HTML5 Microdata and Schema.org

DRAFT NOTE: YKK means things that still need zipped up.

On June 2, 2011 Bing, Google, and Yahoo announced the joint effort Schema.org. When the big search engines talk, web site authors listen.

This is an introduction to Microdata and Schema.org. YKK The tutorial will lead you through implementing these new technologies on a site for discovery of cultural heritage materials. Along the way, you will be introduced to some tools for implementers and some issues with applying this to cultural heritage materials. The conclusion will provide some suggestions on how the cultural heritage sector could . . . YKK

Foundation

HTML5

The HTML5 standard (or HTML Living Standard, depending on who you ask) has brought a lot of changes to web authoring. Amongst all the buzz about HTML5 (some of which is not even part of the HTML spec), HTML5 Microdata, a new semantic markup syntax that *is* part of the HTML standard, often went overlooked. HTML elements often have some semantics. For example, an `ol` element is an ordered list, and by default gets rendered with numbers for the list items. With HTML5 we also have new semantic elements like `header`, `nav`, `article`, and `footer` that allow more expressiveness for page authors. A bunch of `div`s with various class names is no longer the way to divide up much content. These new elements also allow web browser plugins to pull out the article for a cleaner reading experience, or search engines to give more weight to the `article` content rather than the advertising sidebar.

While these new elements provide useful extra information about the content, they provide no deeper understanding of what the `article` is *about*. In many cases you are probably using a nicely normalized relational database or an XML document with a lot of fielded information about your resources. Putting that information in HTML loses a lot of what you already know. The trip from the database or XML into HTML results in lost meaning. Maybe a human can read your field labels to understand your metadata, but that meaning is lost on machines.

One Simple Solution

One solution communicate more of this metadata is to provide an alternative representation of your data separate from the HTML representation. You could even help with auto-detection of that content by providing an alternative link in the `head` of your HTML document.

Here's a simple example of making RIS formatted citation data available:

```
<link rel="alternate" type="application/x-research-info-systems" href="/search?q=cartoons&fo
```

The link provides the type of the alternative representation and a relative URL where it can be found.

This approach can work in many cases, but it has some problems. Techniques like this do not use the visible content of the HTML, so the consumers of your data have to know to look for this particular invisible content.

Embedding Data in Markup

The HTML representation is most visible to users, so it is also the HTML code which gets the most attention from developers. Little-used, overlooked APIs or data feeds are easy to let go stale. If the website goes down, you are likely to hear about it from multiple sources at once. If the OAI-PMH gateway goes down, I am guessing that it would take longer for you to find out about it. Hidden services and content are too easy to get neglected. Data embedded in visible HTML helps keep the representations in sync. This insight has lead to a number of different standards over time which take the approach of embedding structured data with the visible HTML content. Microdata is just one of the syntaxes in use today.

History of Structured Data in HTML

Other efforts that came before Microdata have solved this same problem of marking up the meaning of content. Of the main syntaxes used today, Microformats was not the first but was an early effort to provide "a general approach of using visible HTML markup to publish structured data to the web." Some Microformat specifications like hCard, hCalendar, and rel-license are in common use across the web. Development of the various small Microformat standards take place on a community wiki. Simply put, Microformats usually use the convention of standard class names to provide meaning on a web page. The latest version microformats–2 simplifies and harmonizes these conventions across specifications significantly.

RDFa, a standard of the W3C, has the vision of "Bridging the Human and Data Webs." The idea is to provide attributes and processing rules for embedding RDF (and all of its graph-based, linked data goodness) in HTML. With all that expressive power comes some difficulty, and implementing RDFa has been overly complex for most web developers. Google has supported RDFa in some fashion since 2009, and over that time had discovered a large error rate in the application of RDFa by webmasters. Simplicity is one of the main reasons for the search engines preferring Microdata over RDFa. In part in reaction to greater adoption of Microdata, a simplified profile of RDFa has been created. RDFa Lite 1.1 provides simpler authoring guidelines that mirror more closely the syntax of Microdata.

It also bears mentioning that in the library space, we have developed specifications which tried to solve similar problems of making structured data available to machines through HTML pages. The unAPI specification uses a microformat for exposing (through the deprecated `abbr` method) the presence of identifiers which may resolve to alternative formats through a service. COinS uses empty spans to make OpenURL context objects available for autodiscovery by machines.

