RDFa in XHTML: Syntax and Processing

RDFa in XHTML: Syntax and Processing

A collection of attributes and processing rules for extending XHTML to support RDF

W3C Working Draft 18 October 2007

This version:
http://www.w3.org/TR/2007/WD-rdfa-syntax-20071018
Latest version:
http://www.w3.org/TR/rdfa-syntax
Editors:
Ben Adida, Creative Commons ben@adida.net
Mark Birbeck, x-port.net Ltd. mark.birbeck@x-port.net
Shane McCarron, Applied Testing and Technology, Inc. shane@aptest.com
Steven Pemberton, CWI

This document is also available in these non-normative formats: PostScript version, PDF version, ZIP archive, and Gzip'd TAR archive.

The English version of this specification is the only normative version. Non-normative translations may also be available.

Abstract

The modern Web is made up of an enormous number of documents that have been created using HTML. These documents contain significant amounts of structured data, which is largely unavailable to tools and applications. When publishers can express this data more completely, and when tools can read it, a new world of user functionality becomes available, letting users transfer structured data between applications and web sites, and allowing browsing applications to improve the user experience: an event on a web page can be directly imported into a user's desktop calendar; a license on a document can be detected so that users can be informed of their rights automatically; a photo's creator, camera setting information, resolution, location and topic can be published as easily as the original photo itself, enabling structured search and sharing.

RDFa is a specification for attributes to be used with languages such as HTML and XHTML to express structured data. The rendered, hypertext data of XHTML is reused by the RDFa markup, so that publishers don't need to repeat significant data in the document content. This document only specifies the use of the RDFa attributes with XHTML. The underlying abstract representation is RDF [RDF-PRIMER [p.68]], which lets publishers build their own vocabulary, extend others, and evolve their vocabulary with maximal interoperability over time. The expressed structure is closely tied to the data, so that rendered data can be copied and pasted along with its relevant structure.

The rules for interpreting the data are generic, so that there is no need for different rules for different formats; this allows authors and publishers of data to define their own formats without having to update software, register formats via a central authority, or worry that two formats may interfere with each other.

RDFa shares some use cases with microformats. Whereas microformats specify both a syntax for embedding structured data into HTML documents and a vocabulary of specific terms for each microformat, RDFa specifies only a syntax and relies on independent specification of terms (RDF Classes and Properties) by others. RDFa allows terms from multiple independently-developed vocabularies to be freely intermixed and is designed such that the language can be parsed without knowledge of the specific term vocabulary being used.

This document is a detailed syntax specification for RDFa, aimed at:

- those looking to create an RDFa parser, and who therefore need a detailed description of the parsing rules;
- those looking to recommend the use of RDFa within their organisation, and who would like to create some guidelines for their users;
- anyone familiar with RDF, and who wants to understand more about what is happening 'under the hood', when an RDFa parser runs.

For those looking for an introduction to the use of RDFa and some real-world examples, please consult the RDFa Primer.

How to Read this Document

If you are already familiar with RDFa, and you want to examine the processing rules—perhaps to create a parser—then you'll find the Processing Model [p.23] section of most interest. It contains an overview of each of the processing steps, followed by more detailed sections, one for each rule.

If you are not familiar with RDFa, but you *are* familiar with RDF, then you might find reading the Syntax Overview [p.9] useful, before looking at the Processing Model [p.23] since it gives a range of examples of XHTML mark-up that use RDFa. Seeing some examples first should make reading the processing rules easier.

If you are not familiar with RDF, then you might want to take a look at the section on RDF Terminology [p.13] before trying to do too much with RDFa. Although RDFa is designed to be easy to author—and authors don't need to understand RDF to use it—anyone writing applications that *consume* RDFa will need to understand RDF. There is a lot of material on RDF on the web, and a growing range of tools that will support RDFa, so all we try to do in this document is provide enough background on RDF to make the goals of RDFa clearer.

And finally, if you are not familiar with either RDFa *or* RDF, and simply want to add RDFa to your documents, then you may find the RDFa Primer [RDFaPRIMER [p.68]] to be a better introduction.

Status of this Document

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

This is a First Public Working Draft produced jointly by the Semantic Web Deployment Working Group and the XHTML 2 Working Group. The goals of the Semantic Web Deployment Working Group are discussed in the Semantic web Deployment Working Group Charter and the goals of the XHTML 2 Working Group are discussed in the XHTML 2 Working Group charter.

The design described in this Working Draft is nearly final. The draft contains some editorial notes with distinct markup (see, e.g., the section RDFa Processor Conformance [p.20]). Some open technical issues are also identified with the same markup. These include an open issue on the interpretation of @instanceof when @about is present, and the handling of some unprefixed @rel and @rev values. Some open issues not fully identified in this Working Draft may be described in the RDFa issue tracking data.

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 XHTML 2 group; and another public list of any patent disclosures made in connection with the deliverables of the Semantic Web Deployment Working Group; those pages 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.

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

Please report errors in this specification to public-rdf-in-xhtml-tf@w3.org (archive).

Table of Contents

1. Motivation . . . 7
2. Syntax Overview . . . 9
    2.1. The RDFa Attributes . . . 9
    2.2. Examples . . . 10
3. RDF Terminology . . . 13
    3.1. Statements . . . 13
    3.2. Triples . . . 13
    3.3. URI references . . . 14
    3.4. Plain literals . . . 15
    3.5. Typed literals . . . 15
    3.6. N-Triples . . . 15
    3.7. Graphs . . . 16
    3.8. Compact URIs . . . 16
    3.9. A description of RDFa in RDF terms . . . 16
4. Conformance Requirements . . . 19
    4.1. Document Conformance . . . 19
    4.2. User Agent Conformance . . . 20
    4.3. RDFa Processor Conformance . . . 20
5. Processing Model . . . 23
    5.1. Overview . . . 23
    5.2. Evaluation Context . . . 23
    5.3. Processing . . . 24
6. RDFa Processing in detail . . . 29
    6.1. Changing the evaluation context . . . 30
        6.1.1. Setting the [current resource] . . . 30
    6.2. Object resolution . . . 31
        6.2.1. Literal object resolution . . . 32
        6.2.2. URI object resolution . . . 35
7. CURIE Syntax Definition . . . 39
8. XHTML+RDFa Definition . . . 41
9. Metainformation Attributes Module . . . 43
    9.1. Datatypes . . . 43
    9.2. Metadata Attribute Collection . . . 43
        9.2.1. The about attribute . . . 43
        9.2.2. The content attribute . . . 43
        9.2.3. The datatype attribute . . . 43
        9.2.4. The instanceof attribute . . . 44
        9.2.5. The property attribute . . . 44
        9.2.6. The rel attribute . . . 45
        9.2.7. The resource attribute . . . 47
        9.2.8. The rev attribute . . . 47

A. Other XML Languages 49
B. XHTML+RDFa DTD 51
B.1. XHTML Metainformation Attributes Module 51
B.2. XHTML+RDFa Content Model Module 53
B.3. XHTML+RDFa Driver Module 58
B.4. SGML Open Catalog Entry for XHTML+RDFa 65
C. References 67
C.1. Related Specifications 67
C.2. Related Activities 68
D. Change History 71
E. Acknowledgments 73

1. Motivation

This section is informative.

RDF/XML [RDF-SYNTAX [p.68]] provides sufficient flexibility to represent all of the abstract concepts in RDF [RDF-CONCEPTS [p.68]]. However, it presents two challenges; first it is difficult or impossible to validate documents that contain RDF/XML using XML Schemas or DTDs, which makes it difficult to import RDF/XML into other markup languages. Whilst newer schema languages such as RELAX NG [RELAXNG] [p.69] do provide a way to validate documents that contain arbitrary RDF/XML, it will be a while before they gain wide support.

Second, even if one could add RDF/XML directly into an XML dialect like XHTML, there would be significant data duplication between the rendered data and the RDF/XML structured data. It would be far better to add RDF to a document without repeating the document's existing data. For example, an XHTML document that explicitly renders its author's name in the text—perhaps as a byline on a news site—should not need to repeat this name for the RDF expression of the same concept: it should be possible to supplement the existing markup in such a way that it can also be interpreted as RDF.

Third, as users often want to transfer structured data from one application to another, sometimes to or from a non-web-based application, it is highly beneficial to express the web data's structure "in context." The user experience could then be enhanced, for example by providing contextual information about specific rendered data, perhaps when the user "right-clicks" on an item of interest.

In the past, many attributes were 'hard-wired' directly into the markup language to represent specific concepts. For example, in XHTML 1.1 [XHTML11 [p.67]] and HTML [HTML4 [p.67]] there is @cite; the attribute allows an author to add information to a document which is used to indicate the origin of a quote.

However, these 'hard-wired' attributes make it difficult to define a generic process for extracting metadata from any document since a parser would need to know about each of the special attributes. One motivation for RDFa has been to devise a means by which documents can be augmented with metadata in a general rather than hard-wired manner. This has been achieved by creating a fixed set of attributes and parsing rules, but allowing those attributes to contain properties from any of a number of the growing range of available RDF vocabularies. The *values* of those properties are in most cases the information that is already in an author's XHTML document.

RDFa alleviates the pressure on XML format authors to anticipate all the structural requirements users of their format might have, by outlining a new syntax for RDF that relies only on XML attributes. This specification deals specifically with the use of RDFa in XHTML, and defines an RDF mapping for a number of XHTML attributes, but RDFa can be easily imported into other XML-based markup languages.

2. Syntax Overview

This section is informative.

The following examples are intended to help readers who are not familiar with RDFa to quickly get a sense of how it works. For a more thorough introduction, please read the RDFa Primer [RDFaPRIMER [p.68]].

2.1. The RDFa Attributes

RDFa in XHTML makes use of a number of XHTML attributes, as well as providing a few new ones. Attributes that already exist in XHTML will have the same meaning as in XHTML, although their syntax may be slightly modified. For example, in XHTML, @rel already defines the relationship between one document and another. However, in XHTML there is no clear way to add new values; RDFa sets out to explicitly solve this problem, and does so by allowing URIs as values. It also introduces the idea of 'compact URIs'—referred to as CURIEs in this document—which allow a full URI value to be expressed succinctly.

The XHTML attributes that are relevant are:

@rel,
: a whitespace separated list of CURIE [p.43] s, used for expressing relationships between two resources (a 'predicate' in RDF)

@rev,
: a whitespace separated list of CURIE [p.43] s, used for expressing reverse relationships between two resources (also a 'predicate')

@href,
: a URI for expressing the partner resource of a relationship (the 'object' in RDF)

@src,
: a URI for expressing the partner resource of a relationship when the resource is embedded (also an 'object')

The new—RDFa-specific—attributes are:

@about,
: a URIorCURIE [p.43] , used for stating what the data is about (the 'subject' in RDF terminology)

@property,
: a whitespace separated list of CURIE [p.43] s, used for expressing relationships between the subject and some literal text (also a 'predicate')

@resource,
: a URIorCURIE [p.43] for expressing the partner resource of a relationship that is not intended to be 'clickable' (also an 'object')

@datatype,
: a CURIE [p.43] representing a datatype, to express the datatype of a literal

@content,
: a string, for supplying alternative, machine-readable content for a literal.

