HTML Data Guide

W3C Interest Group Note 08 March 2012

This version:
http://www.w3.org/TR/2012/NOTE-html-data-guide-20120308/
Latest published version:
http://www.w3.org/TR/html-data-guide/
Previous version:
http://www.w3.org/TR/2012/WD-html-data-guide-20120112/
Editor:
Jeni Tennison, Independent

Copyright © 2011-2012 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark and document use rules apply.

Abstract

Microformats, RDFa and microdata all enable consumers to extract data from HTML pages. This data may be embedded within enhanced search engine results, exposed to users through browser extensions, aggregated across websites or used by scripts running within those HTML pages.

This guide aims to help publishers and consumers of HTML data use it well. With several syntaxes and vocabularies to choose from, it provides guidance about how to decide which meets the publisher's or consumer's needs. It discusses when it is necessary to mix syntaxes and vocabularies and how to publish and consume data that uses multiple formats. It describes how to create vocabularies that can be used in multiple syntaxes and general best practices about the publication and consumption of HTML data.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This document was published by the HTML Data Task Force, Semantic Web Interest Group as an Interest Group Note. If you wish to make comments regarding this document, please send them to public-html-data-tf@w3.org (subscribe, archives). All feedback is welcome.

Publication as an Interest Group Note does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

The disclosure obligations of the Participants of this group are described in the charter.

Table of Contents

1. Introduction
    1.1 Scope
    1.2 Terminology
2. Publishers
    2.1 Choosing a Publishing Format
        2.1.1 Syntax Considerations
        2.1.2 Vocabulary Considerations
        2.1.3 Usability Considerations
    2.2 Publishing in Multiple Formats
        2.2.1 Mixing Vocabularies
            2.2.1.1 Mixing Vocabularies in Microformats
            2.2.1.2 Mixing Vocabularies in RDFa

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

2.2.1.3 Mixing Vocabularies in Microdata
2.2.2 Mixing Syntaxes
2.2.2.1 Dates and Times
2.2.2.2 Link relations
2.2.2.3 Microdata and RDFa Equivalencies
2.2.2.4 Properties Within Links
2.2.2.5 Datatypes
2.2.2.6 IRIs
2.3 Good Publishing Practice
2.3.1 Valid HTML
2.3.2 Context Independence
2.3.3 Testing
2.3.4 Clear Licensing
3. Consumers
3.1 Choosing a Syntax to Consume
3.1.1 Application Considerations
3.1.2 Tooling Considerations
3.1.2.1 Microdata/Microformats-2 Data Model
3.1.2.2 RDF Data Model
3.1.3 Data Model Considerations
3.1.4 Usability Considerations
3.2 Consuming Pages with Multiple Formats
3.3 Good Consumption Practice
4. Vocabulary Authors
4.1 Extending Vocabularies
4.1.1 Extending Microformats
4.1.2 Extending RDF Vocabularies
4.1.3 Extending Microdata Vocabularies
4.2 Designing Vocabularies
4.2.1 Syntax-Specific Requirements
4.2.1.1 Microformat Vocabularies
4.2.1.2 Microdata Vocabularies
4.2.1.3 RDFa Vocabularies
4.2.2 Syntax-Neutral Vocabularies
4.2.3 Good Vocabulary Design Practices
A. Acknowledgements
B. Multiple Item Types in Microdata
B.1 Mixing Vocabularies using a Type Property
B.2 Mixing Vocabularies using Repeated Content
C. References
C.1 Normative references
C.2 Informative references

1. Introduction

HTML pages naturally contain a lot of semantic information: the title of the page in the `<title>` element, addresses in `<address>` elements, the source of a quotation in the `@cite` attribute, arbitrary metadata about the page in `<meta>` elements and so on. These mechanisms primarily provide metadata about the HTML page itself, but it is also useful to embed data about *other things* within HTML pages.

The first formal methods of embedding data about things other than the HTML page itself within HTML pages were those pioneered by the microformats community. These sought to regularise the existing use of semantic classes and link relations within HTML markup for common subject areas such as people, organisations and events.

Since then, the practice of embedding HTML data within web pages has gradually grown, particularly bolstered by search engines using embedded data to supplement the appearance of entries within their result pages and by the open linked data community seeking to bridge the gap between documents and data on the web. HTML data is used in a variety of ways, as evinced by the use cases collected during the design of microdata. Consumers of HTML data include:

- scripting libraries
- browsers and browser extensions
- general-purpose search engines
- vertical or domain-specific search engines
- data reusers known and unknown to the publisher of the data

There are currently three main syntaxes for embedding data within HTML pages:

microformats

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

microformats use `@class`, `@rel` and other attributes to encode data using standard HTML markup, and can be used with other markup languages that have `@class` attributes. Traditionally, different microformat vocabularies have followed different parsing rules, but microformats-2 provides a standard parsing algorithm.

RDFa
RDFa reuses existing HTML attributes such as `@href` and `@rel` and adds a few of its own to enable data to be extracted from HTML pages as RDF. RDFa was originally designed for XHTML 1.1; its latest version (RDFa 1.1) is also usable with HTML5 and other markup languages such as SVG.

microdata
Microdata adds attributes to HTML5 to provide machine-readable descriptions of items within the page in terms of properties and values for those properties. It is designed to be used alongside detailed specifications of how these descriptions should be processed by consumers.

The three syntaxes are similar in goals but differ in approach. This document provides guidance about how to choose between them and use them together as well as some good practices for publishing, consuming and designing vocabularies for HTML data. However, it is not intended to be a general-purpose introduction to any of these syntaxes. As well as the specifications themselves, examples and explanations can be found within:

- the microformats wiki
- the RDFa primer
- the schema.org website

1.1 Scope

There are many ways of publishing data on the web that do not necessarily involve HTML at all. This document does not cover how to provide data using other data formats, such as JSON or Turtle. It does not talk about HTTP-level mechanisms for providing information about the relationships between resources on the web, such as the `Link:` header. It does not discuss techniques for embedding data in non-HTML files, such as metadata embedded within PDFs or JPEGs through XMP.

Even with a focus on methods that can be used in HTML, there are many techniques for publishing data such that it can be discovered from HTML pages or used by scripts and stylesheets that operate over your page.

First, publishers may link to alternative versions of a document, using different syntax, through a `link` element. The `@rel` attribute should take the value `alternate` and the `@type` attribute should provide the mime type of the alternative representation. For example:

```
<link rel="alternate" type="text/calendar" value="calendar.ics" />
```

Second, publishers may embed data within the `head` of an HTML document, nested inside a `script` element with an appropriate `@type` attribute. This method can be used for text-based formats, such as JSON or Turtle, as well as XML-based formats. For example:

```
<script type="text/turtle">
  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
  @prefix gr: <http://purl.org/goodrelations/v1#> .
  @prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

  <#company> gr:hasPOS <#store> .

  <#store> a gr:Location ;
    gr:name "Hair Masters" ;
    vcard:adr [
      a vcard:Address ;
      vcard:country-name "USA" ;
      vcard:locality "Sebastpol" ;
      vcard:postal-code "95472" ;
      vcard:street-address "6980 Mckinley Ave" ;
    ] ;
    foaf:page <> ;
    .
</script>
```

Third, data can be embedded through custom data attributes. These must not be used by third parties, but can be useful when the only consumers of the data are scripts and stylesheets used by the page. For example:

```
<div class="spaceship" data-ship-id="92432"
     data-weapons="laser 2" data-shields="50%"
     data-x="30" data-y="10" data-z="90">
 <button class="fire"
         onclick="spaceships[this.parentNode.dataset.shipId].fire()">
  Fire
```


DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

```
  </button>
</div>
```

This document focuses on methods of data markup within HTML that reuse visible data within the page. Embedding data within an HTML page has the advantage of avoiding repetition, enables access through scripts and stylesheets, and is more easily discoverable by browsers and search engines which regularly consume HTML documents.

