RDFa Syntax

A collection of attributes for layering RDF on XML languages

W3C Editor's Draft 04 September 2007

This version:
http://www.w3.org/MarkUp/2007/ED-rdfa-syntax-20070904
Latest version:
http://www.w3.org/TR/rdfa-syntax
Previous Editor's Draft:
http://www.w3.org/MarkUp/2007/ED-rdfa-syntax-20070830
Diff from previous Editor's Draft:
rdfa-syntax-diff.html
Editors:
Mark Birbeck, x-port.net Ltd. mark.birbeck@x-port.net
Steven Pemberton, CWI
Ben Adida, Creative Commons ben@adida.net
Shane McCarron, Applied Testing and Technology, Inc. shane@aptest.com

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 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark and document use rules apply.

Abstract

Current web pages, written in HTML, contain significant inherent structured data. 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. 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, and topic can be published as easily as the original photo itself, enabling structured search and sharing.

RDFa is a syntax for expressing this structured data in XHTML. The rendered, hypertext data of XHTML is reused by the RDFa markup, so that publishers don't repeat themselves. The underlying abstract representation is RDF, 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 do not have to be different rules for different structures; this allows authors and publishers of data to define their own formats without having to update software, or register formats via a central authority.

This document is a detailed syntax specification for RDFa. For a more gentle introduction, please consult the RDFa Primer.

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 an internal draft produced by the Semantic Web Deployment Working Group [SWD-WG] [p.36] , in cooperation with the XHTML 2 Working Group [XHTML2-WG] [p.36] . Initial work on RDFa began with the Semantic Web Best Practices and Deployment Working Group [SWBPD-WG] [p.36] .

At this time, the prose of the document is not complete and is not *quite* consistent with the current agreements the task force has with regard to processing model, etc. The section on Processing Model [p.31] is up-to-date; if there is a discrepancy between the prose and that section, that section is most likely correct.

This document has no official standing within the W3C. It is also a work in progress, which means it may change at any time, without warning, and you shouldn't rely on anything in this document.

Table of Contents

1. Motivation 5
2. Terms and Abbreviations 7
    2.1. Namespaces 7
    2.2. RDF Terminology 7
        2.2.1. N-Triples 8
    2.3. Using `xml:base` 8
    2.4. Compact URIs 9
        2.4.1. CURIE Syntax Definition 9
3. Introduction to the structure of RDFa 11

3.1. Overview 11
3.2. Qualifying document components 13
3.3. Relating document components 14
3.4. Global RDF statements 15
4. RDFa in detail 17
4.1. Processing 17
4.2. Compact URIs 17
4.3. Establishing the subject 17
4.3.1. The `about` attribute 17
4.4. Establishing the predicate 18
4.4.1. The `property` attribute 18
4.4.2. The `rel` attribute 19
4.4.3. The `rev` attribute 19
4.4.4. Using both `rel` and `rev` attribute 19
4.4.5. Multiple Attribute Values 20
4.5. Establishing the object 20
4.5.1. Literal object resolution using the `content` attribute 20
4.5.2. URI object resolution using the `href` attribute 21
4.5.3. `rel` without `href` 21
4.6. Summary 21
5. RDF Concepts 23
5.1. Literals as Objects 23
5.1.1. Without `datatype` 23
5.1.2. With `datatype` 25
5.2. Blank nodes 25
5.3. Reification 27
6. Examples 29
6.1. Creative Commons 29
6.2. FOAF 29
7. Processing Model 31
7.1. Overview 31
7.2. Evaluation Context 31
7.3. Processing 32
A. References 35
A.1. Related Specifications 35
A.2. Related Activities 35
B. Change History 37
C. Acknowledgments 39

1. Motivation

RDF/XML [RDF-SYNTAX] [p.35] provides sufficient flexibility to represent all of the abstract concepts in RDF [RDF-CONCEPTS] [p.35] . However, it presents two challenges; first it is difficult or impossible to validate documents that contain RDF/XML using XML Schemas or DTD's, which makes it difficult to import RDF/XML into other markup languages. Whilst newer schema languages such as RELAX NG [RELAXNG] [p.36] 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 "Mark Birbeck" should not need to repeat this name for RDF expression of the same concept: the existing markup should be augmentable to RDF with minimal data repetition.

Third, as users often want to transfer structured data from one application to another, sometimes to or from a non-web-based application, it is highly beneficial to express the web data's structure "in context." A user can then get contextual information about specific rendered data, for example by "right-clicking" on an item of interest.

In the past, some attributes were 'hard-wired' directly into the markup language to represent specific concepts. For example, in XHTML 1.1 [XHTML11 [p.35]] and HTML [HTML4 [p.35]] there is a `cite` attribute. The attribute allows an author to add information to a document to indicate the origin of a quote.