@instanceof
: a whitespace separated list of CURIE [p.43] s that indicate the RDF type(s) to associate with the subject.

For a normative definition of these attributes see the XHTML Metainformation Attributes Module [p.43] .

2.2. Examples

As an XHTML author you will already be familiar with using `meta` and `link` to add additional information to your documents:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Page 7</title>
    <meta name="author" content="Mark Birbeck" />
    <link rel="prev" href="page6.html" />
    <link rel="next" href="page8.html" />
  </head>
  <body>...</body>
</html>
```

RDFa makes use of this concept, enhancing it with the ability to make use of other vocabularies by using compact URIs:

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
>
  <head>
    <title>My home-page</title>
    <meta property="dc:creator" content="Mark Birbeck" />
    <link rel="foaf:workplaceHomepage" href="http://www.formsPlayer.com/" />
  </head>
  <body>...</body>
</html>
```

Although not widely used, XHTML already supports the use of @rel and @rev on the `a` element. This becomes more useful in RDFa with the addition of support for different vocabularies:

```
This document is licensed under a
<a xmlns:cc="http://creativecommons.org/licenses/"
  rel="cc:license"
  href="http://creativecommons.org/licenses/by/nc-nd/3.0/">
  Creative Commons License
</a>.
```

Not only can URLs in the document be re-used to provide metadata, but so can inline text:

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:cal="http://www.w3.org/2002/12/cal/ical#"
  >
  <head><title>Jo's Friends and Family Blog</title></head>
  <body>
    <p>
      I'm holding
      <span property="cal:summary">
        one last summer Barbecue
      </span>,
      on September 16th at 4pm.
    </p>
  </body>
</html>
```

If some displayed text is different to the actual 'value' it represents, more precise values can be added, which can optionally include datatypes:

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:cal="http://www.w3.org/2002/12/cal/ical#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  >
  <head><title>Jo's Friends and Family Blog</title></head>
  <body>
    <p>
      I'm holding
      <span property="cal:summary">
        one last summer Barbecue
      </span>,
      on
      <span property="cal:dtstart" content="20070916T1600-0500"
            datatype="xsd:datetime">
        September 16th at 4pm
      </span>.
    </p>
  </body>
</html>
```

In many cases a block of mark-up will contain a number of properties that relate to the same item; it's possible with RDFa to indicate the type of that item:

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:cal="http://www.w3.org/2002/12/cal/ical#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  >
  <head><title>Jo's Friends and Family Blog</title></head>
  <body>
    <p instanceof="cal:Vevent">
      I'm holding
      <span property="cal:summary">
        one last summer Barbecue
```

```
      </span>,
      on
      <span property="cal:dtstart" content="20070916T1600-0500"
            datatype="xsd:datetime">
        September 16th at 4pm
      </span>.
    </p>
  </body>
</html>
```

The metadata features available in XHTML only allow information to be expressed about the document itself. RDFa provides a means of referring to other documents and resources:

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:bib="http://example.org/"
>
  <head>
    <title>Books by Marco Pierre White</title>
  </head>
  <body>
    I think
    <span about="urn:ISBN:0091808189" instanceof="bib:book">
      White’s book ’Canteen Cuisine’
    </span>
    is well worth getting since although it’s quite advanced stuff, he
    makes it pretty easy to follow. You might also like his
    <span about="urn:ISBN:1596913614" instanceof="bib:book">
      autobiography
    </span>.
  </body>
</html>
```

3. RDF Terminology

This section is informative.

The previous section gave examples of typical mark-up in order to illustrate what RDFa in XHTML looks like. But what RDFa in XHTML *represents* is RDF. In order to author RDFa in XHTML you do not need to understand RDF, although it would certainly help. However, if you are building a system that consumes the RDF output of an RDFa in XHTML document you will almost certainly need to understand RDF. In this section we introduce the basic concepts and terminology of RDF. For a more thorough explanation of RDF, please refer to the RDF Concepts document [RDF-CONCEPTS [p.68]] and tne RDF Sytax Document [RDF-SYNTAX [p.68]].

3.1. Statements

The structured data that RDFa provides access to is a collection of *statements*. A statement is a basic unit of information that has been constructed in a specific format to make it easier to process. In turn, by breaking large sets of information down into a collection of statements, even very complex metadata can be processed using simple rules.

To illustrate, suppose we have the following set of facts:

```
Albert was born on March 14, 1879, in Germany. There is a picture of him at
the web address, http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg.
```

This would be quite difficult for a machine to interpret, and it is certainly not in a format that could be passed from one data application to another. However, if we convert the information to a set of statements it begins to be more manageable. The same information could therefore be represented by the following shorter 'statements':

```
Albert was born on March 14, 1879.
Albert was born in Germany.
Albert has a picture at
  http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg.
```

3.2. Triples

To make this information machine-processable, RDF defines a structure for these statements. A statement is formally called a *triple*, meaning that it is made up of three components. The first is the *subject* of the triple, and is what we are making our statements *about*. In all of these examples the subject is 'Albert'.

The second part of a triple is the property of the subject that we want to define. In the examples here, the properties would be 'was born on', 'was born in', and 'has a picture at'. These are more usually called *predicates* in RDF.

The final part of a triple is called the *object*. In the examples here the three objects have the values 'March 14, 1879', 'Germany', and 'http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg'.

3.3. URI references

Breaking complex information into manageable units helps us be specific about our data, but there is still some ambiguity. For example, which 'Albert' are we talking about? If another system has more facts about 'Albert', how could we know whether they are about the same person, and so add them to the list of things we know about that person? If we wanted to find people born in Germany, how could we know that the predicate 'was born in' has the same purpose as the predicate 'birthplace' that might exist in some other system? RDF solves this problem by replacing our vague terms with *URI references*.

URIs are most commonly used to identify web pages, but RDF makes use of them as a way to provide unique identifiers for concepts. For example, we could identify the subject of all of our statements (the first part of each triple) by using the DBPedia [?ref] URI for Albert Einstein, instead of the ambiguous string 'Albert':

```
<http://dbpedia.org/resource/Albert_Einstein>
   has the name
   Albert Einstein.
<http://dbpedia.org/resource/Albert_Einstein>
   was born on
   March 14, 1879.
<http://dbpedia.org/resource/Albert_Einstein>
   was born in
   Germany.
<http://dbpedia.org/resource/Albert_Einstein>
   has a picture at
   http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg.
```

URI references are also used to uniquely identify the objects in metadata statements (the third part of each triple). The picture of Einstein is already a URI, but we could also use a URI to uniquely identify the country Germany. At the same time we'll indicate that the name and date of birth really are literals (and not URIs), by putting quotes around them:

```
<http://dbpedia.org/resource/Albert_Einstein>
   has the name
   "Albert Einstein".
<http://dbpedia.org/resource/Albert_Einstein>
   was born on
   "March 14, 1879".
<http://dbpedia.org/resource/Albert_Einstein>
   was born in
   <http://dbpedia.org/resource/Germany>.
<http://dbpedia.org/resource/Albert_Einstein>
   has a picture at
   <http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg>.
```

URI references are also used to ensure that predicates are unambiguous; now we can be sure that 'birthplace', 'place of birth', 'Lieu de naissance' and so on, all mean the same thing:

```
<http://dbpedia.org/resource/Albert_Einstein>
  <http://xmlns.com/foaf/0.1/name>
  "Albert Einstein".
<http://dbpedia.org/resource/Albert_Einstein>
  <http://dbpedia.org/property/dateOfBirth>
  "March 14, 1879".
<http://dbpedia.org/resource/Albert_Einstein>
  <http://dbpedia.org/property/birthPlace>
  <http://dbpedia.org/resource/Germany>.
<http://dbpedia.org/resource/Albert_Einstein>
  <http://xmlns.com/foaf/0.1/depiction>
  <http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg>.
```

3.4. Plain literals

Although URI resources are always used for subjects and predicates, the object part of a triple can be either a URI or a *literal*. In the example triples, Einstein's name is represented by a *plain literal*, which means that it is a basic string with no type or language information:

```
<http://dbpedia.org/resource/Albert_Einstein>
  <http://xmlns.com/foaf/0.1/name> "Albert Einstein".
```

3.5. Typed literals

Some literals, such as dates and numbers, have very specific meanings, so RDF provides a mechanism for indicating the type of a literal. A *typed literal* is indicated by attaching a URI to the end of a plain literal, and this URI indicates the literal's datatype. This URI is usually based on datatypes defined in the XML Schema Datatypes specification [XMLSCHEMA [p.68]]. The following syntax would be used to unambiguously express Einstein's date of birth as a literal of type `http://www.w3.org/2001/XMLSchema#date`:

```
<http://dbpedia.org/resource/Albert_Einstein>
  <http://dbpedia.org/property/dateOfBirth>
  "1879-03-14"^^<http://www.w3.org/2001/XMLSchema#date>.
```

3.6. N-Triples

RDF itself does not have one set way to express triples, since the key ideas of RDF are the triple and the use of URIs, and *not* any particular syntax. However, there are a number of mechanisms, such as RDF/XML, N-Triples [N-TRIPLES [p.68]], and of course RDFa. Most discussions of RDF make use of the *N-Triple* syntax to explain their ideas, since it is quite compact. The examples we have just seen are already using this syntax, and we'll continue to use it throughout this document when we need to talk about the RDF that could be generated from some RDFa. There is one small change that we make to N-Triples, which is to allow long URIs to be abbreviated by using a URI mapping. This is indicated by expressing a compact URI as follows:

```
<http://dbpedia.org/resource/Albert_Einstein>
  foaf:name "Albert Einstein" .
<http://dbpedia.org/resource/Albert_Einstein>
  p:dateOfBirth "1879-03-14"^^xsd:date .
<http://dbpedia.org/resource/Albert_Einstein>
  p:birthPlace <http://dbpedia.org/resource/Germany>.
<http://dbpedia.org/resource/Albert_Einstein>
  foaf:depiction <http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg>.
```

Here 'p:' has been mapped to the URI for DBPedia, and 'foaf:' has been mapped to the URI for the 'Friend of a Friend' taxonomy.

Note that this is merely a way to make examples more compact and the actual triples generated would always use the full URIs.

When writing examples, you will often see the following URI:

```
<>
```

This indicates the 'current document', i.e., the document being processed. In reality there would always be a full URI, but this serves to make examples more compact.

3.7. Graphs

A collection of triples is called a *graph*.

For more information on the concepts described above, see [RDF-CONCEPTS [p.68]]. RDFa additionally defines the following terms:

3.8. Compact URIs

In order to allow for the compact expression of RDF statements, RDFa allows the contraction of all [URI reference]s into a form called a 'compact URI', or [CURIE]. Until recently QNames [QNames] have been the most common way to abbreviate URIs, but there is a well-known limitation that the syntax for QNames does not allow all possible [URI reference]s to be expressed. CURIEs have been specificially designed to look like QNames, but at the same time to get around their syntactic limitations.

Note that CURIEs are only used in the mark-up and N-Triples examples, and will never appear in the generated [triple]s, which will always use [URI reference]s.