1.2 Terminology

Within this document, a ***format*** is a combination of a syntax and types and properties from one or more vocabularies. Traditional microformats do not make the distinction between syntax and vocabulary, but RDFa, microdata and microformats-2 do make this distinction.

In this document, a ***syntax*** is a set of conventions for parsing data from an HTML page into a data structure. The three syntaxes discussed in this document are RDFa, microdata and microformats-2. Each of these can be used with different vocabularies.

A ***vocabulary*** is a set of terms for describing entities within a particular domain. Different mechanisms are used for describing vocabularies. A microformat vocabulary is described within a wiki page. An RDFa vocabulary might be described through an RDFS schema or OWL ontology provided at the vocabulary's URI. A microdata vocabulary must be described within a specification that describes how it is processed.

All three syntaxes follow a similar data model. Each is used to describe ***entities*** — things such as people or events (RDFa calls these resources, microdata calls these items). These entities each have one or more ***types*** which indicate what kind of thing they are and a number of ***properties*** that have ***values***, which provide the data about the entity. The main difference is that in the RDF generated from RDFa, the entities are arranged in a graph, whereas the default data model for microformats and microdata is a tree.

Types, properties and entities can be identified in different ways. Microformats uses short names. RDFa, like RDF, uses IRIs, while microdata uses URLs as defined in HTML5. This document tries to use the appropriate term (IRI or URL) when discussing identifiers, but sometimes uses the term URL to mean a URL or IRI. See also section 2.2.2.6 IRIs for more detail around the use of identifiers in microdata and RDFa.

2. Publishers

If you are publishing HTML data, you are likely to find that the markup within your pages is simpler and easier to maintain if you only use one format (syntax and vocabulary) within each page. To decide which to use, your first consideration has to be which consumers will read the data within your web pages, and which formats they support. These may include:

- scripting libraries
- browsers and browser extensions
- general-purpose search engines
- vertical or domain-specific search engines
- data reusers with whom you have agreements

Your second consideration may be the current state of the tooling to support a particular format. For example:

Are you able to publish using HTML5?
: If you are using a content-management system that doesn't support adding new attributes such as `@itemprop` or `@typeof` then you will be constrained to using microformats.

Are there development tools available?
: Because it is not visible within a web page, it can be hard to tell whether HTML data has been written correctly. Consumers should provide validators that enable you to check that your data has been correctly detected and interpreted, but you may also want to consider tool support for generating the HTML data.

Note

Microdata requires the use of attributes which are introduced by HTML5 and RDFa can be used with XHTML 1.1 or HTML5, while microformats can be used with all versions of HTML. Your organisation's publishing guidelines may need to be brought up to date to sanction use of microdata or RDFa.

Once you have considered both your target consumers and the tooling support that is available, you will be in one of four situations:

1. **with a single choice of format** in which case there are no further choices to be made
2. **unable to publish HTML data that your target consumers understand** in which case you either have to lobby those consumers to add support for the format(s) you can publish in, or consider changing your toolset so that you

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

can publish in something they understand
3. **still with a choice between a number of formats** in which case you will want to pick one to use; this is covered in section 2.1 Choosing a Publishing Format
4. **having to use multiple formats at the same time to provide data to all your target customers** in which case you will need to mix formats within your pages; this is covered in section 2.2 Publishing in Multiple Formats

2.1 Choosing a Publishing Format

This section addresses a situation where all your target consumers recognise a set of formats (each with a particular syntax and vocabulary), your toolset supports publishing in all of them, and you need to make a choice about which of these formats to use. It's assumed that you will want to choose a single format rather than mixing multiple formats as described in section 2.2 Publishing in Multiple Formats, as this will mean less markup in your page and make your publishing task easier.

2.1.1 Syntax Considerations

The different syntaxes — microformats, microdata and RDFa — have different capabilities which may inform your choice.

Structured HTML values

Under appropriate conditions, RDFa and microformats will use markup within the content of an element to provide a property value; in microdata values never retain markup. If property values within your page contain markup (for example a `description` property containing emphasized text, multiple paragraphs, tables and so on), you may want to use RDFa or microformats to ensure that structure is available to consumers of your pages. In RDFa, this is done through adding `datatype="rdf:XMLLiteral"` to the relevant element. In traditional microformats, the handling of the content of an element is determined by the property; in microformats-2, those that retain the HTML structure are named with a `e-*` prefix, such as `e-content`.

Language support

Microformats and RDFa use the language of the HTML elements in the page (from the `@lang` attribute) to indicate the language of relevant values. In microdata, the vocabulary has to provide a separate mechanism to indicate a language. If you have multi-lingual information in your pages, you may find it easier to use microformats or RDFa than microdata.

CSS support

Because microformats generally use classes to mark up data within an HTML page, it is easy to use CSS to style those elements based on their type. For example `.hcard .n { font-weight: bold; }` will enbolden any person's name. This is a little harder with microdata where the selector might be something like

```
[itemtype~="http://microformats.org/profile/hcard"] [itemprop~="n"]
```

or RDFa where it might be

```
[typeof~="foaf:Person"] [property~="foaf:name"]
```

If you are planning to style your page based on the data embedded within it, you may find it easier to use microformats than either microdata or RDFa; if you do style RDFa, you should plan for dependencies between your CSS documents and any prefixes used within it.

Issue

The handling of language by microdata may change in the future.

2.1.2 Vocabulary Considerations

Vocabularies and syntaxes are closely tied together, especially in the case of microformats. Aspects of a vocabulary to bear in mind are:

- How closely does it match with the information that you have?
- How much support does it have? Are there tools for validating and viewing it? Is there good documentation?
- How stable is it? Who has control to make changes to it? How frequently might those changes be made?
- Are other consumers likely to adopt it in the future?

2.1.3 Usability Considerations

The usability of a particular format is likely to depend on your existing expertise and the match between the structure and content of your web pages and the required structure and content of the format. The best thing to do is to try using the format to mark up an example page from your site.

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

2.2 Publishing in Multiple Formats

Publishing in multiple formats can be easy. For example, it may be that different consumers expect HTML data to appear in different places within the page, such as Facebook requiring Open Graph Protocol data to appear within the `head` of an HTML page, while schema.org markup appears in the `body` of the page. Or it may be that the items that you need to mark up on the page appear in different places — events listed in a sidebar while company details are provided in a footer, for example.

Different formats and vocabularies can be used independently in these circumstances. Consumers of the data within your pages might read additional data if it is in a syntax that they recognise — for example, an processor that recognises both RDFa and microdata will interpret all such markup in the page — but it should ignore information that is in a vocabulary that it doesn't understand rather than giving an error.

Publishing can be harder when there are multiple consumers of information that require different formats. If your target consumers will all accept the same syntax, it is usually easiest to use that single syntax in your pages. However, microdata does not support multiple types for a single entity, so if your target consumers expect different vocabularies to be used for the same entities you may find it easier to mix syntaxes or to use RDFa or microformats, which do support multiple vocabularies.

2.2.1 Mixing Vocabularies

Methods for marking up the same data in a page using different vocabularies in the same syntax vary by syntax.

