RDFa Syntax

A collection of attributes for layering RDF on XML languages

W3C Editor’s Draft 12 August 2007

This version:
    http://www.w3.org/MarkUp/2007/ED-rdfa-syntax-20070812
Latest version:
    http://www.w3.org/TR/rdfa-syntax
Previous Editor’s Draft:
    http://www.w3.org/MarkUp/2007/ED-xhtml-rdfa-20070417
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.

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

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

Abstract

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 the user is informed of his 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.

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.35] , in cooperation with the XHTML 2 Working Group [HTML-WG] [p.35] . Initial work on RDFa began with the Semantic Web Best Practices and Deployment Working Group [SWBPD-WG] [p.35] .

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

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.35] 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 and HTML there is a `cite` attribute. The attribute allows an author to add information to a document to indicate the origin of a quote. The following example comes from [HTML] [p.35] , although it has been reformatted as XHTML [XHTML] [p.35] :

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

The problem here is that we have had to add a specific attribute to designate citation, and further, both the browser and some metadata processor need to have knowledge of this attribute, and its position within the mark-up.

The motivation of RDFa is to devise a means by which documents can be augmented with metadata, using property values from the growing range of available taxonomies, reusing existing content from the host language. In RDFa, one way that the example given above could be marked-up is as follows:

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

Or, if the publisher wishes to give the user a clickable link with the same embedded RDF:

```
<blockquote about="#q1">
    taken from <a rel="dc:source"
          href="http://www.example.com/tolkien/twotowers.html">
      Tolkien’s Two Towers</a>.
    <p>They went in single file, running like hounds on a strong scent,
    and an eager light was in their eyes. Nearly due west the broad
    swath of the marching Orcs tramped its ugly slot; the sweet grass
    of Rohan had been bruised and blackened as they passed.</p>
</blockquote>
```

We feel this proposal contributes to standardisation, and takes the pressure off language authors to anticipate all the structural requirements users of their language might have -- in this example we have used "source" from the Dublin Core [DC] [p.35] list, rather than inventing our own citation attribute which would be unknown to other languages. For example, the source could still be determined if the same quote were marked-up in SVG:

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

We feel these aspects of our proposal are crucial to the future of the Semantic Web, and the place of mark-up documents within it.

This proposal therefore outlines 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` [XML-BASE] [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. Using CURIEs

In order to allow for the compact expression of RDF statements, RDFa uses CURIEs (Compact URIs) [CURIE] [p.36] . 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/> .
```

3. Introduction to the structure of RDFa

Need to remove meta and link elements

this introduction should be reworked to feature less (maybe no) LINK and META elements. We should still mention them, but given that there are other ways to express these triples, we should be careful about the focus of this introductory paragraph. We should also explain the core RDF vs. the XHTML-specific profiles.

3.1. General Approach

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, specifically `about`, `rel`, `rev`, `property`, `href`, and `content`. In addition, it should be possible to represent a [triple] using only one XML element. In this way it owes more to 'flat' syntaxes like RDF/N3 [N3-PRIMER] [p.35] than it does to RDF/XML [RDF-SYNTAX] [p.35] , despite its use of XML.

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 .
```

Note that, in the above example, had "Mark Birbeck" not been enclosed in an existing `span` element, a new one could have simply been used with `about` and `property` as its only two attributes.

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> .
```

And it’s also 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"
   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`.

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.

Clickability of href attributes

the clickability of anything with HREF needs to be explored in non XHTML2. Clarification needed.

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" href="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>
```

However, two problems arise: the `href` causes the entire quotation to become a clickable link, which may not be the desired visual effect, and only one triple can be expressed. Thus, RDFa provides another mechanism, using the special `link` or `meta` element without a specified [subject]. In such cases, the [triple] concerns the parent element. This allows the example above to be recast as follows:

```
<html xmlns:dc="http://purl.org/dc/elements/1.1/">
    <head>
        <title>On <em>Crime and Punishment</em></title>
    </head>
    <body>
        <blockquote id="q1">
            <link rel="dc:source" href="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>
```

This syntax (omitting the `about` attribute to refer to the parent element) only applies to the elements `link` and `meta`, which means that even without an `id` attribute, the following statement is still ’about’ the `blockquote` element, and not the document as a whole:

```
<blockquote>
    <link rel="dc:source" href="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>
```

If more than one piece of metadata needs to be attached to the same element, then additional `link` or `meta` elements can be added:

```
<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>
```

Now we have attached two pieces of metadata to the `blockquote` element -- the source of the quote, and its author.