The motivation of RDFa is to devise a means by which documents can be augmented with metadata in a generic rather than hard-wired manner, using property values from the growing range of available taxonomies, reusing existing content from the host language.

This specification takes the pressure off language authors to anticipate all the structural requirements users of their language might have by outlining a new XML syntax for RDF that relies only on XML attributes, and so can be easily imported into other markup languages allowing them to carry arbitrary RDF.

2. Terms and Abbreviations

2.1. Namespaces

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

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

2.2. RDF Terminology

This document uses the following terminology defined in [RDF-CONCEPTS] [p.35] :

- URI reference
- literal
- plain literal
- typed literal
- XML literal
- XML value
- node
- blank node

- triple
- RDF graph

We also add two further concepts, an [RDFa element] and the [context statement] both of which are explained in the processing section.

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

In RDFa, a subject [RDF URI reference] is indicated using the attribute `about` and predicates are represented using one of the attributes `property`, `rel`, or `rev`. Objects which are [RDF URI reference]s are represented using the attribute `href`, whilst objects that are [literal]s are represented either with the attribute `content`, or the content of the element in question.

2.2.1. N-Triples

Most of the examples in this document are shown translated into N-Triples [N-TRIPLES] [p.35] syntax, with a slight variation in that QNames can be used to replace a URI reference. To tell them apart, the QName will have no angle brackets, e.g. the triple:

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

should be read as an abbreviation for the N-Triples syntax:

```
<http://internet-apps.blogspot.com/>
  <http://purl.org/dc/elements/1.1/creator>
  "Mark Birbeck" .
```

Datatypes can also be abbreviated, so the following:

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

should be read as an abbreviation for this N-Triples statement:

```
<> <http://purl.org/dc/elements/1.1/creator>
   "E = mc<sup>2</sup>: The Most Urgent Problem of Our Time"^^<http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral> .
```

2.3. Using `xml:base`

All [RDF URI references] are subject to `xml:base` [XMLBASE] [p.35] . Note that this means that in the absence of an `xml:base` attribute, the document containing the RDF statements is *itself* the base.

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

```
<span xml:base="http://internet-apps.blogspot.com/">
    <link about="" rel="dc:creator" href="http://www.blogger.com/profile/1109404" />
    <meta 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" .
```

2.4. Compact URIs

In order to allow for the compact expression of RDF statements, RDFa uses a variant of QNames [QName] that allows the contraction of all URIs (QNames have a syntactic restriction on the sorts of URI that can be contracted).

These Compact URIs are called CURIEs here.

The `rel`, `rev`, and `property` attributes accept CURIE-only datatypes, while `href` and `about` accept mixed CURIE/URI data. In particular, the following notation is a valid RDFa statement:

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

which generates the following triple, as expected:

```
<> cc:license <http://creativecommons.org/licenses/by/nc-nd/3.0/> .
```

2.4.1. CURIE Syntax Definition

Note that this syntax definition will ultimately be defined in an external document [CURIE [p.??]].

A CURIE is comprised of two components, a *prefix* and a *reference*. The prefix is separated from the reference by a colon (`:`). To disambiguate a CURIE when it appears in a context where a normal [URI [p.??]] may also be used, the entire CURIE is permitted to be enclosed in brackets (`[`, `]`).

```
curie       :=   basic_curie | safe_curie

safe_curie  :=   '[' basic_curie ']'

basic_curie :=   [ prefix ] ':' reference

prefix      :=   NCName

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

When CURIES are used in an XML-based host language, prefix values MUST be able to be defined using the 'xmlns:' syntax specified in [XMLNAMES [p.??]]. Such host languages MAY also provide additional prefix mapping definition mechanisms.

When CURIES are used in a non-XML host language, the host language MUST provide a mechanism for defining the mapping from the `prefix` to an IRI.

A host language MAY define a mechanism for defining a default prefix value. In such a host language, if the `prefix` is omitted from a CURIE, the default prefix value is used.

The concatenation of the prefix associated with a CURIE and its `reference` MUST be an IRI [IRI [p.35]].

The CURIE prefix '_' is reserved. For this reason, prefix declarations using '_' SHOULD be avoided by authors.

Host languages MAY define additional constraints on these syntax rules when CURIES are used in the context of those host languages. Host languages MUST NOT relax the CURIE syntax constraints defined this specification.

3. Introduction to the structure of RDFa

3.1. Overview

The main idea behind the syntax for RDFa is that existing data should be easy to update to convey RDF triples. Thus, the bulk of RDFa can be expressed using only attributes applied to existing elements within the XML document. These are

`about`,
: a URI or CURIE, used for stating what the data is about (the 'subject' in RDF terminology)

`rel`,
: a list of CURIEs, used for expressing relationships between two resources (a 'predicate' in RDF)

`rev`,
: a list of CURIEs, used for expressing reverse relationships between two resources (also a 'predicate')

`property`,
: a list of CURIEs, used for expressing relationships between the subject and some literal text (also a 'predicate')

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

`resource`,
: a URI or CURIE, for expressing the partner resource of a relationship that is not intended to be 'clickable' (also an 'object')

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

`datatype`,
: a schema datatype, to express the datatype of a literal

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

As you will remark, several of these attributes are borrowed directly from HTML, and have largely the same meaning as in HTML.

For example, given an XHTML chunk as follows:

```
This photo was taken by
<span class="author">Mark Birbeck</span>.
```

a simple attribute augmentation can yield an RDF triple:

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

which yields:

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

Similarly, links can be augmented to express RDF triples. Consider an XHTML chunk:

```
This photo was taken by
<a href="http://www.blogger.com/profile/1109404">Mark Birbeck</a>.
```

When the RDF object is a URI, the RDF predicate is designated using `rel`:

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

which yields:

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

It's important to note that the various RDFa attributes can be used on any existing element of the XML dialect. Note also that one can express a reverse relationship using the `rev` attribute. For example, if the photo in question is actually a depiction of Mark, one could write:

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

which would yield:

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

Both relations can be expressed simultaneously:

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

which then yields both triples:

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

It's possible to go further and add the attributes used for denoting statements in which the object is a [literal]:

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

which would then yield:

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

Or going further:

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

which would then yield:

```
<photo1.jpg> dc:creator <http://www.blogger.com/profile/1109404> .
<http://www.blogger.com/profile/1109404> foaf:img <photo1.jpg> .
<photo1.jpg> dc:title "Portrait of Mark" .
```

It's possible to do all of this without ambiguity, since the `property` attribute always denotes a predicate in a statement in which the object is a [literal], whilst the `rel` and `rev` attributes always denote a predicate in a statement in which the object is a [URI reference]. Put a different way, `property` always works with `content`, whilst `rel` and `rev` work with `href` or `resource`.

Of course, the more natural way to express the three above triples is to strive to make all metadata literals and URIs meaningful within the host XML dialect. Specifically, in the case of XHTML2, it makes sense to render as much of the useful metadata as possible and use RDFa to mark up this rendered data. The following XHTML thus generates the same triples shown above.

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

The value of the `about` attribute is inherited from parent elements. The following XHTML thus generates the very same triples as the XHTML above.

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

3.2. Qualifying document components

A second feature of RDFa is that it is possible to use parts of the host document to provide the [subject] of a [triple]. This marks RDFa from other approaches to serialising RDF, in that the the same syntax can now be used to make statements about parts of a document, and external documents.

It is possible to make such statements using the syntax introduced in the examples above:

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

Note that the use of an `href` attribute on an element does not necessarily make it clickable, nor does it affect the display of the element (such as underlining it in blue). These are decisions made by the host language.

3.3. Relating document components

Using qualifying statements, RDFa allows a single XML dialect document to include multiple RDF entities. Relations between the various entities of a given page can also be defined using RDFa notation.

Consider the following XHTML, which defines two RDF entities of type `taxo:topic`, two RDF entities of type `biblio:Publication`, metadata pertinent to each publication, including `dc:title` and `dc:creator`, and relations of type `taxo:topics` between the publications and tags:

```
<html xmlns:dc="http://purl.org/dc/elements/1.1/">
    <head>
        <title>Mark's Publications</title>
    </head>
    <body>

       <h2>Tags</h2>
       <div id="tag_standards">
          <link rel="rdf:type" href="[taxo:topic]" />
          Standards
       </div>
       <div id="tag_xforms">
          <link rel="rdf:type" href="[taxo:topic]" />
          XForms
       </div>

       <h2>Publications</h2>
       <div id="publication_1">
         <link rel="rdf:type" href="[biblio:Publication]" />
         <link rel="dc:creator" href="http://www.blogger.com/profile/1109404" />
         <meta property="dc:title">A Standards-Based Virtual Machine</meta>
         <link rel="taxo:topics" href="#tag_standards" />
       </div>
```

```
        <div id="publication_2">
          <link rel="rdf:type" href="[biblio:Publication]" />
          <link rel="dc:creator" href="http://www.blogger.com/profile/1109404" />
          <meta property="dc:title">XForms and Internet Applications</meta>
          <link rel="taxo:topics" href="#tag_standards" />
          <link rel="taxo:topics" href="#tag_xforms" />
        </div>
    </body>
</html>
```

This yields the expected triples:

```
<#tag_standards> rdf:type taxo:topic .
<#tag_xforms> rdf:type taxo:topic .

<#publication_1> rdf:type biblio:Publication .
<#publication_1> dc:creator <http://www.blogger.com/profile/1109404>
<#publication_1> dc:title "A Standards-Based Virtual Machine"^^rdf:XMLLiteral .
<#publication_1> taxo:topics <#tag_standards> .

<#publication_2> rdf:type biblio:Publication .
<#publication_2> dc:creator <http://www.blogger.com/profile/1109404>
<#publication_2> dc:title "XForms and Internet Applications"^^rdf:XMLLiteral .
<#publication_2> taxo:topics <#tag_standards> .
<#publication_2> taxo:topics <#tag_xforms> .
```

Beyond this theoretical example, this application of RDFa is particularly useful for formats like FOAF. (See examples.)

3.4. Global RDF statements

The previous series of examples may mislead one to think that RDFa statements are only contextual, only meant to qualify existing elements. However, as the first examples implied, a fixed `about` attribute can be used to specify a global subject. It is actually quite easy to make independent, global RDF statements. Statements like:

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

will produce the same triple no matter where they're located in the document:

```
<> cc:license <http://creativecommons.org/licenses/by-nc-nd/3.0/> .
```

4. RDFa in detail

4.1. Processing

An [RDFa element] is defined as any XML element that contains one or more RDFa attributes `about`, `property`, `rel`, `rev`, `href`, `src`, `resource`, `content` (but not `datatype`).

Processing proceeds by examining each [RDFa element] in turn. The [RDFa element] under consideration at any time is called the [current statement], and its parent element (which does not need to be an [RDFa element) is called the [context statement].

As each [RDFa element] is examined, the processor establishes the RDF triples it generates. Each of the attributes `rel`, `rev`, and `property` expresses one or more triples: each accepts space-separated values, and each such value the predicate for one triple. The other attributes are used to express the subject and object of the triples.

4.2. Compact URIs

When the value of an attribute is of type CURIE/URI [REF], it can either be a URI, absolute or relative, or a CURIE in square brackets. Some example values are:

- `#person` : the base URI with `#person` appended to it. Refers to the XML element with `id="person"`, if such an element exists.
- `http://creativecommons.org/licenses/by-nc-nd/3.0/` : the absolute URI as indicated.
- `[finance:GOOG]` : the CURIE `finance:GOOG`.

@@ The syntax is prefix ":" Tail, where Tail is any tail of the production for IRI refs [ref]. URIs are constructed by concatenating the URI associated with the prefix and the Tail. The result MUST be a valid IRI ref.

4.3. Establishing the subject

4.3.1. The `about` attribute

In most cases, the subject of a statement is determined by the `about` attribute, on the element itself, or otherwise from the closest ancestor of that element that has an `about` attribute.

There is one exception: if a closer ancestor element includes a `rel` or `rev` attribute with no `href`, `src` or `resource`, then the subject is the CURIE/URI that corresponds to that parent element (@@ as described later under object resolution. @@ will be moved here)

If neither type of subject ancestor is found then the subject is the current document.

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

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

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

will inherit the `about` attribute from the enclosing `div` and yield the expected triple:

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

@@ Example of using rel and rev to establish the subject

4.4. Establishing the predicate

The predicate of a statement is specified using a `property`, `rel` or `rev` attribute. These attributes can be placed on any element in a document, can co-exist on the same element. Each of these attributes accepts space-separated CURIEs, each of which expresses exactly one triple. The attribute indicates the type of resolution to use for the subject and object of the triple. For simplicity, we assume an attribute value of a single CURIE. We explain the space-separated multiple-values situation later.

4.4.1. The `property` attribute

A `property` attribute designates a predicate whose object is a literal. The object of the triple is determined using [literal] object resolution (Section 4.4). The subject of the triple is determined using subject resolution (Section 4.3). The following example indicates the name of the author responsible for the text being quoted:

```
<blockquote about="#q1">
    <p>
        Rodion Romanovitch! My dear friend! If you go on in this way
        you will go mad, I am positive! Drink, pray, if only a few drops!
    </p>
    <p>
        by <span property="dc:creator">Fyodor Dostoevsky</span>
    </p>
</blockquote>
```

4.4.2. The `rel` attribute

A `rel` attribute designates a predicate whose object is a non-literal. The subject of the triple is determined using subject resolution (Section 4.3). The object of the triple is determined using [URI reference] object resolution (Section 4.4). The following example indicates that one 'FOAF person' knows another:

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

The triple generated is:

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

4.4.3. The `rev` attribute

A `rev` attribute, like its cousin the `rel` attribute, indicates a predicate whose object is a non-literal, though its subject and object resolutions are reversed. The subject of the triple is determined using [URI reference] *object* resolution (Section 4.4). The object of the triple is determined using *subject* resolution (Section 4.3). Note that resolution is effectively the same as if the `rev` attribute had been a `rel` attribute with object and subject reversed. The following example indicates that one 'FOAF person' knows another:

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

and the [triple] generated is essentially a reversal of our previous example:

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

4.4.4. Using both `rel` and `rev` attribute

It is perfectly acceptable to use both `rel` and `rev` attributes within the same element. Predictably, this approach yields two triples, without repeating the subject and object. For example:

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

expresses:

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

The predicates need not be the same, of course.

4.4.5. Multiple Attribute Values

The `rel`, `rev`, and `property` attributes accept multiple space-separated CURIEs as a single attribute value. When there is more than one CURIE, then each expresses the exact same triples it would if it were the single CURIE in the attribute value. For example:

```
This document was authored and published by
<a about="" rel="dc:creator dc:publisher" rel="http://example.org/~markb">
    Mark Birbeck
</a>.
```

is interpreted by performing the normal subject and object resolutions dictated by the `rel` attribute on both the `dc:creator` and `dc:publisher` values. The resulting triples are:

```
<> dc:creator <http://example.org/~markb> .
<> dc:publisher <http://example.org/~markb> .
```

The same exact reasoning applies to the `rev` and `property` attributes.

4.5. Establishing the object

The object of statement can be set using one of the attributes `content` or `href`. Which attribute is used depends on how the predicate is indicated. If the predicate is set using `property` then the object is a [literal], and its value is given by the `content` attribute or the element content. If the predicate is set with the `rel` attribute, then the object is a non-literal whose value depends on the presence and value of `href` attribute. If the predicate is expressed with the `rev` attribute, then the object will be obtained using *subject resolution* as defined in the next section, while this section explains how to set that predicate's *subject*.

4.5.1. Literal object resolution using the `content` attribute

The `content` attribute can be used to indicate a [plain literal] as follows:

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

or, alternatively, using the content of the element (`meta` or other) as an [XMLliteral]:

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

If the element that carries the `property` attribute also carries a `content` attribute and is non-empty, the value of the `content` attribute takes precedence and is taken to be the object of the triple. More details on determining the type of a literal object are provided in Section 5.1.

4.5.2. URI object resolution using the `href` attribute

When a triple predicate has been expressed using the `rel` attribute, the `href` attribute on the [RDFa statement]'s element is used to indicate the object as a [URI reference]. Its type, just like that of the `about` attribute, is CURIE/URI:

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

4.5.3. `rel` without `href`

When a triple predicate has been expressed using the `rel` attribute, and no `href` attribute exists on the same [RDFa element], then the CURIE/URI represented by this element is used as the object. Specifically, this CURIE/URI is affected by the `about` and `id` attributes. When neither is present, the object is a bnode (bnodes are discussed further in Section 5 [REF]). In all cases, the subject resolution for child elements is affected: where they do not override the subject, their subject is this same CURIE/URI here resolved as the object.

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

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

The above yields the following triples:

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

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

4.6. Summary

This section will contain a summary of the syntax.

5. RDF Concepts

Having established the different parts of the syntax of RDFa, we will now look at the various aspects of the RDF Abstract Syntax, and see how they can be represented in RDFa.

5.1. Literals as Objects

When a `property` predicate is used in RDFa, the object is expected to be a literal. This literal can be optionally typed by a `datatype` attribute within the same RDFa element. The absence or presence of this attribute has a significant impact on the interpretation of the literal.

5.1.1. Without `datatype`

Without a `datatype` attribute, the object literal will either be a plain literal or an XML literal, depending on whether the `content` attribute is used. For example, consider the following XHTML with RDFa which designates the author of a web page:

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head property="dc:creator" content="Mark Birbeck">
        <title>Internet Applications</title>
    </head>
...
```

In this case, with the use of the `content` attribute indicates that the object is a plain literal:

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

On the other hand, the following RDFa will yield a slightly different triple:

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Internet Applications</title>
    </head>
    <body>
    <span property="dc:creator">Mark Birbeck</span>
    </body>
...
```

which yields an XML literal object:

```
<> dc:creator "Mark Birbeck"^^rdf:XMLLiteral .
```

5.1.1.1. XML Literals

The default `rdf:XMLLiteral` type plays a significant role. XML documents cannot contain XML mark-up in their attributes, which means it is not possible to represent XML within the `content` attribute. The following would cause an XML parser to generate an error:

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

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

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

RDF does, however, provide a datatype for indicating [XML literal]s. RDFa therefore adds this datatype to any [literal] that is indicated using child text nodes on the [RDFa statement]. For example:

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

would generate the expected triple:

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

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

clarify canonicalization

as per Elias's email, we need to clarify what this canonicalization is.

5.1.1.2. Language Tags

RDF allows [plain literal]s to have a language tag, as illustrated by the following example from [RDFTESTS-RDFMS-XMLLANG-TEST006] [p.36] :

```
<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 the `content` attribute, or by a `datatype` value of `plaintext`:

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

Note that the value can be inherited as defined in [XML-LANG] [p.35] , so the following syntax will give the same triple as above:

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

5.1.2. With `datatype`

RDF allows [literal]s to be given a data type, as illustrated by the following example from [RDFTESTS-DATATYPES-TEST001] [p.36] :

```
<http://example.org/foo>
  <http://example.org/bar>
  "10"^^<http://www.w3.org/2001/XMLSchema#integer> .
```

This can be represented in RDFa as follows:

```
<span about="http://example.org/foo"
      property="ex:bar" content="10" datatype="xsd:integer">ten</span>
```

5.1.2.1. Literal from string value of `meta`

If the `datatype` is specified, but no `content` attribute exists, then the typed literal's value is determined as the concatenation of all textual child elements. For example, the following RDFa:

```
<span about="http://example.org/foo"
      property="dc:creator" datatype="xsd:string">
  <b>M</b>ark <b>B</b>irbeck
</span>.
```

will yield the following triple:

```
<http://example.org/foo> dc:creator "Mark Birbeck"^^xsd:string .
```

EliasT comments

We need to explain which datatypes are allowed and emphasize "plaintext".

5.2. Blank nodes

A [blank node] is generated explicitly when an [RDFa statement] uses a bnode CURIE as its subject. A [blank node] can be generated more implicitly when an XML element without an `about` attribute has `meta` or `link` child elements, also without `about` attributes of their own. In the latter case, the [unique anonymous ID] generated to identify the [blank node] is associated with the [context statement] of the `meta` and `link` elements. This allows a number of statements to be made about the same [blank node].

For example, to establish relationships between a [blank node] and literals or URIs, one can use the implicit [blank node] construction of our earlier example, repeated here:

```
<blockquote>
    <link rel="dc:source" href="urn:isbn:0140449132" />
    <meta property="dc:creator" content="Fyodor Dostoevsky" />
    <p>
        Rodion Romanovitch! My dear friend! If you go on in this way
        you will go mad, I am positive! Drink, pray, if only a few drops!
    </p>
</blockquote>
```

This would generate the following [triple]s:

```
_:a dc:source <urn:isbn:0140449132> .
_:a dc:creator "Fyodor Dostoevsky" .
```

One could also use the more explicit declaration:

```
<blockquote about="[_:a]">
    <p>
        Rodion Romanovitch! My dear friend! If you go on in this way
        you will go mad, I am positive! Drink, pray, if only a few drops!
    </p>
</blockquote>

<link about="[_:a]"
      rel="dc:source" href="urn:isbn:0140449132" />
<meta about="[_:a]"
      property="dc:creator" content="Fyodor Dostoevsky" />
```

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

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

we could use the following XHTML:

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

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

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

```
   </a>.
   He knows Libby.
   <link rel="foaf:knows" href="[_:b]" />
</div>

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

5.3. Reification

RDFa partially supports reification.

During subject resolution (which could be triggered by object resolution for a `rev` attribute), the processor may traverse up the DOM tree in search of an `about` attribute. If a `link` or `meta` element is encountered before an `about` attribute is found, and if this `link` or `meta` element itself does not have an `about` attribute, then the subject (or, again in the case of `rev`, object) is resolved as the [RDFa statement] represented by this `link` or `meta` element.

For example, the following XHTML:

```
<div about="">
  <link rel="cc:license"
    href="http://creativecommons.org/licenses/by-nc-nd/3.0/">
     <meta property="dc:creator" content="Ben Adida" />
  </link>
  <meta property="dc:creator" content="Mark Birbeck" />
</div>
```

will yield the following triples:

```
 <> cc:license <http://creativecommons.org/licenses/by-nc-nd/3.0/> .
 <> dc:creator "Mark Birbeck."
 _:a rdf:type rdf:Statement .
 _:a rdf:subject <> .
 _:a rdf:predicate cc:license .
 _:a rdf:object <http://creativecommons.org/licenses/by-nc-nd/3.0/> .
 _:a dc:creator "Ben Adida" .
```

which means that "Mark Birbeck" is the creator of the current document, that this document is licensed under a Creative Commons license, and that "Ben Adida" is the creator of that licensing statement, *not* of the document itself.

6. Examples

6.1. Creative Commons

One of the advantages of using the same syntax to make general statements as well as statements about a document is that in many cases a document can carry its own metadata. For example, if an XHTML document contains a navigable link to the Creative Commons license, this link can also be used to express metadata:

```
<div about="">
    This document is licensed under a
    <a rel="cc:license"
       href="http://creativecommons.org/licenses/by-sa/2.0/">
        Creative Commons License
    </a>
    which, among other things, requires that you provide
    attribution to the author,
    <a rel="dc:creator" href="http://ben.adida.net">Ben Adida</a>.
</div>
```

This chunk of XHTML will generate the same triples, no matter what other XHTML contains it:

```
<> cc:license <http://creativecommons.org/licenses/by-sa/2.0/> .
<> dc:creator <http://ben.adida.net> .
```

6.2. FOAF

FOAF requires the definition of at least two RDF entities: the FOAF person, and the FOAF homepage, which cannot be the same. Thus, the following XHTML can be used to represent a FOAF record:

```
<html xmlns:geo="http://www.w3.org/2003/01/geo/" ...>
    <head>
      <title property="dc:title">Dan's home page</title>
    </head>
    <body>
      <section id="person">
        <span about="[_:geolocation]">
          Dan is located at latitude
          <meta property="geo:lat">51.47026</meta>
          and longitude
          <meta property="geo:long">-2.59466</meta>
        </span>
        <link rel="rdf:type" href="[foaf:Person]" />
        <link rel="foaf:homepage" href="" />
        <link rel="foaf:based_near" href="[_:geolocation]" />
        <h1 property="foaf:name">Dan Brickley</h1>
      </section>
    </body>
</html>
```

which yields the correct FOAF triples:

```
<> dc:title "Dan's home page"^^rdf:XMLLiteral .
_:geolocation geo:lat "51.47026"^^rdf:XMLLiteral .
_:geolocation geo:long "-2.59466"^^rdf:XMLLiteral .
<#person> rdf:type foaf:Person .
<#person> foaf:homepage <> .
<#person> foaf:based_near _:geolocation .
<#person> foaf:name "Dan Brickley"^^rdf:XMLLiteral .
```

If one wants to make the `foaf:Person` a blank node, then the only change required is taking out the `id="person"` from the `span` element, which then yields the following triples:

```
<> dc:title "Dan's home page" .
_:geolocation geo:lat "51.47026" .
_:geolocation geo:long "-2.59466" .
_:span0 rdf:type foaf:Person .
_:span0 foaf:homepage <> .
_:span0 foaf:based_near _:geolocation .
_:span0 foaf:name "Dan Brickley" .
```

7. Processing Model

This section is informative.

This section looks at a generic set of processing rules for creating a set of triples that represent the metadata 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').

7.1. Overview

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

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

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

7.2. Evaluation Context

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

- The [base]. This will usually be the URL of the document being proessed, but it could be some other URL, set by some other mechanism, such as the HTML `base` element. The important thing is that it establishes a URL against which relative paths can be evaluated.
- The [current resource]. The initial value will be the same as the initial value of `base`, but it will usually change during the course of processing.
- A list of current in-scope [URI mappings].
- The [language]. Note that there is no default language.

7.3. Processing

Processing would normally begin after the document to be parsed has been completely loaded. However, there is no requirement for this to be the case, and it is certainly possible to use a SAX-style processing model to extract the RDFa information. Note that if some approach other than the DOM traversal approach 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, HTML processing rules must still be applied, even if document processing takes place in a non-HTML environment such as a search indexer.)

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

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

Processing then begins with the root element, and all nodes in the tree are processed according to the following rules, depth-first:

1. Any changes to the [current evaluation context] are made first:
   - the [current element] is parsed for [URI mappings] and these are added to the [list of URI mappings]. Note that a [URI mapping] will simply overwrite any current mapping in the list that has the same name;
     These mappings are provided by the `xmlns` atttribute. The value to be mapped is set by the XML namespace prefix, and the value to map is the value of the attribute--a URI. Note that the URI is not processed in any way; in particular if it is a relative path it is not resolved against the [current base]. Authors are advised to follow best practice for using namespaces, which includes not using relative paths. (See [xyz].)
   - the [current element] is parsed for any language information, and [language] is set in the [current evaluation context];
     Language information can be provided using either the general-purpose XML attribute, `xml:lang`, or the HTML attribute `lang`.
   - the [current element] is parsed for any subject information, and it is used to set the [current resource] value, in the [current evaluation context];
     The [current resource] can be set using `@about`. Note that the final value of the [current resource] is an absolute URI, which means that if `@about` contains a relative path the value must be normalised against [base] in the [current evaluation context], using the algorithm defined in RFC 3986.
2. Once the [current evaluation context] has been set, object resolution is carried out, as follows:
   - the [current object resource] is established;
     Since only one [current object resource] is set per element then some attributes will have a higher priority than others. The highest priority is given to the RDFa attribute

`resource`. If there is no `resource` attribute then the HTML `src` attribute is used, and if that is not present, the HTML `href` attribute is used. If none of these are present then a unique identifier or [bnode] is created. Note that the final value of the [current object resource] is an absolute URI, which means that if any of these attributes contain relative paths they must be normalised against [base] in the [current evaluation context], using the algorithm defined in RFC 3986.

- the [current object literal] is established;
  The [current object literal] will be set as a [plain literal] if the `content` attribute is present, *or* the body of the [current element] contains only text (i.e., there are no child elements), *or* the body of the [current element] *does* have child elements but the `datatype` attribute has an empty value. Additionally, if there is a value for [current language] then the value of the [plain literal] should include this language information, as described here:???. The actual literal is either the value of the `content` attribute (if present) *or* a string created by concatenating the inner content of each of the children in turn, of the [current element].
  The [current object literal] will be set as a [typed literal] if the `datatype` attribute is present, and does not have an empty value. The actual literal is either the value of the `content` attribute (if present) *or* a string created by concatenating the inner content of each of the children in turn, of the [current element]. The final string includes the datatype, as described here:???
  The [current object literal] will be set as an [XML literal] if the [current element] has child elements, and the `datatype` attribute is not present. The value of the [XML literal] is a string created from the inner content of the [current element], i.e., not including the element itself.

3. Once object resolution is complete the processor will have two objects, one a resource and the other a literal. These objects can now be used to create triples with the [current resource], depending on the presence of other attributes. This is achieved using the following processing steps:
   - predicates for the [current object literal] are established;
     Predicates for the [current object literal] can be set by using the `property` attribute. If present, the attribute must contain one or more [basic curies], each of which is converted to an absolute URI using CURIE processing rules, and then used to generate a triple as follows:
     subject
     : [current resource]
     predicate
     : expanded value from the basic curie
     object
     : [current object literal]

     Note that *literal* may include language and datatype information as discussed in the section on object resolution.
   - type values for the [current object resource] are established;
     One or more 'types' for the [current object resource] can be set by using the `instanceof` attribute. If present, the attribute must contain one or more [basic curies], each of which is converted to an absolute URI using CURIE processing rules, and then used to generate a triple as follows:

subject
[current object resource]
predicate
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
object
expanded value from the basic curie
If any triples are generated then the [chaining] flag is set to `true`.

- predicates for the [current object resource] are established;
Predicates for the [current object resource] can be set by using one or both of the `@rel` and `@rev` attributes.
If present, the `rel` attribute must contain one or more [basic curies], each of which is converted to an absolute URI using CURIE processing rules, and then used to generate a triple as follows:
subject
[current resource]
predicate
expanded value from the basic curie
object
[current object resource]
If present, the `rev` attribute must contain one or more [basic curies], each of which is converted to an absolute URI using CURIE processing rules, and then used to generate a triple as follows:
subject
[current object resource]
predicate
expanded value from the basic curie
object
[current resource]
If any triples are generated then the [chaining] flag is set to `true`.

4. If the [chaining] flag is set to `true` then the [current resource] is set to the value of the [current object resource], and the [chaining] flag is set to `false`.
5. The [current evaluation context] is pushed onto a stack.

NOTE: The recursive aspect of this needs to be explained a little better, in particular when we get stuff back off the stack.

A. References

A.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
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/.
XMLBASE
"*XML Base*", W3C Recommendation, J. Marsh, *ed.*, 27 June 2001.
Available at: http://www.w3.org/TR/2001/REC-xmlbase-20010627/
XML-LANG
"*Extensible Markup Language (XML) 1.0 (Third Edition)*", W3C Recommendation, T. Bray *et al.*, *eds.*, 4 February 2004.
Available at: http://www.w3.org/TR/2004/REC-xml-20040204

A.2. Related Activities

This section is informative.

DC
Dublin Core Metadata Initiative (DCMI) (See http://dublincore.org/.)
FOAF-PROJECT
The FOAF Project (See http://www.foaf-project.org.)
N-TRIPLES
RDF Test Cases, N-Triples (See http://www.w3.org/TR/rdf-testcases/#ntriples.)
N3-PRIMER
N3 Primer (See http://www.w3.org/2000/10/swap/Primer.)
RDF-CONCEPTS
Resource Description Framework (RDF): Concepts and Abstract Syntax (See http://www.w3.org/TR/rdf-concepts/.)
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 Best Deployment Working Group (See http://www.w3.org/2006/07/SWD/.)

RDFHTML
: RDF-in-HTML Task Force (See http://w3.org/2001/sw/BestPractices/HTML/.)

SWBPD-WG
: Semantic Web Best Practices and Deployment Working Group (See http://w3.org/2001/sw/BestPractices/.)

XHTML2-WG
: XHTML 2 Working Group (See http://w3.org/MarkUp/Group/.)

CURIE
: CURIEs (See http://w3.org/TR/curie.)

B. Change History

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]

C. Acknowledgments

This section is informative.

At the time of publication, the participants in the W3C XHTML 2 Working Group were: