RDFa Primer 1.0

Embedding Structured Data in Web Pages

Editors' Draft

This version:
http://www.w3.org/2006/07/SWD/RDFa/primer/20070910/ $Id: Overview.xml,v 1.2 2007/09/11 00:47:53 adida Exp $
Latest version:
http://www.w3.org/2006/07/SWD/RDFa/primer
Previous version:

- http://www.w3.org/2006/07/SWD/RDFa/primer/20070302/
- http://www.w3.org/2006/07/SWD/RDFa/primer/20070227/
- http://www.w3.org/2006/07/SWD/RDFa/primer/20070104/
- http://www.w3.org/2001/sw/BestPractices/HTML/2006-04-24-rdfa-primer

Editors:
Ben Adida, Creative Commons <ben@adida.net>
Mark Birbeck, x-port.net Ltd. <mark.birbeck@x-port.net>

Abstract

Current web pages, written in XHTML, contain inherent structured data: calendar events, contact information, photo captions, song titles, copyright licensing information, etc. When authors and publishers can express this data precisely, and when tools can read it robustly, 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 desktop calendar. A license on a document can be detected to inform the user of his rights automatically. A photo's creator, camera setting information, resolution, and topic can be published as easily as the original photo itself.

RDFa lets an XHTML author express this structured data using extra XHTML attributes. Where the data is already present on the page, e.g. ~~the~~ a photo~~'s~~ caption, the author need not repeat it. A web publisher can easily reuse concepts, e.g. an event's date, defined by other publishers, or create new ones altogether. RDFa gets most of its expressive power from RDF, though the reader need not understand RDF ~~to read on.~~ before reading this document.

This document introduces XHTML authors to RDFa with simple examples. For more detailed syntax specification, please consult the RDFa Syntax Document.

Status of this Document

This is an internal draft produced by the Semantic Web Deployment Working Group [SWD-WG], in cooperation with the ~~HTML~~ XHTML2 Working Group [HTML-WG]. Initial work on RDFa began with the Semantic Web Best Practices and Deployment Working Group [SWBPD-WG]. and the HTML WG

This document is for internal review only and is subject to change without notice. This document has *no formal standing within the W3C.*

Changes

Since Working Draft #3 of this document:

- Section 3 on Shutr was split up and moved to "advanced concepts", rather than be a semi-syntax'ish thing.
- the `class` attribute is no longer used to declare `rdf:type`, as this was found to be too confusing. We now use the new `instanceof` attribute.
- the updated syntax for chaining (previously referred to as striping) is introduced.

Table of Contents

1 Purpose and Preliminaries
2 Simple Data: Publishing Events and Contacts
2.1 The Basic XHTML
2.2 Publishing An Event
2.3 Publishing Contact Information
2.4 The Complete XHTML with RDFa
2.5 Working Within a Fragment of the XHTML
3 Advanced Concepts: Custom Vocabularies, Document Fragments, Complex Data, ...
3.1 Creating a Custom Vocabulary and Using Compact URIs
3.2 Qualifying Other Documents and Document Chunks
3.3 Data Types
3.4 Layers of Structure — Subresources
3.5 Using src on IMG, and the rev attribute
3.6 Overriding href
4 RDF Correspondence
4.1 Events and Contact Information
* 4.2 ~~Simple Shutr Data~~ Advanced Data (More informative section title)
4.3 Layered Data and Subresources
4.4 Using src on IMG, and the rev attribute
4.5 Overriding href
5 Case Studies
6 Acknowledgments
7 Bibliography

1 Purpose and Preliminaries

Current web pages, written in XHTML, contain inherent structured data: calendar events, contact information, photo captions, song titles, copyright licensing information, etc. When publishers can express this data precisely, and when tools can read it robustly, 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 desktop calendar. A license on a document can be detected to inform the user of his rights automatically. A photo's creator, camera setting information, resolution, and topic can be published as easily as the original photo itself.

