Make your websites smarter with Schema.org, Part 2: The Schema.org syntaxes

Represent your website's Schema.org information model in HTML

Uche Ogbuji

December 05, 2017

The second part of this four-part series shows you how to translate the abstract information model for data in your web pages into one of the three formats supported by Schema.org: RDFa, Microdata, and JSON-LD.

In Part 1 of this series, I introduced Schema.org, describing how it came about and its history. I explained how it helps fulfill the long-standing vision of the "semantic web," which addresses the requirements of autonomous agents such as search engines. Schema.org was created by large search engine and tech companies, but it is not difficult to learn and implement. This second part of the series shows you how to implement Schema.org on your websites.

Schema.org syntax alternatives

As noted in Part 1, Schema.org is based on RDF. There have been quite a few syntax formats for RDF. While these formats were originally designed as stand-alone documents, it quickly became possible to represent RDF within HTML. To offer flexibility, Schema.org supports three different representation options.

- **Microdata.** Created by the WHATWG HTML working group, microdata is not really based on RDF, but its metamodel is similar enough to RDF to work in Schema.org. It probably has the fewest conceptual bells and whistles of the available formats.

- **RDFa Lite.** Resource Description Framework in Attributes (RDFa) is a W3C recommendation that defines a set of attributes for enhancing HTML with machine-friendly metadata. The idea is to juxtapose the richness of the RDF model with all other aspects of HTML, content, style, and hyperlinks. The full power of RDF and RDFa is more than most web publishers need, so **RDFa Lite** is a separate W3C Recommendation that offers a lighter subset of the syntactical features while omitting some of the more arcane aspects of the RDF model. Schema.org supports this variant.

- **JSON-LD.** JavaScript Object Notation for Linked Data (JSON-LD) is a W3C Recommendation for expressing RDF within the popular JSON format. JSON-LD makes the RDF details as
unobtrusive as possible, for easy adoption by the many developers already using JSON. JSON-LD supports the full RDF model, but its use in Schema.org is restricted to a subset compatible with the simpler model of the other two options.

Though Schema.org supports all three of these options, not every application that implements the format supports all three. Some web crawlers and other agents read some of the formats better than others, and some might read only one of the formats. Because of that, you might have to initially choose your format based on the preferences of the applications you are targeting.

The Schema.org documentation uses Microdata for its introductory text and many of its examples, but only as a convenience to get users started. All three formats are equally valid, and there are many RDFa and JSON-LD examples on the site.

"An important consideration is how widespread the support is for your chosen format. The general impression is that the three formats are supported to a similar extent."

**Encode the book club example in HTML**

In Part 1, I presented an example of an information model from an imaginary book club's web page. Figure 1 shows the Schema.org-based model diagram for the book club.

**Figure 1. Book club Schema.org information model**

![Schema.org Information Model Diagram](image)

Let's look at how to represent this model within HTML. We'll start with RDFa.
Book club example in RDFa

In translating the model to RDFa, I created a straightforward page and added the needed attributes to whichever host element made sense to convey the corresponding concept from the information model. Listing 1 shows the result.

Listing 1. bookclub-rdfa.html

```html
<main vocab="http://schema.org/" typeof="Organization">
  <h1 property="name">Geo Book Club</h1>
  <div property="member" typeof="Person" resource="ang">
    Founding member <span property="name">Alice Ng</span> welcomes you!
  </div>
  <div property="event" typeof="Event" resource="GBC_mtg_2">
    Please join us for our next meeting where we shall discuss the novel
    <span property="about" typeof="Book">
      <u property="name">Things Fall Apart</u> by
      <a property="author" typeof="Person" href="http://enwp.org/Chinua_Achebe">
        <span property="name">Chinua Achebe</span>
        <span property="isbn">9780393932195</span>
      </a>
    </span>
    <img property="image" src="TFA_cover.jpg">
  </div>
  We hope you've been able to attend our past meetings
  <ul>
    <li property="event" typeof="Event" resource="GBC_mtg_1">
      …
    </li>
  </ul>
</main>
```

Learn the magic attributes

This example uses the most common attributes for RDFa, which are:

- **vocab**—Specifies one or more vocabularies for the metadata found within the host element. In this case, a single vocabulary, the one for Schema.org, is defined as the default for all the metadata properties expressed. There are vocabularies you can use for RDFa other than Schema.org. You could even define your own.
- **typeof**—Indicates that the host element represents a concept—that is, a resource—with a particular resource type, and specifies the resource type.
- **property**—Generally the most common attribute. Specifies a particular property on the immediately enclosing resource.
- **resource**—Indicates that the host element represents a concept—or resource—and specifies the resource identifier URL.