So what is Microdata?

Microdata came out of a long thread about incorporating RDFa in HTML5. Because RDFa simply was not going to be incorporated into HTML5, something else was needed to fill the gap. Out of that thread and collected use cases Ian "Hixie" Hickson, the editor of the HTML5 specification, showed the first work on Microdata on May 10, 2009 (the syntax has changed some since then). The syntax is designed to be simple for page authors to implement.

In technical Microdata terms, the things being described are items. Each item is made up of one or more key-value pairs. The Microdata syntax is made up of attributes. Only three new HTML attributes are core to the data model:

- `itemscope` says that there is a new item
- `itemtype` specifies the type of item
- `itemprop` gives the item properties and values

A First Example

```
<div itemscope itemtype="unorganization">
  <span itemprop="eponym">code4lib</span>
</div>
```

The user of a browser would only see the text "code4lib". The snippet provides more meaning for machines by asserting that there is an "unorganization" with the "eponym" "code4lib."

The `@itemscope` attribute creates an item and requires no value. The `@itemtype` attribute asserts that the type of thing being described is an "unorganization." The item has a single key-value pair—the property "eponym" with a value of "code4lib."

To make it clear to someone who thinks in JSON, here is what the item looks like:

```
{
  "type": [
    "unorganization"
  ],
  "properties": {
    "eponym": [
      "code4lib"
    ]
  }
}
```

Those are the only three attributes necessary for a complete understanding of the Microdata model. (You'll learn about two more optional ones later.) Pretty simple, right?

What is Schema.org?

While the above snippet is completely valid Microdata,
it uses arbitrary language for its `@itemtype` and `@itemprop` values. If you live in your own little closed world, that may be just fine. But for the most part you probably want other people's machines to understand the meaning of your content. You need to use a shared language so that page authors and consumers can cooperate on how to interpret the meaning.

This is where the Schema.org vocabulary comes in. The search
engines (Bing, Google, Yahoo) created Schema.org and have agreed to support and understand it. It is unrealistic for them to try to support every vocabulary in use. The domain it covers is broad, sometimes called a Web-scale vocabulary or "middle" ontology. One goal of having such a broad schema all in one place is to simplify things for mass adoption and cover the most common use cases. The vocabulary does seem to have a bias towards commercial use cases. You can browse the full hierarchy of the vocabulary to get a feel of the bounds of the world according to search engines.

Schema.org defines a hierarchy of types descending from Thing. Thing has four properties (description, image, name, url) which are inherited by other types. Child types can add their own properites and have their own children types. A property name with a particular meaning has that same meaning when found in every type in the vocabulary. We will get to other specifics as we go through a tutorial.

Microdata and Schema.org have a tight connection, though each can be used without the other. The search engines are currently the main consumers of Schema.org data and have a stated preference for Microdata, which can be seen through the Schema.org examples being written using the Microdata syntax.

Here's the above Microdata example rewritten to make more sense and use the Schema.org `Organization` type.

```
<div itemscope itemtype="http://schema.org/Organization">
  <span itemprop="name">code4lib</span>
</div>
```

Rich Snippets

The most obvious way that the search engines are currently using Microdata is to be able to display rich snippets in search results. If you have done a Google search in the past two years, you have probably seen some examples of this. You can see good examples of how powerful this is by doing a Google Recipe Search for “vegan cupcakes”:

This search result snippet for a recipe includes an image, reviews, cooking time, calorie count, some of the text introducing the recipe, and a list of some of the ingredients needed. This gives a lot more scent to the user for clicking on a particular result. Snippets with this kind of extra information are reported to increase click through rates.

Google started presenting rich snippets in 2009. Using embedded markup like microformats, RDFa or Microdata, page authors can influence what could show up in a search result snippet. Both Microformats and RDFa were promoted for rich snippets in the past and continue to be supported. Google added support for Microdata for Rich Snippets in early 2010. After RDFa Lite was created, the Schema.org partners agreed to support that syntax as well.

Before Schema.org rich snippets were constrained in the types that would trigger them. Reviews, products, and recipes were common types. At the time of this writing there is still not support for all, or even most, of the Schema.org types, but the number of supported types and example rich snippets has been slowly growing. The promise is that many more of the Schema.org types will begin to trigger rich snippets.