RDFa lets an XHTML author express this structured data using extra XHTML attributes. Where the data is already present on the page, e.g. the photo's caption, the author need not repeat it. A web publisher can easily reuse concepts, e.g. an event's date, defined by other publishers, or create new ones altogether. RDFa gets most of its expressive power from RDF, though the reader need not understand RDF to read on.

We note that RDFa makes use of XML namespaces. In this document, we assume, for simplicity's sake, that the following namespaces are defined: `dc` for Dublin Core, `foaf` for FOAF, `cc` for Creative Commons, and `xsd` for XML Schema Definitions:

namespace prefixes are declared

- `dc`: http://purl.org/dc/elements/1.1/
- `foaf`: http://xmlns.com/foaf/0.1/
- `cc`: http://web.resource.org/cc/
- `xsd`: http://www.w3.org/2001/XMLSchema

2 Simple Data: Publishing Events and Contacts

Jo keeps a private blog for her friends and family.

2.1 The Basic XHTML before identifying data

Jo is organizing one last summer Barbecue, which she hopes all of her friends and family will attend. She blogs an announcement of her talk at her private blog, `http://jo-blog.example.org/`. Her blog also includes her contact information:

```
<html>
    <head><title>Jo's Friends and Family Blog</title></head>
    <body>
...
    <p>
        I'm holding one last summer Barbecue, on September 16th at 4pm.
    </p>
...
    <p class="contactinfo">
        Jo Smith. Web hacker
        at
        <a href="http://example.org">
            Example.org
        </a>.
        You can contact me
        <a href="mailto:jo@example.org">
            via email
        </a>.
    </p>
...
    </body>
</html>
```

This short piece of mark-up contains important structured data.

The markup describes an *event*: a Barbecue that Jo is hosting. This Barbecue *starts* at 4pm on September 16th. A *summary* of the event is "one last summer Barbecue." We also have contact information for Jo: she works for the *organization* Example.org, with *job title* of "Web Hacker." She can be contacted at the *email* address "jo@example.org."

At the moment, it is very difficult for software — like web browsers and search engines — to make use of this data's implicit structure. We need a standard mechanism to explicitly express it, so that it can be extracted consistently. This is precisely where RDFa comes in.

2.2 Publishing An Event

Jo would like to label this blog entry so that her friends and family can add her Barbecue directly to their calendar. RDFa allows her to express this structure using a small handful of extra attributes. Since this is a calendar event, Jo will specifically use the iCal vocabulary [ICAL-RDF] to denote the data's structure.

The first step is to reference the iCal vocabulary within the XHTML page, so that Jo's friends' web browsers can look up the calendar concepts and make use of them:

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

then, Jo declares a new event:

```
<p instanceof="cal:Vevent"> ... </p>
```

Note how the `instanceof` attribute is used here to define the kind of ~~the~~ data being expressed. The use of this attribute on the `p` element ensures that, by default, data expressed inside this element refers to the same event. Thus, inside this event declaration, Jo can set up the event fields, reusing the existing XHTML. For example, the event summary can be declared as:

```
I'm holding <span property="cal:summary">
    one last summer Barbecue
</span>
```

suggest line collapsing to clarify whitespace handling

The `property` attribute on the `span` element declares the data field `cal:summary`. The existing content, "one last summer Barbecue", is the value of this field. Sometimes, this isn't the desired effect. Specifically, the start time of the event should be displayed pleasantly — "September 16th" —, but should likely be represented in a machine-parsable way, the standard iCal format: `20070916T1600-0500` (which is clearly not pleasant for human display). In this case, the markup needs only a slight modification:

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

The actual content of the `span` element, "September 16th at 4pm", is ignored by the RDFa portion of the browser: it has been replaced by the explicit `content` attribute. The full markup is then:

```
<html xmlns:cal="http://www.w3.org/2002/12/cal/ical#">
    <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">
            September 16th at 4pm.
        </span>
    </p>
...
    </body>
</html>
```

Note that Jo could have used any other XHTML element, not just `span`, to mark up her event. In other words, when the event information is already laid out in the XHTML using elements such as `h1`, `em`, `div`, etc..., Jo can simply add the `instanceof` data type declaration, the `property` attribute, and optionally the `content` attribute, to mark up the event.