2.2.1.1 Mixing Vocabularies in Microformats

As microformats are simply indicated through classes, it's possible to mix several within the same set of content. An example is the BBC Bangladesh River Journey page which includes hAtom, hCalendar and geo microformats:

```
<li class="hentry vevent xfolkentry postid-f2068841910">
  <h3 class="entry-title summary">
    <a href="http://www.flickr.com/photos/bangladeshboat/2068841910"
       title="The final picture (on Flickr)">The final picture</a>
  </h3>
  <div class="entry-content">
    <p class="photo">
      <a rel="bookmark" class="taggedlink url"
         href="http://www.flickr.com/photos/bangladeshboat/2068841910"
        title="The final picture (on Flickr)">
        <img src="http://farm3.static.flickr.com/2175/2068841910_1162a8086b_s.jpg"
             alt="The final picture (on Flickr)"
             title="The final picture (on Flickr)" width="64" height="64" />
      </a>
    </p>
    <p class="description">As the BBC team prepare to disembark the boat,
      the sun sets overhead, and indeed on the trip itself.</p>
  </div>
  <ul class="meta">
    <li class="date">
      <abbr class="published dtstart" title="2007-11-26T02:11:51+06:00">2 days ago</abbr>
    </li>
    <li class="location">
      <abbr class="geo point-22" title="+22.47157;+89.59534">Mongla, Bangladesh</abbr>
    </li>
  </ul>
</li>
```

2.2.1.2 Mixing Vocabularies in RDFa

RDFa is designed to be used with multiple vocabularies:

- types and properties are given IRIs as names, so do not have to be disambiguated; IRIs do not have to be written out in full (see below)
- an entity can be assigned multiple types from different vocabularies by listing them within the `@typeof` attribute
- attributes that indicate properties (`@property`, `@rel` and `@rev`) can take multiple space-separated properties which may be from different vocabularies

Writing out IRIs in full can clutter HTML so RDFa provides four mechanisms to shorten IRIs:

- There are several built-in prefixes which can be used for popular vocabularies. These are listed as part of the RDFa 1.1 Core initial context. Any IRI within one of these vocabularies can be abbreviated using the `prefix:name` notation.

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

- The `@prefix` attribute can be used to define additional prefixes for other vocabularies.
- The `@vocab` attribute defines a default vocabulary within its scope; any IRIs that begin with this vocabulary can be abbreviated to a short name (the remainder of the IRI after the vocabulary IRI).

Note that if you use any of the last two mechanisms, the shortened IRIs can only be understood when they are within the scope of the relevant attributes. These can be easy to mislay when people copy and paste HTML from one place to another, or as the result of template changes in a content-management system. We therefore recommend that these attributes are avoided where possible — use the built-in prefixes or full IRIs in preference — and, where they are used, placed on elements that represent entities (those with `@about` or `@typeof` attributes) and repeated on each entity element rather than being inherited from an ancestor element. For more details, see section 2.3.2 Context Independence.

2.2.1.3 Mixing Vocabularies in Microdata

Microdata is designed such that each piece of information in a page is assigned types from a single vocabulary, though each entity may have multiple types and have properties from other vocabularies.

Properties in microdata are either short names (in which case they are scoped to the vocabulary of the types of the entity) or URLs. A URL property has no relationship to a given short name property unless that relationship is specified within the vocabulary that defines the properties.

You might find that you need to target two consumers who each recognise items using types from different vocabularies. For example, you might want to both target schema.org and use the vEvent vocabulary when providing data about an event.

In this case there are three options available to you. The first, if consumers support it, is to use a different syntax for one of the vocabularies. For example, the vEvent vocabulary is only supported in microdata but schema.org can be consumed from either microdata or RDFa, so it would be possible to mark up the data using the vEvent vocabulary in microdata and the schema.org vocabulary in RDFa. This approach is described in more detail in section 2.2.2 Mixing Syntaxes. Mixing syntaxes within a single page is rarely a good option but in some circumstances it may be preferable to the other workarounds described here.

The second option is to use a property that is treated by consumers as providing the type for an item, as if the `@itemtype` attribute had been used. This requires vocabulary authors to define such a property for a given vocabulary.

The third option is to repeat the data markup, once in visible content and once in hidden markup (either through `link` and `meta` elements or in a section hidden using CSS).

These two options are described in detail within section B. Multiple Item Types in Microdata.

2.2.2 Mixing Syntaxes

A requirement to support a large range of consumers can mean that it becomes necessary to publish using not only multiple vocabularies but multiple syntaxes.

RDFa, microformats and microdata all share the same basic entity/property/value model, so in many cases it is possible to mirror attributes across the syntaxes. The following example shows the same content marked up with:

- hCalendar (microformat)
- schema.org (RDFa)
- vEvent (microdata)

```
<div class="vevent"
  itemscope itemtype="http://microformats.org/profile/hcalendar#vevent"
  vocab="http://schema.org/" typeof="Event">
  <a class="url" itemprop="url" property="url" href="nba-miami-philadelphia-game3.html">
    NBA Eastern Conference First Round Playoff Tickets:
    <span itemprop="summary" property="name"
         class="summary"> Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1) </span>
  </a>

  <time itemprop="dtstart" property="startDate" content="2016-04-21T20:00:00">
    <abbr class="dtstart" title="2016-04-21T20:00:00">
      Thu, 04/21/16
      8:00 p.m.
    </abbr>
  </time>

  <div class="location" itemprop="location"
       vocab="http://schema.org/" property="location" typeof="Place">
    <a property="url" href="wells-fargo-center.html">
      Wells Fargo Center
    </a>
```


DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

```
      <div property="address" vocab="http://schema.org/" typeof="PostalAddress">
        <span property="addressLocality">Philadelphia</span>,
        <span property="addressRegion">PA</span>
      </div>
    </div>
  </div>
```

A microformats processor will extract the data:

```
{
  "type": [ "vevent" ],
  "properties": {
    "url": [ "http://example.com/nba-miami-philadelphia-game3.html" ],
    "summary": [ " Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1) " ],
    "dtstart": [ "2016-04-21T20:00:00" ],
    "location": [
      "\n    \n      \n      Wells Fargo Center\n      \n      \n        Philadelphia,\n        PA\n      \n    \n  "
    ]
  }
}
```

A microdata processor will extract something very similar, the only difference being the URL of the type:

```
{
  "type": [ "http://microformats.org/profile/hcalendar#vevent" ],
  "properties": {
    "url": [ "http://example.com/nba-miami-philadelphia-game3.html" ],
    "summary": [ " Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1) " ],
    "dtstart": [ "2016-04-21T20:00:00" ],
    "location": [
      "\n    \n      Wells Fargo Center\n    \n    \n      Philadelphia,\n      PA\n    \n  "
    ]
  }
}
```

while processors that map microdata to RDF would extract the following RDF from the microdata markup:

```
@prefix hcal: <http://microformats.org/profile/hcalendar#>

[] a hcal:vevent ;
  hcal:url <http://example.com/nba-miami-philadelphia-game3.html> ;
  hcal:summary " Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1) " ;
  hcal:dtstart "2016-04-21T20:00:00"^^xs:dateTime ;
  hcal:location
    "\n    \n      Wells Fargo Center\n    \n    \n      Philadelphia,\n      PA\n    \n  " ;
  .
```

and an RDFa processor will extract the data provided through the schema.org vocabulary:

```
[] a schema:Event;
  schema:location [
    a schema:Place ;
    schema:address [
      a schema:PostalAddress ;
      schema:addressLocality "Philadelphia" ;
      schema:addressRegion "PA" ;
    ] ;
    schema:url <http://example.com/wells-fargo-center.html> ;
  ] ;
  schema:name " Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1) " ;
  schema:startDate "2016-04-21T20:00:00"^^xsd:dateTime ;
  schema:url <http://example.com/nba-miami-philadelphia-game3.html> ;
  .
```

It is particularly important to check pages in which syntaxes are mixed together using an appropriate validator for each format.

The following guidelines may help when creating pages in which different syntaxes are mixed together.

