RDFa in XHTML: Syntax

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

W3C Editor's Draft 12 September 2007

This version:
http://www.w3.org/MarkUp/2007/ED-rdfa-syntax-20070912
Latest version:
http://www.w3.org/TR/rdfa-syntax
Previous Editor's Draft:
http://www.w3.org/MarkUp/2007/ED-rdfa-syntax-20070906
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.

Abstract

The modern Web is made up of an enormous number of documents that have been created using HTML. These documents contain significant amounts of structured data, which is largely unavailable to tools and applications. When publishers can express this data more completely, and when tools can read it, a new world of user functionality becomes available, letting users transfer structured data between applications and web sites, and allowing browsing applications to improve the user experience. An event on a web page can be directly imported into a user's desktop calendar; a license on a

document can be detected so that users can be informed of their rights automatically; a photo's creator, camera setting information, resolution, location and topic can be published as easily as the original photo itself, enabling structured search and sharing.

RDFa is a 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 is no need for different rules for different formats; this allows authors and publishers of data to define their own formats without having to update software, register formats via a central authority, or worry that two formats may intere with each other.

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

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

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

How to Read this Document

If you are already familiar with RDFa, and you want to examine the processing rules--perhaps to create a parser--then you'll find section x of most interest. Each of the processing steps in this section is outlined in more detail in the sections after it.

If you are not familiar with RDFa, but you *are* familiar with RDF, then you might find reading the overview useful, since it gives a range of examples of XHTML mark-up that use RDFa. Seeing some examples first should make reading the processing rules easier.

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

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

This specification deals specifically with the use of RDFa in HTML-based languages, such as HTML and XHTML, and defines an RDF mapping for a number of HTML attributes.

2. Orientation

NOTE: The idea of this section is to give the reader a quick intro to the features RDFa supports. It's not meant to be exhaustive, but if I have missed anything obvious then do let me know. The idea is to use a simple example, and to make them 'familiar', which is why I chose to use the mark-up from the Primer.

The following examples are intended to help readers who are not familiar with RDFa to quickly get a sense of how it works.

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

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

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

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

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

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

```
    Creative Commons License
  </a>.
```

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

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

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

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

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

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:cal="http://www.w3.org/2002/12/cal/ical#"
```

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

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

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

3. RDF Terminology

The previous section gave examples of typical mark-up in order to illustrate what RDFa in XHTML looks like. But what RDFa in XHTML *represents* is RDF. In order to author RDFa in XHTML you do not need to understood RDF at all, although it would certainly help. However, if you are building a system that consumes the RDF output of an RDFa in XHTML document you will obviously need to understand RDF--but you can also skip this section, since here we provide a very basic introduction to RDF terminology so that we can make use of it in the following sections.

3.1. Statements

The metadata information that RDFa provides access to is generally understood to be a collection of *statements*. A statement is a basic unit of information that has been

constructed in a very specific way to make it easier to process. In turn, by breaking large sets of information down into a collection of statements, even very complex metadata can be made available for processing.

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

```
Albert was born on March 14, 1879, in Germany. There is a picture of him
```

This would be quite difficult for a machine to process, and it is certainly not in a format that could be passed from one system to another. However, if we convert the information to a set of statements it begins to be more manageable. The same information could therefore be represented as follows:

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

3.2. Triples

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

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

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

3.3. URI references

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

URIs are most commonly used to identify web pages, but RDF makes use of them as a way to provide unique identifiers for concepts. For example, we could identify the subject of all of our statements by using the DBPedia URI for Albert Einstein:

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

URI references are also used to uniquely identify the objects in metadata statements (note that the picture of Einstein is already a URI):

```
<http://dbpedia.org/resource/Albert_Einstein> has the name Albert Einste
<http://dbpedia.org/resource/Albert_Einstein> was born on March 14, 1879
<http://dbpedia.org/resource/Albert_Einstein> was born in <http://dbpedi
<http://dbpedia.org/resource/Albert_Einstein> has a picture at <http://e
```

And of course URI references are also used to ensure that predicates are unambiguous:

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

3.4. Plain literals

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

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

3.5. Typed literals

Some literals, such as dates and numbers, have very specific meanings, so RDF provides a mechanism for indicating the type of a literal. A *typed literal* is indicated by attaching a URI to the end of a plain literal which indicates the literal's datatype. This URI is usually based on datatypes defined in the XML Schema Datatypes specification [XML SCHEMA DATATYPES REFERENCE / http://www.w3.org/TR/xmlschema-2/]. The following syntax would be used to unambiguously express Einstein's date of birth as a literal of type `xsd:date`:

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

3.6. N-Triples

RDF does not have one set way to express triples, since the key ideas of RDF are the triple and the use of URIs. However, a number of mechanisms are available, such as RDF/XML, N-Triples [N-TRIPLES], and of course RDFa. Most discussions of RDF make use of the *N-Triple* syntax to explain their ideas, since it's quite compact. The examples we have just seen are already using this syntax, and we'll continue to use it throughout this document, with a slight variation that long URIs can be abbreviated by using a URI mapping. This is indicated by removing the angle brackets from the URI, as follows:

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

Where 'p:' maps to ... and foaf: maps to its extended URI.

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

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