(For the RDF-inclined reader, the RDF triples that correspond to the above markup are available in Section **4.1 Events and Contact Information**.)

2.3 Publishing Contact Information

Now that Jo has published her event in a human-and-machine-readable way, she realizes there is much data on her blog that she can mark up in the same way. Her contact information, in particular, is an easy target:

```
...
    <p class="contactinfo">
        Jo Smith. Web hacker
        at
        <a href="http://example.org">
            Example.org
        </a>.
        You can contact me
        <a href="mailto:jo@example.org">
            via email
        </a>.
    </p>
...
```

Jo discovers the vCard RDF vocabulary [VCARD-RDF], which she adds to her existing page. Since Jo thinks of vCards as a way to publish her contact information, she uses the prefix `contact` to designate this vocabulary. Note that, although Jo already imported the iCal vocabulary, adding the vCard vocabulary is just as easy and does not interfere:

```
<html xmlns:cal="http://www.w3.org/2002/12/cal/ical#"
      xmlns:contact="http://www.w3.org/2001/vcard-rdf/3.0#">
...
```

Jo then sets up her vCard using RDFa, by deciding that the `p` with `class` set to `contactinfo` will be the container for her vcard. She notes that the vCard schema does not require declaring a vCard type. Instead, it is recommended that a vCard refer to a web page that identifies the individual. Jo thus uses RDFa's special attribute `about` for just for this purpose, indicating that all contained XHTML pertains to Jo's designated URI. Note how the `about` attribute is inherited from parent elements in the XHTML: the `about` attribute on the nearest containing element applies.

```
...
    <p class="contactinfo" about="http://example.org/staff/jo">

        <!-- everything here pertains to http://example.org/staff/jo -->

    </p>
...
```

"Simple enough!" Jo realizes. She adds her first vCard fields: name, title, organization and email.

```
...
    <p class="contactinfo"  about="http://example.org/staff/jo">

        <span property="contact:fn">
            Jo Smith
        </span>.

        <span property="contact:title">
            Web hacker
        </span>

        at

        <a rel="contact:org" href="http://example.org">
            Example.org
        </a>.

        You can contact me

        <a rel="contact:email" href="mailto:jo@example.org">
            via email
        </a>.

    </p>
...
```

} suggest collapsing lines to avoid whitespace mismatch w/ triples in §4.1

Notice how Jo was able to use the `rel` attribute directly within the anchor tag for designating her organization and email address. In this case, the `rel` indicates the type of *relationship* between the current URI, designated by `about`, and the target URI, designated by `href`. Specifically, `contact:org` indicates a relationship of type "vCard organization", while `contact:email` indicates a relationship of type "vCard email".

For simplicity's sake, we have slightly abused the vCard vocabulary above: vCard technically requires that the *type* of the email address be specified, e.g. work or home email. In Section **4.3 Layered Data and Subresources**, we show how `rel` can be used without a corresponding `href`, in order to create subresources and provide the correct markup for expressing a true vCard.

2.4 The Complete XHTML with RDFa

Jo's complete XHTML with RDFa is thus:

prob 3

```
<html xmlns:cal="http://www.w3.org/2002/12/cal/ical#"
      xmlns:contact="http://www.w3.org/2001/vcard-rdf/3.0#">
    <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">
            September 16th at 4pm.
        </span>
    </p>
...
    <p class="contactinfo" about="http://example.org/staff/jo">
        <span property="contact:fn">
            Jo Smith
        </span>.
        <span property="contact:title">
            Web hacker
        </span>
        at
        <a rel="contact:org" href="http://example.org">
            Example.org
        </a>.
        You can contact me
        <a rel="contact:email" href="mailto:jo@example.org">
            via email
        </a>.
    </p>
...
    </body>
</html>
```

Summary of her event

If Jo changes her email address link, her organization, or the title of her talk, RDFa-enabled browsers will automatically pick up these changes in the marked up, structured data. The only places where this doesn't happen is when the `content` attribute must override the rendered content, which is inevitable when the human-rendered data and the machine-readable data must differ.