3.9. A description of RDFa in RDF terms

The following is a brief description of RDFa in terms of the RDF terminology introduced here. It may be useful to readers with an RDF background:

The aim of RDFa is to allow a single [RDF graph]s to be carried in XML documents of any type, although this specification deals only with RDFa in XHTML. An [RDF graph] comprises [node]s linked by relationships. The basic unit of a graph is a [triple], in which a subject [node] is linked to an object [node] via a [predicate]. The subject [node] is always either an [URI reference] or a [blank node], the predicate is *always* an [URI reference], and the object of a statement can be an [URI reference], a [literal], or a [blank node].

In RDFa, a subject [URI reference] is generally indicated using @about, and predicates are represented using one of @property, @rel, or @rev. Objects which are [URI reference]s are represented using @href, @resource or @src, whilst objects that are [literal]s are represented either with @content (with an optional [datatype] expressed using @datatype), or the content of the element in question.

4. Conformance Requirements

This section is normative.

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119 [p.67]].

Note that all examples in this document are informative, and are not meant to be interpreted as normative requirements.

4.1. Document Conformance

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

1. The document must conform to the constraints expressed in the schemas in Appendix B - XHTML+RDFa Document Type Definition [p.51] .
2. The local part of the root element of the document must be `html`.
3. The start tag of the root element of the document must explicitly contain an `xmlns` declaration for the XHTML namespace [XMLNAMES [p.68]]. The namespace URI for XHTML is defined to be `http://www.w3.org/1999/xhtml`.

   Sample root element

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" >
   ```

4. There MUST be a DOCTYPE declaration in the document prior to the root element. If present, the public identifier included in the DOCTYPE declaration must reference the DTD found in Appendix B - XHTML+RDFa Document Type Definition [p.51] using its Public Identifier. The system identifier may be modified appropriately.

   ```
   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
       "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
   ```

Example of an XHTML+RDFa 1.0 document

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
    "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd" >
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>Moved to <a href="http://example.org/">example.org</a>.</p>
  </body>
</html>
```

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

XHTML 1.1 documents SHOULD be labeled with the Internet Media Type "application/xhtml+xml" as defined in [RFC3236 [p.67]]. For further information on using media types with XHTML, see the informative note [XHTMLMIME [p.69]].

4.2. User Agent Conformance

A conforming user agent MUST support all of the features required in this specification. A conforming user agent must also support the User Agent conformance requirements as defined in XHTML Modularization [XHTMLMOD [p.68]] section on "XHTML Family User Agent Conformance".

4.3. RDFa Processor Conformance

A conforming RDFa Processor MUST make available to a consuming application a single RDF [graph] containing all possible triples generated by using the rules in the Processing Model [p.23] section. This is the ’default [graph]’.

A conforming RDFa Processor MAY make available additional triples that have been generated using rules not described here, but these triples MUST be made available in one or more additional RDF [graph]s, and not in the default [graph].

A conforming RDFa Processor MUST process whitespace according to the rules of [CSS2 [p.67]]. *Note that this same requirement is imposed upon conforming User Agents via [XHTMLMOD [p.68]].*

Test Suite

We have a test suite - we should likely reference it here, but I need to find the exact way to do that. -Shane

Assertion Annotation

All the assertions in this spec need to be annotated with the appropriate markup (must, should, etc.).

5. Processing Model

This section is normative.

This section looks at a generic set of processing rules for creating a set of triples that represent the structured data present in an XHTML+RDFa document. Processing need not follow the DOM traversal technique outlined here, although the effect of following some other manner of processing must be the same as if the processing outlined here were followed. The processing model is explained using the idea of DOM traversal which makes it easier to describe (particularly in relation to the 'evaluation context').

5.1. Overview

Parsing a document for RDFa triples is carried out by starting at the document object, and then visiting each of its child elements in turn, applying processing rules. Processing is recursive in that for each child element the processor also visits each of *its* child elements, and applies the same processing rules.

(Note that in some environments there will be little difference between starting at the root element of the document, and starting at the document object itself. However, we define it this way since in some environments important information is present at the document object level which is not present on the root element.)

As processing continues, rules are applied which will either generate triples, or change the [evaluation context] information which will be used in subsequent processing. Some of the rules will be determined by the host language—in this case XHTML—and some of the rules will be part of RDFa.

Note that we don't say anything about what should happen to the triples generated, or whether more triples might be generated during processing than are outlined here. However, to be conformant, an RDFa processor needs to act as if at a minimum the rules in this section are applied.

5.2. Evaluation Context

During processing, each rule is applied within an 'evaluation context'. Rules may further modify this evaluation context, or create triples that can be established by making use of this evaluation context. The context itself consists of the following pieces of information:

- The [base]. This will usually be the URL of the document being processed, but it could be some other URL, set by some other mechanism, such as the XHTML `base` element. The important thing is that it establishes a URL against which relative paths can be evaluated.
- The [current resource]. The initial value will be the same as the initial value of `base`, but it will usually change during the course of processing.
- The [current element identifier]. This is an identifier for the current element, which will not always be the same as [current resource].

- A list of current in-scope [URI mappings].
- The [language]. Note that there is no default language.

In addition, during the course of processing a number of locally scoped values are needed, as follows:

- A [recurse] flag. Processing generally continues recursively through the entire tree of elements available. However, if an author indicates that some branch of the tree should be treated as an XML literal, no further processing should take place on that branch, and this flag is used to indicate this.
- A flag to indicate [chaining], the process by which a collection of statements take as their [subject] a [URI reference] which was the [object] of some ancestor statement.
- A value for the [current object literal], the literal to use when creating triples that use a literal value.
- A value for the [current object resource], the resource to use when creating triples that use a literal value.

Note that these locally scoped values do not need to be preserved when traversing the tree.

5.3. Processing

Processing would normally begin after the document to be parsed has been completely loaded. However, there is no requirement for this to be the case, and it is certainly possible to use a stream-based approach, such as SAX [http://en.wikipedia.org/wiki/SAX] to extract the RDFa information. However, if some approach other than the DOM traversal technique defined here is used, it is important to ensure that any `meta` or `link` elements processed in the `head` of the document honour any occurrences of `base` which may appear *after* those elements. (In other words, XHTML processing rules must still be applied, even if document processing takes place in a non-HTML environment such as a search indexer.)

At the beginning of processing, the [current evaluation context] is initialised as follows:

- the [base] is set to either the URL of the document or the value specified in the `base` element, if present;
- the [current resource] is set to the [base] value;
- the [current element identifier] is cleared;
- the [list of URI mappings] is cleared;
- the [language] is cleared.

Processing then begins with the document object, and all elements in the tree are processed according to the following rules, depth-first:

1. First, some of the local values are initialised, as follows:
   - the [recurse] flag is set to true;
   - the [chaining] flag is set to false.
2. Any changes to the [current evaluation context] are made next:

- the [current element] is parsed for [URI mappings] and these are added to the [list of URI mappings]. Note that a [URI mapping] will simply overwrite any current mapping in the list that has the same name;
  These mappings are provided by @xmlns. The value to be mapped is set by the XML namespace prefix, and the value to map is the value of the attribute—a URI. Note that the URI is not processed in any way; in particular if it is a relative path it is not resolved against the [current base]. Authors are advised to follow best practice for using namespaces, which includes not using relative paths. (See [xyz].)
- the [current element] is parsed for any language information, and [language] is set in the [current evaluation context];
  Language information can be provided using the general-purpose XML attribute @xml:lang.
- the [current element] is parsed for any subject information, and it is used to set the [current resource] value, in the [current evaluation context];
  The [current resource] can be set using @about. Note that the final value of the [current resource] is an absolute IRI, which means that if @about contains a relative path the value must be normalised against [base] in the [current evaluation context], using the algorithm defined in RFC 3986. The value can also be provided by a CURIE, and is processed as defined in CURIE Syntax Definition [p.39] . Note also that since this attribute can take both URIs and [curie]s, the latter will have been expressed using the [safe CURIE] syntax.
- the [current element identifier] is set;

  The [current element identifier] is set to the value of @about, if present, *or* the value of @resource, if present, *or* the value of @href, if present, *or* the value of @src, if present, *or* a [blank node]. This means that the value of the [current element identifier] will not necessarily be the same as the [current resource], since in the absence of an attribute to set its value it will be set to a [blank node], whilst [current resource] inherits from its context.

  Note that unless the final value of [current element identifier] is a [blank node] then it will be an absolute IRI, which means that any relative paths must be normalised against [base] in the [current evaluation context], using the algorithm defined in RFC 3986. The value might also have been provided as a [curie], and if so, it is processed as defined in CURIE Syntax Definition [p.39] .

3. Once the [current evaluation context] and the local values have been set, object resolution is carried out, as follows:
   - the [current object resource] is established;
     Since only one [current object resource] is set per element then some attributes will have a higher priority than others. The highest priority is given to @resource. If there is no @resource then @href is used, and if that is not present, @src is used. If none of these are present then a unique identifier or [blank node] is created. Note that unless the final value of the [current object resource] is a [blank node] then it will be an absolute URI, which means that any relative paths must be normalised against [base] in the [current evaluation context], using the algorithm defined in RFC 3986. The value might also have been provided as a [curie], and if so, it is processed as defined in

CURIE Syntax Definition [p.39] .

- the [current object literal] is established;
  The [current object literal] will be set as a [plain literal] if @content is present, *or* all children of the [current element] are text nodes, *or* there are no child nodes, *or* the body of the [current element] *does* have non-text child nodes but @datatype is present, with an empty value. Additionally, if there is a value for [current language] then the value of the [plain literal] should include this language information, as described in [RDFCONCEPTS]. The actual literal is either the value of @content (if present) *or* a string created by concatenating the text content of each of the child elements of the [current element] in document order, and then normalising white-space according to [WHITESPACERULES].

  Whitespace normalising

  So far we defer to CSS2, but I think we should copy the prose into here, so that it's clearer.
  The [current object literal] will be set as an [XML literal] if the [current element] has any child nodes that are not simply text nodes, and @datatype is not present, or is set to `rdf:XMLLiteral`. The value of the [XML literal] is a string created from the inner content of the [current element], i.e., not including the element itself, with the datatype of `rdf:XMLLiteral`.
  The [current object literal] will be set as a [typed literal] if @datatype is present, and does not have an empty value. The actual literal is either the value of @content (if present) *or* a string created by concatenating the inner content of each of the child elements in turn, of the [current element]. The final string includes the datatype, as described in [RDFCONCEPTS].