Let's have a look from the top down. The first element `<main vocab="http://schema.org/" typeof="Organization">` does three RDFa-related things:

- Sets the default vocabulary to `http://schema.org/`.
- Creates an implicit resource to coincide with the main element. It is described by that element and its contents.
• Sets the type of this implicit resource. The value of the `typeof` is appended to the default vocabulary, resulting in a full URL of `http://schema.org/Organization`.

The default vocabulary is prepended to the value of `typeof` attributes if they are URL references (that is, relative URLs) rather than full URLs. Assume, for example, that you change the opening tag to the following snippet:

```html
<main vocab="http://schema.org/" typeof="http://example.org/Organization">
```

In this case, the resource type ignores the default vocabulary, because it's a full URL, not a relative URL. This application of the default vocabulary to relative URLs also affects `property` attributes. (You could also use a special prefix syntax to abbreviate items that are in a vocabulary other than the default, but I'll cover that in a later article.)

**Resources embedded in the HTML**

As I've shown, in this case there is an outermost resource that is of type `http://schema.org/Organization`. You can have as many resources as you like defined in this way. The common way in RDFa (and also Microdata, as we shall see) to express object properties is through nested HTML elements.

```html
<h1 property="name">Geo Book Club</h1>
```

In this case, the `h1` element gives the name of the organization.

```html
<div property="member" typeof="Person" resource="ang">
```

This line defines a resource of type `http://schema.org/Person` and makes this the value of the `http://schema.org/member` property on the organization.

```
Founding member <span property="name">Alice Ng</span> welcomes you!
```

Here the person resource is given a `http://schema.org/name` property.

So far you've seen RDFa attributes on `main`, `h1`, `div`, and `span` elements. You are free to design your HTML however it suits you, and the RDFa attributes can go on whatever element matches the sense of the concepts you're describing.

**Nesting HTML elements**

RDF is a graph model, whereas the nesting of HTML elements is a hierarchy or a tree. Because HTML holds the natural language description and discussion of the resources, it forms a natural framework for most relationships in the abstract data layer.

RDFa lets you take advantage of this convenience and use hierarchy to set the context of what you are describing. The following diagram illustrates this connection between the HTML document hierarchy and the graph relationships of the RDFa properties.
The following line shows how you can assign a resource identifier—in this case to an event resource.

```html
<div property="event" typeof="Event" resource="GBC_mtg_2">
```

The property is what connects this new resource to the organization, as illustrated in the diagram above.

Note that the value of resource is a relative URL, but it is not the default vocabulary used to complete it. The base URL of the page itself is used to complete resource URLs. That means that if you host this book club page at `http://example.com/geobookclub/`, the full resource URL for this event becomes `http://example.com/geobookclub/GBC_mtg_2`. You can tweak this base URL using the HTML base attribute.

The resource ID can also be taken from an href or src attribute (for example, on a, link, img, or object elements). Whether in such cases, or using resource, you can also specify the full URL for the resource. You would normally do this if it is a reference to a resource at another part of the web, as in the following.

```html
<a property="author" typeof="Person" href="http://enwp.org/Chinua_Achebe">
```

The value of the resource ID is here taken from href, and in the following line it's taken from src.

```html
<img property="image" src="TFA_cover.jpg">
```
Book club example in Microdata

The Microdata version is based on the same HTML page design.