(Once again, the RDF-inclined reader will want to consult the resulting RDF triples **4.1 Events and Contact Information**.)

2.5 Working Within a Fragment of the XHTML

What if Jo does not have complete control over the XHTML of her blog? For example, she may be using a templating system which makes it particularly difficult to add the vocabularies in the `html` element at the top of her page without adding it to every page on her site. Or, she may be using a web provider that doesn't allow her to change the header of the page to begin with.

Fortunately, RDFa uses standard XML namespaces, which means that the vocabularies can be imported "locally" to an XHTML element. Jo's blog page could express the exact same structured data with the following markup:

```
<html>
    <head><title>Jo's Friends and Family Blog</title></head>
    <body>
...
    <p instanceof="cal:Vevent" xmlns:cal="http://www.w3.org/2002/12/cal/ical#">
        I'm holding
        <span property="cal:summary">
            one last summer Barbecue,
        </span>
        on
        <span property="cal:dtstart" content="20070916T1600-0500">
            September 16th at 4pm.
        </span>
    </p>
...
    <p class="contactinfo" about="http://example.org/staff/jo"
        xmlns:contact="http://www.w3.org/2001/vcard-rdf/3.0#">
        <span property="contact:fn">
            Jo Smith
        </span>.
        <span property="contact:title">
            Web hacker
        </span>
        at
        <a rel="contact:org" href="http://example.org">
            Example.org
        </a>.
        You can contact me
        <a rel="contact:email" href="mailto:jo@example.org">
            via email
        </a>.
    </p>
...
    </body>
</html>
```

In this case, each `p` only needs one vocabulary: the first uses iCal, the second uses vCard. However, just as before, more than one vocabulary can be imported into any element. This makes copying and pasting XHTML with RDFa much easier. In particular, it allows web widgets to carry their own RDFa in a self-contained manner.

3 Advanced Concepts: Custom Vocabularies, Document Fragments, Complex Data, ...

RDFa can do much more than the simple examples described above. In this section, we explore some of its advanced capabilities. We consider:

- the creation of a custom vocabulary,
- the use of precise datatypes,
- the description of resources beyond the current web page, and
- the definition and annotation of "subresources".

3.1 Creating a Custom Vocabulary and Using Compact URIs

All field names and data types in RDFa are URIs, e.g. `http://purl.org/dc/elements/1.1/title` is the "Dublin Core title" field. In RDFa, we often use compact versions of those URIs, by

- defining a prefix using XML namespaces, and
- using the prefixed notation to designate the URI.

This helps keep the markup short and clean:

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

   <span property="dc:title">Yowl</span>,

   created by

   <span property="dc:creator">Mark Birbeck</span>.

</div>
```

Because concepts are simply URIs, it is trivial to create one's own vocabulary: simply mint new URIs in a domain you control, and use them in RDFa markup. This is, in fact, the power of RDF, RDFa's underlying technology.

Consider a (fictional) photo management web site called *Shutr*, whose web site is `http://www.shutr.net`. Users of Shutr can upload their photos at will, annotate them, organize them into albums, and share them with the world. They can choose to keep these photos private, or make them available for public consumption under licensing terms of their choosing.

Shutr chooses to mark up its photos with RDFa, so that browsers may be able to extract information automatically. Some concepts, such as `dc:title`, `dc:date`, etc. can be clearly reused from the Dublin Core vocabulary, but other concepts, such as lens settings, camera model, and other photographer parameters, may need to be defined from scratch. For this purpose, Shutr defines a vocabulary namespace URI:

`http://shutr.net/vocab/1.0/`

Shutr can then publish terms such as `http://shutr.net/vocab/1.0/takenWithCamera`, `http://shutr.net/vocab/1.0/aperture`, etc.

3.2 Qualifying Other Documents and Document Chunks

Shutr may choose to present many photos in a given XHTML page. In particular, at the URI `http://www.shutr.net/user/markb/album/12345`, all of the album's photos will appear inline. Structured data about each photo can be included simply by specifying an `about` attribute, which indicates the resource that fields refer to within that XHTML element.