Currently, the main reason to be using Microdata with Schema.org is that it is the latest search-engine preferred method for exposing structured data in HTML. While other consumers for structured data using Microdata and/or Schema.org may appear in the future, the most compelling uses cases are currently for use by the search engines. By providing the search engines with more data on your pages, it improves the search experience of users and can draw them to your site. Since most of the users of your site likely come through the search engines, this could be a powerful way to draw more users to your resources.

From a developer’s perspective there are many considerations for choosing a particular syntax. Microdata has a natural fit with HTML and is designed for simplicity and ease of implementation.

Tutorial

This tutorial will lead you through implementing Microdata and Schema.org on a pre-existing site for exposing digitized collections. The example is based on NCSU Libraries' Digital Collections: Rare and Unique Materials. Each step will lead you through the decisions that are made and the problems that are encounted in implementing Microdata and Schema.org.

Before Microdata

Here's a screenshot of the page to mark up. As we add Microdata to the page, it will continue to look exactly like this. The page uses a grid system to place the image on the left and the metadata to the right.

Those students are excited to learn more about HTML5 Microdata and Schema.org!

Here is the basic structure of the main content of the page with some sections and attributes removed for brevity.

```
<div id="main" class="container_12">
  <h2 id="page_name">
    Students jumping in front of Memorial Bell Tower
  </h2>
  <div class="grid_5">
      <img id="main_image" alt="Students jumping in front of Memorial Bell Tower" src="/imag
  </div>
  <div id="metadata" class="grid_7">
    <div id="item" class="info">
      <h2>Photograph Information</h2>
      <dl>
        ...
      </dl>
    </div><!-- item -->

    <div id="building" class="info">
      <h2>Building Information</h2>
      <dl>
        ...
      </dl>
    </div><!-- building -->

    <div id="source" class="info">
      <h2>Source Information</h2>
      <dl>
        ...
      </dl>
    </div><!-- source -->
  </div>
</div>
```

Adding a WebPage

When using the schema.org vocabularies, every page is implicitly assumed to be some kind of WebPage, but the advice is to explicitly declare the type of page. When we find an appropriate type, we will want to choose the most specific type that could accurately represent the thing we want to mark up. In this case it seems appropriate to use ItemPage. ItemPage adds no new properties to WebPage, but it communicates to the search engines that the page refers to a single item rather than a search results page. I can find no proof for this yet, but it may be that using ItemPage will give an extra hint to a crawler that a page should be indexed or treated differently. For the same reason, marking up a SearchResultsPage could give the hint to the search engines to crawl but not index the page.

In some way when you are adding Microdata you are always just describing a web page. Ian Hickson the editor of the Microdata specification, has said that Microdata items exist "in the context of a page and its DOM. It does not have an independent existence outside the page." (Ian Hickson on public-vocabs list.) This is different than the way RDFa may think about embedded structured data in HTML. Microdata isn't so much linked data as it is a description of a single page.

Here we add the ItemPage to the `div#main` on the page and some properties within.

```
<div id="main" class="container_12" itemscope="" itemtype="http:schema.org/ItemPage">
  <h2 id="page_name" itemprop="name">
    Students jumping in front of Memorial Bell Tower
  </h2>
  <div class="grid_5">
      <img id="main_image" alt="Students jumping in front of Memorial Bell Tower"
        src="/images/bell_tower.png" itemprop="image">
  </div>
  <div id="metadata" class="grid_7">
    ...
  </div>
</div>
```

Here we apply the `itemscope` and `itemtype` attributes at a level in the DOM that surrounds everything we want to describe about the page. Since we are describing, the page, we could instead add the `itemscope` and `itemtype` at a higher level. If we need to use some metadata in the `head` of the document, say the `title` element, we could apply this to the `html` element. YKK reports of some problems

Within `div#main` we add the two `itemprop`s for the “name” and “image” properties. Different elements take their `itemprop` value from different places. In this case the “name” property is taken from the text content of `h2` element. For the `img` element the value is taken from the `src` attribute, which is then resolved into an absolute URL. The `a` element uses the absolute URL from the `href` attribute. If you start using Microdata, you will want to consult this list from the spec which details how property values are determind for different elements.

Microdata JSON and DOM API

One of the cool features of Microdata is that it is designed to be extracted into JSON. You can copy any of the HTML snippets with an itemscope from here into Live Microdata to see what the JSON output would look like.

Live Microdata uses MicrodataJS, which is a Javascript (jQuery) implementation of the Microdata DOM API. The Microdata DOM API is another neat feature of HTML5 Microdata which allows you to extract Microdata client-side.

You can test whether your browser implements the Microdata DOM API by running the microdata test suite created by Opera. If you open this page up in Opera Next (at the time of this writing version 12.00 alpha, build 1191), go to this page, and open up the console (Ctrl+Shift+i), you can play with the Microdata DOM API a bit. `document.getItems()` will return a NodeList of all items. There is (and will be) only one top-level item on this page, so the NodeList will contain only one element. It is possible to get all items of a particular type by specifying the type or types as an argument like `document.getItems('http://schema.org/ItemPage')`. This API may change in the future “to match what actually gets implemented”.

```
>>> var itemPage = document.getItems('http://schema.org/ItemPage')
  undefined
>>> itemPage
  NodeList [<html itemscope="" itemtype="http://schema.org/ItemPage" lang="en">]
>>> itemPage[0].properties
  HTMLPropertiesCollection [<h2 id="page_name" itemprop="name">,
    <img itemprop="image" id="main_image" alt="Students jumping in front of Memorial Bell To
```

ItemPage about a Photograph

Now that we have some basic microdata on the page, let’s try to to describe more items on the page. Looking at the valid properties for an ItemPage, it has an `about` property, inherited from CreativeWork. ItemPage inherits different properites from Thing, CreativeWork, and WebPage. In some cases child types add in new properites, but in the case of ItemPage no new properties are defined.

If we add `itemprop="about"` to the `div` containing all metadata, then what would be extracted from the page is just the text content with line breaks and all. Microdata processing does not maintain any of the markup at all. (If you need to maintain some XML or other markup, then consider using microformats or RDfa, which have the ability to maintain XML literals.) Microdata is specified in a way where if the author of the page gets it wrong, the processor may still be able to do something useful with even just that text content. Processors should expect bad data Conformance.

Looking at how the `about` property is defined, the proper value is another `Thing`. In this way Schema.org suggests how to nest items. In this case it allows us to pick any type in the Schema.org hierarchy to create a new item which will become the value of the `about` property. Photograph is most appropriate here.

You implement nesting by using all three attributes (`itemprop`, `itemscope`, `itemtype`) on the same element. Since all of the metadata is describing the photograph, these will be applied to `div#metadata` which contains all of the metadata.

At this point we can also add the subjects and genres, as properties. Subjects maps well enough to the "keywords" property of `Photograph`. These keywords are visible to users, so it is less likely to try to be gamed in the way that the of meta keywords in the `head` a document was. Google advises page authors to not mark up non-visible content on the page, but to stick to adding microdata attributes to what is visible to users.

We could attach the `itemprop` to the `dd` element, but then a processor would extract all the text. The genres have some spaces within each term, so rather than leaving it up to a post-processor to handle that, we apply the same `itemprop` to each term separately to insure the multiword keywords remain intact. At this time irregardless of singulars or plurals for property names, it is allowable in Schema.org to repeat all properties, which then form a list of values. Even singular properties can have multiple values.

We cannot just apply the `itemprop` to the `a` elements either, though, since the value would come from the `href` attribute. To work around this we use the common pattern of adding some extra spans to get at the text content.

Picking Types and Properties (and More Nesting)

We can also begin to state that the Photograph is `about` the Memorial Bell Tower at North Carolina State University. In this case, `LandmarksOrHistoricalBuildings` seems to work well enough as a type.

```
<div id="building" class="info" itemprop="about" itemscope itemtype="http://schema.org/Landm
```

Once we add a type for the building it is easy to add a name and description. We also have the data to add the address and geographic location as nested items.

Picking appropriate types is an issue for describing the historic record. At NCSU Libraries we are describing lots of buildings and making related architectural drawings accessible. Some of these buildings are on the National Register of Historic Places, but many are buildings of less note. Some were never built or have since been demolished. It could be that for most of the records of buildings in the collection, that something more generic like Place would be a more easy fit. For other buildings there are definitely more specific types which could fit, like Airport, CityHall, Courthouse, Hospital, Church.

If we use one of these more specific types, it may be impossible for the search engines to realize that the items refer to the historic record of these places rather than to their current services. We could be giving exact geographic coordinates for a hospital which was demolished or no longer in operation! Also, this is a new facet that we do not currently capture. It would take some retrospective and ongoing work to choose the correct type for every building. For now we have made the decision to always pick the more specific type of `LandmarksOrHistoricalBuildings` for all the buildings we describe as something of a compromise.

Picking appropriate types is one area where Schema.org does not provide anything near the level of granularity at which archives and museums often specify the types of objects they describe. There are types like Painting, Photograph, and Sculpture. So some types of objects like drawings, vases, and suits of armor may have to move back up the hierarchy to use CreativeWork. With the thousands of types of objets that may be held by archives, museums and historical societies, it is not feasible to add every type to Schema.org. One suggestion made by Charles Moad, Director of the Indianapolis Museum of Art Lab, is to extend CreativeWork with an objectType property (private correspondence).

This objectType property in conjunction with Schema.org (or community practice) making the suggestion to use an appropriate vocabulary for the value could go a long way towards expanding the options for cultural heritage organizations to describe their materials. It would be possible to create something like what the product ontology does by reusing the names of Wikipedia articles (http://en.wikipedia.org/wiki/Plate_armour) to identify types of objects.

```
<div itemscope itemtype="http://schema.org/CreativeWork" itemid="http://www.philamuseum.org/
  <span itemprop="name">Gorget (neck defense) and Cuirass (torso defense), for use in the fi
  <link itemprop="objectType" href="http://objectontology.org/id/Plate_armour">
</div>
```

Result so far

Here's what our marked up snippet looks like so far:

```
<div id="main" role="main" class="container_12" itemscope itemtype="http:schema.org/ItemPage
<h2 id="page_name" itemprop="name">
  Students jumping in front of Memorial Bell Tower
</h2>
  <div class="grid_5">
      <img itemprop="image" id="main_image" alt="Students jumping in front of Memorial Bell
  </div>
  <div id="metadata" class="grid_7" itemprop="about" itemscope itemtype="http://schema.org/P
    <div id="item" class="info">
      <h2>Photograph Information</h2>
      <dl>
        <dt>Created Date</dt>
        <dd>
          circa 1981
        </dd>

        <dt>Subjects</dt>
        <dd>
          <a href="#buildings"><span itemprop="keywords">Buildings</span></a><br>
          <a href="#students"><span itemprop="keywords">Students</span></a><br>
        </dd>

        <dt>Genre</dt>
        <dd>
          <a href="#architectural_photos"><span itemprop="genre">Architectural photographs</
          <a href="#publicity_photos"><span itemprop="genre">Publicity photographs</span></a
        </dd>

        <dt>Digital Collection</dt>
        <dd><a href="#uapc">University Archives Photographs</a></dd>
      </dl>
    </div><!-- item -->

    <div id="building" class="info" itemprop="about" itemscope itemtype="http://schema.org/L
      <h2>Building Information</h2>
      <dl>
        <dt>Building Name</dt>
        <dd><a href="#memorial_tower"><span itemprop="name">Memorial Tower</span></a></dd>

        <dt>Description</dt>
        <dd itemprop="description">Memorial Tower honors those alumni who were killed in Wor
          The cornerstone was laid in 1922 and the Tower was dedicated on
          November 11, 1949.</dd>

        <dt>Address</dt>
        <dd itemprop="address" itemscope itemtype="http://schema.org/PostalAddress">
```

```
            <span itemprop="streetAddress">2701 Sullivan Drive</span><br>
            <span itemprop="addressLocality">Raleigh</span>, <span itemprop="addressLocality
          </dd>

          <dt>Latitude, Longitude</dt>
          <dd itemprop="geo" itemscope="" itemtype="http://schema.org/GeoCoordinates"><span it

        </dl>
      </div><!-- building -->

      <div id="source" class="info">
        <h2>Source Information</h2>
        ...
      </div><!-- source -->
    </div>
</div>
```

All told there are five Microdata items (ItemPage, Photograph, LandmarksOrHistoricalBuildings, PostalAddress, GeoCoordinates) on the page. In part, this snippet now basically says something like this in English:

> This page is an ItemPage about a Photograph. The Photograph is about a LandmarksOrHistoricalBuildings. The Itempage has a "name" of "Students jumping in front of Memorial Bell Tower" and an "image" at "http://example.com//images/bell_tower.png." The Photograph has some "keywords" and "genre." The LandmarksOrHistoricalBuildings has a "name" of "Memorial Tower" and a "description." The LandmarksOrHistoricalBuildings has an "address" which is a PostalAddress item (with its own properties), as well as, a "geo" property which is a GeoCoordinates item.