4. Once object resolution is complete the processor will have two objects, one a resource and the other a literal. These objects can now be used to create triples with the [current resource], depending on the presence of other attributes. This is achieved using the following processing steps:
   - Predicates for the [current object literal] are established;
     Predicates for the [current object literal] can be set by using @property. If present, the attribute must contain one or more [curie]s, each of which is converted to an absolute URI using CURIE processing rules as defined in CURIE Syntax Definition [p.39] , and then used to generate a triple as follows:
     subject
       [current resource]
     predicate
       expanded value from the curie
     object
       [current object literal]
     Note that *literal* may include language and datatype information as discussed in the section on object resolution. Once the triple has been created, the [recurse] flag is set to false.
   - predicates for the [current object resource] are established. If any triples are generated then the [chaining] flag is set to `true`.;
     Predicates for the [current object resource] can be set by using one or both of the @rel

and @rev attributes. If present, @rel must contain one or more [CURIE]s, each of which is converted to an absolute URI using CURIE processing rules as defined in CURIE Syntax Definition [p.39] , and then used to generate a triple as follows:

subject
: [current resource]

predicate
: expanded value from the curie

object
: [current object resource]

If present, @rev must contain one or more [CURIE]s, each of which is converted to an absolute URI using CURIE processing rules as defined in CURIE Syntax Definition [p.39] , and then used to generate a triple as follows:

subject
: [current object resource]

predicate
: expanded value from the curie

object
: [current resource]

5. Type values for the [current element identifier] are established. If any triples are generated then the [chaining] flag is set to `true`.;
   One or more 'types' for the [current element identifier] can be set by using @instanceof. If present, the attribute must contain one or more [curie]s, each of which is converted to an absolute URI using CURIE processing rules, and then used to generate a triple as follows:
   subject
   : [current element identifier]

   predicate
   : http://www.w3.org/1999/02/22-rdf-syntax-ns#type

   object
   : expanded value from the curie
6. If the [chaining] flag is set to `true` then the [current resource] is set to the value of the [current object resource], and the [chaining] flag is set to `false`.
7. If the [recurse] flag is true, all elements that are children of the [current element] are processed using the rules described here. Before processing a child element the [current evaluation context] is preserved, and after processing the child the context is restored.

6. RDFa Processing in detail

This section is normative.

This section provides an in-depth examination of the processing steps described in the previous section. It also includes examples which may help clarify some of the steps involved.

@instanceof situation

This section still needs the detail on whether @instanceof should use @about if it is present, or use the subject from chaining.

NOTE: There isn't quite enough detail on chaining yet.

In the following examples, for brevity assume that the following namespace prefixes have been defined:

cc:	http://creativecommons.org/ns#
dc:	http://purl.org/dc/elements/1.1/
ex:	http://example.org/
foaf:	http://xmlns.com/foaf/0.1/
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
p:	http://dbpedia.org/property/
rdfa:	http://www.w3.org/ns/rdfa/
svg:	http://www.w3.org/2000/svg
xh11:	http://www.w3.org/1999/xhtml
xsd:	http://www.w3.org/2001/XMLSchema#
biblio:	http://example.org/biblio/0.1
taxo:	http://purl.org/rss/1.0/modules/taxonomy/

The key to processing is that a triple is generated whenever a predicate/object combination is detected. The actual triple generated will include a subject that may have been set previously, so this is tracked in the [current evaluation context] and is called the [current resource]. Since the subject will default to the current document if it hasn't been set explicitly, then a predicate/object combination is always enough to generate one or more triples.

The attributes for setting a predicate are @rel, @rev and @property, whilst the attributes for setting an object are @resource, @href, @content, and @src. Note that @instanceof sets both a predicate and object at the same time.

6.1. Changing the evaluation context

6.1.1. Setting the [current resource]

When triples are created they will always be in relation to the [current resource]. When parsing begins the [current resource] will be the URI of the document being parsed, or a value as set by `base`.

Metadata about the document itself is usually placed in the `head`:

```
<html>
  <head>
    <title>Jo's Friends and Family Blog</title>
    <link rel="foaf:primaryTopic" href="#bbq" />
    <meta property="dc:creator" content="Jo" />
  </head>
  <body>
    ...
  </body>
</html>
```

although it is possible for the data to appear elsewhere:

```
<html>
  <head>
    <title>Jo's Blog</title>
  </head>
  <body>
    <h1><span property="dc:creator">Jo</span>'s blog</h1>
    <p>
      Welcome to my blog.
    </p>
  </body>
</html>
```

The value of `base` may change the initial value of [current resource]:

```
<html>
  <head>
    <title>Jo's Friends and Family Blog</title>
    <link rel="foaf:primaryTopic" href="#bbq" />
    <meta property="dc:creator" content="Jo" />
    <base href="http://www.example.org/jo/blog" />
  </head>
  <body>
    ...
  </body>
</html>
```

As processing progresses, any @about attributes will change the [current resource]. The value of @about is a URI or a CURIE. If it is a relative URI then it needs to be resolved against the current [base] value. In this mark-up the properties `cal:summary` and `cal:dtstart` become part of the ’event’ object, rather than referring to the document:

```
<html>
  <head>
    <title>Jo’s Friends and Family Blog</title>
    <link rel="foaf:primaryTopic" href="#bbq" />
    <meta property="dc:creator" content="Jo" />
  </head>
  <body>
    <p about="#bbq" instanceof="cal:Vevent">
      I’m holding
      <span property="cal:summary">
        one last summer Barbecue
      </span>,
      on
      <span property="cal:dtstart" content="20070916T1600-0500"
            datatype="xsd:datetime">
        September 16th at 4pm
      </span>.
    </p>
  </body>
</html>
```

Other kinds of resources can be used to set the [current resource], not just references to web-pages:

```
Daniel knows
<a about="mailto:daniel.brickley@bristol.ac.uk"
  rel="foaf:knows" href="mailto:libby.miller@bristol.ac.uk">Libby</a>.

Libby knows
<a about="mailto:libby.miller@bristol.ac.uk"
  rel="foaf:knows" href="mailto:ian.sealy@bristol.ac.uk">Ian</a>.

<div about="photo1.jpg">
  <span class="attribution-line">this photo was taken by
    <span property="dc:creator">Mark Birbeck</span>
  </span>
</div>
```

6.2. Object resolution

There are two types of object, [URI resource]s and [literal]s.

A [literal] object can be set by using @property to express a [predicate], and then using either @content, or the inline text of the element that @property is on.

A [URI resource] object can be set using one of @rel or @rev to express a [predicate], and then using one of @href, @resource or @src to provide the object.

6.2.1. Literal object resolution

An [object literal] will be generated when @property is present. @property provides the predicate, and the following sections describe how the actual literal to be generated is determined.

6.2.1.1. Plain Literals

@content can be used to indicate a [plain literal], as follows:

```
<meta about="http://internet-apps.blogspot.com/"
      property="dc:creator" content="Mark Birbeck" />
```

The [plain literal] can also be specified by using the content of the element:

```
<span about="http://internet-apps.blogspot.com/"
      property="dc:creator">Mark Birbeck</span>
```

Both of these examples give the following triple:

```
  <http://internet-apps.blogspot.com/>
    dc:creator "Mark Birbeck" .
```

The value of @content attribute is given precedence over any element content, so the following would give exactly the same triple:

```
<span about="http://internet-apps.blogspot.com/"
      property="dc:creator" content="Mark Birbeck">John Doe</span>
```

6.2.1.1.1. Language Tags

RDF allows [plain literal]s to have a language tag, as illustrated by the following example from [RDFTESTS-RDFMS-XMLLANG-TEST006] [p.68] :

```
<http://example.org/node>
  <http://example.org/property> "chat"@fr .
```

In RDFa the XML language attribute @xml:lang is used to add this information, whether the plain literal is designated by @content, or by the inline text of the element:

```
<meta about="http://example.org/node"
  property="ex:property" xml:lang="fr" content="chat" />
```

Note that the language value can be inherited as defined in [XML-LANG] [p.67] , so the following syntax will give the same triple as above:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ex="http://www.example.com/ns/" xml:lang="fr">
  <head>
    <title xml:lang="en">Example</title>
    <meta about="http://example.org/node"
      property="ex:property" content="chat" />
  </head>
  ...
</html>
```

6.2.1.2. Typed literals

[Literal]s can be given a data type using @datatype.

This can be represented in RDFa as follows:

```
<span property="cal:dtstart" content="20070916T1600-0500"
      datatype="xsd:datetime">
  September 16th at 4pm
</span>.
```

The triples that this mark-up generates include the datatype after the literal:

```
<>
  cal:dtstart "20070916T1600-0500"^^xsd:datetime .
```

6.2.1.3. XML Literals

XML documents cannot contain XML mark-up in their attributes, which means it is not possible to represent XML within @content (the following would cause an XML parser to generate an error):

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

It does not help to escape the content, since the output would simply be a string of text containing numerous ampersands:

```
<>
dc:title "E = mc&lt;sup&gt;2&amp;lt;/sup&gt;: The Most Urgent Problem of Our Time" .
```

RDFa therefore supports the use of normal mark-up to express XML literals, by using @datatype:

```
    <h2 property="dc:title" datatype="rdf:XMLLiteral">
      E = mc<sup>2</sup>: The Most Urgent Problem of Our Time
    </h2>
```

This would generate the following triple, with the XML preserved in the literal:

```
<>
  dc:title "E = mc<sup>2</sup>: The Most Urgent Problem of Our Time"^^rdf:XMLLiteral .
```

Note that this requires that a URI mapping for the prefix `rdf` has been defined. To make authoring easier, if there are child elements and no @datatype attribute, then the effect is the same as if @datatype have been explicitly set to `rdf:XMLLiteral`:

```
<h2 property="dc:title">
  E = mc<sup>2</sup>: The Most Urgent Problem of Our Time
</h2>
```

In the examples we've been using the `sup` element is actually part of the meaning of the literal, but there will be situations where the extra mark-up means nothing, and can therefore be ignored. In this situation an empty @datatype value can be used to override the XML literal behaviour:

```
<p>You searched for <strong>Einstein</strong>:</p>
<p about="http://dbpedia.org/resource/Albert_Einstein">
  <span property="foaf:name" datatype="">Albert <strong>Einstein</strong></span>
  (March 14, 1879 â&#8364;" April 18, 1955) was a German-born theoretical physicist.
</p>
```

Although the rendering of this page has highlighted the term the user searched for, setting @datatype to nothing ensures that the data is interpreted as a plain literal, giving the following triples:

```
<http://dbpedia.org/resource/Albert_Einstein>
  foaf:name "Albert Einstein" .
```

Note that the value of this [XML Literal] is the exclusive canonicalization of the RDFa element's value.

Although the RDFa processing model requires visiting each element in the tree, if the processor meets an [XML literal] then it MUST NOT process any further down the tree. This is to prevent triples being generated from mark-up that is not actually in the hierarchy. For example, we might want to set the title of something to some XHTML that includes RDFa:

```
<h2 property="dc:title">
  Example 3: <span about="#bbq" instanceof="cal:Vevent">...</span>
</h2>
```

This does effectively mean that the presence of @property without @content will inhibit any further processing, so authors should watch out for stray attributes, especially if they find that they are getting fewer triples than they had expected.

6.2.2. URI object resolution

Slightly out of sync

The contents of this small sub-section are slightly out of sync with current task force thinking, and will be updated in the next draft.

One or more [URI object]s are needed when @rel or @rev is present. Each attribute will cause triples to be generated when used with @href, @resource or @src.

@rel and @rev are essentially the inverse of each other; whilst @rel establishes a relationship between the [current resource] as subject, and the [object resource] as the object, @rev does the exact opposite, and uses the [object resource] as the subject, and the [current resource] as the object.

6.2.2.1. Using @resource to set the object

RDFa provides the @resource attribute as a way to set the object of statements. This is particularly useful when referring to resources that are not themselves navigable links:

```
<html>
  <head>
    <title>On Crime and Punishment</title>
    <base href="http://www.example.com/candp.xhtml" />
  </head>
  <body>
    <blockquote about="#q1" rel="dc:source" resource="urn:isbn:0140449132" >
      <p id="q1">
        Rodion Romanovitch! My dear friend! If you go on in this way
        you will go mad, I am positive! Drink, pray, if only a few drops!
      </p>
    </blockquote>
  </body>
</html>
```

The `blockquote` element generates the following triple:

```
<http://www.example.com/candp.xhtml#q1>
  <http://purl.org/dc/elements/1.1/source> <urn:isbn:0140449132>.
```

6.2.2.2. Using @href

If no @resource is present, then @href can be used to set the object.

When a triple predicate has been expressed using @rel, the @href on the [RDFa statement]'s element is used to identify the object with a [URI reference]. Its type is a URI:

```
<link about="mailto:daniel.brickley@bristol.ac.uk"
      rel="foaf:knows" href="mailto:libby.miller@bristol.ac.uk" />
```

It’s also possible to use both @rel and @rev at the same time on an element. This is particularly useful when two things stand in two different relationships with each, for example when a picture is taken *by* Mark, but that picture also *depicts* him:

```
<p>This photo was taken by
<a about="photo1.jpg" rel="dc:creator" rev="foaf:img"
   href="http://www.blogger.com/profile/1109404">Mark Birbeck</a>.</p>
```

which then yields two triples:

```
<photo1.jpg>
  dc:creator <http://www.blogger.com/profile/1109404> .
<http://www.blogger.com/profile/1109404>
  foaf:img <photo1.jpg> .
```

6.2.2.3. Using @about to set the subject

```
This photo, entitled
<span about="photo1.jpg" property="dc:title">Portrait of Mark</span>
was taken by
<a about="photo1.jpg" rel="dc:creator" rev="foaf:img"
  href="http://www.blogger.com/profile/1109404">Mark himself</a>.
```

The value of @about sets the subject for any nested triples which means that the same triples can be expressed using this, more compact, syntax:

```
  <div about="photo1.jpg">
    This photo, entitled
    <span property="dc:title">Portrait of Mark</span>
    was taken by
    <a rel="dc:creator" rev="foaf:img"
      href="http://www.blogger.com/profile/1109404">Mark himself</a>.
  </div>
```

6.2.2.4. Using a blank node to set the object

When a triple predicate has been expressed using @rel, and no @href, @src, or @resource exists on the same [RDFa element], then the CURIE [p.43] represented by this element is used as the object. This CURIE [p.43] is affected by @about, but if none is present the object is a [blank node] (blank nodes are discussed further in @@@section bnode [REF]@@@). In all cases, the subject resolution for child elements is affected: where they do not override the subject, their subject is this same CURIE [p.43] here resolved as the object.

Consider, for example, a simple fragment of XHTML for describing the creator of a web page, with further information about the creator, including his name and email address:

```
<div rel="dc:creator">
  <span property="foaf:name">Ben Adida</span>
  (<a property="foaf:mbox" href="mailto:ben@adida.net">ben@adida.net</a>)
</div>
```

The above yields the following triples:

```
<>
   dc:creator _:div0 .

_:div0
   foaf:name "Ben Adida" .
_:div0
   foaf:mbox <mailto:ben@adida.net> .
```

6.2.2.4.1. Referencing Blank Nodes

To establish relationships between [blank node]s, the [unique anonymous ID] must be set explicity using a CURIE [blank node] as subject or object. For example, if our desired output is the following [triple]s:

```
_:a
  foaf:mbox <mailto:daniel.brickley@bristol.ac.uk> .
_:b
  foaf:mbox <mailto:libby.miller@bristol.ac.uk> .
_:a
  foaf:knows _:b .
```

we could use the following XHTML:

```
<link about="[_:a]" rel="foaf:mbox"
  href="mailto:daniel.brickley@bristol.ac.uk" />
<link about="[_:b]" rel="foaf:mbox"
  href="mailto:libby.miller@bristol.ac.uk" />
<link about="[_:a]" rel="foaf:knows"
  href="[_:b]" />
```

or, alternatively, if we wish to partly render the information in XHTML:

```
<div about="[_:a]">
  DanBri can be reached via
  <a rel="foaf:mbox"
    href="mailto:daniel.brickley@bristol.ac.uk">
    email
  </a>.
  <span rel="foaf:knows" resource="[_:b]">He knows Libby.</span>
</div>

<div about="[_:b]">
  Libby can be reached via
  <a rel="foaf:mbox"
    href="mailto:libby.miller@bristol.ac.uk">
    email
  </a>
</div>
```

7. CURIE Syntax Definition

This section is normative.

The key component of RDF is the URI, but they are usually long and unwieldy. RDFa therefore supports a mechanism by which URIs can be abbreviated, called 'compact URIs' or simply, CURIEs.

A CURIE is comprised of two components, a *prefix* which maps to a URI, and a *reference*. The prefix is separated from the reference by a colon (`:`). It is possible to omit the prefix, and make use of the default prefix. It is also possible to omit both the prefix *and* the colon, leaving just a *reference*.

```
curie       :=   [ [ prefix ] ':' ] reference

prefix      :=   NCName

reference   :=   irelative-ref (as defined in [IRI])
```

In some situations an attribute will allow either a CURIE, or a normal IRI. Since it is difficult to distinguish between CURIEs and IRIs, RDFa adds the notion of a [safe CURIE]. The syntax is simply to surround the CURIE with square brackets:

```
safe_curie  :=   '[' curie ']'
```

To evaluate CURIEs during processing the following context needs to be set:

- a set of mappings from prefixes to URIs;
  The prefix mappings are provided by the current in-scope namespace declarations of the [current element] during parsing.
- a mapping to use with the default prefix (for example, `:p`);
  The mapping to use with the default prefix is the current default namespace.
- a mapping to use when there is no prefix (for example, `p`);
  The mapping to use when there is no prefix is `http://www.w3.org/1999/xhtml#`.
- a mapping to use with the '_' prefix, which is used to generate unique identifiers (for example, `_:p`).
  the mapping to use with the '_' prefix is not explicitly stated, but should be chosen by the processor to ensure that there is no possibility of collision with other documents.

clarify the 'no prefix' situation

The advantage of setting the 'no prefix' mapping to the XHTML namespace is that we no longer need a preprocessing step to handle XHTML link types, such as a `next`. However, this does have the effect of moving all other values into the XHTML namespace, such as `openid.delegate`. An alternative is to prohibit unprefixed CURIEs, other than those defined by XHTML.

A CURIE is a representation of a full IRI. This IRI is obtained by taking the currently in-scope mapping that is associated with `prefix`, and concatenating it with the `reference`. The result MUST be a syntactically valid IRI [IRI [p.67]].

8. XHTML+RDFa Definition

This section is normative.

The XHTML+RDFa document type is a fully functional document type with rich semantics. It is a superset of [XHTML11 [p.67]]. See that document for the details of the underlying language.

The XHTML+RDFa 1.0 document type is made up of the following XHTML modules. The elements, attributes, and content models associated with these modules are defined in "XHTML Modularization" [XHTMLMOD [p.68]]. The elements are listed here for information purposes, but the definitions in "XHTML Modularization" should be considered authoritative. In the on-line version of this document, the module names in the list below link into the definitions of the modules within the current versions of "XHTML Modularization".

Structure Module
: `body, head, html, title`

Text Module
: `abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var`

Hypertext Module
: `a`, and @href is available on all elements.

List Module
: `dl, dt, dd, ol, ul, li`

Object Module
: `object, param`

Presentation Module
: `b, big, hr, i, small, sub, sup, tt`

Edit Module
: `del, ins`

Bidirectional Text Module
: `bdo`

Forms Module
: `button, fieldset, form, input, label, legend, select, optgroup, option, textarea`

Table Module
: `caption, col, colgroup, table, tbody, td, tfoot, th, thead, tr`

Image Module
: `img`

Client-side Image Map Module
: `area, map`

Server-side Image Map Module
: Attribute `ismap` on `img`

Intrinsic Events Module
: Events attributes

Metainformation Module
    `meta`
Scripting Module
    `noscript, script`
Stylesheet Module
    `style` element
Style Attribute Module *Deprecated*
    @style
Target Module
    @target
Link Module
    `link`
Base Module
    `base`
Metainformation Attributes Module [p.43]
    @about, @content, @datatype, @instanceof, @property

XHTML+RDFa also uses the Ruby Annotation module as defined in [RUBY [p.67]]:

Ruby Annotation Module
    `ruby, rbc, rtc, rb, rt, rp`

There are no additional definitions required by this document type. An implementation of this document type as an XML DTD is defined in Appendix B [p.51] .

9. Metainformation Attributes Module

This section is *normative*.

The Metainformation Attributes Module defines the `Metainformation` attribute collection. This collection allows elements to be annotated with metadata throughout an XHTML-family document. When this module is included in a markup language, this collection is added to the `Common` attribute collection as defined in [XHTMLMOD [p.68]].

9.1. Datatypes

Some of the attributes in this section use the following datatype:

Data type	Description
CURIE	A Compact URI or CURIE [p.39] .
URIorCURIE	A URI or safe_curie [p.39] .

9.2. Metadata Attribute Collection

9.2.1. The about attribute

This attribute specifies a URIorCURIE [p.43] that indicates which resource has a specified property.

```
<meta about="http://www.example.com/" property="dc:created">2004-03-20</meta>
```

9.2.2. The content attribute

This attribute specifies a value of type CDATA that defines the metadata associated with an element. If not specified, then the metadata for an element is its content. If it is specified, and there is no `property` attribute, then the property is considered to be `reference`.

```
<meta about="http://www.example.com/" property="dc:created" content="2004-03-20"/>
```

9.2.3. The datatype attribute

This attribute defines as a CURIE [p.43] the datatype of the content metadata of the element. If the attribute is not specified, then the default value is `string` as defined by [XMLSCHEMA [p.68]].

```
<meta about="http://www.example.com/" property="dc:created" datatype="xsd:date">2004-03-20</meta>
```

9.2.4. The instanceof attribute

This attribute indicates the `rdf:type` of the associated resource(s).

9.2.5. The property attribute

This attribute specifies a space-separated list of CURIE [p.43] s that indicates which property is being defined by the element.

```
<meta about="http://www.example.com/" property="dc:creator">John Smith</meta>
```

The list of predefined values (in the XHTML namesapce) is given below. Users may extend this collection of relationships, however new values must be defined in their own vocabulary, and the relationship names must be referenced in documents as CURIEs (e.g., `dc:creator` for the Dublin Core "creator" relationship).