```
<ul>
  <li> <img src="/user/markb/photo/23456_thumbnail" />,

    <span about="/user/markb/photo/23456" property="dc:title">
      Sunset in Nice
    </span>

  </li>

  <li> <img src="/user/markb/photo/34567_thumbnail" />,

    <span about="/user/markb/photo/34567" property="dc:title">
      W3C Meeting in Mandelieu
    </span>

  </li>
</ul>
```

} collapse lines

This same approach applies when the field value is a URI. For example, each photo in the album has a creator and may have its own copyright license. We can use the convenient inheritance of the `about` attribute to refer to the photo once, and add as many fields as we need:

```
<ul>
  <li about="/user/markb/photo/23456">

    <img src="/user/markb/photo/23456_thumbnail" />,

    <span property="dc:title">
      Sunset in Nice
    </span>

    taken by photographer

    <a property="dc:creator"
       href="/user/markb">
      Mark Birbeck
    </a>,

    licensed under a

    <a rel="cc:license"
       href="http://creativecommons.org/licenses/by-nc/3.0/">
      Creative Commons Non-Commercial License
    </a>.

  </li>

  <li about="/user/markb/photo/34567">

    <img src="/user/markb/photo/34567_thumbnail" />

    <span property="dc:title">
      W3C Meeting in Mandelieu
    </span>

    taken by photographer

    <a property="dc:creator"
       href="/user/stevenp">
         Steven Pemberton
    </a>,

    licensed under a

    <a rel="cc:license"
       href="http://creativecommons.org/licenses/by/3.0/">
         Creative Commons Commercial License
    </a>.

  </li>
</ul>
```

While it makes sense for Shutr to have a whole web page dedicated to each photo album, it might not make as much sense to have a single page for each camera owned by a user. A single page that describes *all* cameras belonging to a single user is more appropriate. For this purpose, RDFa provides ways to mark up document fragments using natural XHTML constructs.

Consider the page `http://www.shutr.net/user/markb/cameras`, which, as its URI implies, lists

Mark Birbeck's cameras. Its XHTML contains:

```
<ul>
  <li id="nikon_d200"> Nikon D200, 3 pictures/second.
  </li>

  <li id="canon_sd550"> Canon Powershot SD550, 5 pictures/second.
  </li>
</ul>
```

and the photo page will then include information about which camera was used to take each photo:

```
<ul>
  <li>
    <img src="/user/markb/photo/23456_thumbnail" />
    ...
    using the <a href="/user/markb/cameras#nikon_d200">Nikon D200</a>,
    ...
  </li>
...
</ul>
```

The RDFa syntax for formally specifying the relationship is exactly the same as before, as expected:

```
<ul>
  <li about="/user/markb/photo/23456">
    <img src="/user/markb/photo/23456_thumbnail" />
    ...
    using the <a rel="shutr:takenWithCamera"
         href="/user/markb/cameras#nikon_d200">Nikon D200</a>,
    ...
  </li>
...
</ul>
```

Then, the XHTML snippet at `http://www.shutr.net/user/markb/cameras` is:

```
<ul>
  <li id="nikon_d200" about="#nikon_d200">

    <span property="dc:title">
      Nikon D200
    </span>

    <span property="shutr:shutterSpeed">
      3 pictures/second
    </span>

  </li>

  <li id="canon_sd550" about="#canon_sd550">

    <span property="dc:title">
      Canon Powershot SD550
    </span>

    <span property="shutr:shutterSpeed">
```

} collapse lines

frameRate

```
        5 pictures/second
      </span>

    </li>
  </ul>
```

Notice, again, how text can serve both for the human and machine readable versions: there is no need to keep a separate file up-to-date. See § 4.2 for details of the RDF produced.

3.3 Data Types

When dealing with fields of structured data, one may well want (or need) to specify a data type so that computer programs that read the data can make sense of it. Consider the expression of a date. We have already seen how the human-rendered and machine-readable data may not be the same, and how we can use `content` to provide a machine-readable value. Adding a datatype is only one more attribute: `datatype`. For example, when expressing the date on which a photo was taken:

```
      <ul>
        <li about="/user/markb/photo/23456">

          ...

          take on
          <span property="dc:date" content="2007-05-12" datatype="xsd:date">
            May 12th, 2007
          </span>

          ...

        </li>

      </ul>
```

Note how we use XML data types. XSD ref

3.4 Layers of Structure — Subresources

Sometimes, one may need to mark up a resource with a number of fields, all without giving it a URL, or even a fragment identifier. Consider the case where Shutr decides to let users annotate photos in order to indicate which individuals are depicted in the photo. The barebones XHTML is:

```
  This photo depicts Mark Birbeck (mark@example.org) and Steven Pemberton (steven@example
```

The simplest way to mark this up without attempting to resolve unique identities for photo subjects is to define *subresources*, effectively new resources that are not given a name. (In RDF, we call these blank nodes.) The following markup will do just that, thanks to the FOAF (Friend-Of-A-Friend) vocabulary which includes the handy field `foaf:depicts` for just this purpose:

```
  <div about="/user/markb/photo/23456">
    This photo depicts
    <span rel="foaf:depicts">
      <span property="foaf:firstname">Mark</span>
      <span property="foaf:lastname">Birbeck</span>
      (<span property="foaf:mbox">mark@example.org</span>)
    </span>

    and

    <span rel="foaf:depicts">
      <span property="foaf:firstname">Steven</span>
      <span property="foaf:lastname">Pemberton</span>
      (<span property="foaf:mbox">steven@example.org<span>).
    </span>

  </div>
```

The use of the `rel` attribute without an `href` triggers the definition of a new subresource, which is then the value of the `foaf:depicts` field. Then, all contained markup applies to this new subresource. In RDFa-speak, we call this "chaining," as it allows one to easily chain one resource to a subresource (each of the ~~[illegible]~~ two persons).
& (the photo in this example)

3.5 Using `src` on `IMG`, and the `rev` attribute

Shutr authors may notice that, in a number of cases, the URI of the photos it displays inline using the `IMG` element is actually the same as the `about` attribute for marking up the photo's fields. In order to minimize markup, RDFa allows ~~Shutr~~ authors to make use of the `src` attribute on an `IMG` element: it behaves just like `href`.

Consider Mark's profile page on Shutr, which lists all of his albums and cameras. This page will likely include a picture of Mark himself:

```
<div>
        <h1>Mark Birbeck's Photos</h1>
        <img src="/user/markb/profile_photo.jpg" />
  ...
</div>
```

Shutr may want to indicate that this is Mark's photo, using the FOAF field `foaf:img` defined specifically for this purpose. This can be accomplished as follows:

```
<div about="/user/markb">
        <h1>Mark Birbeck's Photos</h1>
        <img rel="foaf:img" src="/users/markb/profile_photo.jpg" />
  ...
</div>
```

Shutr then notes that the profile photo isn't only Mark's profile photo, it also happens to depict Mark, since Mark obviously appears in his own profile photo (hopefully). This requires expressing an *inverse relationship*, where the field is actually added to the image's URI, not to Mark's profile.

For this purpose, ~~Shutr uses~~ RDFa provides the `rev` attribute, which can be used on `IMG` or any other element. The attribute `rev` functions much like `rel`, except the ~~opposite relationship is expressed:~~ direction of the relationship is reversed

```
<div about="/user/markb">
```

```
        <h1>Mark Birbeck's Photos</h1>
        <img rel="foaf:img" rev="foaf:depicts" src="/user/markb/profile_photo.jpg" />
   ...
</div>
```

In other words, Mark has, as his main image, `/user/markb/profile_photo.jpg`, which of course happens to depict Mark.

3.6 Overriding `href`

When the displayed content is not quite the correct machine-readable data, we used the `content` attribute to override it. In some cases, the clickable link is not quite the right machine-readable data, either. Consider, for example, the case where the displayed photo on Mark Birbeck's profile is, in fact, just a thumbnail, while his official FOAF image is the full-sized version. In this case, and in any case where `href` or `src` appears and needs to be overridden by another URI, RDFa offers the attribute `resource`.

The XHTML written above can then be transformed to:

```
<div about="/user/markb">
        <h1>Mark Birbeck's Photos</h1>
        <img rel="foaf:img"
             resource="/user/markb/profile_photo.jpg"
             src="/user/markb/profile_photo_thumbnail.jpg" />
   ...
</div>
```

Here, the loaded image will the thumbnail, but an RDFa-aware browser will know that the machine-readable data only cares about the full-sized version specified in `resource`.

The `resource` attribute can be particularly useful in cases where the URI is not clickable in any way, e.g. a book's ISBN number represented as `URN:ISBN:0-395-36341-1`.

4 RDF Correspondence

RDF [RDF] is the W3C's standard for interoperable structured data. Though one need not be versed in RDF to understand the basic concepts of RDFa, it helps to know that RDFa is effectively the embedding of RDF in XHTML.

Briefly, RDF is an abstract generic data model. An RDF statement is a triple, composed of a subject, a predicate, and an object. For example, the following triple has `/photos/123` as subject, `dc:title` as predicate, and the literal "Beautiful Sunset" as object:

```
    </photos/123> dc:title "Beautiful Sunset" .
```

A triple effectively relates its subject and object by its predicate: the document `/photos/123` has, as title, "Beautiful Sunset". Structured data in RDF is represented as a set of triples. The notation above is called N3 [N3]. URIs are written using angle brackets, literals are written in quotation marks, and compact URIs are written directly.

All subjects and predicates are nodes, while objects can be nodes or literals. Nodes can be URIs, or they can be blank, in which case they are not addressable by other documents. Blank nodes, denoted `_:bnodename`, are particularly useful when expressing layered data without having to assign URIs to intermediate nodes.

4.1 Events and Contact Information

In Section **2.2 Publishing An Event**, Jo published an event without giving it a URI. The RDF triples extracted from her markup are:

```
_:bn0
        rdf:type cal:Vevent;
        cal:summary "last summer Barbecue";
        cal:dtstart "20070916T1600-0500" .
```

one

w/ whitespace

In Section **2.3 Publishing Contact Information**, Jo published contact information. The RDFa is parsed to generate the following RDF triples:

```
<http://example.org/staff/jo>
          contact:fn "Jo Smith";
          contact:title "Web Hacker";
          contact:org <http://example.org>;
          contact:email <mailto:jo@example.org>.
```

+ whitespace

Advanced

4.2 Simple Shutr Data

The XHTML+RDFa in the first Shutr example in §3.2 yields the following triples:

```
</user/markb/photo/23456> dc:title "Sunset in Nice" .

</user/markb/photo/34567> dc:title "W3C Meeting in Mandelieu" .
```

The more complete example, including licensing information, yields the following triples:

```
</user/markb/photo/23456>
    dc:title "Sunset in Nice" ;
    dc:creator "~~Mark Birbeck~~" ;
    cc:license <http://creativecommons.org/licenses/by-nc/3.0/> .