2.2.2.1 Dates and Times

Microformats do not use `link` or `meta` elements within the content of the page and in some cases require particular elements to be used to encode information. In particular, `abbr` must be used to support the datetime-design-pattern. Conversely, properties that hold dates and times must be marked up using the `time` element in microdata. Using the `time` element is also advantageous in RDFa, as it automatically confers the appropriate datatype to the value. So when using both microformats and RDFa or microdata, you must nest a `time` element within a `abbr` element or vice versa, as shown here:

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

```
<time itemprop="dtstart" property="startDate" content="2016-04-21T20:00:00">
  <abbr class="dtstart" title="2016-04-21T20:00:00">
    Thu, 04/21/16 8:00 p.m.
  </abbr>
</time>
```

RDFa vocabularies are typically stricter in the range of values that they accept for properties that take dates and times; it is best to use the syntax `YYYY-MM-DD` for dates, `hh:mm:ss` for times and `YYYY-MM-DDThh:mm:ss` for dateTimes to be compliant with the XML Schema dates and times which RDFa-based vocabularies will typically use.

Issue

It is likely that the HTML5 `time` element will accept types of values that do not have an equivalent XML Schema datatype. These should be avoided when using RDFa. See bug 14881.

2.2.2.2 Link relations

In (X)HTML5 markup, unprefixed values in the `@rel` attribute will usually be ignored by RDFa processing unless there is a `@vocab` attribute in scope, the exceptions being `describedby`, `license` and `role` which will be recognised as being part of the HTML vocabulary. In RDFa in XHTML 1.1, some additional unprefixed values are recognised as known terms and used to create triples.

Link relations required in certain microformats, particularly XFN, clash with the use of RDFa's `@vocab` attribute. For example:

```
<a vocab="http://purl.org/dc/terms/"
   rel="date" href="http://reference.data.gov.uk/id/day/2011-11-15">15th November 2011</a>
```

will result in a `dc:date` relationship based on RDFa processing, but XFN processing will assume that the link is to someone whom the author of the HTML page is dating.

To avoid the `@rel` attribute being misinterpreted, it is best to avoid using `@vocab` on any ancestor of an element that contains a `@rel` attribute: use `@property` instead to provide RDFa properties, and if you need to use `@rel` attributes on your links, use prefixes instead of `@vocab` in the RDFa markup.

2.2.2.3 Microdata and RDFa Equivalencies

When marking up RDFa alongside microdata, the following equivalencies between attributes generally hold true:

- `@itemid` = `@resource`
- `@itemtype` = `@typeof` (+ `@vocab` to enable the use of short names for properties)
- `@itemprop` + `@itemscope` = `@property` + an empty `@typeof` if there's no `@itemtype`
- `@itemprop` otherwise = `@property`

2.2.2.4 Properties Within Links

When using RDFa, any `@property` attributes within an element with a `@href` (ie a link) will be taken as providing properties of the entity identified by the URL in that `@href`. This is not the case in microdata or microformats, where the `@href` attribute is only ever used to provide a value for a property. For example, the microdata:

```
<div itemscope itemtype="http://schema.org/AggregateRating">
  Ratings:
  <a href="ratings"
     title="23,201 IMDb users have given an average vote of 7.2/10">
    <span itemprop="ratingCount">23,201</span> users</a>
</div>
```

will generate an `http://schema.org/AggregateRating` whose `ratingCount` is `23,201`. However, the similar RDFa:

```
<div vocab="http://schema.org/" typeof="AggregateRating">
  Ratings:
  <a href="ratings"
     title="23,201 IMDb users have given an average vote of 7.2/10">
    <span property="ratingCount">23,201</span> users</a>
</div>
```

creates two unconnected statements:

```
[] a schema:AggregateRating .
<http://example.com/ratings> schema:ratingCount "23,201" .
```


DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

If the link doesn't have a `@rel` attribute, as in this example, you can avoid the `@href` attribute creating a new subject by adding an empty `@property` attribute to the link:

```
<div vocab="http://schema.org/" typeof="AggregateRating">
  Ratings:
  <a href="ratings" property=""
     title="23,201 IMDb users have given an average vote of 7.2/10">
    <span property="ratingCount">23,201</span> users</a>
</div>
```

If the link *does* have a `@rel` attribute, it is usually easiest to move the relevant property outside the link, for example:

```
<div vocab="http://schema.org/" typeof="AggregateRating">
  Ratings:
  <span property="ratingCount" content="23201"></span>
  <a href="ratings" rel="nofollow"
     title="23,201 IMDb users have given an average vote of 7.2/10">
    23,201 users</a>
</div>
```

The alternative is to identify the subject explicitly using a `@resource` attribute on both the outer element and the link element:

```
<div vocab="http://schema.org/" typeof="AggregateRating"
     resource="_:rating">
  Ratings:
  <a href="ratings" rel="nofollow" resource="_:rating"
     title="23,201 IMDb users have given an average vote of 7.2/10">
    <span property="ratingCount">23,201</span> users</a>
</div>
```

These three methods all generate the same RDF:

```
[] a schema:AggregateRating ;
  schema:ratingCount "23,201" ;
  .
```

2.2.2.5 Datatypes

The `@datatype` attribute might be required for some RDFa vocabularies/consumers; others will coerce values into the appropriate datatype based on the property itself. However, if a property takes a structured value, the property element must have `datatype="rdf:XMLLiteral"` for that structure to be preserved.

2.2.2.6 IRIs

HTML defines some attributes, such as `@href` and `@src`, as holding URLs. The currently specified processing of these URLs results in non-URI characters within IRIs being percent-encoded. This also happens with microdata attributes such as `@itemid` and `@itemtype`.

This normalisation does not happen in attributes defined in RDFa, such as `@resource` and `@property`: IRIs provided in these attributes will be passed into the extracted RDF as IRIs.

This discrepancy means that when using RDFa, you have to be careful to use URIs only (by percent-encoding IRIs) or avoid using the HTML-defined attributes such as `@href` or `@src`. For example:

```
<p resource="#menu">
  <a property="eg:wine" href="#rosé">Rosé</a>
  ...
</p>
...
<p resource="#rosé">
  <span property="eg:description">This Californian wine...</span>
</p>
```

will result in the RDF:

```
<#menu> eg:wine <#ros%E9> .
<#rosé> eg:description "This Californian wine..." .
```

The URL in the `@href` attribute is percent-encoded, while the one from the `@resource` attribute is not; while the URLs appear identical in the HTML, in the RDF, they refer to distinct entities.

This can be avoided by percent-encoding the non-URI characters within the original HTML:

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

```
<p resource="#menu">
  <a property="eg:wine" href="#ros%E9">Rosé</a>
  ...
</p>
...
<p resource="#ros%E9">
  <span property="eg:description">This Californian wine...</span>
</p>
```

which will result in:

```
<#menu> eg:wine <#ros%E9> .
<#ros%E9> eg:description "This Californian wine..." .
```

or by using the `@resource` attribute to provide the IRI value for a property:

```
<p resource="#menu">
  <a property="eg:wine" resource="#rosé" href="#rosé">Rosé</a>
  ...
</p>
...
<p resource="#rosé">
  <span property="eg:description">This Californian wine...</span>
</p>
```

which will result in:

```
<#menu> eg:wine <#rosé> .
<#rosé> eg:description "This Californian wine..." .
```

Similar considerations apply when mixing microdata or microformats with RDFa, since the identifiers used within the microdata or microformats will be URIs rather than IRIs.

It is good practice for vocabulary authors to state whether any further normalisation occurs when interpreting URL values, and to either avoid using IRIs for property names or state explicitly equivalence between IRIs and the percent-encoded URI versions of property and type identifiers that will be generated from microdata markup.

2.3 Good Publishing Practice

There are a number of practices which can help ensure good quality HTML Data that can be easily reused by consumers.

2.3.1 Valid HTML

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. For example, in microdata the item to which an element with an `@itemprop` attribute assigns a property is usually the closest ancestor element with a `@itemscope` attribute.