You can see the extracted JSON at Live Microdata under the JSON tab. YKK redo this link with the final version.

Problems with Rich Snippets

Even though we have all these items marked up, at the time of this writing Google Rich Snippets only supports several of the Schema.org types (Applications, Authors, Events, Movie, Music, People, Products, Products with many offers, Recipes, Reviews, TV Series, not all of which are Schema.org types). No rich snippet would show up in search results or the Rich Snippets Testing Tool for this example right now. But using the Structured Data Linter we get this possible preview for the LandmarksOrHistoricalBuildings.

This is exactly the kind of rich snippet we want users to see for our digitized resources. Hopefully the search engines will beging showing snippets for some of the types cultural heritage organizations are most likely to be making available.

itemref

One thing we have not marked up yet, is to associate the Photograph with the image on the page. We need to fix that, because a rich snippet for a Photograph is unlikely to show up for . The problem is that the image is not nested within the same `div` where the Photograph is defined.

> Valid HTML is particularly important in pages that contain embedded markup. All methods of embedding data within HTML use the structure of the HTML to determine the meaning of the additional markup. http://www.w3.org/wiki/Choosing_an_HTML_Data_Format#Good_Publishing_Practice

While we have the content on our page relatively well organized to contain our items, our layout and grid system mean that the image of the photograph is not nested within the same `div` as the metadata about the photograph. When coding new pages it is important to think not just about presentation but also about how to structure the page with both presentation and data in mind. It makes marking up data easier if the properties of a thing are grouped together.

While in most cases it will be easier to arrange our markup so that the properties of an item are all within the same block on the page, microdata provides a rather simple mechanism for including properties that are outside of that scope.

Microdata uses the `itemref` attribute to make this more convenient.

> Note: The itemref attribute is not part of the microdata data model. It is merely a syntactic construct to aid authors in adding annotations to pages where the data to be annotated does not follow a convenient tree structure. Microdata specification itemref attribute

In our example above we already have an `id` of main_image and an `itemprop` with the value “image” on the image. All that we need to give our Photograph an image property is add `itemref="main_image"` to `div#metadata`. This adds it to the queue of locations in the DOM to check for properties.

```
<img id="main_image" alt="Students jumping in front of Memorial Bell Tower"
       src="/images/bell_tower.png" itemprop="image">
...
<div id="metadata" class="grid_7" itemprop="about" itemscope itemtype="http://schema.org/Pho
  ...
</div>
```

time and Datatypes

Another piece of data which would be good to add to the Photograph is the created date (`dateCreated`) with an expected value of `Date`.

The Microdata specification (unlike RDFa and Microformats–2) does not provide a generic mechanism for specifying data types. Instead Microdata again relies on a vocabulary to define expected data types. Schema.org has four basic types: Boolean, Date, Number, and Text, but none of them are very well documented.

HTML5 has some new elements to allow inclusion of machine-readable data in HTML and these can be used for the Schema.org datatypes. The `time` and `data` elements have been added to HTML5. The specification of `time` has not been stable. In fact it was removed with some strong objections and lots of exciting specification drama.

So while `time` is in the specification again, the Schema.org and Google documentation adds its own confusion to the matter, by using the `meta` element instead when it needs a date. Lots of examples around look like this:

```
<meta itemprop="startDate" content="2016-04-21T20:00">
  Thu, 04/21/16
  8:00 p.m.
```

Using the `meta` and `link` elements within the `body` body of a document is allowed in HTML5 when used with an `itemprop` attribute. These can sometimes be useful for Microdata in expressing the meaning of content or a URL which has no usefulness to a human reader. The `meta` element here is used to give the machine readable date near the date visible to the user. It is hidden on the page, so goes against the general recommendation to not use hidden markup for Microdata. Further, the `meta` element does not even surround the free text version of the date disassociating them from each other.

You could alternatively mark up the same text with `<time>` like so:

```
<time itemprop="startDate" datetime="2016-04-21T20:00">
  Thu, 04/21/16
  8:00 p.m.
</time>
```

The `time` element has the correct semantics for our Photograph:

```
<time itemprop="dateCreated" datetime="1981">
  circa 1981
</time>
```

I have made no attempt to handle the approximate date ("circa"). The current specification processing rules does not handle many valid ISO8601 dates. As dates and ambiguity about dates is important for describing cultural heritage materials, hopefully the HTML5 processing rules can be adjusted to handle all valid ISO8601 dates. It seems as if the WHATWG has accepted a proposal to support year only dates, which is a start.

(YKKK Move this to a footnote or remove? Also the MicrodataJS parser in use on this site erroneously takes the text value of the `time` element rather than the value of the `@datetime` attribute. As `time` is currently specified, the value for purposes of Microdata of `time` is only falls back to the text content if there is no `@datetime` attribute. Up until some time in December of 2011 the Google Rich Snippets Testing Tool, was throwing a warning that just a year was not a valid ISO8601 date, though it seems to be. The HTML5 specification has stricter processing rules.)

itemid

The Building the Memorial Bell Tower is a unique resource which can be represented by this Freebase URI:

http://www.freebase.com/m/026twjv

In Microdata the `itemid` attribute can be used to associate an item with a globally unique URL identifier, "so that it can be related to other items on pages elsewhere on the web." This is the main mechanism by which Microdata natively supports something like linked data. ("Item types are opaque identifiers, and user agents must not dereference unknown item types, or otherwise deconstruct them, in order to determine how to process items that use them." Items) The meaning of the `@itemid` attribute is determined by the vocabulary.

Unfortunately, Schema.org does not document any use or support for the `itemid` attribute at this time, though they "strongly encourage the use of itemids". The semantics of `itemid` in the Schema.org context seems uncertain and overlapping with the consistent use of url properties.

We'll add an `itemid` in any case.

```
<div id="building" class="info" itemprop="about" itemscope="" itemtype="http://schema.org/La
```

Extending Schema.org

Another kind of property a cultural heritage organization might like to add to a landmark or building like the Memorial Tower are the events related to the building. In this case the cornerstone was laid in 1922 and the tower dedicated on November 11, 1949. Other buildings could have events in their history like the dates they were designed or dates of renovations, derived from the drawings and project records. Museums may be interested in various events in the history of a painting including provenance and restorations. History museums and historical societies may also want to refer to various historical events that relate to their exhibits. Each of these may also want to promote various events happening in the future like movie screenings, limited time exhibits, and tours. So it may be important to be able to disambiguate whether an event is in the future or of some historical significance.

Schema.org has an Event type defined as: "An event happening at a certain time at a certain location." That seems broad enough to apply it to either future or historic events. But the message from Google's support page on Rich Snippets for events gives more specific guidance:

> The goal of events snippets is to provide users with additional information about specific events not to promote complementary products or services. Event markup can be used to mark up events that occur on specific future dates, such as a musical concert or an art festival.

Google has a different view of the world than cultural heritage organizations. Because of the focus on the future, we may not want to try to mark up historic events, as rich snippets may be unlikely to show up for past events.

Another option may be to use the Schema.org Extension Mechanism to possibly make it clearer to the search engines that certain types of events are different. As Schema.org is intended as a Web-scale schema, there is no possibility of having it fit every kind of data on the web. The basic mechanism for extending an item type is to take any Schema.org type URL, add a forward slash to the end, and then add the camel cased name of the extension. So one possibility to handle historic events would be to extend Event:

```
http://schema.org/Event/HistoricEvent
```

At least the search engines will understand that these items are some type of event. If enough other folks use the same extension and the search engines notice, then the search engines may start using the data in a meaningful way. There is not a good, public, formal process for how to share extensions or advocate for their inclusion in Schema.org proper. There is still work to be done to have a clear central location to share extensions or have a community process to work out new extensions.

For properties there are two options for mixing in a new property for an existing (or extended type). Schema.org prefers page authors to just add the new property name as if it were defined by Schema.org. So our rights statement could be given this markup:

```
<dd itemprop="rights">
      Reproduction and use of this material requires permission from
      North Carolina State University.</dd>
```

The other option provided by the Microdata specification is to add a full URL for the property like this:

```
<dd itemprop="http://purl.org/dc/elements/1.1/rights">
      Reproduction and use of this material requires permission from
      North Carolina State University.</dd>
```

Another way forward for the cultural heritage sector?

These are just some of the ways in which Schema.org may not fit the needs of cultural heritage organizations very well. Other communities have already voiced their concerns that their domains are not adequately specified.

For instance, the rNews standard of the International Press and Telecommunications Council was added to schema.org. This was the first industry organization to work with the Schema.org partners to publish an expansion of Schema.org to make it more robust and useful in a particular domain. The adoption of much of rNews has resulted in news and publishing-related types being more expressive and better meeting the needs of news organizations.

Another change to the schema was the result of a collaboration between Schema.org and the United States Office of Science and Technology Policy to add support for job postings to schema.org. One immediate use this was put to was to create a job search widget for use on government web sites to highlight job listings from employers who commit to hiring veterans. (See the Veterans Job Bank.)

So this kind of partnership with domain experts seems like a clear way forward for other groups. Libraries, museums, archives, and other cultural heritage organizations could start to enumerate the places where the vocabulary could be modified to better fit their data. There has been some suggestion for future discussions with the cultural sector to this end.

Some of this work may already be available. Work is ongoing to map other vocabularies to Schema.org. This provides a simpler way for organizations to expose their data through Microdata and Schema.org while still maintaining their data in their current schema. The Dublin Core Metadata Initiative has begun an effort to do such a mapping. In each of these mappings there are certainly some areas where there is not overlap, so there is potential for expanding Schema.org in those directions.

Conclusion

Semantic markup of data embedded in HTML is a rapidly changing area. Whether or not you implement Microdata and/or Schema.org, understanding . . . For cultural heritage organizations to YKK

Appendix: Resources

Examples from Cultural Heritage Organizations

- NCSU Libraries' Digital Collections: Rare and Unique Materials The example in this tutorial is based on this site.
- Indianapolis Museum of Art Uses CreativeWork and extends the type by adding the following properties: accessionNumber, collection, copyright, creditLine, dimensions, materials.
- Biodiversity Heritage Library Adds an "OCLC" property for a Book.
- Sudoc French academic union catalogue Seems to only show the microdata representation to crawlers? more information
- Sindice Search This search can be adjusted to find a specific schema.org type (class here). Faceting by the Microdata does not always seem to find sites that predominantly use Microdata rather than RDFa.

Tools

These are tools which I have regularly used.

- Rich Snippets Testing Tool Note that while it may not currently show a *rich* snippet example for every schema.org type, you can use the data at the bottom of the page to insure that your Microdata is being parsed as you intended. The format here breaks every item out and shows references between them in a flat way.
- Structured Data Linter The best feature of this tool is the way that it displays your nested microdata as nested tables, making it easy to spot problems. If the Rich Snippets Testing Tool doesn't show a rich snippet for your content, this is a good alternative to see what your snippets *might* look like. The snippets here do not cover every type either, but they cover a few different types from what the Rich Snippets Testing Tool does, for instance it will show images for more types. The code is open source, so you can run your own instance to be able to check your syntax while you are in development. This is written by folks who have been part of the conversations around web vocabularies and structured data in HTML.

- Live Microdata A good open-source tool for testing snippets of HTML marked up with Microdata. The MicrodataJS source code allows you to implement the Microdata DOM API on your own site, similar to how this tutorial outputs the JSON from parsing the page.
- HTML5 Living Validator
- schema.rdfs.org list of tools

Mappings

There are many efforts to map Schema.org to other vocabularies. If you maintain your metadata in one of these vocabularies, you can use these to help expose your data in a way that the search engines understand.

- Dublin Core Alignment Task Group
- Partial BIBO mapping
- Alignment between rNews and Schema.org
- schema.rdfs.org list of mappings

Other Tutorials

- Google Rich Snippets Videos: This series of short tutorial videos uses Microdata and Schema.org.

- Getting started with schema.org

- HTML Data Guide: This guide helps producers and consumers determine which structured data syntax to use. It covers Microformats, RDFa, and Microdata. Highly Recommended.

- Why rNews? is a good tutorial on how organizations often have good structured data that when made accessible through HTML loses its meaning.

- Dive Into HTML5: "Distributed," "Extensibility," And Other Fancy Words
- Spoonfeeding Library Data to Search Engines
- Extending microdata vocabularies

Discussions

- Public Vocabs list Schema.org discussion has moved over to this list.
- W3C HTML Data Task Force
- W3C Web Schemas Task Force