</user/markb/photo/34567>
    dc:title "W3C Meeting in Mandelieu"
    dc:creator "~~Steven Pembert~~on" ;
    cc:license <http://creativecommons.org/licenses/by/3.0/> .
```

/usr/MarkB

/usr/stevenp

The example that links a photo to the camera it was taken with corresponds to the following triple:

```
</user/markb/photo/23456> shutr:takenWith </user/markb/cameras#nikon_d200> .
```

Camera

while the complete camera descriptions yields:

```
<#nikon_d200>
  dc:title "Nikon D200" ;
  shutr:shutterSpeed "3 pictures/second" .

<#canon_sd550>
```

"frameRate"

```
    dc:title "Canon Powershot SD550" ;
    shutr:shutterspeed "5 pictures/second" .
```

Finally, the `datatype` attribute indicates a datatype as follows:

```
</user/markb/photo/23456> dc:date "2007-05-12"^^xsd:date .
```

4.3 Layered Data and Subresources

The subresources example, in §3.4 with photos annotated with the individuals depicted, correspond to RDF bnodes as follows:

```
</user/markb/photo/23456>
    foaf:depicts _:span0 ;
    foaf:decpits _:span1 ;

_:span0
    foaf:firstname "Mark" ;
    foaf:lastname "Birbeck" ;
    foaf:mbox "mark@example.org" .

_:span1
    foaf:firstname "Steven" ;
    foaf:lastname "Pemberton" ;
    foaf:mbox "steven@example.org" .
```

Note specifically how the bnode, in this example, is both the object of a first triple, and the subject of a number of follow-up triples. This is specifically the point of the layered markup approach: to create an unnamed subresource.

4.4 Using `src` on `IMG`, and the `rev` attribute

The use of `src` on an image in §3.5 yields exactly the same triple as if the `src` were `href`:

```
</user/markb> foaf:img </user/markb/profile_photo.jpg> .
```

The `rev` attribute specifies a triple with the subject and object reversed:

```
</user/markb> foaf:img </user/markb/profile_photo.jpg> .

</user/markb/profile_photo.jpg> foaf:depicts </user/markb> .
```

4.5 Overriding `href`

Where `resource` is present in §3.6 the same triples are generated, with the value of `resource` replacing the value of `href`. Even though the `href` points to `/user/markb/profile_photo_thumbnail.jpg`, the corresponding triple is:

```
</user/markb> foaf:img </user/markb/profile_photo.jpg> .
```

5 Case Studies

A number of RDFa Case Studies are under development and available at http://rdfa.info/rdfa-case-studies/.

6 Acknowledgments

+Shane

This document is the work of the RDF-in-HTML Task Force, including (in alphabetical order) Ben Adida, Mark Birbeck, Jeremy Carroll, Michael Hausenblas, Steven Pemberton, Ralph Swick, and Elias Torres. This work would not have been possible without the help of the Semantic Web Deployment and Best Practices Working Group and the subsequent Semantic Web Deployment Working Group, in particular chairs Guus Schreiber and David Wood, and Ivan Herman, head of the Semantic Web Activity. Earlier versions of this document were officially reviewed by Gary Ng and David Booth, both of whom provided insightful comments that significantly improved the work. Manu Sporny provided crucial feedback in the end stages.

BPD

Add XHTML WG & Shane McCarron

7 Bibliography

FOAF
The Friend of a Friend (FOAF) Project (See http://www.foaf-project.org/.)

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

SWD-WG
Semantic Web Best Deployment Working Group (See http://www.w3.org/2006/07/SWD/.)

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

HTML-WG
HTML Working Group (See http://www.w3.org/MarkUp/Group/.)

XHTML2 ("HTML WG" prior to 2007)

ICAL-RDF
RDF Calendar Interest Group Note (See http://www.w3.org/TR/rdfcal/.)

VCARD-RDF
Representing vCard Objects in RDF/XML (See http://www.w3.org/TR/vcard-rdf.)