Note:

We say nothing here about how this metadata is used. In the previous example, the information may be of use to an RDFa-aware browser, and it could be made available to the user accessing the page via a mechanism such as tooltips. But it may also be the case that the document is parsed by some external processor and the output stored as a set of [triple]s. In the latter case the [triple]s generated by the previous example would have a [unique anonymous ID] as the subject of each statement, as follows:

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

If one wishes a non-anonymous node to represent the blockquote, one only needs to add an additional attribute to the `blockquote`, namely the usual XML `id` attribute:

```
<blockquote id="q1">
       <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>
```

which then yields:

```
<#q1> dc:source <urn:isbn:0140449132> .
<#q1> dc:creator "Fyodor Dostoevsky" .
```

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

In this section we look in more detail at the [triple]s that are generated when using RDFa attributes and elements. We've already said that the aim is to make it possible to generate a [triple] with one element. However, we also saw that parent elements may have an effect on the triple represented by their children elements. We therefore need to understand how the subject, predicate and object parts of a [triple] are established from our syntax.

4.1. Processing

An [RDFa element] is defined as any XML element that contains one or more RDFa attributes: `about`, `property`, `rel`, `rev`, `href`, `content`. Processing proceeds by examining each [RDFa element] in turn. The [RDFa element] under consideration at any time is the [current statement], and its parent element is the [context statement]. Note that the [context statement] does not need to be an [RDFa element]. RDFa also includes a `datatype` attribute. The presence of that attribute does not by itself designate an [RDFa element].

As each [RDFa element] is examined, the processor tries to establish the RDF triples it generates. Each of the attributes `rel`, `rev`, and `property` can express one or more triples: each accepts space-separated values, and each such value the predicate for one triple. It makes sense, then, for the processor to start with identifying the predicate of a triple, then to figure out the triple's subject and object.

Need section on associating CURIEs with RDFa elements

A small section here on associating a CURIE/URI with any [RDFa element] could be useful, as that is used at least twice.

4.2. 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, and -- although readability may suffer -- can even 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.

4.2.1. Using 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>
```

4.2.2. Using 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> .
```

4.2.3. Using 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> .
```

4.2.4. 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.

4.2.5. 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.

4.3. Establishing the object

The object of the 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*.

4.3.1. 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.

4.3.2. 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" />
```

4.3.3. `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>)
</div>
```

The above yields the following triples:

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

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

4.4. Establishing the subject

While the predicate and object resolution are relatively local to the RDFa element on which the corresponding attributes reside, subject resolution can be a little bit more complicated. Importantly, the [context statement] is highly relevant in determining the subject. Note that, when the predicate is determined by the `rev` attribute, the subject resolution described here applies to the object of the triple under consideration, while the subject is determined using object resolution, as described in the previous section.

At a high level, the subject of a statement is determined by the `about` attribute, either on the element or on the closest parent of that element. Two exceptions to that rule exist. First, if a closer parent element includes a `rel` or `rev` attribute with no `href`, then the subject is the CURIE/URI that corresponds to that parent element (as described previously in object resolution.) Second, if the [RDFa element] under consideration is a `META` or `LINK` without an `about`, then the subject is the immediate parent element's CURIE/URI equivalent.

4.4.1. The `about` attribute

The subject of a triple is usually indicated using the `about` attribute, as follows:

```
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>.
```

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

- `#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`.

4.4.2. Inheriting the `about` attribute

Note that this section does not apply to [RDFa statement]s whose predicates are defined in `meta` or `link` elements. Section 4.3.3 deals with `meta` and `link` specifically.

If the [RDFa element] that includes the predicate attribute does not have an `about` attribute, then the subject of the [triple] is determined by the [context statement]'s `about` attribute. The resolution of the `about` attribute is recursive: if the [context statement] has no such attribute, then the processor must continue up the DOM tree to find the closest ancestor with an `about` attribute.

(The recursive search for the closest ancestor may be interrupted if an element is encountered with a `rel` or `rev` without a corresponding `href`. This is described in the next subsection.)

For example, the following XHTML:

```
<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 .
```

If no such parent is ever found all the way up the DOM tree, then the default value for the `about` attribute is the empty string, which effectively indicates the current document.

4.4.3. `rel` and `rev` attributes in ancestor elements

During the ancestor element traversal, one may encounter an element with a `rel` or `rev` attribute without a corresponding `href` attribute. As described in object resolution, this situation defines a new subject for all its children elements, in particular the currently considered [RDFa element]. Specifically, the CURIE/URI associated with this ancestor element becomes the subject.

4.4.4. `meta` and `link` elements

If an [RDFa statement] is generated by a predicate attribute of a `meta` or `link` element, and this element does not contain an explicit `about` attribute, subject resolution is slightly different. Only the immediate [context statement] is considered, whether or not it has its own `about` attribute.

The subject is then the CURIE/URI that corresponds to the immediate [context statement]. Specifically, the [context statement] may have an `about` attribute, in which case the [RDFa statement]'s subject is resolved as the value of this attribute (exactly as if the current [RDFa statement] weren't a `link` or `meta`.) However, if the [context statement] does not have an `about` attribute, the subject of the current [RDFa statement] is the parent element itself. If this parent element is not identified with `xml:id`, it is treated as a [bnode].

For example, the following XHTML:

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

will generate one triple corresponding to the `property` attribute of the `meta` element, whose subject is resolved as the value of the `about` attribute of the immediate parent element (`div`).

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

However, the following, slightly different XHTML which includes an extra element between the `about` and the `META`:

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

will yield the possibly unexpected:

```
_:span0 dc:creator "Mark Birbeck" .
```

The `meta` and `link` elements should be thought of as mechanisms for applying metadata to any existing element in the XML document. Thus, to recover the previous triple, one could simply switch the `meta` to a `span`, which will trigger the recursive search up the DOM tree for the closest `about` attribute (this is exactly the same example as that of the previous section):

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

4.4.4.1. `meta` or `link` inside the head in XHTML2

A `meta` or `link` without its own `about` attribute and positioned directly within the `head` of a document will automatically apply to the document itself. Effectively, the `head` of an XHTML2 document contains an implicit `about=""`.

Note also that this isn’t exactly the same thing as having the subject eventually default -- all the way up the DOM tree -- to the current document. A `meta` or `link` element applies to the parent element, never inheriting further up the tree than the `head` element, in this case. Thus, this additional detail about the `head` element is not redundant with any other instruction in this document.

4.5. 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;lt;sup&amp;gt;2&amp;lt;/sup&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.35] :

```
<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.35] :

```
<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` children 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. Rules

(not updated yet.)

A. References

DC
Dublin Core Metadata Initiative (DCMI) (See http://dublincore.org/.)
FOAF-PROJECT
The FOAF Project (See http://www.foaf-project.org.)
HTML
HTML 4.01 Specification (See http://www.w3.org/TR/html401/.)
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/.)
XHTML
XHTML 1.0 (See http://www.w3.org/TR/xhtml1.)
XHTML-2.0-LINKTYPES
XHTML 2.0 Link Types (See http://www.w3.org/MarkUp/Group/2003/WD-xhtml2-20031029/abstraction.html#dt_LinkTypes.)
XML-BASE
XML Base (See http://www.w3.org/TR/xmlbase/.)
XML-LANG
Extensible Markup Language (XML) 1.0 (Third Edition), Language Identification (See http://www.w3.org/TR/REC-xml/#sec-lang-tag.)
XPOINTER-FRAMEWORK
XPointer Framework (See http://www.w3.org/TR/xptr-framework/.)
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/.)
HTML-WG
HTML Working Group (See http://w3.org/MarkUp/Group/.)

CURIE
CURIEs (See http://w3.org/2001/sw/BestPractices/HTML/2005-10-27-CURIE.)

B. Change History

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: