

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
 - It can also be expressed as JSON-LD
 - Vocabularies are controlled and hosted at [schema.org](#)
- * Web Hypertext Application Technology Working Group

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
 - Also a standard way to encode Microdata as RDFa
- Sanctioned vocabularies at schema.org and include a small number of very useful ones: people, movies, events, recipes, 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

```
<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: 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

[] a schema:Movie .

```
<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

```
<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 use a schema:Movie ;
- The *itemtype* attribute specifies the schema:genre "Science fiction" ;
- An *itemprop* attribute gives a schema:name "Avatar" ;

```
[ ] a schema:Movie ;  
schema:genre "Science fiction" ;  
schema:name "Avatar" ;  
schema:trailer <avatar-trailer.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 *itemscope* makes the value an object

```
<div itemscope itemtype="http://schema.org/Movie">  
  <h1 itemprop="name">Avatar</h1>  
  <div itemprop="director"  
    <strong 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

- An itemprop immediately follows the value an object

```
[ ] 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> .
```

```
<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](#)
- 797 classes, 1457 properties, 14 Data Types as of Nov. 2022
- **Data types:** Boolean, Date, DateTime, Number, Text, Time
- **Objects:** Rooted at Thing with two ‘metaclasses’ (Class and Property) and eight subclasses
- See [github repo](#) for examples & code

Object Hierarchy

- ▶ Action +
- ▶ BioChemEntity +
- ▶ CreativeWork +
- ▶ Event +
- ▶ Intangible +
- ▶ MedicalEntity +
- ▶ Organization +
- ▶ Person +
- ▶ Place +
- ▶ Product +
- Taxon

Datatypes

- ▶ Boolean +
- Date
- DateTime
- ▶ Number +
- ▶ Text +
- Time

Schemas as rdfs and owl?

See the [schema.org developer page](http://schema.org/developer)

The screenshot displays a web browser window with the URL `http://schema.org/`. The browser's address bar shows the path `schema (http://schema.org/) : [/Users/finin/Downloads/schemaorg.owl]`. The browser's tabs include `Active Ontology`, `Entities`, `Classes`, `Object Properties`, `Data Properties`, `Individuals by class`, `OWLviz`, and `ROWLTab`. The main content area is divided into several panels:

- Class hierarchy:** Shows a tree view of classes. The `schema:CreativeWork` class is highlighted, and its subclasses are listed: `schema:Action`, `schema:CreativeWork`, `schema:Event`, `schema:Intangible`, `schema:MedicalEntity`, `schema:Organization`, and `schema:Person`.
- Object property hierarchy:** Shows a list of object properties. The `schema:color` property is highlighted, and its subclasses are listed: `schema:code`, `schema:codeRepository`, `schema:codeSampleType`, `schema:codingSystem`, `schema:colleague`, `schema:colleagues`, `schema:color`, `schema:colorist`, `schema:comment`, `schema:commentCount`, `schema:commentText`, `schema:commentTime`, `schema:competitor`, `schema:composer`, `schema:comprisedOf`, `schema:connectedTo`, and `schema:contactlessPayment`.
- Annotations:** Shows the annotations for the `schema:color` class. The annotations are: `rdfs:label` [language: en] `color`, `rdfs:comment` [language: en] `The color of the product.`, and `rdfs:isDefinedBy` `https://schema.org/color`.
- Characteristics:** Shows a list of characteristics for the `schema:color` class. The characteristics are: `Functional`, `Inverse functional`, `Transitive`, `Symmetric`, `Asymmetric`, `Reflexive`, and `Irreflexive`.
- Description:** Shows the description for the `schema:color` class. The description is: `Equivalent To` `schema:Product`, `SubProperty Of`, `Inverse Of`, `Domains (intersection)` `schema:Product`, `Ranges (intersection)` `schema:Role or schema:Text or schema:URL`, `Disjoint With`, and `SuperProperty Of (Chain)`.

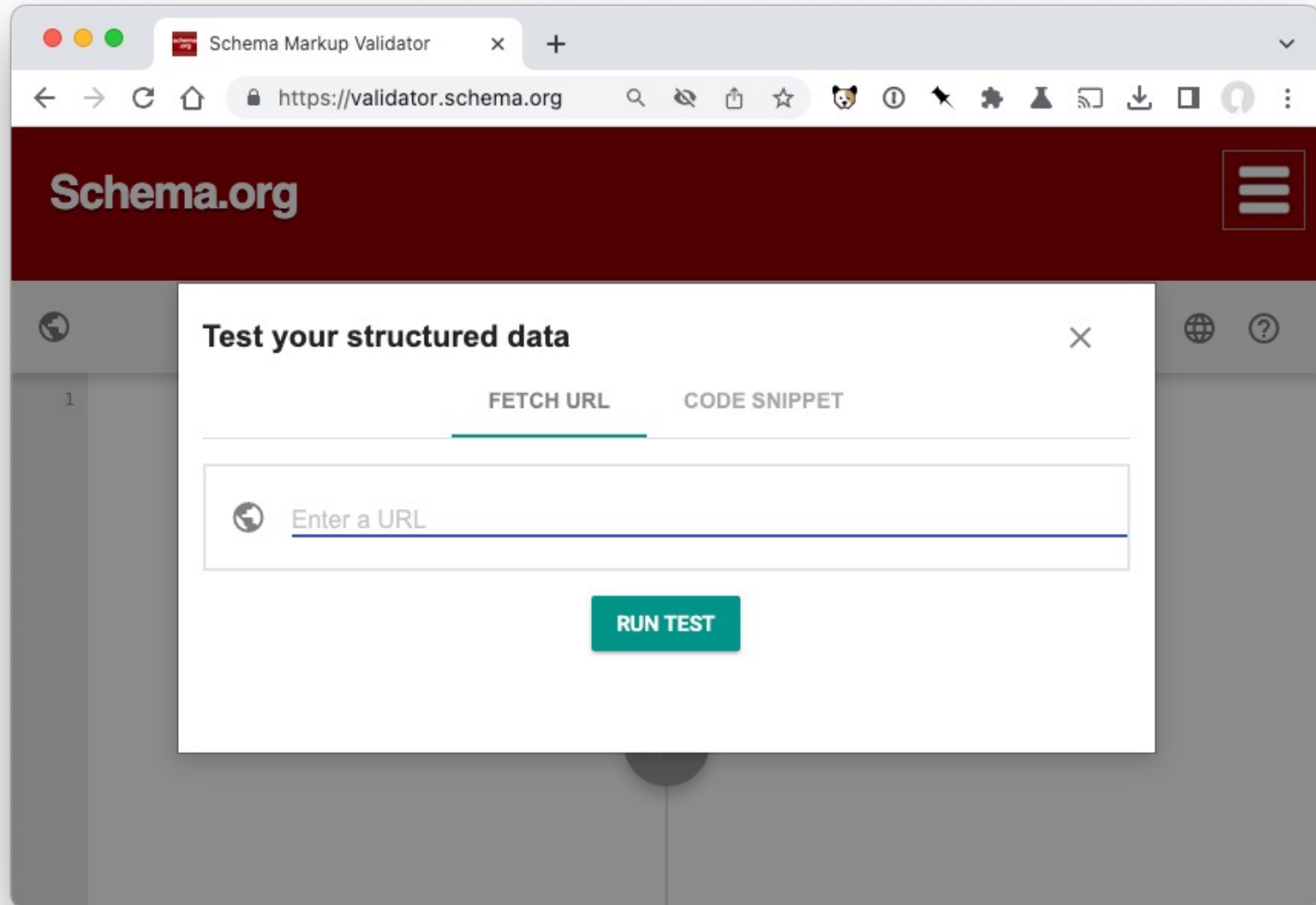
At the bottom of the browser window, there is a footer that reads: `To use the reasoner click Reasoner > Start reasoner` and `Show Inferences` (checked).

<http://www.schema.org/Recipe>

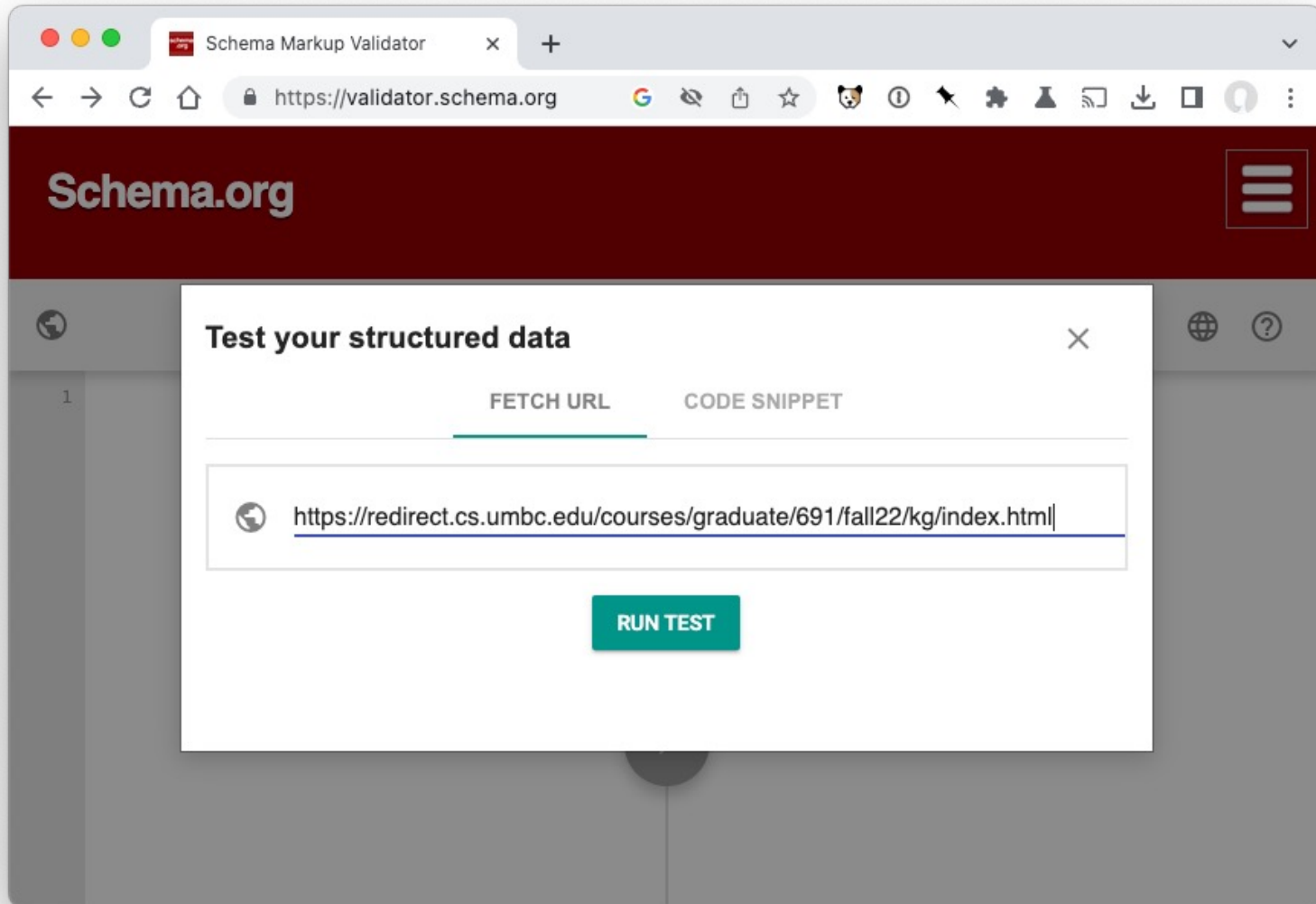
The screenshot shows a web browser window with the address bar displaying "www.schema.org/Recipe". The page header features the "schema.org" logo on the left and a search bar on the right. Below the header, there are navigation links for "Home", "Schemas", and "Documentation". The main content area is titled "Thing > CreativeWork > Recipe" and includes a brief description: "A recipe." Below this, there is a table listing the properties of the Recipe schema, categorized into "Properties from Thing" and "Properties from CreativeWork".

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 CreativeWork		
about	Thing	The subject matter of the content.
accountablePerson	Person	Specifies the Person that is legally accountable for the CreativeWork.
aggregateRating	AggregateRating	The overall rating, based on a collection of reviews or ratings, of the item.
alternativeHeadline	Text	A secondary title of the CreativeWork.
associatedMedia	MediaObject	The media objects that encode this creative work. This property is a synonym for encodings.
audience	Audience	The intended audience of the item, i.e. the group for whom the item was created.
audio	AudioObject	An embedded audio object.
author	Organization or Person	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.
award	Text	An award won by this person or for this creative work.
awards	Text	Awards won by this person or for this creative work. (legacy spelling; see singular form, award)
comment	UserComments	Comments, typically from users, on this CreativeWork.

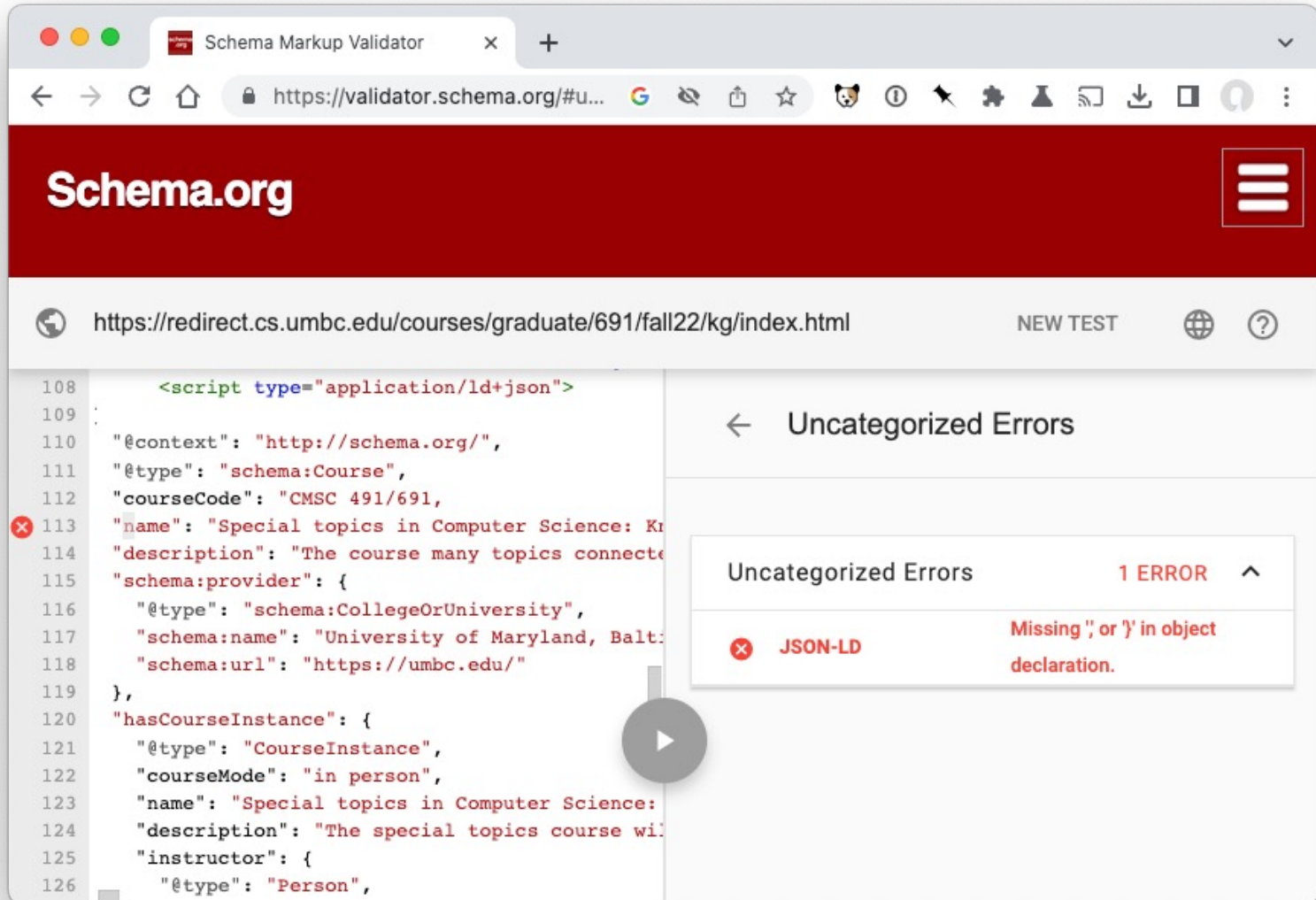
Google's Schema Markup Validator (1)



Google's Schema Markup Validator (2)



Google's Schema Markup Validator (3)



The screenshot shows the Schema Markup Validator interface. The browser address bar displays the URL `https://validator.schema.org/#u...`. The page title is "Schema.org". The URL being validated is `https://redirect.cs.umbc.edu/courses/graduate/691/fall22/kg/index.html`. The interface shows a code editor on the left with the following JSON-LD snippet:

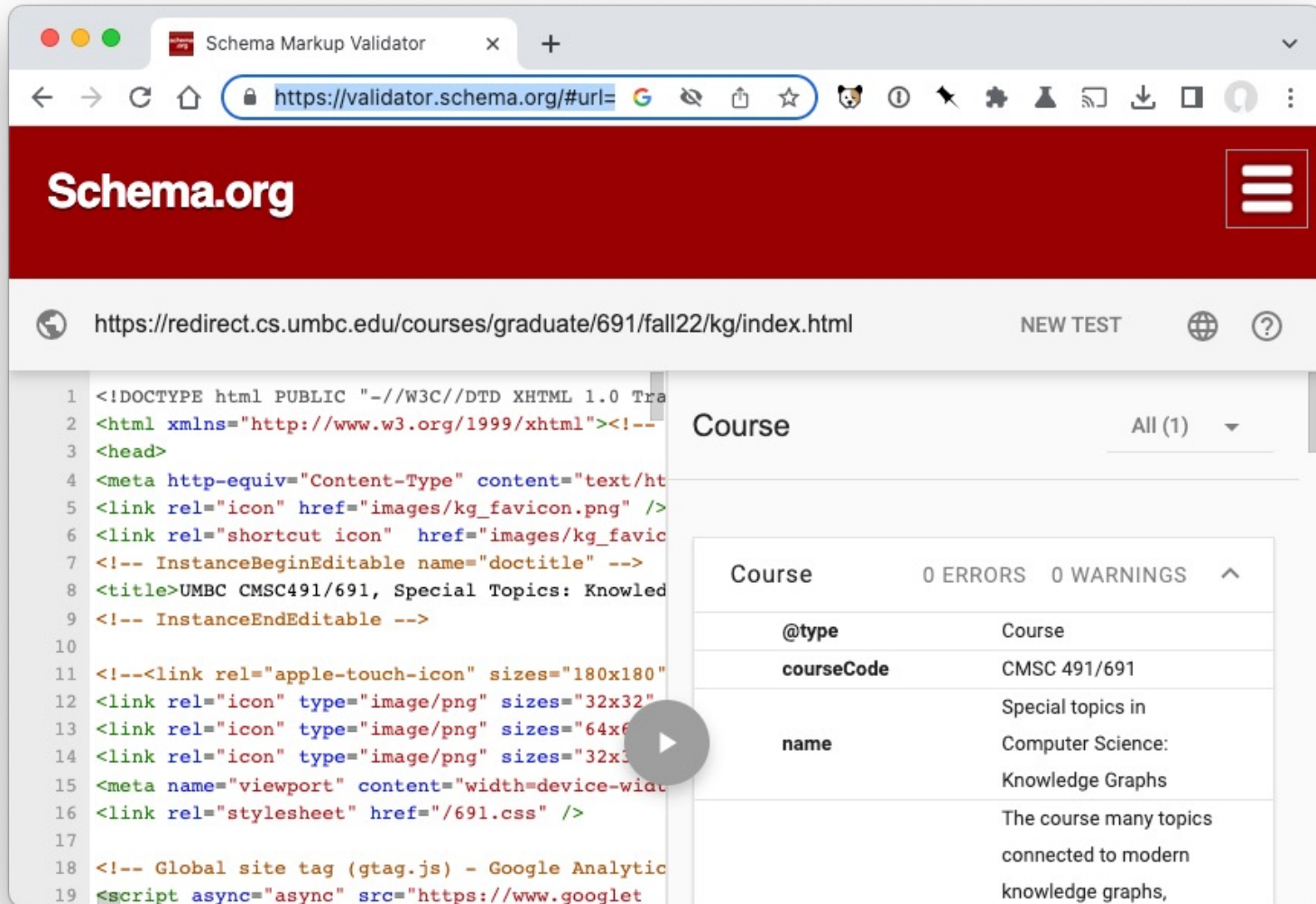
```
108 <script type="application/ld+json">
109 :
110 "@context": "http://schema.org/",
111 "@type": "schema:Course",
112 "courseCode": "CMSC 491/691",
113 "name": "Special topics in Computer Science: K
114 "description": "The course many topics connecte
115 "schema:provider": {
116   "@type": "schema:CollegeOrUniversity",
117   "schema:name": "University of Maryland, Balt
118   "schema:url": "https://umbc.edu/"
119 },
120 "hasCourseInstance": {
121   "@type": "CourseInstance",
122   "courseMode": "in person",
123   "name": "Special topics in Computer Science:
124   "description": "The special topics course wi
125   "instructor": {
126     "@type": "Person",
```

The error panel on the right, titled "Uncategorized Errors", shows one error:

Uncategorized Errors		1 ERROR
JSON-LD	Missing ',' or '}' in object declaration.	

A play button icon is visible over the code editor.

Google's Schema Markup Validator (4)



The screenshot shows the Schema Markup Validator interface. The browser address bar displays the URL `https://validator.schema.org/#url=https://redirect.cs.umbc.edu/courses/graduate/691/fall22/kg/index.html`. The page title is "Schema.org". The URL being validated is `https://redirect.cs.umbc.edu/courses/graduate/691/fall22/kg/index.html`. The left pane shows the HTML source code, and the right pane displays the detected schema type and its details.

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml"><!--
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <link rel="icon" href="images/kg_favicon.png" />
6 <link rel="shortcut icon" href="images/kg_favicon.png" />
7 <!-- InstanceBeginEditable name="doctitle" -->
8 <title>UMBC CMSC491/691, Special Topics: Knowledge Graphs</title>
9 <!-- InstanceEndEditable -->
10
11 <!--<link rel="apple-touch-icon" sizes="180x180" href="images/apple-touch-icon.png" />
12 <link rel="icon" type="image/png" sizes="32x32" href="images/favicon-32x32.png" />
13 <link rel="icon" type="image/png" sizes="64x64" href="images/favicon-64x64.png" />
14 <link rel="icon" type="image/png" sizes="32x32" href="images/favicon-32x32.png" />
15 <meta name="viewport" content="width=device-width, height=device-height, initial-scale=1, maximum-scale=1" />
16 <link rel="stylesheet" href="/691.css" />
17
18 <!-- Global site tag (gtag.js) - Google Analytics -->
19 <script async="async" src="https://www.google-analytics.com/gtag.js"></script>
```

Course All (1) ▾

Course	
@type	Course
courseCode	CMSC 491/691
name	Special topics in Computer Science: Knowledge Graphs
	The course many topics connected to modern knowledge graphs,



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 relation like subPropertyOf

Mixing vocabularies

- Microdata is intended to work with just one vocabulary: the one at schema.org
- Advantages: simple and controlled
 - Simple, organized, well designed
 - Controlled by the schema.org group
- Disadvantages: too simple, too 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

Hosted Extensions

- auto.schema.org
- bib.schema.org
- health-lifesci.schema.org
- iot.schema.org
- meta.schema.org
- pending.schema.org

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)
- With no namespace mechanism, your Person/Engineer and mine can be confused and might mean different things
 - Is a Computer Scientist an engineer?
- Extensions not generally adopted by schema.org

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 an increasingly popular accepted encoding
- Search engines look for all of these, e.g., Microdata, RDFa and JSON-LD
- Schema.org considers RDFa to be the “canonical machine representation of schema.org”
- But Google recommends using JSON-LD

Conclusions

- Microdata is an effort by search companies to use a simple, controlled semantic language
 - Its 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
 - Supported vocabularies and
 - their use by Search companies
- ⇒ Immediate motivation for using semantic markup