RDFa in XHTML: Syntax and Processing

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

W3C Editor's Draft 17 February 2008

This version:
http://www.w3.org/MarkUp/2008/ED-rdfa-syntax-20080217
Latest version:
http://www.w3.org/TR/rdfa-syntax
Previous Editor's Draft:
http://www.w3.org/MarkUp/2008/ED-rdfa-syntax-20080125
Diff from previous Editor's Draft:
rdfa-syntax-diff.html
Editors:
Ben Adida, Creative Commons ben@adida.net
Mark Birbeck, webBackplane mark.birbeck@webBackplane.com
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.

Copyright © 2007-2008 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark and document use rules apply.

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.80]], 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 (often called vocabularies or taxonomies) 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.15] 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 about RDF on the web, and a growing range of tools that 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.80]] 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 Last Call 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 working groups invite comments on this draft through 18 March 2008.

An sample test harness is available to assist in evaluating this specification at http://rdfa.digitalbazaar.com/rdfa-test-harness/.

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 15
3.1. Statements 15
3.2. Triples 15
3.3. URI references 16
3.4. Plain literals 17
3.5. Typed literals 17
3.6. Turtle 17
3.7. Graphs 18
3.8. Compact URIs 18
3.9. A description of RDFa in RDF terms 19
4. Conformance Requirements 21
4.1. Document Conformance 21
4.2. User Agent Conformance 22
4.3. RDFa Processor Conformance 22
5. Processing Model 23
5.1. Overview 23
5.2. Evaluation Context 23
5.3. Chaining 24
5.4. CURIE and URI Processing 26
5.4.1. Scoping of Prefix Mappings 27
5.4.2. Converting a CURIE to a URI 27
5.4.3. General Use of CURIEs in Attributes 27
5.4.4. Use of CURIEs in Specific Attributes 28
5.5. Sequence 29
6. RDFa Processing in detail 37
6.1. Changing the evaluation context 37
6.1.1. Setting the [current subject] 37
6.2. Completing 'incomplete triples' 43
6.3. Object resolution 46
6.3.1. Literal object resolution 47
6.3.2. URI object resolution 49
7. CURIE Syntax Definition 53
8. XHTML+RDFa Definition 55
9. Metainformation Attributes Module 57
9.1. Datatypes 57
9.2. Metainformation Attributes Collection 57
9.3. @property attribute values 58

9.4. @rel/@rev attribute values 58
A. Other XML Languages 61
B. XHTML+RDFa DTD 63
B.1. XHTML Metainformation Attributes Module 63
B.2. XHTML+RDFa Content Model Module 65
B.3. XHTML+RDFa Driver Module 70
B.4. SGML Open Catalog Entry for XHTML+RDFa 77
C. References 79
C.1. Related Specifications 79
C.2. Related Activities 80
D. Change History 83
E. Acknowledgments 85

1. Motivation

This section is informative.

RDF/XML [RDF-SYNTAX [p.80]] provides sufficient flexibility to represent all of the abstract concepts in RDF [RDF-CONCEPTS [p.80]]. However, it presents a number of challenges; first it is difficult or impossible to validate documents that contain RDF/XML using XML Schemas or DTDs, which therefore makes it difficult to import RDF/XML into other markup languages. Whilst newer schema languages such as RELAX NG [RELAXNG [p.81]] 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.

Another reason for aligning the rendered data with the structured data is that it is highly beneficial to express the web data's structure 'in context'; as users often want to transfer structured data from one application to another, sometimes to or from a non-web-based application, the user experience can be enhanced. For example, information about specific rendered data could be presented to the user via '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.79]] and HTML [HTML4 [p.79]] 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.80]].

For brevity, in the following examples and throughout this document, assume that the following namespace prefixes have been defined:

biblio:	http://example.org/biblio/0.1
cc:	http://creativecommons.org/ns#
dbp:	http://dbpedia.org/property/
dbr:	http://dbpedia.org/resource/
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#
rdfa:	http://www.w3.org/ns/rdfa/
rdfs:	http://www.w3.org/2000/01/rdf-schema#
taxo:	http://purl.org/rss/1.0/modules/taxonomy/
xhv:	http://www.w3.org/1999/xhtml/vocab#
xsd:	http://www.w3.org/2001/XMLSchema#

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.57] s, used for expressing relationships between two resources ('predicates' in RDF terminology);

@rev
: a whitespace separated list of CURIE [p.57] s, used for expressing reverse relationships between two resources (also 'predicates');

@name
: a token for expressing a value which is a plain literal (also a 'predicate');

@content
: a string, for supplying machine-readable content for a literal (a 'plain literal object', in RDF terminology);

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

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

The new—RDFa-specific—attributes are:

@about
: a URIorCURIE [p.57] , used for stating what the data is about (the 'subject' in RDF terminology);

@property
: a whitespace separated list of CURIE [p.57] s, used for expressing relationships between the subject and some literal text (also a 'predicate');

@resource
: a URIorCURIE [p.57] for expressing the partner resource of a relationship that is not intended to be 'clickable' (also an 'object');

@datatype
: a CURIE [p.57] representing a datatype, to express the datatype of a literal;

@instanceof
: a whitespace separated list of CURIE [p.57] 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.57] .

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:biblio="http://example.org/"
>
```

```
  <head>
    <title>Books by Marco Pierre White</title>
  </head>
  <body>
    I think
    <span about="urn:ISBN:0091808189" instanceof="biblio: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="biblio: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.80]] and the RDF Sytax Document [RDF-SYNTAX [p.80]].

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 [http://dbpedia.org] 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 [p.17]], and this URI indicates the literal's datatype. This URI is usually based on datatypes defined in the XML Schema Datatypes specification [XMLSCHEMA [p.80]]. 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. Turtle

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 for expressing triples, such as RDF/XML, Turtle [TURTLE [p.81]], and of course RDFa. Many discussions of RDF make use of the *Turtle* 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. Turtle allows long URIs to be abbreviated by using a URI mapping, which can be used to express a compact URI as follows:

```
@prefix dbp: <http://dbpedia.org/property/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://dbpedia.org/resource/Albert_Einstein>
  foaf:name "Albert Einstein" .
<http://dbpedia.org/resource/Albert_Einstein>
  dbp:birthPlace <http://dbpedia.org/resource/Germany> .
```

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

Any URI in Turtle could be abbreviated in this way. This means that we could also have used the same technique to abbreviate the identifier for Einstein, as well as the datatype indicator:

```
@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
dbr:Albert_Einstein dbp:dateOfBirth "1879-03-14"^^xsd:date .
dbr:Albert_Einstein
  foaf:depiction <http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg> .
```

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

```
<>
```

This indicates the 'current document', i.e., the document being processed. In reality there would always be a full URI based on the document's location, but this abbreviation serves to make examples more compact. Note in particular that the whole technique of abbreviation is merely a way to make examples more compact, and the actual triples generated would always use the full URIs.

3.7. Graphs

A collection of triples is called a *graph*.

For more information on the concepts described above, see [RDF-CONCEPTS [p.80]]. 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 [p.16]]s into a form called a 'compact URI', or CURIE [p.57] . A detailed discussion of this this mechanism is in section CURIE and URI Processing [p.26] .

Note that CURIEs are only used in the mark-up and Turtle examples, and will never appear in the generated [triple [p.15]]s, which are defined in RDF to use [URI reference [p.16]]s.

Full details on how CURIEs are processed is in the section titled CURIE Processing [p.26] .

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 [p.19]] to be carried in various types of document mark-up. However, this specification deals only with RDFa in XHTML (see Appendix A [p.61] for some comments on using RDFa in other markup languages). An [RDF graph] comprises [node]s linked by relationships. The basic unit of an [RDF graph [p.19]] is a [triple [p.15]], in which a subject [node [p.19]] is linked to an object [node [p.19]] via a [predicate [p.19]]. The [subject] [node [p.19]] is always either an [URI reference [p.16]] or a [blank node (or bnode)], the [predicate] is *always* an [URI reference [p.16]], and the object of a statement can be an [URI reference [p.16]], a [literal [p.17]], or a [bnode [p.19]].

In RDFa, a subject [URI reference [p.16]] is generally indicated using @about, and predicates are represented using one of @property, @rel, or @rev. Objects which are [URI reference [p.16]]s are represented using @href, @resource or @src, whilst objects that are [literal [p.17]]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.79]].

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.63] .

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.80]]. 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 SHOULD 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.63] 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">
   ```

5. There SHOULD be a @profile attribute on the `head` element that includes the value `http://www.w3.org/1999/xhtml/vocab`.

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 profile="http://www.w3.org/ns/rdfa">
    <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 + RDFa documents SHOULD be labeled with the Internet Media Type "application/xhtml+xml" as defined in [RFC3236 [p.79]]. For further information on using media types with XHTML family markup languages, see the informative note [XHTMLMIME [p.81]].

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.80]] 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 [p.19]] containing all possible triples generated by using the rules in the Processing Model [p.23] section. This is called 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 NOT be made available in the [default graph [p.22]]. (Whether these additional triples are made available in one or more additional [RDF graph [p.19]]s is implementation-specific, and therefore not defined here.)

Since XHTML+RDFa is based upon XHTML Modularization [XHTMLMOD [p.80]], and since XHTML Modularization requires that whitespace is preserved, conforming processors must preserve whitespace in both [plain literal [p.17]]s and [XML literals [p.48]]. However, it may be the case that the architecture in which a processor operates does not make all whitespace available. It is therefore advisable for authors who would like to make their documents consumable across different processors, to remove any unnecessary whitespace in their mark-up.

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 [p.29]]).

Note that in this section, explanations about the processing model or guidance to implementors are enclosed in sections like this.

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, in document order, 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 may generate triples, and may also change the [evaluation context [p.29]] information that will then be used when processing descendant elements.

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, and a single [RDF graph] produced. As described in the RDFa Processor Conformance [p.22] section, any additional triples generated MUST NOT appear in the [default graph [p.22]].

5.2. Evaluation Context

During processing, each rule is applied using information provided by an [evaluation context [p.29]]. An initial context is created when processing begins, with the following set of values:

- 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 resolved.
- The [current subject]. The initial value will be the same as the initial value of [base [p.23]], but it will usually change during the course of processing.

- The [parent object]. In some situations the object of a statement becomes the subject of any nested statements, and this property is used to convey this value. Note that this value may be a bnode, since in some situations a number of nested statements are grouped together on one bnode. This means that the bnode must be set in the containing statement and passed down, and this property is used to convey this value.
- A list of current, in-scope [URI mappings [p.??]].
- A list of [incomplete triple]s. A triple can be incomplete when no object resource is provided alongside a predicate that requires a resource (i.e., @rel or @rev). The triples can be completed when a resource becomes available, which will be when the next subject is specified (part of the process called [chaining [p.24]]).
- The [language]. Note that there is no default language.

During the course of processing new [evaluation context [p.29]]s are created which are passed to each child element. The rules described below will determine the values of the items in the context. Additionally, some rules will cause new triples to be created by combining information provided by an element with information from the [evaluation context [p.29]].

During the course of processing a number of locally scoped values are needed, as follows:

- An initially empty list of [URI mappings [p.??]], called the [local list of URI mappings].
- An initially empty list of [incomplete triple [p.24]]s, called the [local list of incomplete triples].
- An initially empty [language [p.24]] value.
- 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 setting this flag to `false` would have that effect.
- A [new subject] value, which once calculated will set [current subject [p.23]] in the evaluation context, as well as being used to complete any [incomplete triples [p.??]], as described in the next section.
- A value for the [current object literal], the literal to use when creating triples that have a literal object.
- A value for the [current object resource], the resource to use when creating triples that have a resource object.

5.3. Chaining

RDFa has the notion of [chaining] which aims to combine statements together in as intuitive a way as possible, so as avoid unnecessary repetition of mark-up. For example, if an author were to add statements as children of an object that was a resource, these statements should be interpreted as being about that resource:

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <span property="foaf:name">Albert Einstein</span>
  <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
  <div rel="dbp:birthPlace" resource="http://dbpedia.org/resource/Germany">
    <span property="dbp:conventionalLongName">Federal Republic of Germany</span>
  </div>
  <img rel="foaf:depiction" src="http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg" />
</div>
```

In this example we can see that an object resource ('Germany'), has become the subject for nested statements. This mark-up also illustrates the basic chaining pattern of 'A has a B has a C' (i.e., Einstein has a birth place of Germany, which has a long name of "Federal Republic of Germany").

It's also possible for the subject of nested statements to provide the object for *containing* statements—essentially the reverse of the example we have just seen. To illustrate, we'll take an example of the type of chaining just described, and show how it could be marked up more efficiently. To start, we mark up the fact that Albert Einstein had both German and American citizenship:

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <div rel="dbp:citizenship" resource="http://dbpedia.org/resource/Germany"></div>
  <div rel="dbp:citizenship" resource="http://dbpedia.org/resource/United_States"></div>
</div>
```

Now, we show the same information, but this time we create an [incomplete triple [p.24]] from the citizenship part, and then use any number of further subjects to 'complete' that triple, as follows:

```
<div about="http://dbpedia.org/resource/Albert_Einstein" rel="dbp:citizenship">
  <span about="http://dbpedia.org/resource/Germany"></span>
  <span about="http://dbpedia.org/resource/United_States"></span>
</div>
```

In this example, the [incomplete triple [p.24]] actually gets completed twice, once for Germany and once for the USA, giving exactly the same information as we had in the earlier example:

```
<http://dbpedia.org/resource/Albert_Einstein>
  dbp:citizenship <http://dbpedia.org/resource/Germany> .
<http://dbpedia.org/resource/Albert_Einstein>
  dbp:citizenship <http://dbpedia.org/resource/United_States> .
```

Chaining can also involve relatively minimal mark-up, where an element might contain only one resource, or only one predicate. For example:

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <div rel="dbp:citizenship">
    <span about="http://dbpedia.org/resource/Germany"></span>
    <span about="http://dbpedia.org/resource/United_States"></span>
  </div>
</div>
```

When such minimal mark-up is used, any of the resource-related attributes could act as a subject or an object in the chaining.

5.4. CURIE and URI Processing

Since RDFa is ultimately a means for transporting RDF, then a key concept is the *resource* and its manifestation as a URI. Since RDF deals with complete URIs (not relative paths), then when converting RDFa to triples, any relative URIs will need to be resolved relative to the base URI, using the algorithm defined in section 5 of RFC 3986 [URI [p.79]], *Reference Resolution*.

Many of the attributes that hold URIs are also able to carry 'compact URIs' or CURIEs. A CURIE is a convenient way to represent a long URI, by replacing a leading section of the URI with a substitution token. It's possible for authors to define a number of substitution tokens as they see fit; the full URI is obtained by locating the mapping defined by a token from a list of in-scope tokens, and then simply concatenating the second part of the CURIE onto the mapped value.