In some cases, elements can be moved when HTML is parsed into a DOM. This can lead to properties unexpectedly referring to the wrong entity, and, if you are serving your documents as XHTML (with a `application/xhtml+xml` mime type), it can cause discrepancies between the data gleaned by XML-based consumers and HTML-aware consumers. There are two causes for this:

- Error correction in HTML parsing can restructure invalid HTML is restructured to make it valid, for example non-table markup within a table is moved to before the table. This includes `link` and `meta` elements that are directly within the `table` element. You can avoid this restructuring by making sure that your HTML is valid so that it is not needed.
- Firefox 3.5 and 3.6 move `meta` elements in the `body` of an HTML document to within the `head` element, because they cannot not validly appear within the body in older versions of HTML. If you are targeting consumers which run within these old browsers, such as scripts or extensions, you can avoid this restructuring by using empty `span` or other elements instead of `link` or `meta`; other consumers should be using an up-to-date HTML5 parser which will not do this.

2.3.2 Context Independence

One of the ways in which people learn how to publish information on the web is to view the source of other web pages and copy portions of their contents into their own pages. It is also common for web pages to be constructed from templates and for these to change as the result of site redesigns. In both these situations, it can be easy to lose any context information that is used to interpret the HTML Data embedded within the page.

To help preserve relevant context information:

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

- when using microformats, use microformats-2 if possible as the prefixed classnames are less likely to be changed during site redesigns; use the top-most microformat class as near as possible to the properties of the relevant entity
- when using RDFa, avoid using namespace declarations, `@prefix` or `@vocab`; if you do use them, add them as close to the elements that use the prefixes or vocabulary as possible
- when using microdata, add the `@itemscope` attribute as closely as possible to the data and use `@itemtype` where a relevant type is available rather than relying on consumers to infer the type

2.3.3 Testing

It is good practice to test the data that you expose within your page against a parser that will show you the data your page contains. It is also good practice to test the data that you expose using a tool that understands the vocabulary you are using. Consumers may provide testing tools and validators for this purpose, or you may need to check the way that vocabulary-specific tools behave with your data.

If you are constructing your page from a database, another good testing approach is to compare the data extracted from the page with the data extracted directly from the database.

2.3.4 Clear Licensing

The goal of publishing HTML data is to enable consumers to reuse it. To make it clear how the HTML data you publish can be reused, you should include information about the rights holder and license that the information is made under. There are a number of vocabularies that enable you to do this, such as schema.org, rel-license, Creative Commons and Dublin Core. Your target consumers should indicate which formats they understand when it comes to expressing licensing information and which licenses they know about, and you should choose a relevant format in the same way as you do for the core data that you are publishing.

3. Consumers

You will find it easier to consume and combine data published using a single format (syntax and vocabulary). To decide which to consume, you should first look at what formats your target publishers are currently using. It may be that these contain sufficient information for your application.

If the publishers whom you are targeting are already publishing using multiple formats, you may want to consume from all those formats (see section 3.2 Consuming Pages with Multiple Formats) in order to maximise the data that you can collect while minimising the impact on the publishers who are providing that information. If you are consuming microdata and storing the results as RDF, you should follow a standard mapping.

If current formats do not encode the information you need to the detail you need it for your application, publishers will be more likely to publish extra data for you to consume if you:

- extend existing common vocabularies they are already using
- consume data from a syntax they already use

If you cannot simply extend an existing vocabulary, you will need to create your own vocabulary and choose which syntaxes to support with that vocabulary.

3.1 Choosing a Syntax to Consume

As you choose syntax, you should take into account the following considerations.

3.1.1 Application Considerations

Microdata, RDFa and microformats-2 all use a generic syntax, which means that it's possible to have generic parsers operate over them to extract data. In the case of microdata and microformats-2, the data has a JSON structure; data extracted from RDFa has a RDF structure (microdata can also be converted into RDF).

Generic applications can work in the browser to do things such as highlighting markup that follows a particular syntax or enabling users to download the data embedded within a page into a separate file. These can also use the context in which the HTML data is found to provide additional features. For example, generic consumers may detect that each row in a table is associated with a distinct entity, and each cell with a particular property, and enable users to sort that table based on property values. In this case, a consumer could ensure that when values are marked up as dates, times or durations using the `time` element, the items are sorted by date/time/duration rather than alphabetically.

Both microformats-2 and RDFa provide additional facilities that enable publishers to indicate the datatypes of values to support generic consumers. Microformats-2 properties have a prefix that can indicate when a value is a URL (`u-*`), a date/time (`dt-*`), extended HTML (`e-*`) or a string (`p-*`). RDFa supports a `@datatype` attribute that publishers can use to

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

indicate the datatype of a value, usually an XML Schema datatype such as `xsd:integer` or `xsd:language`. Note that once microformats-2 data is extracted from a page into JSON, these prefixes are no longer available, so a consumer of the JSON has to know the vocabulary to tell whether a given value should be interpreted as a string or as HTML markup, for example. In contrast, the datatypes used to annotate RDFa values are carried within the RDF data.

RDFa also adheres to a follow-your-nose principle, whereby vocabulary authors are encouraged to provide a machine-readable description of types and properties at the URL used for the type or property. This can enable generic processors to automatically pick up additional information about the type or property such as labels, help text, supertypes, property cardinality and ranges and so on. While microdata also uses URLs for types and properties, microdata consumers are not permitted to dereference URLs that they do not already recognise.

3.1.2 Tooling Considerations

Applications vary widely in terms of the tooling that they need. A script that runs in a publisher's page needs easy access to data through a DOM API. A crawler that creates a store of data from a set of distributed pages requires a server-side parser and good storage and querying support.

As a consumer, you will be led by the requirements you have for your application and the experience that you have with different technology sets. It's important, however, to also consider the experience and capabilities of the publishers that are providing you with data, and which formats they will find easy to publish given their tooling. You should also consider the ease with which you can provide support tools for the format, such as validators or previewers that make it easy for publishers to tell whether they have published data correctly within their pages.

There are several specifications that can be used to provide standard mechanisms for accessing, manipulating, querying and validating data gleaned from HTML pages. However, you should check what has been implemented in your environment: it may be that there isn't an implementation that follows a standard, but there is one that provides its own API which enables you to do what you need to do.

3.1.2.1 Microdata/Microformats-2 Data Model

Microdata and microformats-2 can be mapped to the same basic (JSON) data model. Processing JSON into native programming structures, in Javascript and other languages, is usually very easy. Vocabularies are usually described in specification prose rather than a formal language.

- microdata DOM API — part of microdata specification (W3C Last Call Working Draft)
- JSON Schema — schema language for JSON (IETF Internet Draft)

3.1.2.2 RDF Data Model

RDFa processors extract an RDF data model and processors can also generate RDF from microdata. There are a number of standards for alternative serialisations of RDF graphs that target different toolchains, formally expressing RDF vocabularies and querying RDF, and drafts in progress for DOM-based manipulation of RDFa content.

- RDF/XML — XML-based serialisation of RDF (W3C Recommendation)
- Turtle — text-based serialisation of RDF (W3C Working Draft)
- JSON-LD — JSON-based serialisation of RDF (Unofficial Draft)
- RDFS — vocabulary description language for RDF (W3C Recommendation)
- OWL — ontology language for RDF (W3C Recommendation)
- SPARQL — query language for RDF (W3C Recommendation)
- SPARQL 1.1 — W3C Working Draft
- RDFa API — W3C Working Draft

3.1.3 Data Model Considerations

Microdata uses a JSON-based data model of a tree of objects which may be identified through a URI, with properties whose values are strings. microformats-2 uses a similar JSON-based data model of a tree of objects, but they do not have identifiers and their property values may be strings, URLs, date/times or structured HTML values. RDFa uses RDF as its data model, which is a graph of objects identified by URLs with properties whose values may be other objects, lists or literal values which can be tagged with a language or any datatype. These different models have different capabilities.