```
<html ....  xmlns:dc="http://purl.org/dc/elements/1.1/">
```

description
: Gives a description of the resource.

generator
: Identifies the software used to generate the resource.

keywords
: Gives a comma-separated list of keywords describing the resource.

reference
: The default value, gives no explicit information about the relationship with the resource.

robots
: Gives advisory information intended for automated web-crawling software. This specification does not define values for this property.

title
: Specifies a title for the resource.

Note that previous versions of XHTML included an `author` property; this has now been replaced with the Dublin Core `creator` property.

Note that the @title attribute is just a shorthand for a common case:

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

is equivalent to:

```
<h2 about="#jakob" property="title">Author biography</h2>
<p><a href="Jakob.html" id="jakob">Jakob Nielsen</a>'s Alertbox for January 11, 1998</p>
```

This allows you to specify richer, marked-up text for a title when needed.

9.2.6. The rel attribute

This attribute describes the relationship between the resource specified by @about (or its default value) and the resource referred to by @href as defined in XHTML. The type for this attribute is a space-separated list of CURIE [p.43] s.

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

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

```
<link href="doc.ps"
      rel="alternate"
      media="print"
      type="application/postscript" />
```

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

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

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

Users may extend this collection of relationships. However, extensions must be defined in their own vocabulary, and the relationship names must be referenced in documents as CURIE [p.43] s (e.g., dc:creator for the Dublin Core "creator" relationship).

Note that in order to reference relationship definitions via CURIE, their prefix must be defined via an xmlns attribute somewhere suitable:

```
<html ....  xmlns:dc="http://purl.org/dc/elements/1.1/">
```

alternate
: Designates alternate versions for the document.

appendix
: Refers to a resource serving as an appendix in a collection.

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

cite
: Refers to a resource that defines a citation. In the following example, the `cite` is used to reference the book from which the quotation is taken:

  cite as book reference

```
As Gandalf the White said in
<span rel="cite" about="http://www.example.com/books/the_two_towers">
    The Two Towers
</span>,
<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
<span property="cite" about="http://www.w3.org/TR/REC-xml">[XML]</cite>.
```

chapter
: Refers to a resource serving as a chapter in a collection.

contents
: Refers to a resource serving as a table of contents.

copyright
: Refers to a copyright statement for the resource.

glossary
: Refers to a resource providing a glossary of terms.

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

icon
: Refers to a resource that represents an icon.

index
: Refers to a resource providing an index.

meta
: Refers to a resource that provides metadata, for instance in RDF.

next
: Refers to the next resource (after the current one) in an ordered collection.

p3pv1
: Refers to a P3P Policy Reference File. See [P3P [p.67]].

prev
: Refers to the previous resource (before the current one) in an ordered collection.

role
: Indicates the purpose of the resource. For some possible values, see [XHTMLROLE [p.67]] module.

section
: Refers to a resource serving as a section in a collection.

subsection
: Refers to a resource serving as a subsection in a collection.

start
: Refers to the first resource in a collection of resources. A typical use case might be a collection of chapters in a book.

need a license value

We agreed sometime ago with Creative Commons that we would have a reserve value of "license".

No end or last value

We have a value of "start", but no corresponding "end" value. Do we need one?

up

Refers to the resource "above" in a hierarchically structured set.

9.2.7. The resource attribute

This attribute takes a URIorCURIE [p.43] , and can be used to define the resource referenced by a @rel, @rev, or @property attribute. When provided, the value of @resource supercedes any value for the @href attribute on the same element.

9.2.8. The rev attribute

This attribute is the complement of the @rel attribute and describes the reverse relationship between the resource specified by the @about attribute (or its default value) and the resource referred to by the @href attribute. Its value is a space-separated list of CURIE [p.43] s. For a list of relationship names, see the @rel attribute.

```
<link href="doc.html" rev="contents"/>
```

This example states that the current document is the table of contents for the referenced document.

An implementation of this module can be found in Appendix B [p.51] .

A. Other XML Languages

This section is informative.

While outside the scope of this specification, RDFa is intended to be extensible for use in languages beyond XHTML 1.1. The module defined in Metainformation Attibutes Module [p.43] is compatible with XHTML Modularization [XHTMLMOD [p.68]], and can be used to construct other such languages.

If a language includes @xml:base [XMLBASE [p.67]], an RDFa parser for that host language must process it, and use its value to set [base].

An example follows to show how @xml:base affects the subject:

```
<span xml:base="http://internet-apps.blogspot.com/">
  <span about="" rel="dc:creator" href="http://www.blogger.com/profile/1109404" />
  <span about="" property="dc:title" content="Internet Applications" />
</span>
```

The triples generated would be as follows:

```
<http://internet-apps.blogspot.com/>
  dc:creator <http://www.blogger.com/profile/1109404> .
<http://internet-apps.blogspot.com/>
  dc:title "Internet Applications" .
```

B. XHTML+RDFa DTD

This appendix is *normative*.

This appendix includes an implementation of the XHTML+RDFa 1.0 language as an XML DTD. It is implemented by by combining the XHTML 1.1 DTD with the XHTML Metainformation Attribute Module. This is done by using a content model module, and then a driver module:

B.1. XHTML Metainformation Attributes Module

```
<!-- ...................................................................... -->
<!-- XHTML MetaAttributes Module  ......................................... -->
<!-- file: xhtml-metaAttributes-1.mod

     This is XHTML-RDFa, modules to annotate XHTML family documents.
     Copyright 2007 W3C (MIT, ERCIM, Keio), All Rights Reserved.
     Revision: $Id: rdfa-syntax.ps,v 1.11 2007/10/18 21:45:02 jigsaw Exp $

     This DTD module is identified by the PUBLIC and SYSTEM identifiers:

       PUBLIC "-//W3C//ENTITIES XHTML MetaAttributes 1.0//EN"
       SYSTEM "http://www.w3.org/MarkUp/DTD/xhtml-metaAttributes-1.mod"

     Revisions:
     (none)
     ....................................................................... -->

<!-- Common Attributes

     This module declares a collection of meta-information related
     attributes.

     %NS.decl.attrib; is declared in the XHTML Qname module.

     This file also includes declarations of "global" versions of the
     attributes.  The global versions of the attributes are for use on
     elements in other namespaces.
-->

<!ENTITY % QName.datatype "CDATA" >
<!ENTITY % QNames.datatype "CDATA" >

<!ENTITY % about.attrib
     "about        %URI.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.about.attrib
     "%XHTML.prefix;:about           %URI.datatype;        #IMPLIED"
>
]]>

<!ENTITY % instanceof.attrib
     "instanceof        %QName.datatype;             #IMPLIED"
```

```
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.instanceof.attrib
     "%XHTML.prefix;:instanceof           %QName.datatype;         #IMPLIED"
>
]]>

<!ENTITY % property.attrib
     "property        %QNames.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.property.attrib
     "%XHTML.prefix;:property           %QNames.datatype;        #IMPLIED"
>
]]>

<!ENTITY % resource.attrib
     "resource        %URI.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.resource.attrib
     "%XHTML.prefix;:resource           %URI.datatype;        #IMPLIED"
>
]]>

<!ENTITY % content.attrib
     "content        CDATA             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.content.attrib
     "%XHTML.prefix;:content           CDATA        #IMPLIED"
>
]]>

<!ENTITY % datatype.attrib
     "datatype        %QName.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.datatype.attrib
     "%XHTML.prefix;:datatype           %QName.datatype;        #IMPLIED"
>
]]>

<!ENTITY % rel.attrib
     "rel        %QNames.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.rel.attrib
     "%XHTML.prefix;:rel           %QNames.datatype;        #IMPLIED"
>
]]>
```

```
<!ENTITY % rev.attrib
     "rev        %QNames.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.rev.attrib
     "%XHTML.prefix;:rev           %QNames.datatype;        #IMPLIED"
>
]]>

<!ENTITY % Metainformation.extra.attrib "" >

<!ENTITY % Metainformation.attrib
     "%about.attrib;
      %content.attrib;
      %datatype.attrib;
      %instanceof.attrib;
      %property.attrib;
      %rel.attrib;
      %resource.attrib;
      %rev.attrib;
      %Metainformation.extra.attrib;"
>

<!ENTITY % XHTML.global.metainformation.extra.attrib "" >

<![%XHTML.global.attrs.prefixed;[

<!ENTITY % XHTML.global.metainformation.attrib
     "%XHTML.global.about.attrib;
      %XHTML.global.content.attrib;
      %XHTML.global.datatype.attrib;
      %XHTML.global.instanceof.attrib;
      %XHTML.global.property.attrib;
      %XHTML.global.rel.attrib;
      %XHTML.global.resource.attrib;
      %XHTML.global.rev.attrib;
      %XHTML.global.metainformation.extra.attrib;"
>
]]>

<!ENTITY % XHTML.global.metainformation.attrib "" >


<!-- end of xhtml-metaAttributes-1.mod -->
```

B.2. XHTML+RDFa Content Model Module

```
<!-- ....................................................................... -->
<!-- XHTML+RDFa Document Model Module  ..................................... -->
<!-- file: xhtml-rdfa-model-1.mod

     This is XHTML+RDFa.
     Copyright 1998-2007 W3C (MIT, ERCIM, Keio), All Rights Reserved.
     Revision: $Id: rdfa-syntax.ps,v 1.11 2007/10/18 21:45:02 jigsaw Exp $ SMI
```

```
     This DTD module is identified by the PUBLIC and SYSTEM identifiers:

       PUBLIC "-//W3C//ENTITIES XHTML+RDFa Document Model 1.0//EN"
       SYSTEM "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-model-1.mod"

     Revisions:
     (none)
     ....................................................................... -->

<!-- XHTML+RDFa Document Model

     This module describes the groupings of elements that make up
     common content models for XHTML elements.

     XHTML has three basic content models:

         %Inline.mix;  character-level elements
         %Block.mix;   block-like elements, eg., paragraphs and lists
         %Flow.mix;    any block or inline elements

     Any parameter entities declared in this module may be used
     to create element content models, but the above three are
     considered 'global' (insofar as that term applies here).

     The reserved word '#PCDATA' (indicating a text string) is now
     included explicitly with each element declaration that is
     declared as mixed content, as XML requires that this token
     occur first in a content model specification.
-->
<!-- Extending the Model

     While in some cases this module may need to be rewritten to
     accommodate changes to the document model, minor extensions
     may be accomplished by redeclaring any of the three *.extra;
     parameter entities to contain extension element types as follows:

         %Misc.extra;    whose parent may be any block or
                         inline element.

         %Inline.extra;  whose parent may be any inline element.

         %Block.extra;   whose parent may be any block element.