For example, the full URI for Albert Einstein on DPPedia is:

```
http://dbpedia.org/resource/Albert_Einstein
```

This can be shortened by authors to make the information easier to manage, using a CURIE. The first step is for the author to create a prefix mapping that links a prefix to some leading segment of the URI. In RDFa these mappings are expressed using the XML namespace syntax:

```
<div xmlns:db="http://dbpedia.org/">
  ...
</div>
```

Once the prefix has been established, an author can then use it to shorten a URI as follows:

```
<div xmlns:db="http://dbpedia.org/">
  <div about="[db:resource/Albert_Einstein]">
    ...
  </div>
</div>
```

The author is free to break the URI at any point, as long as it begins at the left end. However, since a common use of CURIEs is to make available libraries of terms and values, the prefix will usually be mapped to some common segment that provides the most re-use, often provided by those who manage the library of terms. For example, since DBPedia contains an enormous list of resources, it is more efficient to create a prefix mapping that uses the base location of the resources:

```
<div xmlns:dbr="http://dbpedia.org/resource/">
  <div about="[dbr:Albert_Einstein]">
    ...
  </div>
  <div about="[dbr:Baruch_Spinoza]">
    ...
  </div>
</div>
```

5.4.1. Scoping of Prefix Mappings

Since CURIE mappings are created by authors via the XML namespace syntax [XMLNS [p.79]] then an RDFa processor MUST take into account the hierarchical nature of prefix declarations. For example, the URIs expressed by the following two CURIEs are different, despite the common prefix, because the prefix mappings are locally scoped:

```
<div xmlns:dbr="http://dbpedia.org/resource/">
  <div about="[dbr:Albert_Einstein]">
    ...
  </div>
</div>
<div xmlns:dbr="http://someotherdb.org/resource/">
  <div about="[dbr:Albert_Einstein]">
    ...
  </div>
</div>
```

5.4.2. Converting a CURIE to a URI

Since a CURIE is merely a means for abbreviating a URI, then its *value* is a URI, rather than the abbreviated form. Obtaining a URI from a CURIE involves the following steps:

1. Split the CURIE at the colon to obtain the prefix and the resource.
2. Using the prefix and the current in-scope mappings, obtain the URI that the prefix maps to.
3. Concatenate the mapped URI with the resource value, to obtain an absolute URI.

Note that it is generally considered a good idea not to use relative paths in namespace declarations, but since it is possible that an author may ignore this guidance, it is further possible that the URI obtained from a CURIE is relative. However, since all URIs must be resolved relative to [base [p.23]] before being used to create triples, the use of relative paths should not have any effect on processing.

5.4.3. General Use of CURIEs in Attributes

There are a number of ways that attributes will make use of CURIEs, and they need to be dealt with differently. These are:

1. An attribute may be CURIE-only, disallowing other types of values. In this case any value that is not a 'curie' according to the definition in the section CURIE Syntax Definition [p.53] should not affect processing in any way; this means that not only will there be no error reporting, but also the RDFa processor should act as if the value simply did not exist.
2. An attribute may allow CURIEs, as well as a full URI. In this case any value that is not surrounded by square brackets, as defined by 'safe_curie' in the section CURIE Syntax Definition [p.53] , will be processed as if it was a URI. If the value *is* surrounded by square brackets, then the inner content must conform to the 'curie' definiton, and as before, if it does not then the value should have no effect on processing.

An example of an attribute that can contain CURIE and non-CURIE values is @about. As described, any CURIEs expressed in the attribute must follow the format of a [safe CURIE [p.53]]. So to express a URI directly, an author might do this:

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  ...
</div>
```

whilst to express a CURIE they would do this:

```
<div about="[dbr:Albert_Einstein]">
  ...
</div>
```

Since non-CURIE values MUST be ignored, the following value in @about would *not* set a new subject, since the CURIE has no prefix separator:

```
<div about="[Albert_Einstein]">
  ...
</div>
```

However, this mark-up *would* set a subject, since it is not a CURIE, but a valid relative URI:

```
<div about="Albert_Einstein">
  ...
</div>
```

There is one exception to this; @rel and @rev can also take any value from the list in the section on The rel attribute [p.??] , and any matching value MUST be treated as if it was a full URI with the XHTML vocabulary as its prefix mapping. This is discussed further in the next section.

5.4.4. Use of CURIEs in Specific Attributes

The general rules discussed in the previous section apply to the RDFa attributes in the following ways:

- @about and @resource support either a URI or a CURIE (expressed as a [safe CURIE [p.53]]).
- @href and @src support only a URI.
- @property, @datatype and @instanceof support only CURIE values.
- @rel and @rev support both XHTML link types and CURIEs.

Note that unlike @about and @resource, @rel and @rev do not differentiate their two types of data by using [safe CURIE [p.53]]s. Instead, any value that matches an entry in the list of link types in the section The rel attribute [p.??] , MUST be treated as if it was a URI within the XHTML vocabulary, and all other values must be CURIEs. This means that either of the following examples:

```
<link rel="next" href="http://example.org/page2.html" />
<link rel="xhv:next" href="http://example.org/page2.html" />
```

would generate this triple:

```
<> <http://www.w3.org/1999/xhtml/vocab#next> <http://example.org/page2.html> .
```

Note that only values in the link type list have this special behaviour, which means that any value that is not in the list *and* is not a valid CURIE MUST not generate triples in the [default graph [p.22]] . For example, no triples would be generated in the [default graph [p.22]] by the following mark-up:

```
<link rel="foobar" href="http://example.org/page7.html" />
```

5.5. Sequence

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, an initial [evaluation context] is created, as follows:

- the [base [p.23]] is set to either the URL of the document or the value specified in the `base` element, if present;
- the [current subject [p.23]] is set to the [base [p.23]] value;
- the [parent object [p.24]] is set to null;
- the [list of URI mappings [p.??]] is empty;
- the [list of incomplete triples [p.??]] is empty;
- the [language [p.24]] is set to null.

Processing begins by applying the processing rules below to the document object, in the context of this initial [evaluation context [p.29]]. All elements in the tree are also processed according to the rules described below, depth-first, although the [evaluation context [p.29]] used for each set of rules will be based on previous rules that may have been applied.

The processing rules are:

1. First, the local values are initialized, as follows:
   - the [recurse [p.24]] flag is set to true;
   - [new subject [p.24]] is set to null;
   - [current object resource [p.24]] is set to null;
   - the [local list of URI mappings [p.24]] is set to the list of URI mappings from the [evaluation context [p.29]];

- the [local list of incomplete triples [p.24]] is set to null;
- the [current language [p.32]] value is set to the [language [p.24]] value from the [evaluation context [p.29]].

Note that some of the local variables are temporary containers for values that will be passed to descendant elements via an [evaluation context [p.29]]. In some cases the containers will have the same name, so to make it clear which is being acted upon in the following steps, the local version of an item will generally be referred to as such.

2. Next the [current element] is parsed for [URI mapping]s and these are added to the [local list of URI mappings [p.24]]. Note that a [URI mapping [p.30]] will simply overwrite any current mapping in the list that has the same name;
   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 [p.23]]. Authors are advised to follow best practice for using namespaces, which includes not using relative paths.
3. The [current element [p.30]] is also parsed for any language information, and if present, [current language [p.32]] is set accordingly;
   Language information can be provided using the general-purpose XML attribute @xml:lang.
4. If the [current element [p.30]] contains no valid @rel or @rev URI, obtained according to the section on CURIE and URI Processing [p.26] , then the next step is to establish a value for [new subject [p.24]]. Any of the attributes that can carry a resource can set [new subject [p.24]];
   [new subject [p.24]] is set to the URI obtained from the first match from the following rules:
   - by using the URI from @about, if present, obtained according to the section on CURIE and URI Processing [p.26] ;
   - *otherwise*, by using the URI from @src, if present, obtained according to the section on CURIE and URI Processing [p.26] .
   - *otherwise*, by using the URI from @resource, if present, obtained according to the section on CURIE and URI Processing [p.26] ;
   - *otherwise*, by using the URI from @href, if present, obtained according to the section on CURIE and URI Processing [p.26] .

   If no URI is provided by a resource attribute, then the first match from the following rules will apply:
   - if the element is the `head` or `body` element then act as if there is an empty @about present, and process it according to the rule for @about, above;
   - if @instanceof is present, obtained according to the section on CURIE and URI Processing [p.26] , then [new subject [p.24]] is set to be a newly created [bnode [p.19]];
   - *otherwise*, if [parent object [p.24]] is present, [new subject [p.24]] is set to that;
5. If the [current element [p.30]] *does* contain a valid @rel or @rev URI, obtained according to the section on CURIE and URI Processing [p.26] , then the next step is to establish *both* a value for [new subject [p.24]] and a value for [current object resource [p.24]]:
   [new subject [p.24]] is set to the URI obtained from the first match from the following rules:
   - by using the URI from @about, if present, obtained according to the section on CURIE and URI Processing [p.26] ;

- *otherwise*, by using the URI from @src, if present, obtained according to the section on CURIE and URI Processing [p.26] .

If no URI is provided then the first match from the following rules will apply:

- if the element is the `head` or `body` element then act as if there is an empty @about present, and process it according the rule for @about, above;
- if @instanceof is present, obtained according to the section on CURIE and URI Processing [p.26] , then [new subject [p.24]] is set to be a newly created [bnode [p.19]];
- *otherwise*, if [parent object [p.24]] is present, [new subject [p.24]] is set to that.

Then the [current object resource [p.24]] is set to the URI obtained from the first match from the following rules:

- by using the URI from @resource, if present, obtained according to the section on CURIE and URI Processing [p.26] ;
- *otherwise*, by using the URI from @href, if present, obtained according to the section on CURIE and URI Processing [p.26] .
- *otherwise*, [current object resource [p.24]] is set to be a newly created [bnode [p.19]];

Note that final value of the [current object resource [p.24]] will either be null (from initialization), a full URI, or a bnode.

6. If in any of the previous steps a [new subject [p.24]] was set to a non-null value, it is now used to provide a subject for type values;
   One or more 'types' for the [new subject [p.24]] can be set by using @instanceof. If present, the attribute must contain one or more URIs, obtained according to the section on URI and CURIE Processing [p.26] , each of which is used to generate a triple as follows:
   subject
   [new subject [p.24]]
   predicate
   http://www.w3.org/1999/02/22-rdf-syntax-ns#type
   object
   full URI of 'type'
   Note that none of this block is executed if there is no [new subject [p.24]] value, i.e., [new subject [p.24]] remains null.
7. If in any of the previous steps a [current object resource [p.24]] was set to a non-null value, it is now used to generate triples:
   Predicates for the [current object resource [p.24]] can be set by using one or both of the @rel and @rev attributes. If present, @rel will contain one or more URIs, obtained according to the section on CURIE and URI Processing [p.26] each of which is used to generate a triple as follows:
   subject
   [current subject [p.23]]
   predicate
   full URI
   object
   [current object resource [p.24]]
   If present, @rev will contain one or more URIs, obtained according to the section on CURIE and URI Processing [p.26] each of which is used to generate a triple as follows:

subject
: [current object resource [p.24]]

predicate
: full URI

object
: [current subject [p.23]]

8. If however [current object resource [p.24]] was set to null, but there are predicates present, then they must be stored as [incomplete triples [p.??]], pending the discovery of a subject that can be used as the object;
   Predicates for [incomplete triples [p.??]] can be set by using one or both of the @rel and @rev attributes. If present, @rel must contain one or more URIs, obtained according to the section on CURIE and URI Processing [p.26] each of which is added to the [local list of incomplete triples [p.24]] as follows:
   predicate
   : full URI

   direction
   : forward

   If present, @rev must contain one or more URIs, obtained according to the section on CURIE and URI Processing [p.26] , each of which is added to the [local list of incomplete triples [p.24]] as follows:
   predicate
   : full URI

   direction
   : reverse
9. The next step of the iteration is to establish any [current object literal [p.24]];
   Predicates for the [current object literal [p.24]] can be set by using @property. If present, one or more URIs are obtained according to the section on CURIE and URI Processing [p.26] , and then the actual literal value is obtained as follows:
   - as a [plain literal [p.17]] if:
     - @content is present;
     - *or* all children of the [current element [p.30]] are text nodes;
     - *or* there are no child nodes;
     - *or* the body of the [current element [p.30]] *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 [p.17]] should include this language information, as described in [RDF-CONCEPTS [p.80]]. The actual literal is either the value of @content (if present) *or* a string created by concatenating the text content of each of the descendant elements of the [current element [p.30]] in document order.
   - as an [XML literal [p.48]] if:
     - the [current element [p.30]] has any child nodes that are not simply text nodes, and @datatype is not present, or is present, but is set to `rdf:XMLLiteral`.

     The value of the [XML literal [p.48]] is a string created by serializing to text, all nodes that are descendants of the [current element [p.30]], i.e., not including the element itself, and giving it a datatype of `rdf:XMLLiteral`.
   - as a [typed literal [p.17]] 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 value of all descendant text nodes, of the [current element [p.30]] in turn. The final string includes the datatype URI, as described in [RDF-CONCEPTS [p.80]], which will have been obtained according to the section on CURIE and URI Processing [p.26] .

The [current object literal [p.24]] is then used with each predicate to generate a triple as follows:

subject
: [current subject [p.23]]

predicate
: full URI

object
: [current object literal [p.24]]

Once the triple has been created, if the [datatype] of the [current object literal [p.24]] is `rdf:XMLLiteral`, then the [recurse [p.24]] flag is set to `false`.

10. If the [recurse [p.24]] flag is 'true', all elements that are children of the [current element [p.30]] are processed using the rules described here, using an [evaluation context [p.29]] set as follows:
    - the [base [p.23]] is set to the [base [p.23]] value of the current [evaluation content [p.??]];
    - the [current subject [p.23]] is set to the value of [new subject [p.24]], if non-null, *or* the value of the [current subject [p.23]] of the current [evaluation content [p.??]];
    - the [parent object [p.24]] is set to value of [current object resource [p.24]], if non-null, *or* the value of [new subject [p.24]], if non-null, or the value of the [current subject [p.23]] of the current [evaluation content [p.??]];
    - the [list of URI mappings [p.??]] is set to the [local list of URI mappings [p.24]];
    - the [list of incomplete triples [p.??]] is set to the [local list of incomplete triples [p.24]];
    - [language [p.24]] is set to the value of [current language [p.32]].

    Note that this step is pivotal to chaining. The final step of processing (step 12, below) involves returning a flag to indicate whether it is worth completing any [incomplete triples [p.??]] that might exist at this level. If the flag is true for *any* of the child elements, then those triples are completed in the next step (step 11). The reason for setting this flag is that it allows a processor to avoid generating triples for chained bnodes that have no non-bnode properties. An example would be:

```
<div rel="foaf:knows">
  <div rel="foaf:knows">
    <div rel="foaf:knows">
      ...
    </div>
  </div>
</div>
```

Although this *could* generate the following triples:

```
<> foaf:knows _:a .
_:a foaf:knows _:b .
_:b foaf:knows _:c .
```

there is no point in doing so, since the triples are useless. The return flag from the recursion step is used to spot this condition. However, the moment a true value is returned from *anywhere* in the hierarchy, it becomes worth completing the [incomplete triple [p.24]]s. For example, if a name is added to the most deeply nested person:

```
<div rel="foaf:knows">
  <div rel="foaf:knows">
    <div rel="foaf:knows">
      <span property="foaf:name">Ivan Herman</span>
    </div>
  </div>
</div>
```

the triples become useful:

```
<> foaf:knows _:a .
_:a foaf:knows _:b .
_:b foaf:knows _:c .
_:c foaf:name "Ivan Herman" .
```

Therefore, setting a return value of 'true' when the `foaf:name` triple is created causes all [incomplete triple [p.24]]s to be completed. This completion happens in the next step.

11. If the previous step resulted in a 'true' flag, *or* [new subject [p.24]] was set to a non-null and non-bnode value, then any [incomplete triple [p.24]]s *within the current context* should be completed:
    The [list of incomplete triples [p.??]] from the current [evaluation context [p.29]] (*not* the [local list of incomplete triples [p.24]]) will contain zero or more predicate URIs. This list is iterated, and each of the predicates is used with [current subject [p.23]] and [new subject [p.24]] to generate a triple. Note that if [new subject [p.24]] is a bnode, then the process of completion will only happen if triples were created during the course of processing descendant elements. Note also that at each level there are *two*, lists of [incomplete triple [p.24]]s; one for the current processing level (which is passed to each child element in the previous step), and one that was received as part of the [evaluation context [p.29]]. It is the latter that is used in processing during this step.
    Note that each [incomplete triple [p.24]] has a [direction] value that it used to determine what will become the subject, and what will become the object, of each generated triple. If [direction [p.34]] is 'forward' then the following triple is generated:
    subject
        [current subject [p.23]]
    predicate
        the predicate from the iterated [incomplete triple [p.24]]
    object
        [new subject [p.24]]
    If [direction [p.34]] is not 'forward' then this is the triple generated:

subject
: [new subject [p.24]]

predicate
: the predicate from the iterated [incomplete triple [p.24]]

object
: [current subject [p.23]]

12. If any triples were created during the current level of processing, *or* [new subject [p.24]] was set to a non-null and non-bnode value, *or* a value of 'true' was returned from the recursion step (step 10, above), then a value of 'true' should be returned from this level of processing. Otherwise a value of false should be returned. This returned value is forwarded to step 11 and determines whether to complete any [incomplete triple [p.24]]s after having recursed into the processing of descendants.

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.

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 [p.??]] and is called the [current subject [p.23]]. 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. @instanceof is unique in that it sets *both* a predicate and an object at the same time. Inline content might also set an object, if @content is not present, but @property is.

6.1. Changing the evaluation context

6.1.1. Setting the [current subject]

When triples are created they will always be in relation to the [current subject [p.23]].

6.1.1.1. The current document

When parsing begins, the [current subject [p.23]] will be the URI of the document being parsed, or a value as set by `base` according to normal XHTML processing rules. This means that any metadata found in the `head` of the document will concern the document itself:

```
<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>
```

This would generate the following triples:

```
<> foaf:primaryTopic <#bbq> .
<> dc:creator "Jo" .
```

It is possible for the data to appear elsewhere in the document:

```
<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>
```

which would still generate the triple:

```
<> dc:creator "Jo" .
```

The value of `base` may change the initial value of [current subject [p.23]]:

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

A parser should now generate the following triples, regardless of the URL from which the XHTML document is served:

```
<http://www.example.org/jo/blog> foaf:primaryTopic <#bbq> .
<http://www.example.org/jo/blog> dc:creator "Jo" .
```

6.1.1.2. Using @about

As processing progresses, any @about attributes will change the [current subject [p.23]]. 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 [p.23]] value. To illustrate how this affects the statements, note in this mark-up how the properties inside the `body` element become part of a new calendar event object, rather than referring to the document as they do in the head of 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>
```

With this mark-up a parser should generate the following triples:

```
<> foaf:primaryTopic <#bbq> .
<> dc:creator "Jo" .
<#bbq> rdf:type cal:Vevent .
<#bbq> cal:summary "one last summer barbecue" .
<#bbq> cal:dtastart "20070916T1600-0500"^^xsd:datetime .
```

Other kinds of resources can be used to set the [current subject [p.23]], not just references to web-pages. Although not advised, email addresses might be used to represent a person:

```
John knows
<a about="mailto:john@example.org"
  rel="foaf:knows" href="mailto:sue@example.org">Sue</a>.

Sue knows
<a about="mailto:sue@example.org"
  rel="foaf:knows" href="mailto:jim@example.org">Jim</a>.
```

This should generate the following triples:

```
<mailto:john@example.org> foaf:knows <mailto:sue@example.org> .
<mailto:sue@example.org> foaf:knows <mailto:jim@example.org> .
```

Similarly, authors may make statements about images:

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

which should generate the following triples:

```
<photo1.jpg> dc:creator "Mark Birbeck" .
```

6.1.1.3. Using @src

If @about is not present, then @src is next in priority order, for setting the subject of a statement. A typical use would be to indicate the licensing type of an image:

```
<img src="photo1.jpg" rel="license" resource="http://creativecommons.org/licenses/by/2.0/" />
```

Since there is no difference between @src and @about, then the information expressed in the last example in the section on @about (the *creator* of an image), could be expressed as follows:

```
<img src="photo1.jpg"
  rel="license" resource="http://creativecommons.org/licenses/by/2.0/"
  property="dc:creator" content="Mark Birbeck"
/>
```

Since normal chaining rules will apply, the image URL can also be used to complete hanging triples:

```
<div about="http://www.blogger.com/profile/1109404" rel="foaf:img">
  <img src="photo1.jpg"
    rel="license" resource="http://creativecommons.org/licenses/by/2.0/"
    property="dc:creator" content="Mark Birbeck"
  />
</div>
```

The complete mark-up yields three triples:

```
<http://www.blogger.com/profile/1109404> foaf:img <photo1.jpg> .
<photo1.jpg> xh:license <http://creativecommons.org/licenses/by/2.0/> .
<photo1.jpg> dc:creator "Mark Birbeck" .
```

6.1.1.4. Creating a new item with @instanceof

Whilst @about explicitly creates a new context for statements, @instanceof does so implicitly. @instanceof works differently to other ways of setting a predicate since the predicate is always `rdf:type`, which means that the processor only requires one attribute, the value of the type.

Since @instanceof is setting the type of an item, this means that if no item exists one should automatically be created. This involves generating a new bnode, and is examined in more detail below; it is mentioned here because the bnode used by the new item will become the subject for further statements.

For example, an author may wish to create mark-up for a person using the FOAF vocabulary, but without having a clear identifier for the item:

```
<div instanceof="foaf:Person">
  <span property="foaf:name">Albert Einstein</span>
  <span property="foaf:givenname">Albert</span>
</div>
```

This mark-up would cause a bnode to be created which has a 'type' of `foaf:Person`, as well as name and given name properties:

```
_:a rdf:type foaf:Person .
_:a foaf:name "Albert Einstein" .
_:a foaf:givenname "Albert" .
```

A bnode is simply a unique identifier that is only available to the processor, not to any external

software. By generating values internally, the processor is able to keep track of properties for `_:a` as being distinct from `_:b`. But by not exposing these values to any external software, it is possible to have complete control over the identifier, as well as preventing further statements being made about the item.

6.1.1.5. Inheriting a subject

As described in the previous two sections, @about will always take precedence and mark a new subject, but if no @about value is available then @instanceof will do the same job, although using an implied identifier, i.e., a bnode.

But if neither @about or @instanceof are present, there are a number of ways that the subject could be arrived at. One of these is to 'inherit' the subject from the containing statement, with the value to be inherited set either explicitly, or implicitly.

6.1.1.5.1. Using an explicit object

The most usual way that an inherited subject might get set would be when the parent statement has an object that is a resource. Returning to the earlier example, in which the long name for Germany was added, the following mark-up was used:

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <span property="foaf:name">Albert Einstein</span>
  <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
  <div rel="dbp:birthPlace" resource="http://dbpedia.org/resource/Germany" />
  <span about="http://dbpedia.org/resource/Germany"
    property="dbp:conventionalLongName">Federal Republic of Germany</span>
</div>
```

In an earlier illustration the subject and object for Germany were elided by removing the @resource, relying on the @about to set the object:

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <span property="foaf:name">Albert Einstein</span>
  <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
  <div rel="dbp:birthPlace">
    <span about="http://dbpedia.org/resource/Germany"
      property="dbp:conventionalLongName">Federal Republic of Germany</span>
  </div>
</div>
```

but it is also possible for authors to achieve the same effect by removing the @about and leaving the @resource:

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <span property="foaf:name">Albert Einstein</span>
  <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
  <div rel="dbp:birthPlace" resource="http://dbpedia.org/resource/Germany">
    <span property="dbp:conventionalLongName">Federal Republic of Germany</span>
  </div>
</div>
```

In this situation, all statements that are 'contained' by the object resource representing Germany (the value in @resource) will have the same subject, making it easy for authors to add additional statements:

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <span property="foaf:name">Albert Einstein</span>
  <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
  <div rel="dbp:birthPlace" resource="http://dbpedia.org/resource/Germany">
    <span property="dbp:conventionalLongName">Federal Republic of Germany</span>
    <span rel="dbp:capital" resource="http://dbpedia.org/resource/Berlin" />
  </div>
</div>
```

Looking at the triples that a parser would generate, we can see that we actually have two groups of statements; the first group are set to refer to the @about that contains them:

```
<http://dbpedia.org/resource/Albert_Einstein> foaf:name "Albert Einstein" .
<http://dbpedia.org/resource/Albert_Einstein> dbp:dateOfBirth "1879-03-14"^^xsd:date .
<http://dbpedia.org/resource/Albert_Einstein> dbp:birthPlace <http://dbpedia.org/resource/Germany> .
```

whilst the second group refer to the @resource that contains them:

```
<http://dbpedia.org/resource/Germany>
  dbp:conventionalLongName "Federal Republic of Germany" .
<http://dbpedia.org/resource/Germany>
  dbp:capital <http://dbpedia.org/resource/Berlin> .
```

Note also that the same principle described here applies to @src and @href.

6.1.1.5.2. Using an implicit object

There will be occasions when the the author wants to elide the subject and object as shown above, but is not concerned to name the resource that is common to the two statements (i.e., the object of the first statement, which is the subject of the second). For example, to indicate that Einstein was influenced by Spinoza the following mark-up could well be used:

```
<div about="http://dbpedia.org/resource/Baruch_Spinoza" rel="dbp:influenced">
  <div about="http://dbpedia.org/resource/Albert_Einstein">
    <span property="foaf:name">Albert Einstein</span>
    <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
  </div>
</div>
```

A parser should generate the following triples:

```
<http://dbpedia.org/resource/Baruch_Spinoza>
  dbp:influenced <http://dbpedia.org/resource/Albert_Einstein> .
<http://dbpedia.org/resource/Albert_Einstein> foaf:name "Albert Einstein" .
<http://dbpedia.org/resource/Albert_Einstein> dbp:dateOfBirth "1879-03-14"^^xsd:date .
```

However, an author could just as easily say that Spinoza influenced *something by the name of Albert Einstein, that was born on March 14th, 1879*:

```
<div about="http://dbpedia.org/resource/Baruch_Spinoza" rel="dbp:influenced">
  <div>
    <span property="foaf:name">Albert Einstein</span>
    <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
  </div>
</div>
```

In RDF terms, the item that 'represents' Einstein is *anonymous*, since it has no URI to identify it. However, the item is given an automatically generated bnode, and it is onto this idenfifier that all child statements are attached:

A parser should generate the following triples:

```
<http://dbpedia.org/resource/Baruch_Spinoza> dbp:influenced _:a .
_:a foaf:name "Albert Einstein" .
_:a dbp:dateOfBirth "1879-03-14"^^xsd:date .
```

Note that the `div` is superfluous, and a parser should create the intermediate object even if the element is removed:

```
<div about="http://dbpedia.org/resource/Baruch_Spinoza" rel="dbp:influenced">
  <span property="foaf:name">Albert Einstein</span>
  <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
</div>
```

An alternative pattern is to *keep* the `div` and move the @rel onto it:

```
<div about="http://dbpedia.org/resource/Baruch_Spinoza">
  <div rel="dbp:influenced">
    <span property="foaf:name">Albert Einstein</span>
    <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
  </div>
</div>
```

From the point of view of the mark-up, this latter layout is to be preferred, since it draws attention to the 'hanging rel'. But from the point of view of a parser, all of these permutations need to be supported.

6.2. Completing 'incomplete triples'

When a new subject is calculated, it is also used to complete any incomplete triples that are pending. This situation arises when the author wants to 'chain' a number of statements together. For example, an author could have a statement that Albert Einstein was born in Germany:

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <span property="foaf:name">Albert Einstein</span>
  <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
  <div rel="dbp:birthPlace" resource="http://dbpedia.org/resource/Germany" />
</div>
```

and then a further statement that the 'long name' for Germany is the *Federal Republic of Germany*:

```
<span about="http://dbpedia.org/resource/Germany"
  property="dbp:conventionalLongName">Federal Republic of Germany</span>
```

RDFa allows authors to insert this statement as a self-contained unit into other contexts:

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <span property="foaf:name">Albert Einstein</span>
  <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
  <div rel="dbp:birthPlace" resource="http://dbpedia.org/resource/Germany" />
  <span about="http://dbpedia.org/resource/Germany"
    property="dbp:conventionalLongName">Federal Republic of Germany</span>
</div>
```

But it also allows authors to avoid unnecessary repetition and to 'normalise' out duplicate identifiers, in this case the one for Germany:

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <span property="foaf:name">Albert Einstein</span>
  <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
  <div rel="dbp:birthPlace">
    <span about="http://dbpedia.org/resource/Germany"
      property="dbp:conventionalLongName">Federal Republic of Germany</span>
  </div>
</div>
```

When this happens the @rel for 'birth place' is regarded as a 'hanging rel' because it has not yet generated any triples, but these 'incomplete triples' are completed by the @about that appears on the next line. The first step is therefore to store the two parts of the triple that the parser *does* have, but without an object:

```
<http://dbpedia.org/resource/Albert_Einstein> dbp:birthPlace ? .
```

Then as processing continues, the parser encounters the subject of the statement about the long name for Germany, and this is used in two ways. First it is used to complete the 'incomplete triple':

```
<http://dbpedia.org/resource/Albert_Einstein> dbp:birthPlace <http://dbpedia.org/resource/Germany> .
```

and second it is used to generate its own triple:

```
<http://dbpedia.org/resource/Germany> dbp:conventionalLongName "Federal Republic of Germany" .
```

Note that each occurrence of @about will complete any incomplete triples. For example, to mark up the fact that Albert Einstein had both German and American citizenship, an author need only specify one @rel value that is then used with multiple @about values:

```
<div about="http://dbpedia.org/resource/Albert_Einstein" rel="dbp:citizenship">
  <span about="http://dbpedia.org/resource/Germany" />
  <span about="http://dbpedia.org/resource/United_States" />
</div>
```

In this example there is one incomplete triple:

```
<http://dbpedia.org/resource/Albert_Einstein> dbp:citizenship ? .
```

When the processor meets each of the @about values, this triple is completed, giving:

```
<http://dbpedia.org/resource/Albert_Einstein>
  dbp:citizenship <http://dbpedia.org/resource/Germany> .
<http://dbpedia.org/resource/Albert_Einstein>
  dbp:citizenship <http://dbpedia.org/resource/United_States> .
```

These examples show how @about completes triples, but there are other situations that can have the same effect. For example, when @instanceof creates a new bnode (as described above), that will be used to complete any 'incomplete triples'. To illustrate, to indicate that Spinoza influenced both Einstein and Schopenhauer, the following mark-up could be used:

```
<div about="http://dbpedia.org/resource/Baruch_Spinoza">
  <div rel="dbp:influenced">
    <div instanceof="foaf:Person">
      <span property="foaf:name">Albert Einstein</span>
      <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
    </div>
    <div instanceof="foaf:Person">
      <span property="foaf:name">Arthur Schopenhauer</span>
      <span property="dbp:dateOfBirth" datatype="xsd:date">1788-02-22</span>
    </div>
  </div>
</div>
```

First the following incomplete triple is stored:

```
<http://dbpedia.org/resource/Baruch_Spinoza> dbp:influenced ? .
```

Then when the parser processes the two occurences of @instanceof, each generates a bnode, which is used to both complete the 'incomplete triple', and to set the subject for further statements:

```
<http://dbpedia.org/resource/Baruch_Spinoza"> dbp:influenced _:a .
_:a rdf:type foaf:Person .
_:a foaf:name "Albert Einstein" .
_:a dbp:dateOfBirth "1879-03-14"^^xsd:date .
<http://dbpedia.org/resource/Baruch_Spinoza"> dbp:influenced _:b .
_:b rdf:type foaf:Person .
_:b foaf:name "Arthur Schopenhauer" .
_:b dbp:dateOfBirth "1788-02-22"^^xsd:date .
```

Triples are also 'completed' if any one of @property, @rel or @rev are present. However, unlike the situation when @about or @instanceof are present, all predicates are attached to one bnode:

```
<div about="http://dbpedia.org/resource/Baruch_Spinoza" rel="dbp:influenced">
  <span property="foaf:name">Albert Einstein</span>
  <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
  <div rel="dbp:citizenship">
    <span about="http://dbpedia.org/resource/Germany" />
    <span about="http://dbpedia.org/resource/United_States" />
  </div>
</div>
```

This example has two 'hanging rels', and so two situations when 'incomplete triples' will be created. Processing would proceed as follows; first an incomplete triple is stored:

```
<http://dbpedia.org/resource/Baruch_Spinoza> dbp:influenced ? .
```

Next, the parser processes the predicate values for `foaf:name`, `dbp:dateOfBirth` and `dbp:citizenship`, but note that only the first needs to 'complete' the 'hanging rel'. So processing `foaf:name` generates two triples:

```
<http://dbpedia.org/resource/Baruch_Spinoza> dbp:influenced _:a .
_:a foaf:name "Alber Einstein" .
```

but processing `dbp:dateOfBirth` generates only one:

```
_:a dbp:dateOfBirth "1879-03-14"^^xsd:date .
```

Processing `dbp:citizenship` also uses the same bnode, but note that it also generates its own 'incomplete triple':

```
  _:a dbp:citizenship ? .
```

As before, the two occurrences of @about complete the 'incomplete triple', once each:

```
_:a dbp:citizenship <http://dbpedia.org/resource/Germany> .
_:a dbp:citizenship <http://dbpedia.org/resource/United_States> .
```

The entire set of triples that a parser should generate are as follows:

```
<http://dbpedia.org/resource/Baruch_Spinoza> dbp:influenced _:a .
_:a foaf:name "Alber Einstein" .
_:a dbp:dateOfBirth "1879-03-14"^^xsd:date .
_:a dbp:citizenship <http://dbpedia.org/resource/Germany> .
_:a dbp:citizenship <http://dbpedia.org/resource/United_States> .
```

6.3. Object resolution

Although objects have been discussed in the previous sections, as part of the explanation of subject resolution, chaining, evaluation contexts, and so on, this section will look at objects in more detail.

There are two types of object, [URI resource [p.47]]s and [literal [p.17]]s.

A [literal [p.17]] object can be set by using @property to express a [predicate [p.19]], 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 [p.19]], and then *either* using one of @href, @resource or @src to provide an object resource explicitly, *or* using the chaining techniques described above to obtain an object from a nested subject, or from a bnode.

6.3.1. 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.

6.3.1.1. Plain Literals

@content can be used to indicate a [plain literal [p.17]], as follows:

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

The [plain literal [p.17]] 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 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>
```

6.3.1.1.1. Language Tags

RDF allows [plain literal [p.17]]s to have a language tag, as illustrated by the following example from [RDFTESTS-RDFMS-XMLLANG-TEST006] [p.81] :

```
<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.80]], 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>
```

6.3.1.2. Typed literals

Literals 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 .
```

6.3.1.3. 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>
  <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 given here 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>
  (b. March 14, 1879, d. 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 [p.48]] 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 [p.48]] 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 itself includes RDFa:

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

In this example the nested RDFa should not be parsed. This effectively means 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.

6.3.2. URI object resolution

Most of the rules governing the processing of objects that are resources are to be found in the processing descriptions given above, since they are important for establishing the subject. This section aims to highlight general concepts, and anything that might have been missed.

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, or with the subject value of any nested statement if none of these attributes are present.

@rel and @rev are essentially the inverse of each other; whilst @rel establishes a relationship between the [current subject [p.23]] as subject, and the [current object resource [p.24]] as the object, @rev does the exact opposite, and uses the [current object resource [p.24]] as the subject, and the [current subject [p.23]] as the object.

6.3.2.1. 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> .
```

6.3.2.2. Using @href

If no @resource is present, then @href is next in priority order, for setting the object.

When a 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:john@example.org"
      rel="foaf:knows" href="mailto:sue@example.org" />
```

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 other, for example when a picture is taken *by* Mark, but that picture also *depicts* him:

```
<img src="photo1.jpg" rel="dc:creator" rev="foaf:img"
   href="http://www.blogger.com/profile/1109404" />
```

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> .
```

6.3.2.3. Incomplete triples

When a triple predicate has been expressed using @rel or @rev, but no @href, @src, or @resource exists on the same element, there is a 'hanging rel'. This causes the current subject and all possible predicates (with an indicator of whether they are 'forwards, i.e., @rel values, or not, i.e., @rev values), to be stored as 'incomplete triples' pending discovery of a subject that could be used to 'complete' those triples.