```
<>
```

This indicates the 'current document', i.e., the document being processed.

3.7. Graphs

A collection of triples is called a *graph*.

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

3.7.1. Compact URIs

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

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

3.8. A description of RDFa in RDF terms

The following is a description of RDFa that uses RDF terminology, which may be useful to readers with an RDF background:

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

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

4. Processing Model

This section is normative.

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

4.1. 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, 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.

4.2. Evaluation Context

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

- The [base]. This will usually be the URL of the document being processed, but it could be some other URL, set by some other mechanism, such as the 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.

4.3. 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 IRI, which means that if `@about` contains a relative path the value must be normalised against [base] in the [current evaluation context], using the algorithm defined in RFC 3986. The value can also be provided by a CURIE, and is

processed as defined in section curie. Note that since this attribute can take both URIs and CURIEs, the latter will have been expressed using the [safe CURIE] syntax.

- the [recurse] flag is set to true;

  Processing will generally continue recursively through the entire tree of nodes available. However, if an author indicates that some branch of the tree should be treated as an XML literal, no further processing should take place on that branch. This flag is used to inhibit this processing.

- the [chaining] flag is set to false;

  If a collection of statements is contained by a [URI reference] then this may become the subject of further statements.

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. Note also that since `@resource` can take both URIs and CURIEs, the latter will have been expressed using the [safe CURIE] syntax.

   - the [current object literal] is established;

[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. 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`.

6. If the [recurse] flag is true, the [current evaluation context] is pushed onto a stack, and all nodes that are children of the [current element] are processed using the rules described here. Once all of the children have been processed then the [current evaluation context] is popped back off the stack.

5. RDFa in detail

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

@instanceof situation

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

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

In the following examples, for brevity assume that the following namespace prefixes

have been defined:

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

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

The attributes for setting a predicate are `rel`, `rev` and `property`, whilst the attributes for setting an object are `resource`, `href`, content, src

5.1. Changing the evaluation context

5.1.1. Setting the [current resource]

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

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

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

although it is possible for the data to appear elsewhere:

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

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

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

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

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

Other kinds of resources can be used to set the [current resource]:

```
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>
```

Using xml:base

```
<p>
  <strong>NOTE:</strong> The formatting will probably be odd here, but what I'm
  'box' like you would have in a magazine. The intention is not to break the ma:
  to this point.
</p>
  <p>XHTML does not include the <code>xml:base</code> <a
    href="#ref_XMLBASE">[XMLBASE]</a> attribute by default. However, if
    it is part of the host language an RDFa parser must process it, and
    use its value to set [base].</p>

  <p>An example follows to show how <code>xml:base</code> affects the
    subject:</p>

  <example>
    <span xml:base="http://internet-apps.blogspot.com/">
      <span about="" rel="dc:creator" href="http://www.blogger.com/profile/1109
      <span about="" property="dc:title" content="Internet Applications" />
    </span>
  </example>

  <p>The triples generated would be as follows:</p>

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

5.2. Object resolution

There are two types of object, [URI resources] and [literals].

A [literal] object can be set by using the `property` attribute to express a [predicate], and

then using either the attribute `content`, or the inline text of the element that `property` is on.

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

5.2.1. Literal object resolution

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

5.2.1.1. Plain Literals

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" />
```

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

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

Both of these examples give the following triple:

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

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

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

5.2.1.1.1. Language Tags

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

```
<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 the inline text of

the element:

```
<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], 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>
```

5.2.1.2. Typed literals

[Literal]s can be given a data type using the `datatype` attribute:

This can be represented in RDFa as follows:

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

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

EliasT comments

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

5.2.1.3. XML Literals

XML documents cannot contain XML mark-up in their attributes, which means it is not possible to represent XML within 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;: T
```

RDF does, however, provide a datatype for indicating [XML literal]s. RDFa therefore adds this datatype to any [literal] that has child elements. For example:

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

would generate the expected triple:

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

There will be situations where the extra mark-up is not actually part of the meaning of the literal, and can be ignored. In this situation an empty `datatype` value can be used to override the XML literal behaviour:

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

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

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

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

reference

clarify canonicalization

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

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

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

Note that this does effectively mean that the presence of `@property` inhibits any further processing, so authors should watch out for stray attributes, if they find they are getting fewer triples than they had expected.

5.2.2. URI object resolution

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

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

5.2.2.1. Using `@resource` to set the object

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

```
<html>
  <head>
    <title>On Crime and Punishment</title>
  </head>
  <body>
    <blockquote 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>
```

5.2.2.2. Using the `href` attribute

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

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 a URI:

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

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

```
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 two triples:

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

5.2.2.3. Using `@src` to set the object

wrong title for this section

```
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 sets the subject for any nested triples which means that the same triples can be expressed using this, more compact, syntax:

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

5.2.2.4. Using a bnode to set the object

When a triple predicate has been expressed using the `rel` attribute, and no `href`, `src`, or `resource` attribute exists on the same [RDFa element], then the CURIE represented

by this element is used as the object. This CURIE is affected by the `about` attribute, but if none is present the object is a bnode (bnodes are discussed further in Section bnode [REF]). In all cases, the subject resolution for child elements is affected: where they do not override the subject, their subject is this same CURIE 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> .
```

5.2.2.4.1. Referencing Bnodes

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
```

NOTE: The following language-independent prose will be removed shortly, once we have finalised this.

CURIEs can be used in any language, including non-XML languages. Any language that wishes to make use of CURIEs must provide a context which consists of:

- a set of mappings from prefixes to URIs;
- a mapping to use with the default prefix (for example, `:p`);
- a mapping to use when there is no prefix (for example, `p`);
- a mapping to use with the '_' prefix, which is used to generate unique identifiers (for example, `_:p`).

When CURIEs are used in RDFa in XHTML, the context is set as follows:

- the prefix mappings are provided by the current in-scope namespace declarations of the [current element] during parsing;
- the mapping to use with the default prefix is the current default namespace;
- the mapping to use when there is no prefix is `http://www.w3.org/1999/xhtml#`.
- the mapping to use with the '_' prefix is not explicitly stated, but should be chosen by the processor to ensure that there is no possibility of collision with other documents.

clarify the 'no prefix' situation

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

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

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/.)