Implementing microdata in your page / the `itemscope`,`itemtype` and `itemprop`attributes

BASIC STEPS

Adding microdata to an HTML page is a really simple task and requires only three attributes: `itemscope`, `itemtype` and`itemprop`.

1 - Define a container element by adding an`itemscope` attribute

First, you need to add an `itemscope` attribute to an HTML element. This will define the "global object" for which we will define properties. This element can be of different types that we will describe later, but for now let us keep looking at the same example we used in previous sections:

```
<section itemscopeitemtype="http://schema.org/Person">
...
</section>
```

We will look at the `itemtype` attribute later. Now that we have defined a global wrapper object/element (a Person in this case), we can add properties inside this element to define the first name, last name, etc.

2 - Specify the vocabulary used for your microdata with the `itemtype` attribute of the container element

HTML5 proposes semantic elements for representing sections, articles, headers, etc, but it does not propose any specific elements or attributes to describe an address, a product, a person, etc.

We need a special vocabulary to represent a person or a physical address. With microdata you can define your own vocabulary or better, reuse one of the existing popular vocabularies, such as these: http://www.schema.org.

Microdata works with properties defined as name/value pairs. The names are defined in the corresponding vocabulary. For example, the vocabulary for representing a Person, available athttp://schema.org/Person defines a set of property names, as illustrated by the following screenshot:

Thing > Person

A person (alive, dead, undead, or fictional).

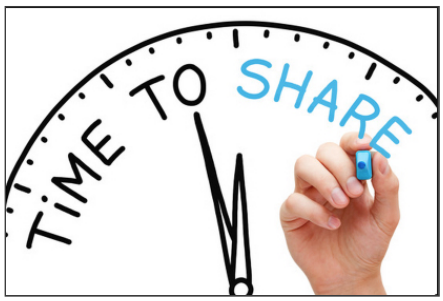
Property	Expected Type	Description
Properties from Thing		
additionalType	URL	An additional type for the item, typically used for adding more specific types from external vocabularies in microdata syntax. This is a relationship between something and a class that the thing is in. In RDFa syntax, it is 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.
description	Text	A short description of the item.
image	URL	URL of an image of the item.
name	Text	The name of the item.
url	URL	URL of the item.
Properties from Person		
additionalName	Text	An additional name for a Person, can be used for a middle name.
address	PostalAddress	Physical address of the item.
affiliation	Organization	An organization that this person is affiliated with. For example, a school/university, a club, or a team.
alumniOf	EducationalOrganization	An educational organizations that the person is an alumni of.

As you can see in this small extract from the vocabulary (also called a "schema"), a Person can have a name (some text), an Address (the type is defined by another vocabulary named PostalAddress), an affiliation (defined by another vocabulary named Organization) and so on.

We notice that one property, such as the address of a Person, may use another vocabulary. Yes, a vocabulary may link to another vocabulary! There is also inheritance between vocabularies! The above screenshot shows that the Person vocabulary inherits from a Thing vocabulary, and the five first properties of the table come from this vocabulary that describes things.

If you are a developer and if you are familiar with object oriented programming, think of properties as class attributes and think of vocabularies as classes.

Vocabularies are meant to be shared



If one of the existing vocabularies available at the schema.org Web site fits your needs, you should reuse it, as the most popular vocabularies are becoming de facto standards and will be taken into account by Web crawlers, browsers, and browser extensions.

However, if you do not find a vocabulary corresponding to your needs, keep in mind that anyone can define a microdata vocabulary and start embedding custom properties in their own Web pages. You need to define a namespace and put a description of your vocabulary in a Web page that has the name of your vocabulary. For example, if you own japaneserobots.com, you may define a vocabulary for describing Mech Warrior robots at http://japaneserobots/MechWarrior in the same way as http://schema.org/Person describes the properties of a person.

3 - Add properties using the `itemprop` attribute in HTML elements inside the container

Basics:

Now that you have defined a container element, you may add properties to the HTML inside:

```
<section itemscopeitemtype="http://schema.org/Person">
    <h1>Contact Information</h1>
    <dl>
        <dt>Name</dt>
        <dd itemprop="name">Michel Buffa</dd>
        <dt>Position</dt>
        <dd><span itemprop="jobTitle">
              Professor/Researcher/Scientist
            </span> for
            <spanitemprop="affiliation">University of Nice,
                  France
            </span>
         </dd>
    </dl>
    <h1>My different online public accounts</h1>
    <ul>
        <li><ahref="http://www.twitter.com/micbuffa"
            itemprop="url">Twitter profile</a></li>
        <li><ahref="http://www.blogger.com/micbuffa"
            itemprop="url">Michel Buffa's blog</a></li>
    </ul>
</section>
```

In this example the container is a `<section>` that corresponds to a Person (we have one clue here: the name of the vocabulary given by the `itemtype` attribute), and each property defined inside this section is identified by the value of the `itemprop`attribute of sub-elements.

The line:

```
<dd itemprop="name">Michel Buffa</dd>
```

...defines a property called "name" that has a value of "Michel Buffa" (the text value between the opening and closing tags of the `<dd>` element)

Nesting microdata items:

As we saw with the Person/Address example at the beginning of this chapter, it is possible to nest microdata items inside one another.

Give an element inside a microdata container its own`itemscope` attribute with the

recommended `itemtype`attribute for indicating the name of the vocabulary used by the nested microdata.

Again, look at the Person/Address example:

```
    ...
    </dl>
    <!-- SURFACE ADDRESS GOES HERE -->
    <dd itemprop="address" itemscope
       itemtype="http://schema.org/PostalAddress">
         <span itemprop="streetAddress">10promenade des anglais</span><br>
         <span itemprop="addressLocality">Nice</span>,
         <span itemprop="addressRegion">Alpesmaritimes, France</span>
11.      <span itemprop="postalCode">06410</span><br>
         <span itemprop="addressCountry"itemscope
               itemtype="http://schema.org/Country">
              <span itemprop="name">France</span>
         </span>
    </dd>
    <h1>My different online publicaccounts</h1>
    ...
```

The properties at lines 8-12 refer to the address nested microdata (they are defined in the Address vocabulary, not the Person vocabulary), and "France" (line 14) is a property that refers to the Country vocabulary.

Several properties with the same name but different values

It is possible to use the same property name several times in one microdata object, but with different values:

```
    ...
     <h1>My different online publicaccounts</h1>
     <ul>
     <li><a href="http://www.twitter.com/micbuffa"itemprop="url">Twitter
          profile</a></li>
     <li><a href="http://www.blogger.com/micbuffa"itemprop="url">Michel
          Buffa's blog</a></li>
    </ul>
```

This will define the fact that Michel Buffa has two online accounts, and the two properties have the name `url`, each with its own value.

It is possible to set more than one property at once, with the same value

Here are some microdata that represent a song. In this example, at line 5 we set two different properties: `genre` and `keywords` with the same value (see the MusicRecording schema definition at http://schema.org/MusicRecording):

```
<div itemscopeitemtype="http://schema.org/MusicRecording">
 <h2>The song I just published</h2>
 <ul>
 <li>Name: <span itemprop="name">Please buy me on itunes, I need money!</span>
</li>
 <li>Band: <span itemprop="genre keywords">Punk, Ska</span></li>
 </ul>
</div>
```

And so on...

Now let's see what elements are compatible with the `itemprop` attribute and where the values of the properties are located, depending on each element type.

THE HTML ELEMENTS COMPATIBLE WITH THE `ITEMPROP` ATTRIBUTE

Extract from : http://www.sencha.com/blog/the-html5-family-microdata-overview/

If the `itemprop` attribute appears on a:

Elements that can be associated with microdata

HTML5 elements	microdata value associated
`<a>`, `<area>`, `<audio>`, `<embed>`, `<iframe>`, `<img>`, `<link>`, `<object>`, `<source>`, or `<video>` element	The data is the url in the element's `href`, `src`, or `data` attribute, as appropriate. For example, an image element inside a container of personal contact information can be recognized as that person's photo and downloaded accordingly.
`<time>` element	The data is the time in the element's `datetime` attribute. This lets you, for example, just say "last week" in your text content but still indicate exact date and time.
`<meta>` element	The data is whatever appears in the content attribute of the `<meta>` element. This is

	used when you need to include some data that isn't actually in the text of your page.
anything else	The data is whatever is in the text of the element.

For example, the value of a property defined in an `<img>`element will be the value of the `src` attribute:

```
<img itemprop="image"src="MichelBuffa.png" alt="A great professor">
```

Or for a `<time>`, it will be the value of the `datetime` attribute:

```
<time itemprop="birthday" datetime="1965-04-16">April 16, 1965</time>
```

Or for an `<a>` element, the value will be the value of the `href`attribute:

```
<a href="http://www.twitter.com/micbuffa"itemprop="url">profile</a>
```

KNOWLEDGE CHECK 1.5.3 (NOT GRADED)

What is the correct schema from schema.org for describing a person's address?

- http://schema.org/PostalAddress
- http://schema.org/Address
- http://schema.org/SurfaceAddress
- http://schema.org/LocalAddress