Microdata and schema.org

Basics

- **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/
### HTML5

- 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

### 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](http://schema.org) and include a small number of very useful ones: people, movies, etc.

### HTML taxonomy and status

### An example

```html
<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

```html
<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


Microdata <-> RDF

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>
```

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 itemprop="director">Director: James Cameron (born 1954)</span>
  <span itemprop="genre">Science fiction</span>
  <a itemprop="trailer" href="avatar-trailer.html">Trailer</a>
</div>
```

An example: embedded items

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

```html
<div itemscope itemtype="http://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <div 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 itemprop="trailer" href="avatar-trailer.html">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> (<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
- 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

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
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

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

Schema <-> RDF

http://schema.rdfs.org

Extending the schema.org ontology

- 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 serialization
  - 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