Structured HTML values

Under appropriate conditions, RDFa and microformats will use markup within the content of an element to provide a property value; in microdata values never retain markup. If you wish to consume data that may contain markup — be it structures such as multiple paragraphs, list items, tables, or inline markup such as emphases, links or ruby markup — you will need publishers to use RDFa or microformats to mark up that data. In RDFa, this is done by publishers adding `datatype="rdf:XMLLiteral"` to elements whose markup should be preserved. In microformats, the

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

handling of the content of an element is determined by the property; in microformats-2, those that retain the HTML structure are named with a `e-*` prefix, such as `e-content`.

Language support

Microformats and RDFa use the language of the HTML elements in the page (from the `@lang` attribute) to indicate the language of relevant values. In microdata, the vocabulary has to provide a separate mechanism to indicate a language. If you are consuming information about the same things from pages that use different languages, or anticipate publishers using multiple languages in their pages to describe a particular entity, you can automatically pick up the language of the content of the page if publishers use microformats or RDFa. If you consume microdata, you need to provide specific properties in your vocabulary that publishers can use to indicate the language of the content.

Issue

The handling of language by microdata may change in the future.

3.1.4 Usability Considerations

Publishing data within HTML can be a challenge for publishers, simply because the structure of the data that they publish is not immediately visible within their pages. The publishers you are targeting will have different levels of skill and experience, which may influence your choice of syntax and the way in which you design your vocabulary. If you can, you should try to work closely with a few target publishers to better understand their requirements and constraints. Experimenting with marking up a few of their existing pages will often highlight issues with both syntax and vocabulary.

Some usability issues may be addressed by restricting the set of attributes that you instruct publishers how to use, or by restricting their location to provide more consistency. For example:

- RDFa 1.1 Lite is an authoring profile of RDFa 1.1 that is sufficient for most data publishing
- most microdata markup does not require `@itemid` or `@itemref`
- constraining data markup to the `head` of an HTML document can make it easier to author and protect it from templating changes, although it also runs the risk of getting out of sync with the content of the page, increases repetition, and is hard to use for anything but flat data structures

Profiling microdata and RDFa is useful for documentation, but consumers should still recognise and understand the full set of syntactic constructs described by the standards. This ensures that those publishers who find that they need the more advanced constructs to mark up their pages can do so, and means that publishers can use general-purpose tools and documentation rather than just those that you provide.

3.2 Consuming Pages with Multiple Formats

In attempting to provide information to multiple consumers, publishers may use several formats within a single page. Consumers should ignore data in vocabularies that they do not recognise and only raise errors for unexpected properties in those vocabularies.

Consumers of HTML data may recognise several formats embedded within a given page, and even within the same part of a page. In these cases, consumers should merge from the different formats; in the example above, a consumer should recognise that the data in vEvent, hCalendar and schema.org is about is a single event rather than interpreting it as three events and merge property values so that the event ends up having a single URL rather than several. Different formats may provide information about different aspects of an entity to different levels of fidelity — in the example above, the schema.org RDFa provided extra details about the location of the event t to the vEvent or hCalendar formats — and consumers should seek to use whatever gives them the most detailed information.

3.3 Good Consumption Practice

It is good practice for a consumer to provide tools that help publishers to see how the data within their pages is interpreted by the consumer and that highlight any errors in the markup, such as invalid values or missing required properties.

It is good practice for consumers to ignore markup that uses syntax or vocabularies that they do not understand. Properties and types in unrecognised vocabularies should be ignored by consumers.

The presence of HTML data within a website does not imply that the data can be used without restriction. Publishers may license the information provided through HTML data, for example to restrict it to non-commercial use or to use only with attribution. Legally, consumers must honour licenses and it is good practice for consumers to indicate to publishers which formats they recognise for expressing licensing information within HTML pages, and which licenses they recognise as indicating that the data within the page is consumable. Typical vocabularies for expressing this information are schema.org, rel-license, Creative Commons or Dublin Core.

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

Even when the use of data is unrestricted, it is good practice for consumers to record the source of the information that they use and, when republishing that data, provide metadata about the rights holder, source and license under which the information is available, using the same vocabularies as those listed above.

Working out how much to believe data gathered from the web may be complex. Consumers may use a variety of metrics based on the reliability of the publisher, the quality of the data itself and so on, to determine the extent to which the published data can be trusted. This is particularly important when combining data about the same entity from multiple publishers, where data from the same origin as the entity identifier may be given higher weight. These methods are outside the scope of this document.

4. Vocabulary Authors

Designing vocabularies is a complex craft, and this document does not cover all aspects of how to go about it. There are several existing more general resources for vocabulary creators, such as:

- the microformats process
- SWIG Web Schemas Task Force

4.1 Extending Vocabularies

There are already many vocabularies in existence, particularly for common domains such as people, organisations, events, products, reviews, recipes and so on. Reusing these vocabularies benefits consumers because it saves design time and means they do not have to create supporting tools and materials such as validators, previewers or documentation. It also benefits publishers because it increases the likelihood that the data within their pages can be consumed by other useful tools. It is therefore good practice to extend existing vocabularies rather than creating new ones, where possible.

This section describes some of the issues that vocabulary authors who extend existing vocabularies need to be aware of.

4.1.1 Extending Microformats

Microformats are developed using an iterative process whereby proposals for extensions are brainstormed and eventually either accepted or rejected by the microformats community. It is not appropriate to create unilateral extensions to microformats. On the other hand, publishers should use semantic classes within their HTML, whether or not they are used within current microformats. Evidence of use of semantic classes within HTML pages is one input to the microdata standardisation process.

4.1.2 Extending RDF Vocabularies

RDF vocabularies, which are used within RDFa, use IRIs for types and properties. Any resource in RDFa can be extended by adding new types to the `@typeof` attribute and/or adding new properties from different vocabularies. However, it is not general practice to allow RDF vocabularies themselves to be extended with new types or properties by third parties.

One pattern that is quite common is for one vocabulary to accept a string for a property, such as an address, and for an extension to provide more structure for that property. In this case, a useful pattern is to nest the more structured property inside the textual property within the HTML. For example:

```
<div property="location">
  <address property="http://example.org/address"
          vocab="http://example.org/" typeof="Address">
    <span property="name">The White House</span><br>
    <span property="street">1600 Pennsylvania Avenue NW</span><br>
    <span property="city">Washington</span>,
    <span property="state">DC</span>
    <span property="zip">20500</span>
  </address>
</div>
```

This pattern also works for properties whose values are XML literals; in this case, the XML literal will include the RDFa markup.

4.1.3 Extending Microdata Vocabularies

Microdata items can have both properties that are scoped to the type of the item and properties that have absolute URLs. There are two ways you can extend a type by adding new properties:

- use a property that is an absolute URL

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

- if the vocabulary allows it, use a new short-name property

Third parties who wish to extend an existing type with new properties should check the constraints of the type being extended to work out whether it's possible to use a non-URL property or not. Note that there is always a possibility, if you do use a non-URL property name, that your extension will conflict with an extension made by someone else; properties whose names are absolute URLs do not have this issue but are more verbose when used in markup.

Microdata does not allow items to have multiple types from different vocabularies. Some vocabularies, such as schema.org, may permit third parties to freely extend existing types within that vocabulary. In this case, items should be assigned both the supertype and the extension type within the `@itemtype` attribute. For example, schema.org describes a method of extending its vocabulary that involves identifying an appropriate supertype or superproperty and appending a `/` and then the name of a subtype or subproperty. Schema.org also permits anyone to create additional non-URL properties on these new types. To extend schema.org's types with a type for a member of parliament, a vocabulary author might use the URI `http://schema.org/Person/MP`, and mark up their page with

```
<p itemscope itemtype="http://schema.org/Person http://schema.org/Person/MP">
  <span itemprop="name">David Cameron</span> is the member of parliament
  for <span itemprop="constituency">Witney</span>.
</p>
```

Here, both `http://schema.org/Person` and `http://schema.org/Person/MP` are given as types, and the non-URL `constituency` property is used despite it not being defined within the schema.org vocabulary.

Other microdata vocabularies do not enable third parties to extend the vocabulary. In these cases, third parties should use a URL property to specify the additional type for the item. For compatibility with RDF, we recommend using `http://www.w3.org/1999/02/22-rdf-syntax-ns#type` for this property, and using a full URL for the type. An alternative to the example above that didn't use the schema.org extension mechanism would be:

```
<p itemscope itemtype="http://schema.org/Person">
  <link itemprop="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"
        href="http://gov.example.org/uk/MP">
  <span itemprop="name">David Cameron</span> is the member of parliament
  for <span itemprop="http://gov.example.org/uk/constituency">Witney</span>.
</p>
```

More details about the use and limitations of this technique can be found in section 2.2.1.3 Mixing Vocabularies in Microdata.

The technique described for RDFa above, of nesting a property that contains more structure within a property that has less, can also be used with microdata content.

4.2 Designing Vocabularies

This section looks at the particular requirements of different HTML data syntaxes on vocabularies, and how to create vocabularies that can be used across HTML data syntaxes.

4.2.1 Syntax-Specific Requirements

Each HTML data syntax brings with it a set of constraints on both how vocabularies are designed and their documentation.

4.2.1.1 Microformat Vocabularies

The microformats 2 page describes the constraints on the design of microformat vocabularies, and the microformats process describes additional procedural guidelines on how to create a new microformat.

4.2.1.2 Microdata Vocabularies

Microdata vocabularies must define, within a specification for that vocabulary, processing rules to be followed by consumers of that vocabulary, using the terms given by the microdata specification. These include:

- what types the vocabulary includes
- which types support `@itemid` to provide global identifiers for items
- whether and how two items described using microdata should be considered a single item by a consumer (such as when they have the same `@itemid`) and if so, how two items within an HTML page should be merged
- whether URL values that have the same value as an `@itemid` should be treated the same as if the item had been nested within the page
- which non-URL properties (**defined property names**) are permitted on each of those types, whether there are equivalent URL properties for them, and how properties will be merged if both are used

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

- how many and what kinds of values are allowed for each property, and what consumers should do if there are more or fewer values than required, how the values are parsed, and what happens when the values are of the wrong type
- whether items that are the value of a property must explicitly have a type or if this can be inferred by consumers
- what to do when an item has a property that it should not have
- whether type and property URLs can be dereferenced
- how consumers should recognise items belonging to the vocabulary (whether purely by `@itemtype` or through some other mechanism)

An example of a microdata vocabulary description is available for GoodRelations. There are also example microdata vocabularies within the WHATWG version of the microdata specification.

Microdata does not support the use of the HTML `@lang` attribute to provide language information for textual values; if this is important, a microdata vocabulary must provide a mechanism for supplying a language separately. This can be done by:

- having a property that indicates the language used in the data for the item; this only works if all the data uses the same language
- defining a `LanguageString` type that has properties for both content and language and specifying the use of items of that type as a value for any appropriate property

Microdata does not support structured HTML values. Where these need to be captured, vocabularies can instead use URLs that reference fragments of HTML in the page. For example:

```
<link itemprop="breadcrumb" href="#breadcrumb">
<div id="breadcrumb">
  <a href="category/books.html">Books</a> >
  <a href="category/books-literature.html">Literature & Fiction</a> >
  <a href="category/books-classics">Classics</a>
</div>
```

4.2.1.3 RDFa Vocabularies

RDFa is used to create RDF graphs, so vocabularies used within RDFa should bear in mind the constraints and conventions that commonly apply to RDF vocabularies. These include:

- types should be named using CapitalCamelCase, and properties using lowerCamelCase
- types and properties in the same vocabulary should share a IRI prefix — the vocabulary IRI — which should end in a `#` or a `/`; the local part of a type or property IRI, after this prefix, should be a valid NCName so that it can be used within RDF/XML serialisations
- the IRIs used for types and properties should resolve into documentation and/or (through content negotiation) an RDFS schema or OWL ontology that describes the types and properties

In addition, the authors of vocabularies designed to be used with RDFa should specify whether IRIs and percent-encoded URIs should be treated as equivalent when used for property and type identifiers or values.

More guidelines and patterns for modelling using RDF are available within Linked Data Patterns.

4.2.2 Syntax-Neutral Vocabularies

Syntax-neutral vocabularies must have variants for each syntax that meet the requirements for the syntax as described above, but the capabilities of each variant do not have to be identical.

For example, a syntax-neutral review vocabulary could specify a required `reviewLanguage` property to give the language of a review in microdata, but say that if microformats or RDFa were used, and this were left unspecified, the language would be assumed. Publishers who had content that included multiple languages in the review itself (which couldn't be represented using a property providing a language for the entire review) would be able to use microformats or RDFa to mark up the review.

There are a number of measures that make it easier for vocabularies to be used across syntaxes in ways that make it easier for consumers to combine data whichever syntax is used.

Naming Conventions

Adopt consistent names across syntaxes, even if the naming conventions between the syntaxes differs. For example, microformats uses lowercase-hyphenated-names whereas RDF uses lowerCamelCase; all that is needed is a clear mapping between them. Although microdata allows defined property names to contain any character except `:` and `.`, non-URL properties should have names that are NCNames so that they can be used in microformats and RDFa. Note that microdata's restrictions mean that `.`s should be avoided in these names.

Entity Identity

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

Microformats and microdata have a limited notion of entity identity: entities may have identifiers (in microdata, from the `@itemid` attribute) but these are not used within the data model to combine entities or link them together into graphs. Syntax-neutral vocabularies use the RDF concept of identity whereby entities with the same identifier are the same entity, and references to that entity's identifier serve to create a graph of entities. This should be reflected in the definition of the microdata variant of the vocabulary, which should allow `@itemid` on all items, and specify that consumers should combine and link to items to create a graph.

An example of a syntax-neutral vocabulary is GoodRelations, which can be used in both microdata and RDFa as well as various other syntaxes that are not usually embedded within HTML.

4.2.3 Good Vocabulary Design Practices

It is good practice for vocabulary creators to collaborate with others who are consuming or publishing information in the relevant domains in order to create a vocabulary that can be used widely across an industry.

It is good practice for vocabulary creators to make available a validation tool that enables publishers who use a vocabulary to check that their HTML pages contain data that is valid against that vocabulary.

It is good practice for vocabulary creators to make available test suites that enable implementers to check the behaviour of their implementations. These test suites should cover error handling as well as the correct interpretation of valid data.

A. Acknowledgements

Many thanks to the members of the HTML Data Task Force for their contributions to this document.

B. Multiple Item Types in Microdata

As discussed in section 2.2.1.3 Mixing Vocabularies in Microdata, microdata does not support providing multiple types from different vocabularies to a given item within the `@itemtype` attribute. There are two work-arounds for this, which are discussed here using the example of targetting both schema.org and use the vEvent vocabulary with the original HTML:

```
<a href="nba-miami-philadelphia-game3.html">
  NBA Eastern Conference First Round Playoff Tickets:
  Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1)
</a>

Thu, 04/21/16
8:00 p.m.

<a href="wells-fargo-center.html">
  Wells Fargo Center
</a>
Philadelphia, PA
```

B.1 Mixing Vocabularies using a Type Property

Some vocabularies may define a property through which types from that vocabulary can be assigned to items that are in a different vocabulary. For example, schema.org could define a `http://schema.org/type` property. It could say that the value of `http://schema.org/type` must be the URL for a schema.org type. And further, that if the property `http://schema.org/type` has the value `http://schema.org/Person`, say, then the item will be interpreted exactly as if the `@itemtype` attribute held the value `http://schema.org/Person`.

Note

At time of writing schema.org does not specify a `http://schema.org/type` property, and this explanation is hypothetical.

When using this technique, the types specified in the `@itemtype` attribute are the **primary types** of the item and those specified through the type property are the **secondary types**.

If the schema.org vocabulary also stated that property URLs that begin with `http://schema.org/` must be treated in the same way as equivalent short-name properties on items with a schema.org type, the schema.org vocabulary could be mixed in with an item marked up using vEvent:

```
<div itemscope itemtype="http://microformats.org/profile/hcalendar#vevent">
  <link itemprop="http://schema.org/type" href="http://schema.org/Event">
  <a itemprop="url http://schema.org/url" href="nba-miami-philadelphia-game3.html">
    NBA Eastern Conference First Round Playoff Tickets:
    <span itemprop="summary http://schema.org/name"> Miami Heat at Philadelphia 76ers
    - Game 3 (Home Game 1) </span>
```


DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

```
    </a>

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

    <div itemprop="location">
      <div itemprop="http://schema.org/location"
           itemscope itemtype="http://schema.org/Place">
        <a itemprop="url" href="wells-fargo-center.html">
        Wells Fargo Center
        </a>
        <div itemprop="address" itemscope itemtype="http://schema.org/PostalAddress">
          <span itemprop="addressLocality">Philadelphia</span>,
          <span itemprop="addressRegion">PA</span>
        </div>
      </div>
    </div>
  </div>
```

Note

The vEvent `location` property takes text while the schema.org `location` property takes structured information about the location. These are combined by having an element for the property which requires structured information nested within the property that requires text.

This generates the JSON:

```
{
  "type": [ "http://microformats.org/profile/hcalendar#vevent" ],
  "properties": {
    "http://schema.org/type": [ "http://schema.org/Event" ],
    "url": [ "http://example.com/nba-miami-philadelphia-game3.html" ],
    "http://schema.org/url": [ "http://example.com/nba-miami-philadelphia-game3.html" ],
    "summary": [ " Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1) " ],
    "http://schema.org/name": [ " Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1) " ],
    "dtstart": [ "2016-04-21T20:00" ],
    "http://schema.org/startDate": [ "2016-04-21T20:00" ],
    "location": [
      "\n    \n      \n      Wells Fargo Center\n      \n      \n        Philadelphia,\n        PA\n      \n    \n  "
    ],
    "http://schema.org/location": [{
      "type": [ "http://schema.org/Place" ],
      "properties": {
        "url": [ "http://example.com/wells-fargo-center.html" ],
        "address": [{
          "type": [ "http://schema.org/PostalAddress" ],
          "properties": {
            "addressLocality": [ "Philadelphia" ],
            "addressRegion": [ "PA" ]
          }
        }]
      }
    }]
  }
}
```

The schema.org consumer would ignore the vEvent vocabulary but recognise the use of the `http://schema.org/type` property, and therefore treat this data in the same way as if the JSON were:

```
{
  "type": [ "http://schema.org/Event" ],
  "properties": {
    "url": [ "http://example.com/nba-miami-philadelphia-game3.html" ],
    "name": [ " Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1) " ],
    "startDate": [ "2016-04-21T20:00" ],
    "location": [{
      "type": [ "http://schema.org/Place" ],
      "properties": {
        "url": [ "http://example.com/wells-fargo-center.html" ],
        "address": [{
          "type": [ "http://schema.org/PostalAddress" ],
          "properties": {
            "addressLocality": [ "Philadelphia" ],
            "addressRegion": [ "PA" ]
          }
        }]
      }
    }]
```


DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

```
    }
  }
```

Also note that in this example the `http://schema.org/type` property is only used where necessary, on the item which needs to be marked as an event in both vocabularies. Where possible, the schema.org type for an entity is provided explicitly through the `@itemtype` attribute.

This method of mixing vocabularies requires vocabularies to specify how consumers should recognise items of a particular type. It is recommended that vocabulary authors define an `@itemtype`-equivalent property, and that, for better integration with RDF tools, this property is `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`.

A particular disadvantage of this approach is that there is no support within the microdata API for retrieving items based on the value of a property. In the example above, it would be possible to retrieve the event using:

```
document.getItems('http://microformats.org/profile/hcalendar#vevent')
```

but not through:

```
document.getItems('http://schema.org/Event')
```

Scripts that extract microdata information using the DOM will be faster if they can use the primary types for an item, specified within the `@itemtype` attribute, so you should specify types accessed through scripts within `@itemtype` rather than through a property wherever possible.

B.2 Mixing Vocabularies using Repeated Content

The second method of supporting multiple properties is to have the entity represented by two (or more) microdata items on the page. To enable dragging and dropping the data from these items, they should be nested inside each other. Properties can be set on the outer element using `link` and `meta` elements which are hidden from users, while the visible content of the page is marked up by the inner element.

```
<div itemscope itemtype="http://microformats.org/profile/hcalendar#vevent">
  <link itemprop="url" href="nba-miami-philadelphia-game3.html">
  <meta itemprop="summary"
        content="Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1)">
  <meta itemprop="dtstart" content="2016-04-21T20:00">
  <meta itemprop="location" content="Wells Fargo Center, Philadelphia, PA">
  <div itemscope itemtype="http://schema.org/Event">
    <a itemprop="url" href="nba-miami-philadelphia-game3.html">
      NBA Eastern Conference First Round Playoff Tickets:
      <span itemprop="name"> Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1) </span>
    </a>

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

    <div itemprop="location" itemscope itemtype="http://schema.org/Place">
      <a itemprop="url" href="wells-fargo-center.html">
      Wells Fargo Center
      </a>
      <div itemprop="address" itemscope itemtype="http://schema.org/PostalAddress">
        <span itemprop="addressLocality">Philadelphia</span>,
        <span itemprop="addressRegion">PA</span>
      </div>
    </div>
  </div>
</div>
```

This generates two items:

```
{
  "items": [{
    "type": [ "http://microformats.org/profile/hcalendar#vevent" ],
    "properties": {
      "url": [ "http://example.com/nba-miami-philadelphia-game3.html" ],
      "summary": [ "Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1)" ],
      "dtstart": [ "2016-04-21T20:00" ],
      "location": [ "Wells Fargo Center, Philadelphia, PA" ]
    }
  }, {
    "type": [ "http://schema.org/Event" ],
    "properties": {
      "url": [ "http://example.com/nba-miami-philadelphia-game3.html" ],
      "name": [ " Miami Heat at Philadelphia 76ers - Game 3 (Home Game 1) " ],
      "startDate": [ "2016-04-21T20:00" ],
      "location": [{
```


DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756

```
            "type": [ "http://schema.org/Place" ],
            "properties": {
              "url": [ "http://example.com/wells-fargo-center.html" ],
              "address": [{
                "type": [ "http://schema.org/PostalAddress" ],
                "properties": {
                  "addressLocality": [ "Philadelphia" ],
                  "addressRegion": [ "PA" ]
                }
              }]
            }
          }]
        }
      }]
    }
```

This method does not require any special properties to be defined in the vocabularies used to mark up the page, and the two items are directly assigned the relevant type and are thus accessible to scripts through the `document.getItems()` method.

The disadvantages of this method are that the page contains more items than there are entities (in the above example, two items representing the same event), and it requires repetition of data within the page.

C. References

C.1 Normative references

No normative references.

C.2 Informative references

No informative references.

DoDots Exhibit 2017
Samsung v. DoDots - IPR2023-00756