This process is described in more detail in Completing 'Incomplete Triples' [p.43] .

6.3.2.4. Referencing Blank Nodes

To establish relationships between [bnode [p.19]]s, the [unique anonymous ID] must be set explicity using a CURIE [bnode [p.19]] as subject or object. For example, if our desired output is the following [triple [p.15]]s:

```
_:a foaf:mbox <mailto:john@example.org> .
_:b foaf:mbox <mailto:sue@example.org> .
_:a foaf:knows _:b .
```

we could use the following XHTML:

```
<link about="[_:a]" rel="foaf:mbox"
  href="mailto:john@example.org" />
<link about="[_:b]" rel="foaf:mbox"
  href="mailto:sue@example.org" />
<link about="[_:a]" rel="foaf:knows"
  resource="[_:b]" />
```

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

```
<div about="[_:a]">
  JohnSmith can be reached via
  <a rel="foaf:mbox"
    href="mailto:john@example.org">
    email
  </a>.
  <span rel="foaf:knows" resource="[_:b]">He knows Sue.</span>
</div>

<div about="[_:b]">
  Sue can be reached via
  <a rel="foaf:mbox"
```

```
      href="mailto:sue@example.org">
      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* 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 mapping (`http://www.w3.org/1999/xhtml/vocab#`). 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 URI. Since it is difficult to distinguish between CURIEs and URIs, 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;
- a mapping to use with the default prefix (for example, `:p`);
- a mapping to use when there is no prefix (for example, `p`);
- a mapping to use with the '_' prefix, which is used to generate unique identifiers (for example, `_:p`).

These values are defined in RDFa as follows:

- the **set of mappings from prefixes to URIs** is provided by the current in-scope namespace declarations of the [current element [p.30]] during parsing;
- the **mapping to use with the default prefix** is the current default mapping;
- the **mapping to use when there is no prefix** is not defined;
- the **mapping to use with the '_' prefix**, is not explicitly stated, but since it is used to generate [bnode [p.19]]s, its implementation needs to be compatible with the RDF definition.

A CURIE is a representation of a full URI. This URI is obtained by taking the currently in-scope mapping that is associated with `prefix`, and concatenating it with the `reference`. The resulting URI MUST be a syntactically valid IRI [IRI [p.79]]. For a more detailed explanation see CURIE and URI Processing [p.26]

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.79]]. 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.80]]. 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.57]
    @about, @content, @datatype, @instanceof, @property, @rel, @resource, @rev

XHTML+RDFa also uses the Ruby Annotation module as defined in [RUBY [p.79]]:

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.63] .

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.80]].

9.1. Datatypes

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

Data type	Description
CURIE	A Compact URI or CURIE [p.53] .
CURIEs	A whitespace separated list of CURIE [p.57] s.
URIorCURIE	A URI or safe_curie [p.53] .

9.2. Metainformation Attributes Collection

The following attributes are included in the attribute collection, and take values in the associated datatype:

Attributes	Notes
about (URIorCURIE [p.57])	
content (CDATA)	
datatype (CURIE [p.57])	If not specified, then the default value is `string` as defined in [XMLSCHEMA [p.80]].
instanceof (CURIEs [p.57])	
property (reserved word | CURIE [p.57])+	See the reserved values list in @property attribute values [p.58]
rel (reserved word | CURIE [p.57])+	See the reserved values list in @rel/@rev Attribute Values [p.58]
resource (URIorCURIE [p.57])	
rev (reserved word | CURIE [p.57])+	See the reserved values list in @rel/@rev attribute values [p.58]

An implementation of this module can be found in Appendix B [p.63] .

9.3. @property attribute values

The list of predefined values (in the XHTML vocabulary space) is given below.

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.

9.4. @rel/@rev attribute values

The list of reserved values for @rel and @rev are:

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" resource="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>
```

which would generate the following triples:

```
<> xhv:cite <http://www.example.com/books/the_two_towers> .
```

`cite` is also useful for referencing specifications:

cite to reference another specification

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

which would generate the following triples:

```
<> xhv:cite <http://www.w3.org/TR/REC-xml> .
```

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.

last
: Refers to the last resource in a collection of resources.

license
: Refers to a resource that defines the license associated with this document.

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.79]].

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.79]] module.

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

stylesheet
: Refers to a resource acting as a stylesheet for this document.

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.

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

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.57] is compatible with XHTML Modularization [XHTMLMOD [p.80]], and can be used to construct other such languages.

If a language includes @xml:base [XMLBASE [p.79]], an RDFa parser for that host language must process it, and use its value to set [base [p.23]].

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 2007-2008 W3C (MIT, ERCIM, Keio), All Rights Reserved.
     Revision: $Id: rdfa-syntax.ps,v 1.3 2008/02/18 01:54:54 smccarro 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 1998-2008 W3C (MIT, ERCIM, Keio), All Rights Reserved.
     Revision: $Id: rdfa-syntax.ps,v 1.3 2008/02/18 01:54:54 smccarro 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 1998-2008 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  "XHTML+RDFa 1.0" >

<!-- 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 % html.attlist  "IGNORE" > -->
<!ENTITY % html.attlist  "IGNORE" >
<!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;'"
>
<!ENTITY % XHTML.version.attrib
     "version      %FPI.datatype;           #FIXED '%XHTML.version;'"
>
<!ATTLIST %html.qname;
```

```
      %Common.attrib;
      %XSI.schemaLocation.attrib;
      %XHTML.version.attrib;
>
<!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.80]] of the public identifiers for XHTML+RDFa 1.0.

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

--  XHTML+RDFa Catalog Data File

    Revision:  $Revision: 1.3 $

    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.

[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

[URI]
"*Uniform Resource Identifier (URI): Generic Syntax*", RFC 3986, T. Berners-Lee, R. Fielding, L. Masinter, January 2005.
Available at: http://www.ietf.org/rfc/rfc3986.txt

[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/

[XMLNS]
"*Namespaces in XML*", W3C Recommendation, T. Bray *et al.*, *eds.*, 14 January 1999.
Available at: http://www.w3.org/TR/1999/REC-xml-names-19990114

[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
[DBPEDIA]
DBPedia (See http://dbpedia.org/.)
[DC]
Dublin Core Metadata Initiative (DCMI) (See http://dublincore.org/.)
[FOAF-PROJECT]
The FOAF Project (See http://www.foaf-project.org.)
[RDFHTML]
RDF-in-HTML Task Force (See http://w3.org/2001/sw/BestPractices/HTML/.)
[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/.)
[TURTLE]
Turtle: Terse RDF Triple Language, David Beckett, Tim Berners-Lee, January 2008. See: http://www.w3.org/TeamSubmission/turtle/
[XHTML2-WG]
XHTML 2 Working Group (See http://w3.org/MarkUp/.)
[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.

2008-01-23: Updated to reflect latest task-force thinking re- the processing of legacy values in @rel and @rev. As part of this work, made the whole processing of CURIEs and URIs much clearer. [MarkBirbeck]

2008-01-03: Updated to reflect latest task-force thinking re- the processing model, in particular regarding 'chaining', and the behaviour of @instanceof. [MarkBirbeck]

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: