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.

* Web Hypertext Application Technology Working Group
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

- Worked on [HTML5](https://www.w3.org/html), developed [Microdata](https://www.w3.org/Microdata) spec
HTML5

- Started by WHATWG as an alternative to XHTML, joined by W3C
  - HTML5 recommendation, October 2014
  - HTML5.1 recommendation, November 2016
  - WHATWG will evolve it as a “living standard”

- HTML5 ≈ HTML + CSS + js

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

- Current support in major browsers
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. Also a standard way to encode Microdata as RDFa.

Sanctioned vocabularies at [schema.org](http://schema.org) and include a small number of very useful ones: people, movies, events, recipes, etc.
An example

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

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

```html
<div itemprop>
  <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

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 online 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:

http://rdf-translator.appspot.com/
Microdata <-> RDF

Examples: RDFa - Microdata - RDF/XML - N3 - N-Triples - RDF/JSON - JSON-LD

Input Microdata Output N3

@prefix bcalendar: <http://microformats.org/profile/bcalendar#> .
@prefix bcard: <http://microformats.org/profile/bcard#> .
@prefix md: <http://www.w3.org/ns/md#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfa: <http://www.w3.org/2000/01/rdfa-syntax#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema> .
<> rdfa:usesVocabulary schema: .

[] a schema:Movie .

REST API

This online service provides an easy-accessible API which allows for a couple of access methods:
An example: itemtype

- An `itemscope` attribute identifies 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>
```
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>
```

[ ] a schema:Movie ;
  schema:genre "Science fiction" ;
  schema:name "Avatar" ;
  schema:trailer <avatar-trailer.html> .
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 itemprop 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>
```
schema.org vocabulary

- Full type hierarchy in one file
- 590 classes, >700 properties (Nov ‘16)
- **Data types**: Boolean, Date, DateTime, Number, Text, Time
- **Objects**: Rooted at Thing with two ‘metaclasses’ (Class and Property) and eight subclasses
- See [github repo](https://github.com) for examples and code
### 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>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Testing Structured Data in HTML
Perfect Apple Pie

A classic apple pie takes a shortcut with easy Pillsbury® unroll-fill refrigerated pie crust.

Ingredients

Crust
- 1 box Pillsbury™ refrigerated pie crusts, softened as directed on box

Filling*
- 6 cups thinly sliced, peeled apples (8 medium)

Try These Next

- Caramel Apple Pie Cookies
- Mini Apple Pies
- French Cranberry-Apple Pie
- Caramel Apple Pie with Pecans

*Use any variety of quartered, peeled, sliced apples.
Testing Structured Data in HTML
Microdata as a KR language

● More than RDF, less than RDFS
● Properties have an expected type (range)
   – Can be a list of types, any of which are OK
   – Might be a string for many properties ("some data better than none")
● Properties attached ≥ 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
Mixing vocabularies

- 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
Extending schema.org ontology

- **Extensions**: hosted vs. external
  - Hosted: managed & published by schema.org project

- You can subclass existing classes
  - Person/Engineer
  - Person/Engineer/ElectricalEngineer

- Subclass existing properties
  - musicGroupMember/leadVocalist
  - musicGroupMember/leadGuitar1
  - musicGroupMember/leadGuitar2
Extension Problems

- Hard to establish 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
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 an increasingly popular accepted encoding

Search engines look for Microdata, RDFa and JSON-LD

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

- Microdata is an effort by a group of 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