Microdata and schema.org
Microdata is a simple semantic markup scheme that’s an alternative to RDFa

Developed by WHATWG and supported by major search companies (Google, Microsoft, Yahoo, Yandex)

Like RDFa, it uses HTML tag attributes to host metadata

Vocabularies are controlled and hosted at schema.org
What is WHATWG?

- **Web Hypertext Application Technology Working Group**
  - Community interested in evolving the Web with focus on HTML and Web API development
  - Ian Hickson is a key person, now at Google

- Founded in 2004 by individuals from Apple, Mozilla and Opera after a W3C workshop
  - Concern about W3C's embrace of XHTML

- Current work on **HTML5**

- Developed **Microdata** spec
http://whatwg.org/
• Started by WHATWG as an alternative to XHTML, joined by W3C
  – A W3C candidate recommendation in 2012 (draft)
  – WHATWG will evolve it as a “living standard”

• HTML5 ≈ HTML + CSS + js

• Native support for graphics, video, audio, speech, semantic markup, ...

• Current partial support in major browsers & extensions
HTML taxonomy and status

**HTML5**

Taxonomy & Status on January 20, 2013

- W3C Recommendation
- Proposed Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated

HTML5 & related technologies:

- MathML 3.0
- SVG
- Selectors L1
- Navigation Timing
- Web Open Font
- HTML + RDFa
- RDFa
- Geo Location
- Touch Events
- Animation Timing
- Device Orientation
- Contacts API
- Indexed Database
- Media Capture
- File API
- Drag and Drop
- Web Sockets
- HTML Markup
- Web Workers
- Microdata
- Web Storage
- Web Messaging
- Canvas 2D
- Web SQL
- HTTP Caching
- Messaging API
- Calendar API
- XMLHttpRequest

by Sergey Mavrody  © BY-SA
Microdata

- The microdata effort has two parts:
  - A markup scheme
  - A set of vocabularies/ontologies
- The markup is similar to RDFa in providing ways to identify subjects, types, properties & objects.
  There’s also a standard way to encode microdata as RDFa.
- The sanctioned vocabularies are found at schema.org and include a small number of very useful ones: people, movies, etc.
An example

<div>
<h1>Avatar</h1>
<span>Director: James Cameron (born 1954)</span>
<span>Science fiction</span>
<a href="avatar-trailer.html">Trailer</a>
</div>
An example: itemscope

- An *itemscope* attribute identifies a content *subtree* that is the subject about which we want to say something

```
<div itemscope>
  <h1>Avatar</h1>
  <span>Director: James Cameron (born 1954)</span>
  <span>Science fiction</span>
  <a href="avatar-trailer.html">Trailer</a>
</div>
```
An example: `itemtype`

- An `itemscope` attribute identifies a content `subtree` that is the subject about which we want to say something.
- The `itemtype` attribute specifies the subject’s type.

```html
<div itemscope itemtype="http://schema.org/Movie">
  <h1>Avatar</h1>
  <span>Director: James Cameron (born 1954)</span>
  <span>Science fiction</span>
  <a href="avatar-trailer.html">Trailer</a>
</div>
```
Microdata <-> RDF

http://rdf-translator.appspot.com/

**RDF Translator** is a multi-format conversion tool for structured markup. It provides translations between data formats ranging from RDF/XML to RDFa or Microdata. The service allows for conversions triggered either by URI or by direct text input. Furthermore, it comes with a straightforward REST API for developers.

REST API
This on-line service provides an easily accessible API which allows for a couple of access methods:

1. Request raw code snippet served using the proper media type for the target data format:

   ```
   http://rdf-translator.appspot.com/convert/〈source〉/〈target〉/〈uri〉
   ```

   Examples:
Microdata <-> RDF

http://rdf-translator.appspot.com/
An example: `itemtype`

- An `itemscope` attribute identifies a content `subtree` that is the subject about which we want to say something.
- The `itemtype` attribute specifies the subject’s type.

```html
<div itemscope itemtype="http://schema.org/Movie">
  <h1>Avatar</h1>
  <span>Director: James Cameron (born 1954)</span>
  <span>Science fiction</span>
  <a href="avatar-trailer.html">Trailer</a>
</div>
```

[ ] a schema:Movie.
An example: itemprop

- An *itemscope* attribute identifies a content *subtree* that is the subject about which we want to say something
- The *itemtype* attribute specifies the subject’s type
- An *itemprop* attribute gives a property of that type

```html
<div itemscope itemtype="http://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <span>Director: James Cameron (born 1954)</span>
  <span itemprop="genre">Science fiction</span>
  <a href="avatar-trailer.html" itemprop="trailer">Trailer</a>
</div>
```
An example: itemprop

- An `itemscope` attribute identifies a content `subtree` that is the subject about which we want to say something.
- The `itemtype` attribute specifies the subject's type.
- An `itemprop` attribute gives a property of that type.

```html
<div itemscope itemtype="http://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <span>Director: James Cameron (born 1954) </span>
  <span itemprop="genre">Science fiction</span>
  <a href="avatar-trailer.html" itemprop="trailer">Trailer</a>
</div>
```
An example: embedded items

- An itemprop immediately followed by another itemcope makes the value an object.

```html
<div itemscope itemtype="http://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <div itemprop="director"
    itemscope itemtype="http://schema.org/Person">
    Director: <span itemprop="name">James Cameron</span>
    (born <span itemprop="birthDate">1954</span>)
  </div>
  <span itemprop="genre">Science fiction</span>
  <a href="avatar-trailer.html" itemprop="trailer">Trailer</a>
</div>
```
An example: embedded items

- An itemprop immediately followed by another makes the value an object.

```html
<div itemscope itemtype="http://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <div itemprop="director" itemscope itemtype="http://schema.org/Person">
    Director: <span itemprop="name">James Cameron</span> (born <span itemprop="birthDate">1954</span>)
  </div>
  <span itemprop="genre">Science fiction</span>
  <a href="avatar-trailer.html" itemprop="trailer">Trailer</a>
</div>
```

[ ] a schema:Movie;
schema:director [ a schema:Person;
schema:birthDate "1954";
schema:name "James Cameron" ];
schema:genre "Science fiction";
schema:name "Avatar";
schema:trailer <avatar-trailer.html>.
Full type hierarchy in one file

548 classes, 711 properties (5/4/14)

Data types: Boolean, Date, DateTime, Number (Float, Integer) Text (URL), Time

Objects: Rooted at Thing with two ‘metaclasses’ (Class and Property) and eight subclasses
http://www.schema.org/Recipe

### Thing > CreativeWork > Recipe

A recipe.

<table>
<thead>
<tr>
<th>Property</th>
<th>Expected Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>additionalType</td>
<td>URL</td>
<td>An additional type for the item, typically used for adding more specific types from external vocabularies in microdata syntax. This is a relationship between something and a class that the thing is in. In RDFa syntax, it is better to use the native RDFa syntax – the 'typeof' attribute – for multiple types. Schema.org tools may have only weaker understanding of extra types, in particular those defined externally.</td>
</tr>
<tr>
<td>description</td>
<td>Text</td>
<td>A short description of the item.</td>
</tr>
<tr>
<td>image</td>
<td>URL</td>
<td>URL of an image of the item.</td>
</tr>
<tr>
<td>name</td>
<td>Text</td>
<td>The name of the item.</td>
</tr>
<tr>
<td>url</td>
<td>URL</td>
<td>URL of the item.</td>
</tr>
</tbody>
</table>

### Properties from CreativeWork

<table>
<thead>
<tr>
<th>Property</th>
<th>Expected Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>about</td>
<td>Thing</td>
<td>The subject matter of the content.</td>
</tr>
<tr>
<td>accountablePerson</td>
<td>Person</td>
<td>Specifies the Person that is legally accountable for the CreativeWork.</td>
</tr>
<tr>
<td>aggregateRating</td>
<td>AggregateRating</td>
<td>The overall rating, based on a collection of reviews or ratings, of the item.</td>
</tr>
<tr>
<td>alternativeHeadline</td>
<td>Text</td>
<td>A secondary title of the CreativeWork.</td>
</tr>
<tr>
<td>associatedMedia</td>
<td>MediaObject</td>
<td>The media objects that encode this creative work. This property is a synonym for encodings.</td>
</tr>
<tr>
<td>audience</td>
<td>Audience</td>
<td>The intended audience of the item, i.e. the group for whom the item was created.</td>
</tr>
<tr>
<td>audio</td>
<td>AudioObject</td>
<td>An embedded audio object.</td>
</tr>
<tr>
<td>author</td>
<td>Organization or Person</td>
<td>The author of this content. Please note that author is special in that HTML 5 provides a special mechanism for indicating authorship via the rel tag. That is equivalent to this and may be used interchangeably.</td>
</tr>
<tr>
<td>award</td>
<td>Text</td>
<td>An award won by this person or for this creative work.</td>
</tr>
<tr>
<td>awards</td>
<td>Text</td>
<td>Awards won by this person or for this creative work. (legacy spelling; see singular form, award)</td>
</tr>
<tr>
<td>comment</td>
<td>UserComments</td>
<td>Comments, typically from users, on this CreativeWork.</td>
</tr>
</tbody>
</table>
Microdata as a KR language

- More than RDF, less than RDFS
- Properties have an *expected* type (range)
  - Might be a string
  - A list of types, any of which are OK
- Properties attached $\geq 1$ types (domain)
- Classes can have multiple parents and inherit (properties) from all of them
- No axioms (e.g., disjointness, cardinality, etc.)
- No subPropertyOf like relation
Microdata is intended to work with just one vocabulary – the one at schema.org

Advantages
- Simple, organized, well designed
- Controlled by the schema.org people

Disadvantages: too simple, controlled
- Too simple, narrow, mono-lingual
- Controlled by the schema.org people

Schema.rdfs.org defines mappings between schema.org and popular RDF ontologies
Schema <-> RDF

http://schema.rdf.org

What is Schema.RDFS.org?

In early June 2011, the three big search engines Bing, Google and Yahoo! introduced Schema.org, a collection of terms that webmasters can use to markup their pages to improve the display of search results. This site is a complementary effort by people from the Linked Data community to support Schema.org deployment and usage with a special focus on Linked Data:

- We provide markup examples and tutorials about publishing & consuming data with Schema.org terms.
- We maintain mappings from Web Data vocabularies such as the DBpedia ontology to Schema.org terms.
- We list tools and libraries that are able to consume or produce Schema.org-based data.
- We automatically scrape the Schema.org terms on a daily basis and generate the following formats:

  - RDF/Turtle
  - RDF/XML
  - RDF/NTriples
  - JSON
  - CSV: all classes, all properties

Note that the official OWL version of the terms is directly maintained at Schema.org and independent from the above presented formats. Various tools that use or produce Schema.org terms are already available or in preparation. Stay tuned!
Extending the schema.org ontology

- [http://www.schema.org/docs/extension.html](http://www.schema.org/docs/extension.html)
- You can subclass existing classes
  - Person/Engineer
  - Person/Engineer/ElectricalEngineer
- Subclass existing properties
  - musicGroupMember/leadVocalist
  - musicGroupMember/leadGuitar1
  - musicGroupMember/leadGuitar2
Extension Problems

- Do agreed upon meaning
  - Through axioms supported by the language (e.g., equivalence, disjointness, etc.)
  - No place for documentation (annotations, labels, comments)

- Without a namespace mechanism, your Person/Engineer and mine can be confused and might mean different things
Serialization

- Schema.org has a **data model** and serializations
  - Microdata is the original, native sterilization
  - RDFa is more expressive and works with the RDF stack
  - Everyone agrees that *RDFa Lite* is a good encoding: as simple as Microdata but more expressive
  - JSON-LD is also an accepted encoding

- Search engines look for Microdata and RDFa encodings and are beginning to look for JSON-LD

- Schema.org considers RDFa to be the “canonical machine representation of schema.org”
Conclusions

- Microdata is a good effort by the search companies to use a simple semantic language.
- The semantics is pragmatic:
  - e.g., expected types: A string is accepted where a thing is expected – “some data is better than none”
- The real value is in:
  - the supported vocabularies and
  - their use by Search companies
- => Immediate motivation for using semantic markup