Listing 2. bookclub-udata.html

```html
<main itemscope itemtype="http://schema.org/Organization">
  <h1 itemprop="name">Geo Book Club</h1>
  <div itemscope itemprop="member" itemtype="http://schema.org/Person" id="ang">
    Founding member <span itemprop="name">Alice Ng</span> welcomes you!
  </div>

  <div itemprop="event" itemscope itemtype="http://schema.org/Event" id="GBC_mtg_2"/>
  Please join us for our next meeting where we shall discuss the novel
  <u itemprop="about" itemscope itemtype="http://schema.org/Book">
    Things Fall Apart
  </u>
  by
  <a itemprop="author" itemscope itemtype="http://schema.org/Person" href="http://enwp.org/Chinua_Achebe">
    Chinua Achebe
  </a>
  (ISBN: 9780393932195)
  <img itemprop="image" src="TFA_cover.jpg">
</main>

We hope you've been able to attend our past meetings

<ul>
  <li itemprop="event" itemscope itemtype="http://schema.org/Event" id="GBC_mtg_1">
    ...
  </li>
</ul>

Learn the magic attributes

There are fewer special attributes in Microdata. This example uses three of the five defined attributes. Microdata doesn't conform to the RDF model and is just approximated for purposes of Schema.org. The data item is the main unit of description in Microdata.

- itemscope—Has no value, but flags the attribute in question as a data item.
- itemtype—Specifies a type for the item indicated by itemscope. Generally specified as a full URL.
- itemprop—Specifies a property on an item. Generally specified as a relative URL, and interpreted with a vocabulary relative to the item type.

This example uses the HTML id attribute, and this touches on the fact that Microdata has an odd ambiguity in connection to identifiers. It defines an itemid attribute as well, supposedly for use across the web, but in a way that doesn't really connect to URL concepts. For example, if you want to refer to an item from elsewhere in the same document, you must use id rather than itemid. Even the sole example in the Microdata spec uses a Uniform Resource Name (URN) rather than a URL.

Book club example in JSON-LD

JSON-LD is a completely different approach. While Schema.org recommends embedding it in HTML, this integration doesn't happen seamlessly as it does in RDFa and Microdata. Rather, you create an island of the separate JSON format within a script tag.
Listing 3. bookclub.json

```json
<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "Organization",
  "name": "Geo Book Club",
  "member": [{
    "@type": "Person",
    "@id": "ang",
    "name": "Alice Ng"
  }],
  "event": [{
    "@type": "Event",
    "@id": "GBC_mtg_2",
    "about": {
      "@type": "Book",
      "name": "Things Fall Apart",
      "isbn": "9780393932195",
      "author": {
        "@id": "http://enwp.org/Chinua_Achebe",
        "@type": "Person",
        "name": "Chinua Achebe"
      },
      "image": {
        "@id": "TFA_cover.jpg"
      }
    }
  },
  {"@type": "Event",
   "@id": "GBC_mtg_1"
  }
}]
</script>
```

You can insert this `script` element wherever you like in the document, though I would suggest putting it within the `head` element unless you have reason to do otherwise.

JSON-LD is a full RDF format. The `@context` key establishes the vocabulary for types and properties. Properties are expressed as JSON fields except for those whose names start with @; these have special meanings. The `@id` and `@type` fields provide resource IDs and types, respectively. Resource-to-resource relationships are expressed by having JSON objects as field values, leading to a nesting similar to that of HTML. Multiple properties are represented using JSON lists.

**Picking a format**

You'll certainly want to learn and focus on one Schema.org format. So, the question is: Which one do you choose?

I recommend starting with RDFa because it gives you the richest, most coherent RDF model. Microdata is marginally simpler, but not enough to make it significantly easier to maintain or find developers who can work with it.

You could also get the full RDF model by using JSON-LD, but then you are truly dealing with a separate format from HTML, with the problems that implies for collaboration and staffing. You also would be separating the metadata from the content, making it more likely that the two will diverge
over time and lose consistency. With RDFa and Microdata, the process of adding attributes and piggy-backing on HTML element nesting makes the connection between the content and data more evident. So, you're less likely to neglect dealing with them in tandem.

One thing you should definitely consider: How widespread is the support for your chosen format? Right now, it seems as thought all three formats are supported to a similar extent. JSON-LD was probably the last to come into prominence, but support for it has accelerated in recent years.

**Conclusion**

In Part 2, I've introduced three different ways to express the abstract information model for Schema.org data on a web page: JSON-LD, RDFa, and Microdata. RDFa Lite is strictly an RDF format and, offering more expressive power but with a bit more complexity. The next things for you to get familiar with are the different sorts of information you can encode into your chosen syntax. Schema.org provides many vocabularies for many areas of interest. In the next article, I'll introduce you to a few of these vocabularies and show you how to use the Schema.org documentation to figure out how to express what your own web pages are all about.