML Text Module [p.47]	A subscript
summary [p.122]	XHTML Tables Module [p.119]	Set the summary for a table
sup [p.53]	XHTML Text Module [p.47]	A superscript
table [p.123]	XHTML Tables Module [p.119]	Define a table
tbody [p.133]	XHTML Tables Module [p.119]	Define the body for a table
td [p.133]	XHTML Tables Module [p.119]	Define a table cell
tfoot [p.138]	XHTML Tables Module [p.119]	Define the footer for a table
th [p.133]	XHTML Tables Module [p.119]	Define a table header
thead [p.138]	XHTML Tables Module [p.119]	Define the heading for a table
title [p.35]	XHTML Document Module [p.33]	Title for document

tr [p.139]	XHTML Tables Module [p.119]	Define a table row
ul [p.60]	XHTML List Module [p.57]	Unordered list
var [p.53]	XHTML Text Module [p.47]	A variable

K. List of Attributes

This appendix is *informative*.

Attribute Name	Module	Description
abbr [p.133]	XHTML Tables Module [p.119]	Abbreviated form
align	XHTML Applet Module	Define the verital alignment
alt	XHTML Applet Module	Define alternate description
archive	XHTML Applet Module	Define list of object archives
archive [p.103]	XHTML Object Module [p.103]	Define list of object archives
axis [p.133]	XHTML Tables Module [p.119]	Define a category for a cell
cite [p.67]	XHTML Hypertext Attributes Module [p.67]	Set the URI for a citation
class [p.63]	XHTML Core Attributes Module [p.63]	Set the class of an element
code	XHTML Applet Module	No title
codebase	XHTML Applet Module	Set the URI base for applet code
cols	XHTML Frames Module	Define the width of a frameset
colspan [p.133]	XHTML Tables Module [p.119]	Set the number of columns a table cell should span
content-length [p.103]	XHTML Object Module [p.103]	Indicate the size of a link's target
coords [p.93]	XHTML Image Map Attributes Module [p.91]	Define coordinates for am element
datetime [p.81]	XHTML Edit Attributes Module [p.81]	Define date and time
declare [p.103]	XHTML Object Module [p.103]	Indicate that an object should only be loaded
dir [p.75]	XHTML Bi-directional Text Attribute Module [p.75]	Define the direction of text
disabled [p.113]	XHTML Style Sheet Module [p.113]	Indicate that an item is disabled
edit [p.81]	XHTML Edit Attributes Module [p.81]	Indicate how an item was edited
encoding [p.83]	XHTML Embedding Attributes Module [p.83]	Define the encoding of an external source
frameborder	XHTML Frames Module	Define whether a frame has a border

full [p.48]	XHTML Text Module [p.47]	Refer to the full version of an abbreviation
headers [p.134]	XHTML Tables Module [p.119]	Define the header cells that relate to this cell
height	XHTML Applet Module	Define the initial height
href [p.67]	XHTML Hypertext Attributes Module [p.67]	Define a URI target when the element is activated
hreflang [p.67]	XHTML Hypertext Attributes Module [p.67]	Indicate the base language of the target
hrefmedia [p.68]	XHTML Hypertext Attributes Module [p.67]	Indicate the target media of an href attribute
hreftype [p.68]	XHTML Hypertext Attributes Module [p.67]	Indicate the content type of an href attribute
hspace	XHTML Applet Module	Define the horizontal space around an element
id [p.64]	XHTML Core Attributes Module [p.63]	Define the ID for the element
ismap [p.92]	XHTML Image Map Attributes Module [p.91]	Indicate whether this is an imagemap
ismap	XHTML Server-Side Image Map Module	Indicate whether this is an imagemap
layout [p.64]	XHTML Core Attributes Module [p.63]	Indicate whether whitespace is relevant
longdesc	XHTML Frames Module	Reference to an external description
marginheight	XHTML Frames Module	Define the top and bottom frame margins
marginwidth	XHTML Frames Module	Define the left and right frame margins
media [p.95]	XHTML Media Attribute Module [p.95]	Define the applicable media
media [p.113]	XHTML Style Sheet Module [p.113]	Define the applicable media
name	XHTML Applet Module	Set the name of an applet parameter
name [p.108]	XHTML Object Module [p.103]	Set the name of an applet parameter
nextfocus [p.68]	XHTML Hypertext Attributes Module [p.67]	Define the order in which this control is accessed

noresize	XHTML Frames Module	Inidicated the frame cannot be resized
object	XHTML Applet Module	Define the serialzed version of an applet
prevfocus [p.70]	XHTML Hypertext Attributes Module [p.67]	Define the order in which this control is accessed
profile [p.34]	XHTML Document Module [p.33]	Define the location of the metadata profiles
rows	XHTML Frames Module	Define the height of a frameset
rowspan [p.134]	XHTML Tables Module [p.119]	Define the number of rows a cell spans
scope [p.134]	XHTML Tables Module [p.119]	Define the scope of a header
scrolling	XHTML Frames Module	Define whether a frame can scroll
shape [p.93]	XHTML Image Map Attributes Module [p.91]	Define the shape of a map
span [p.120]	XHTML Tables Module [p.119]	Define the number of columns a colgroup spans
src [p.84]	XHTML Embedding Attributes Module [p.83]	Define the URI for an external source for the element
src	XHTML Frames Module	Define the URI for an external source for the element
srctype [p.84]	XHTML Embedding Attributes Module [p.83]	Indicate the content type of a src attribute
style [p.117]	XHTML Style Attribute Module [p.117]	Set the style to use on the element
target [p.70]	XHTML Hypertext Attributes Module [p.67]	Set the target window for a link
title [p.64]	XHTML Core Attributes Module [p.63]	Set the title for an element
type	XHTML Applet Module	Set the type of an applet value
type [p.108]	XHTML Object Module [p.103]	Define the type of a referenced value element
usemap [p.92]	XHTML Image Map Attributes Module [p.91]	Set the name of an image map to use
value [p.61]	XHTML List Module [p.57]	Set the value for a list item
value	XHTML Applet Module	Set the value for an applet parameter
value [p.108]	XHTML Object Module [p.103]	Set the value for an applet parameter