     If used, these parameter entities must be an OR-separated
     list beginning with an OR separator ("|"), eg., "| a | b | c"

     All block and inline *.class parameter entities not part
     of the *struct.class classes begin with "| " to allow for
     exclusion from mixes.
-->

<!-- ..............  Optional Elements in head  .................. -->

<!ENTITY % HeadOpts.mix
     "( %script.qname; | %style.qname; | %meta.qname;
      | %link.qname; | %object.qname; )*"
```

```
>

<!-- ................  Miscellaneous Elements  ................. -->

<!-- ins and del are used to denote editing changes
-->
<!ENTITY % Edit.class "| %ins.qname; | %del.qname;" >

<!-- script and noscript are used to contain scripts
     and alternative content
-->
<!ENTITY % Script.class "| %script.qname; | %noscript.qname;" >

<!ENTITY % Misc.extra "" >

<!-- These elements are neither block nor inline, and can
     essentially be used anywhere in the document body.
-->
<!ENTITY % Misc.class
     "%Edit.class;
      %Script.class;
      %Misc.extra;"
>

<!-- ...................  Inline Elements  ..................... -->

<!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
     "| %tt.qname; | %i.qname; | %b.qname; | %big.qname;
      | %small.qname; | %sub.qname; | %sup.qname;" >

<!ENTITY % I18n.class "| %bdo.qname;" >

<!ENTITY % Anchor.class "| %a.qname;" >

<!ENTITY % InlSpecial.class
     "| %img.qname; | %map.qname;
      | %object.qname;" >

<!ENTITY % InlForm.class
     "| %input.qname; | %select.qname; | %textarea.qname;
      | %label.qname; | %button.qname;" >

<!ENTITY % Inline.extra "" >

<!ENTITY % Ruby.class "| %ruby.qname;" >

<!-- %Inline.class; includes all inline elements,
     used as a component in mixes
-->
<!ENTITY % Inline.class
```

```
     "%InlStruct.class;
      %InlPhras.class;
      %InlPres.class;
      %I18n.class;
      %Anchor.class;
      %InlSpecial.class;
      %InlForm.class;
      %Ruby.class;
      %Inline.extra;"
>

<!-- %InlNoRuby.class; includes all inline elements
     except ruby, used as a component in mixes
-->
<!ENTITY % InlNoRuby.class
     "%InlStruct.class;
      %InlPhras.class;
      %InlPres.class;
      %I18n.class;
      %Anchor.class;
      %InlSpecial.class;
      %InlForm.class;
      %Inline.extra;"
>

<!-- %NoRuby.content; includes all inlines except ruby
-->
<!ENTITY % NoRuby.content
     "( #PCDATA
      | %InlNoRuby.class;
      %Misc.class; )*"
>

<!-- %InlNoAnchor.class; includes all non-anchor inlines,
     used as a component in mixes
-->
<!ENTITY % InlNoAnchor.class
     "%InlStruct.class;
      %InlPhras.class;
      %InlPres.class;
      %I18n.class;
      %InlSpecial.class;
      %InlForm.class;
      %Ruby.class;
      %Inline.extra;"
>

<!-- %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  ..................... -->

<!-- In the HTML 4.0 DTD, heading and list elements were included
     in the %block; parameter entity. The %Heading.class; and
     %List.class; parameter entities must now be included explicitly
     on element declarations where desired.
-->

<!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 % Table.class "| %table.qname;" >

<!ENTITY % Form.class  "| %form.qname;" >

<!ENTITY % Fieldset.class  "| %fieldset.qname;" >

<!ENTITY % BlkStruct.class "%p.qname; | %div.qname;" >

<!ENTITY % BlkPhras.class
     "| %pre.qname; | %blockquote.qname; | %address.qname;" >

<!ENTITY % BlkPres.class "| %hr.qname; " >

<!ENTITY % BlkSpecial.class
     "%Table.class;
      %Form.class;
      %Fieldset.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;
      %BlkSpecial.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;"
>

<!-- end of xhtml-rdfa-model-1.mod -->
```

B.3. XHTML+RDFa Driver Module

```
<!-- ....................................................................... -->
<!-- XHTML 1.1 + RDFa DTD  ................................................. -->
<!-- file: xhtml-rdfa-1.dtd
-->

<!-- XHTML 1.1 + RDFa DTD

     This is an example markup language combining XHTML 1.1 and the RDFa
     modules.

     XHTML+RDFa
     Copyright 1998-2007 World Wide Web Consortium
        (Massachusetts Institute of Technology, European Research Consortium
         for Informatics and Mathematics, Keio University).
         All Rights Reserved.

     Permission to use, copy, modify and distribute the XHTML DTD 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 the DTD for any purpose.

     It is provided "as is" without expressed or implied warranty.

-->
<!-- This is the driver file for version 1 of the XHTML + RDFa DTD.

     Please use this public identifier to identify it:

         "-//W3C//DTD XHTML+RDFa 1.0//EN"
-->
<!ENTITY % XHTML.version  "-//W3C//DTD XHTML+RDFa 1.0//EN" >

<!-- Use this URI to identify the default namespace:

         "http://www.w3.org/1999/xhtml"

     See the Qualified Names module for information
```

```
          on the use of namespace prefixes in the DTD.

          Note that XHTML namespace elements are not prefixed by default,
          but the XHTML namespace prefix is defined as "xhtml" so that
          other markup languages can extend this one and use the XHTML
          prefixed global attributes if required.

     -->
     <!ENTITY % NS.prefixed "IGNORE" >
     <!ENTITY % XHTML.prefix "xhtml" >

     <!-- Be sure to include prefixed global attributes - we don't need
          them, but languages that extend XHTML 1.1 might.
     -->
     <!ENTITY % XHTML.global.attrs.prefixed "INCLUDE" >

     <!-- Reserved for use with the XLink namespace:
     -->
     <!ENTITY % XLINK.xmlns "" >
     <!ENTITY % XLINK.xmlns.attrib "" >

     <!-- For example, if you are using XHTML 1.1 directly, use the public
          identifier in the DOCTYPE declaration, with the namespace declaration
          on the document element to identify the default namespace:

            <?xml version="1.0"?>
            <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
                                  "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
            <html xmlns="http://www.w3.org/1999/xhtml"
                  xml:lang="en">
            ...
            </html>

          Revisions:
          (none)
     -->

     <!-- reserved for future use with document profiles -->
     <!ENTITY % XHTML.profile  "" >

     <!-- ensure XHTML Notations are disabled -->
     <!ENTITY % xhtml-notations.module "IGNORE" >

     <!-- Bidirectional Text features
          This feature-test entity is used to declare elements
          and attributes used for bidirectional text support.
     -->
     <!ENTITY % XHTML.bidi  "INCLUDE" >

     <!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

     <!-- Pre-Framework Redeclaration placeholder  .................... -->
     <!-- this serves as a location to insert markup declarations
          into the DTD prior to the framework declarations.
     -->
     <!ENTITY % xhtml-prefw-redecl.module "IGNORE" >
     <!ENTITY % xhtml-prefw-redecl.mod "" >
```

```
<![%xhtml-prefw-redecl.module;[
%xhtml-prefw-redecl.mod;
<!-- end of xhtml-prefw-redecl.module -->]]>

<!-- we need the datatypes now -->
<!ENTITY % xhtml-datatypes.module "INCLUDE" >
<![%xhtml-datatypes.module;[
<!ENTITY % xhtml-datatypes.mod
     PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-datatypes-1.mod" >
%xhtml-datatypes.mod;]]>

<!-- bring in the RDFa attributes cause we need them in Common -->
<!ENTITY % xhtml-metaAttributes.module "INCLUDE" >
<![%xhtml-metaAttributes.module;[
<!ENTITY % xhtml-metaAttributes.mod
     PUBLIC "-//W3C//ENTITIES XHTML MetaAttributes 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-metaAttributes-1.mod" >
%xhtml-metaAttributes.mod;]]>

<!ENTITY % xhtml-events.module "INCLUDE" >

<!ENTITY % Common.extra.attrib
   "href         %URI.datatype;           #IMPLIED
    %Metainformation.attrib;"
>

<!-- Inline Style Module  ....................................... -->
<!ENTITY % xhtml-inlstyle.module "INCLUDE" >
<![%xhtml-inlstyle.module;[
<!ENTITY % xhtml-inlstyle.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Inline Style 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-inlstyle-1.mod" >
%xhtml-inlstyle.mod;]]>

<!-- declare Document Model module instantiated in framework
-->
<!ENTITY % xhtml-model.mod
     PUBLIC "-//W3C//ENTITIES XHTML+RDFa Document Model 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-model-1.mod" >

<!-- Modular Framework Module (required) ........................ -->
<!ENTITY % xhtml-framework.module "INCLUDE" >
<![%xhtml-framework.module;[
<!ENTITY % xhtml-framework.mod
     PUBLIC "-//W3C//ENTITIES XHTML Modular Framework 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-framework-1.mod" >
%xhtml-framework.mod;]]>

<!-- Post-Framework Redeclaration placeholder  .................. -->
<!-- this serves as a location to insert markup declarations
     into the DTD following the framework declarations.
-->
<!ENTITY % xhtml-postfw-redecl.module "IGNORE" >
<!ENTITY % xhtml-postfw-redecl.mod "">
<![%xhtml-postfw-redecl.module;[
%xhtml-postfw-redecl.mod;
```

```
<!-- end of xhtml-postfw-redecl.module -->]]>



<!-- Text Module (Required)  ..................................... -->
<!ENTITY % xhtml-text.module "INCLUDE" >
<![%xhtml-text.module;[
<!ENTITY % xhtml-text.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Text 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod" >
%xhtml-text.mod;]]>

<!-- Hypertext Module (required) ................................. -->
<!ENTITY % a.attlist  "IGNORE" >
<!ENTITY % xhtml-hypertext.module "INCLUDE" >
<![%xhtml-hypertext.module;[
<!ENTITY % xhtml-hypertext.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Hypertext 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-hypertext-1.mod" >
%xhtml-hypertext.mod;]]>
<!ATTLIST %a.qname;
      %Common.attrib;
      charset      %Charset.datatype;       #IMPLIED
      type         %ContentType.datatype;   #IMPLIED
      accesskey    %Character.datatype;     #IMPLIED
      tabindex     %Number.datatype;        #IMPLIED
>

<!-- Lists Module (required)  .................................... -->
<!ENTITY % xhtml-list.module "INCLUDE" >
<![%xhtml-list.module;[
<!ENTITY % xhtml-list.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Lists 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod" >
%xhtml-list.mod;]]>

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- Edit Module  ................................................ -->
<!ENTITY % xhtml-edit.module "INCLUDE" >
<![%xhtml-edit.module;[
<!ENTITY % xhtml-edit.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Editing Elements 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-edit-1.mod" >
%xhtml-edit.mod;]]>

<!-- BIDI Override Module  ....................................... -->
<!ENTITY % xhtml-bdo.module "%XHTML.bidi;" >
<![%xhtml-bdo.module;[
<!ENTITY % xhtml-bdo.mod
     PUBLIC "-//W3C//ELEMENTS XHTML BIDI Override Element 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-bdo-1.mod" >
%xhtml-bdo.mod;]]>

<!-- Ruby Module  ................................................ -->
<!ENTITY % Ruby.common.attlists "INCLUDE" >
<!ENTITY % Ruby.common.attrib "%Common.attrib;" >
```

```
<!ENTITY % xhtml-ruby.module "INCLUDE" >
<![%xhtml-ruby.module;[
<!ENTITY % xhtml-ruby.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Ruby 1.0//EN"
            "http://www.w3.org/TR/ruby/xhtml-ruby-1.mod" >
%xhtml-ruby.mod;]]>

<!-- Presentation Module  ........................................ -->
<!ENTITY % xhtml-pres.module "INCLUDE" >
<![%xhtml-pres.module;[
<!ENTITY % xhtml-pres.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Presentation 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-pres-1.mod" >
%xhtml-pres.mod;]]>

<!ENTITY % link.attlist  "IGNORE" >
<!-- Link Element Module  ........................................ -->
<!ENTITY % xhtml-link.module "INCLUDE" >
<![%xhtml-link.module;[
<!ENTITY % xhtml-link.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Link Element 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-link-1.mod" >
%xhtml-link.mod;]]>

<!ATTLIST %link.qname;
      %Common.attrib;
      charset      %Charset.datatype;       #IMPLIED
      hreflang     %LanguageCode.datatype;  #IMPLIED
      type         %ContentType.datatype;   #IMPLIED
      media        %MediaDesc.datatype;     #IMPLIED
>