valuetype	XHTML Applet Module	Set the type of an applet value
valuetype [p.108]	XHTML Object Module [p.103]	Set the type of an applet value
version [p.34]	XHTML Document Module [p.33]	Set the XHTML document version
vspace	XHTML Applet Module	Set the amount of vertical whitespace for an element
width	XHTML Applet Module	Set the width for a table column
xml:base [p.70]	XHTML Hypertext Attributes Module [p.67]	Set the base URI for the element
xml:id [p.64]	XHTML Core Attributes Module [p.63]	Define the ID for the element
xml:lang [p.73]	XHTML I18N Attribute Module [p.73]	Set the language of the element
xsi:schemaLocation [p.34]	XHTML Document Module [p.33]	Set the location for an XML Schema

L. Cross-reference Index

This appendix is *informative*.

This appendix will contain a detailed index of this document, with links to the indexed terms.

M. References

This appendix is *normative*.

M.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 Effects Module*", W3C Working Draft, E. J. Etemad, *ed.*, 27 June 2005, *work in progress*.
Available at: http://www.w3.org/TR/2005/WD-css3-text-20050627/
The latest version is available at: http://www.w3.org/TR/css3-text/

[CURIE]
"*CURIE Syntax 1.0*", W3C Candidate Recommendation, M. Birbeck *et al.*, 9 January 2009.
Available at: http://www.w3.org/TR/2009/CR-curie-20090109

[DCORE]
"*DCMI Metadata Terms*", 13 June 2005.
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

[DOMEVENTS]
"*Document Object Model (DOM) Level 2 Events Specification*", W3C Recommendation, T. Pixley, ed., 13 November 2000.
Available at: http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113
A list of DOM Level 2 specifications can be found at: http://www.w3.org/DOM/DOMTR#dom2

[IRI]
"*Internationalized Resource Identifiers (IRIs)*", RFC 3987, M. Dürst and M. Suignard, January 2005.
Available at: http://www.rfc-editor.org/rfc/rfc3987.txt

[ITS]
"*The Internationalization Tag Set (ITS) Version 1.0*", W3C Recommendation, C. Lieske, F. Sasaki, 3 April 2007.
Available at: http://www.w3.org/TR/2007/REC-its-20070403/

[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/
[RDFASYNTAX]
"*RDFa in XHTML: Syntax and Processing*", W3C Recommendation, J. Miller *et al.*, 14 October 2008.
Available at: http://www.w3.org/TR/2008/REC-rdfa-syntax-20081014
[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, 25 March 2005.
Available at: http://www.unicode.org/reports/tr9/tr9-15.html
The latest version of UAX #9 is available at: http://www.unicode.org/reports/tr9/
[URI]
"*Uniform Resource Identifiers (URI): Generic Syntax*", RFC 3986, T. Berners-Lee *et al.*, January 2005.

Available at: http://www.rfc-editor.org/rfc/rfc3986.txt.

[XFORMS]
"*XForms 1.1*", W3C Candidate Recommendation, John Boyer, *eds.*, 29 November 2007.
Available at: http://www.w3.org/TR/2007/CR-xforms11-20071129/

[XHTMLACCESS]
"*XHTML Access Module*", W3C Candidate Recommendation, M. Birbeck *et al.*, 15 January 2009.
Available at: http://www.w3.org/TR/2009/CR-xhtml-access-20090115

[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

[XHTMLROLE]
"*XHTML Role Attribute Module*", W3C Candidate Recommendation, M. Birbeck *et al.*, 15 January 2009.
Available at: http://www.w3.org/TR/2009/CR-xhtml-role-20090115

[XHTMLVOCAB]
"*XHTML Vocabulary*", Steven Pemberton, 21 October 2008.
Available at: http://www.w3.org/1999/xhtml/vocab/

[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 2*", W3C Working Draft, S. McCarron *et al.*, *eds.*, 16 February 2007.
Available at: http://www.w3.org/TR/2007/WD-xml-events-20070216

[XMLID]
"*xml:id Version 1.0*", W3C Recommendation, J. Marsh, D. Veillard, N. Walsh, *eds.*, 9 September 2005.
Available at: http://www.w3.org/TR/2005/REC-xml-id-20050909/

[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 Second Edition*", W3C Recommendation, H. S. Thompson *et al.*, *eds.*, 28 October 2004.
Available at: http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
"*XML Schema Part 2: Datatypes Second Edition*", W3C Recommendation, P. V. Biron, A. Malhotra, *eds.*, 28 October 2004.
Available at: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

M.2. Informative References

[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/

[SVG]
"*Scalable Vector Graphics (SVG) 1.1 Specification*", Jon Ferraiolo *et al.*
Available at: http://www.w3.org/TR/SVG/

[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

N. 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.