<!-- Document Metainformation Module  ........................... -->
<!ENTITY % meta.attlist  "IGNORE" >
<!ENTITY % xhtml-meta.module "INCLUDE" >
<![%xhtml-meta.module;[
<!ENTITY % xhtml-meta.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Metainformation 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-meta-1.mod" >
%xhtml-meta.mod;]]>
<!ATTLIST %meta.qname;
      %Common.attrib;
      http-equiv   NMTOKEN                  #IMPLIED
      name         NMTOKEN                  #IMPLIED
      scheme       CDATA                    #IMPLIED
>

<!-- Base Element Module  ........................................ -->
<!ENTITY % xhtml-base.module "INCLUDE" >
<![%xhtml-base.module;[
<!ENTITY % xhtml-base.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Base Element 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-base-1.mod" >
%xhtml-base.mod;]]>

<!-- Scripting Module  ........................................... -->
<!ENTITY % xhtml-script.module "INCLUDE" >
```

```
<![%xhtml-script.module;[
<!ENTITY % xhtml-script.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Scripting 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-script-1.mod" >
%xhtml-script.mod;]]>

<!-- Style Sheets Module  ......................................... -->
<!ENTITY % xhtml-style.module "INCLUDE" >
<![%xhtml-style.module;[
<!ENTITY % xhtml-style.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Style Sheets 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-style-1.mod" >
%xhtml-style.mod;]]>

<!-- Image Module  ............................................... -->
<!ENTITY % xhtml-image.module "INCLUDE" >
<![%xhtml-image.module;[
<!ENTITY % xhtml-image.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Images 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod" >
%xhtml-image.mod;]]>

<!-- Client-side Image Map Module  ............................... -->
<!ENTITY % area.attlist  "IGNORE" >
<!ENTITY % xhtml-csismap.module "INCLUDE" >
<![%xhtml-csismap.module;[
<!ENTITY % xhtml-csismap.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Client-side Image Maps 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-csismap-1.mod" >
%xhtml-csismap.mod;]]>
<!ATTLIST %area.qname;
      %Common.attrib;
      shape        %Shape.datatype;         'rect'
      coords       %Coords.datatype;        #IMPLIED
      nohref       ( nohref )               #IMPLIED
      alt          %Text.datatype;          #REQUIRED
      tabindex     %Number.datatype;        #IMPLIED
      accesskey    %Character.datatype;     #IMPLIED
>

<!-- Server-side Image Map Module  ............................... -->
<!ENTITY % xhtml-ssismap.module "INCLUDE" >
<![%xhtml-ssismap.module;[
<!ENTITY % xhtml-ssismap.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Server-side Image Maps 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-ssismap-1.mod" >
%xhtml-ssismap.mod;]]>

<!-- Param Element Module  ...................................... -->
<!ENTITY % xhtml-param.module "INCLUDE" >
<![%xhtml-param.module;[
<!ENTITY % xhtml-param.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Param Element 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-param-1.mod" >
%xhtml-param.mod;]]>

<!-- Embedded Object Module  .................................... -->
```

```
<!ENTITY % xhtml-object.module "INCLUDE" >
<![%xhtml-object.module;[
<!ENTITY % xhtml-object.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Embedded Object 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-object-1.mod" >
%xhtml-object.mod;]]>

<!-- Tables Module ............................................... -->
<!ENTITY % xhtml-table.module "INCLUDE" >
<![%xhtml-table.module;[
<!ENTITY % xhtml-table.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Tables 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod" >
%xhtml-table.mod;]]>

<!-- Forms Module  ............................................... -->
<!ENTITY % xhtml-form.module "INCLUDE" >
<![%xhtml-form.module;[
<!ENTITY % xhtml-form.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Forms 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-form-1.mod" >
%xhtml-form.mod;]]>

<!-- Target Attribute Module  ................................... -->
<!ENTITY % xhtml-target.module "INCLUDE" >
<![%xhtml-target.module;[
<!ENTITY % xhtml-target.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Target 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-target-1.mod" >
%xhtml-target.mod;]]>

<!-- Legacy Markup ............................................... -->
<!ENTITY % xhtml-legacy.module "IGNORE" >
<![%xhtml-legacy.module;[
<!ENTITY % xhtml-legacy.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Legacy Markup 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-legacy-1.mod" >
%xhtml-legacy.mod;]]>

<!-- Document Structure Module (required)  ...................... -->
<!ENTITY % head.attlist  "IGNORE" >
<!ENTITY % title.attlist  "IGNORE" >
<!ENTITY % xhtml-struct.module "INCLUDE" >
<![%xhtml-struct.module;[
<!ENTITY % xhtml-struct.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Document Structure 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-struct-1.mod" >
%xhtml-struct.mod;]]>
<!ENTITY % profile.attrib
     "profile      %URI.datatype;           '%XHTML.profile;'"
>
<!ATTLIST %head.qname;
      %Common.attrib;
      %profile.attrib;
>
<!ATTLIST %title.qname;
      %Common.attrib;
```

```
>

<!-- end of XHTML-RDFa DTD  ................................................. -->
<!-- ....................................................................... -->
```

B.4. SGML Open Catalog Entry for XHTML+RDFa

This section contains the SGML Open Catalog-format definition [CATALOG [p.68]] of the public identifiers for XHTML+RDFa 1.0.

```
-- .......................................................................... --
-- File catalog  ............................................................ --

--  XHTML+RDFa Catalog Data File

    Revision:  $Revision: 1.11 $

    See "Entity Management", SGML Open Technical Resolution 9401 for detailed
    information on supplying and using catalog data. This document is available
    from OASIS at URL:

        <http://www.oasis-open.org/html/tr9401.html>
--

-- .......................................................................... --
-- SGML declaration associated with XHTML  .................................. --

OVERRIDE YES

SGMLDECL "xml1.dcl"

-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --

-- XHTML+RDFa modules          .............................................. --


PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"       "xhtml-rdfa-1.dtd"


PUBLIC "-//W3C//ENTITIES XHTML+RDFa Document Model 1.0//EN"      "xhtml-rdfa-model-1.mod"

PUBLIC "-//W3C//ENTITIES XHTML MetaAttributes 1.0//EN"           "xhtml-metaAttributes-1.mod"

-- End of catalog data  ..................................................... --
-- .......................................................................... --
```

C. References

C.1. Related Specifications

This section is normative.

CSS2
"*Cascading Style Sheets, level 2 (CSS2) Specification*", W3C Recommendation, B. Bos, H. W. Lie, C. Lilley, I. Jacobs, *eds.*, 12 May 1998.
Available at: http://www.w3.org/TR/1998/REC-CSS2-19980512

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

IRI
"*Internationalized Resource Identifiers (IRI)*", RFC 3987, M.Duerst, M. Suignard January 2005.
Available at: http://www.ietf.org/rfc/rfc3987.txt

[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/

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

RFC3236
"The 'application/xhtml+xml' Media Type", M. Baker, P. Stark, January 2002.
Available at: http://www.ietf.org/rfc/rfc3236.txt

RUBY
Ruby Annotation, W3C Recommendation, Marcin Sawicki, et al., 31 May 2001.
See: http://www.w3.org/TR/2001/REC-ruby-20010531

XHTML 1.1
"*XHTML 1.1 - Module-based XHTML*", W3C Recommendation, M. Altheim, S. McCarron, 31 May 2001.
Available at: http://www.w3.org/TR/2001/REC-xhtml11-20010531/.

XHTMLROLE
"*XHTML Role Attribute Module*", W3C Working Draft, M. Birbeck, et. al. 4 October 2007.
Available at: http://www.w3.org/TR/2007/WD-xhtml-role-20071004.

XMLBASE
"*XML Base*", W3C Recommendation, J. Marsh, *ed.*, 27 June 2001.
Available at: http://www.w3.org/TR/2001/REC-xmlbase-20010627/

XML-LANG
"*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

XHTMLMOD
: *XHTML Modularization 1.1*, W3C Working Draft, Shane McCarron, et al., 5 July 2006 See: http://www.w3.org/TR/2006/WD-xhtml-modularization-20060705

XMLNAMES
: "*Namespaces in XML*", W3C Recommendation, Tim Bray, Dave Hollander, Andrew Layman, 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/
See also "*XML Schema Part 2: Datatypes Second Edition*", available at: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

C.2. Related Activities

This section is informative.

CATALOG
: *Entity Management: OASIS Technical Resolution 9401:1997 (Amendment 2 to TR 9401)*, Paul Grosso, Chair, Entity Management Subcommittee, SGML Open, 10 September 1997. See: http://www.oasis-open.org/html/a401.htm

DC
: Dublin Core Metadata Initiative (DCMI) (See http://dublincore.org/.)

FOAF-PROJECT
: The FOAF Project (See http://www.foaf-project.org.)

N-TRIPLES
: RDF Test Cases, N-Triples (See http://www.w3.org/TR/rdf-testcases/#ntriples.)

N3-PRIMER
: N3 Primer (See http://www.w3.org/2000/10/swap/Primer.)

RDFa Primer
: RDFa Primer 1.0 - Embedding Structured Data in Web Pages (see http://www.w3.org/2006/07/SWD/RDFa/primer.)

RDF-CONCEPTS
: Resource Description Framework (RDF): Concepts and Abstract Syntax (See http://www.w3.org/TR/rdf-concepts/.)

RDF-PRIMER
: RDF Primer (See http://www.w3.org/TR/rdf-primer/.)

RDF-SYNTAX
: RDF/XML Syntax and Grammar (See http://www.w3.org/TR/rdf-syntax-grammar/.)

RDFTESTS-DATATYPES-TEST001
: datatypes/test001.nt (See http://www.w3.org/2000/10/rdf-tests/rdfcore/datatypes/test001.nt.)

RDFTESTS-RDFMS-XMLLANG-TEST006
: rdfms-xmllang/test006.nt (See http://www.w3.org/2000/10/rdf-tests/rdfcore/rdfms-xmllang/test006.nt.)

RELAXNG
RELAX NG Home Page (See http://www.relaxng.org/.)
SWD-WG
Semantic Web Deployment Working Group (See http://www.w3.org/2006/07/SWD/.)
RDFHTML
RDF-in-HTML Task Force (See http://w3.org/2001/sw/BestPractices/HTML/.)
XHTML2-WG
XHTML 2 Working Group (See http://w3.org/MarkUp/Group/.)
XHTMLMIME
"*XHTML Media Types*", Masayasu Ishikawa, 1 August 2002.
Latest version available at: http://www.w3.org/TR/xhtml-media-types

D. Change History

This section is informative.

2007-10-19: Updated to reflect latest task force thinking re: processing model. Integrated XHTML Module definition and hybrid markup language. Completed development as First Public Working Draft. [ShaneMcCarron], [MarkBirbeck]

2007-09-04: Migrated to XHTML 2 Working Group Publication System. Converted to a format that is consistent with REC-Track documents. Updated to reflect current processing model. Added normative definition of CURIEs. Started updating prose to be consistent with current task force agremeents. [ShaneMcCarron], [StevenPemberton], [MarkBirbeck]

2007-04-06: fixed some of the language to talk about "structure" rather than metadata. Added note regarding space-separated values in predicate-denoting attributes. [BenAdida]

2006-01-16: made the use of CURIE type for @rel, @rev, @property consistent across document (particularly section 2.4 was erroneous). [BenAdida]

E. Acknowledgments

This section is informative.

This section is informative.

At the time of publication, the participants in the XHTML 2 Working Group were: