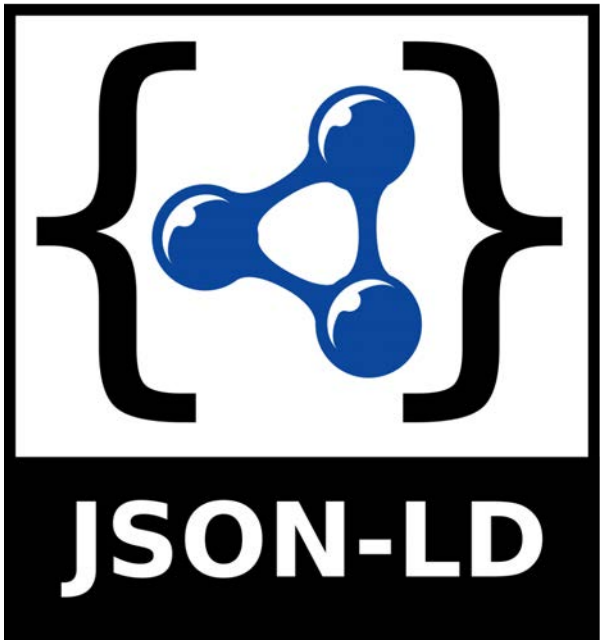
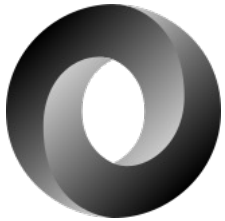
JSON-LD



JSON for Linked Data:
a standard for serializing RDF using JSON

JSON as an XML Alternative



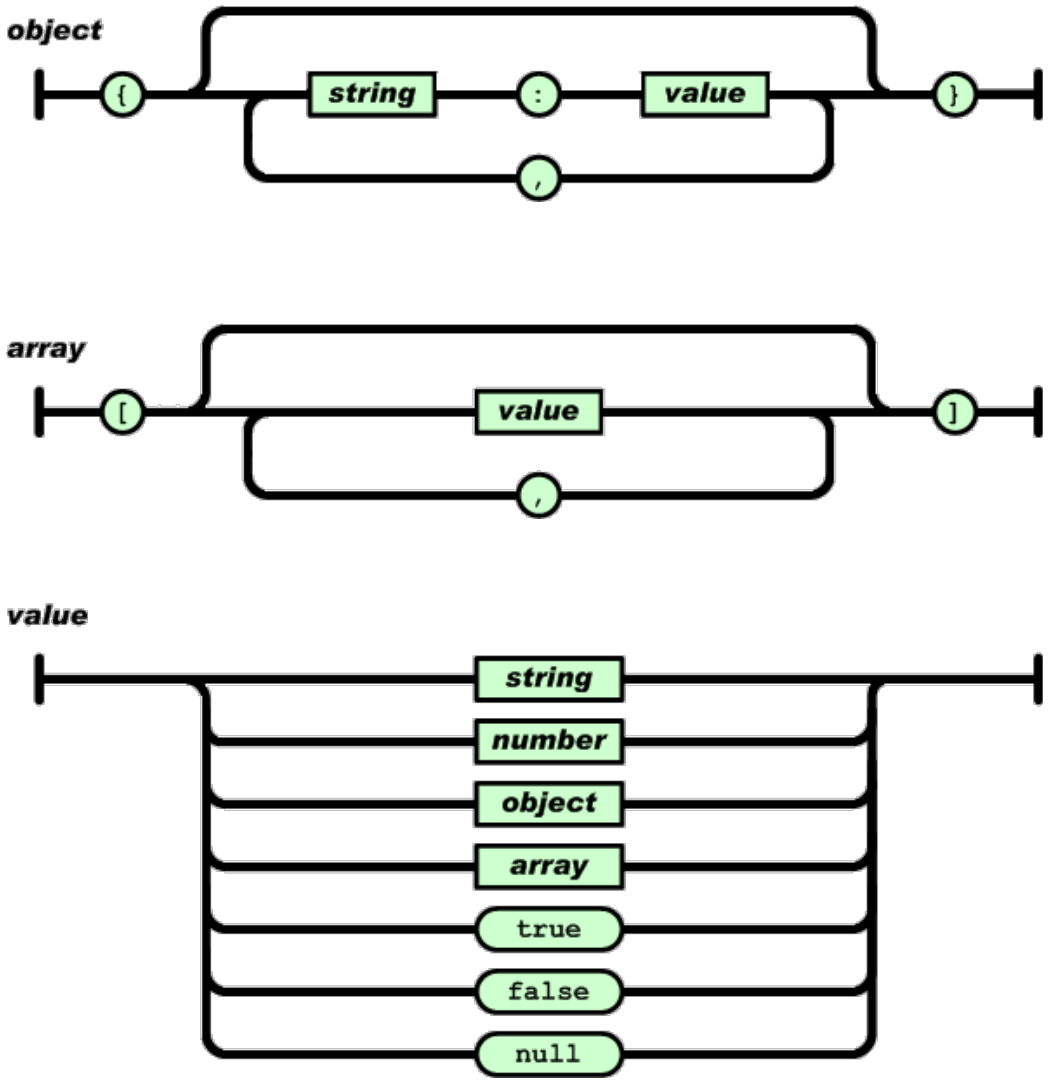
- Light-weight XML alternative for data-interchange
- JSON = JavaScript Object Notation
  - It's really language independent
  - Most programming languages can easily read it and instantiate objects
- Defined in RFC 4627
- Started gaining traction ~2006, now widely used
- http://json.org/ has more information

Example

```
{"firstName": "John",
 "lastName" : "Smith",
 "age"         : 25,
 "address"   :
   {"streetAdr” : "21 2nd Street",
    "city"        : "New York",
    "state"       : "NY",
    ”zip"         : "10021"},
 "phoneNumber":
   [{"type"  : "home",
    "number": "212 555-1234"},
   {"type"  : "fax",
    "number” : "646 555-4567"}]
}
```

- This is a JSON object with five key-value pairs
- Objects are wrapped by curly braces
- There are no object IDs
- Keys are strings
- Values are numbers, strings, objects or arrays
- Arrays are wrapped by square brackets

The BNF is simple



Evaluation

- JSON is simpler than XML and more compact
  - No closing tags, but after compressing XML and JSON the difference is not so great
  - XML parsing is hard because of its complexity
- JSON has a better fit for OO systems than XML, but not as extensible
- Preferred for simple data exchange by many
- MongoDB: 'NoSQL' database for JSON objects
- ElasticSearch: Lucene-based IR system using JSON to represent documents

Dict to Graph (1)

- JSON objects: like key-value stores where the values can be atomic, lists or JSON objects
- These map to Python simple types, lists and dictionaries

```
>>> j_string = """{"firstName":"John", "lastName":"Smith" ... }"""
>>> j_obj = json.loads(j_string)
>>> j_obj
{'firstName':'John','lastName':'Smith','age':25,'address':{'streetAdr':'21
2ndStreet','city':'NewYork','state':'NY','zip':'10021'},'phoneNumber':[{'typ
e':'home','number':'212-555-1234'},{'type':'fax','number':'646-555-
4567'}]}
>>> j_obj['phoneNumber'][0]['number']
'212-555-1234'
```

Dict to Graph (2)

Using JSON for knowledge graphs requires us to develop conventions to:

- Represent a general **graph structure** with this **tree-oriented** data structure
- Encode aspects for a RDF knowledge graph, like namespaces, prefixes, distinguishing URI representing objects from those representing web pages

JSON-LD Status

- JSON-LD: 2014 W3C recommendation for representing RDF **data** as JSON objects
- Current version: JSON-LD 1.1 (2020)
- Google, Bing, and Yandex look for embedded JSON-LD data in web pages
  - and use the information they understand, e.g., statements using **schema.org** terms
  - Google: "Google recommends using JSON-LD for structured data whenever possible."

JSON-LD => RDF

```
{"@context": {
   "name": "http://xmlns.com/foaf/0.1/name",
   "homepage": {
     "@id": "http://xmlns.com/foaf/0.1/workplaceHomepage",
     "@type": "@id”  },
   "Person": http://xmlns.com/foaf/0.1/Person   },
 "@id": "http://me.markus-lanthaler.com",
 "@type": "Person",
 "name": "Markus Lanthaler",
 "homepage": http://www.tugraz.at/  }
```

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
<http://me.markus-lanthaler.com> a foaf:Person ;
    foaf:name "Markus Lanthaler"^^xsd:string ;
    foaf:workplaceHomepage <http://www.tugraz.at/> .
```

In the beginning


```
{
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

A bit better

```
{
  "http://schema.org/name": "Manu Sporny",
  "http://schema.org/url": { "@id": "http://manu.sporny.org/" }
  "http://schema.org/image":
       { "@id": "http://manu.sporny.org/images/manu.png" }
}
```

- The '@id' keyword means 'This value is an identifier that is an IRI’
- i.e., it’s not just a reference to a web page

Define a context

A context lets you define things that apply to the entire JSON object, such as full versions of some terms and (we will see) namespace prefixes and other properties

```
{ "@context":
  {
    "name": "http://schema.org/name",  % [1]
    "image": {
      "@id": "http://schema.org/image",  % [2]
      "@type": "@id"                     % [3]
    },
    "homepage": {
      "@id": "http://schema.org/url",    % [4]
      "@type": "@id"                     % [5]
    } } }
```

[1] means 'name' is short for 'http://schema.org/name'

[2] means 'image' is short for 'http://schema.org/image'

[3] means a string value associated with 'image' should be interpreted as an identifier that is an IRI

[4] means 'homepage' short for 'http://schema.org/url'

[5] means string value associated with 'homepage' to be interpreted as an identifier that is an IRI

Reference an external context

A context can be specified by a URL that points to a JSON object

```
{
  "@context": "http://json-ld.org/contexts/person.jsonld",
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

More typically: add context inline

```
{"@context":
    {
      "name": "http://schema.org/name",
      "image": {
          "@id": "http://schema.org/image",
          "@type": "@id"
           },
      "homepage": {
       "@id": "http://schema.org/url",
       "@type": "@id"
       }
     },
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

Making assertions about things

The **"@id"** JSON property means value is (1) a reference to an RDF object, not a literal, and (2) the subject of all other property/values pairs in this object. @type is rdf:type

```
{"@context": {
    …
   "Restaurant": "http://schema.org/Restaurant",
   "Brewery": "http://schema.org/Brewery"
  }
  "@id": "http://example.org/places#BrewEats",
  "@type": [ "Restaurant", "Brewery" ],
  …}
```

Adding a default vocabulary

The **"@vocab"** JSON property means any unqualified properties (e.g., “name”) or objects (e.g., “Restaurant”) come from its value’s vocabulary

```
{"@context": {
   "@vocab": "http://schema.org/"
 }
 "@id": "http://example.org/places#BrewEats",
 "@type": "Restaurant",
 "name": "Brew Eats"
 ...}
```

Mixing vocabularies

```
{"@context":
  { "xsd": "http://www.w3.org/2001/XMLSchema#",
    "foaf": "http://xmlns.com/foaf/0.1/",
    "foaf:homepage": { "@type": "@id" },
    "picture": { "@id": "foaf:depiction", "@type": "@id" }
 },
  "@id": "http://me.markus-lanthaler.com/",
  "@type": "foaf:Person",
  "foaf:name": "Markus Lanthaler",
  "foaf:homepage": "http://www.markus-lanthaler.com/",
  "picture": "http://twitter.com/account/profile_image/mlanthaler"
}
```

Mixing vocabularies

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
<http://me.markus-lanthaler.com/> a foaf:Person ;
  foaf:depiction <http://twitter.com/account/profile_image/mlanthaler> ;
  foaf:homepage <http://www.markus-lanthaler.com/> ;
  foaf:name "Markus Lanthaler"^^xsd:string .
```

```
{"@context":
 { "xsd": "http://www.w3.org/2001/XMLSchema#",
   "foaf": "http://xmlns.com/foaf/0.1/",
   "foaf:homepage": { "@type": "@id" },
   "picture": { "@id": "foaf:depiction", "@type": "@id" }
 },
 "@id": "http://me.markus-lanthaler.com/",
 "@type": "foaf:Person",
 "foaf:name": "Markus Lanthaler",
 "foaf:homepage": "http://www.markus-lanthaler.com/",
 "picture": http://twitter.com/account/profile_image/mlanthaler }
```

Embedding other objects

```
{
  ...
  "name": "Manu Sporny",
  "foaf:knows":
  {
    "@type": "Person",
    "name": "Gregg Kellogg",
  }
...
}
```

Produces a blank node

Search Engines looks for JSON-LD

- Google, Bing and Yandex all looks for and use JSON-LD markup
- Only schema.org vocabulary is “understood” by ther search engines
- Put a JSON-LD object in head or body of web page wrapped with script tags:

```
<script type="application/ld+json">
{...}
</script>
```


UMBC ebiquity research gr
ebiquity.umbc.edu
UMBC ebiquity
search UMBC ebiquity research group
GO
g+1 26
US RESEARCH PEOPLE PUBLICATIONS NEWS PHOTOS EVENTS BLOG TAGS INTERNAL MORE
ebiquity
group
You are on the UMBC subnet and probably at school. Go outside! It's a beautiful day...
84°F
eBiquity Blog
do not be a gl***hole, use face-block.me!
google mooc: making sense of data
stardog unleashed: md
Latest Paper Additions
usm community cloud: vcl-based academic cloud
cloud data management policies: security and privacy checklist
Latest eBiquity News
ebiquity student karuna joshi's ibm ph.d. fellowship renewed for 2012-13
ebiquity student karuna joshi receives ibm ph.d. fellowship
Latest eBiquity Events
prob: a tool for tracking provenance and reproducibility of big data experiments
tutorials by center for hybrid multicore productivity research students

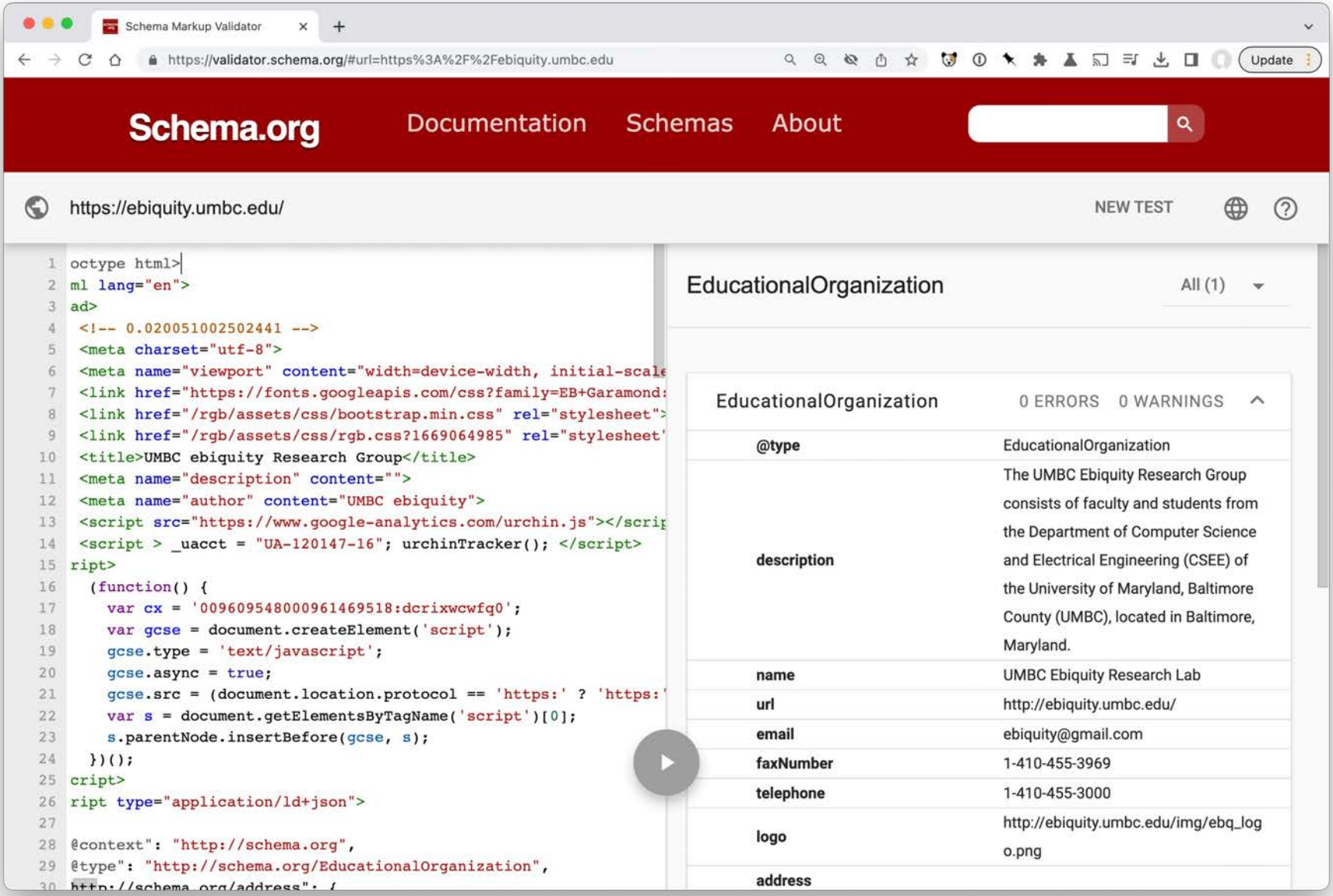
view-source:ebiquity.umbc ×

view-source:ebiquity.um...

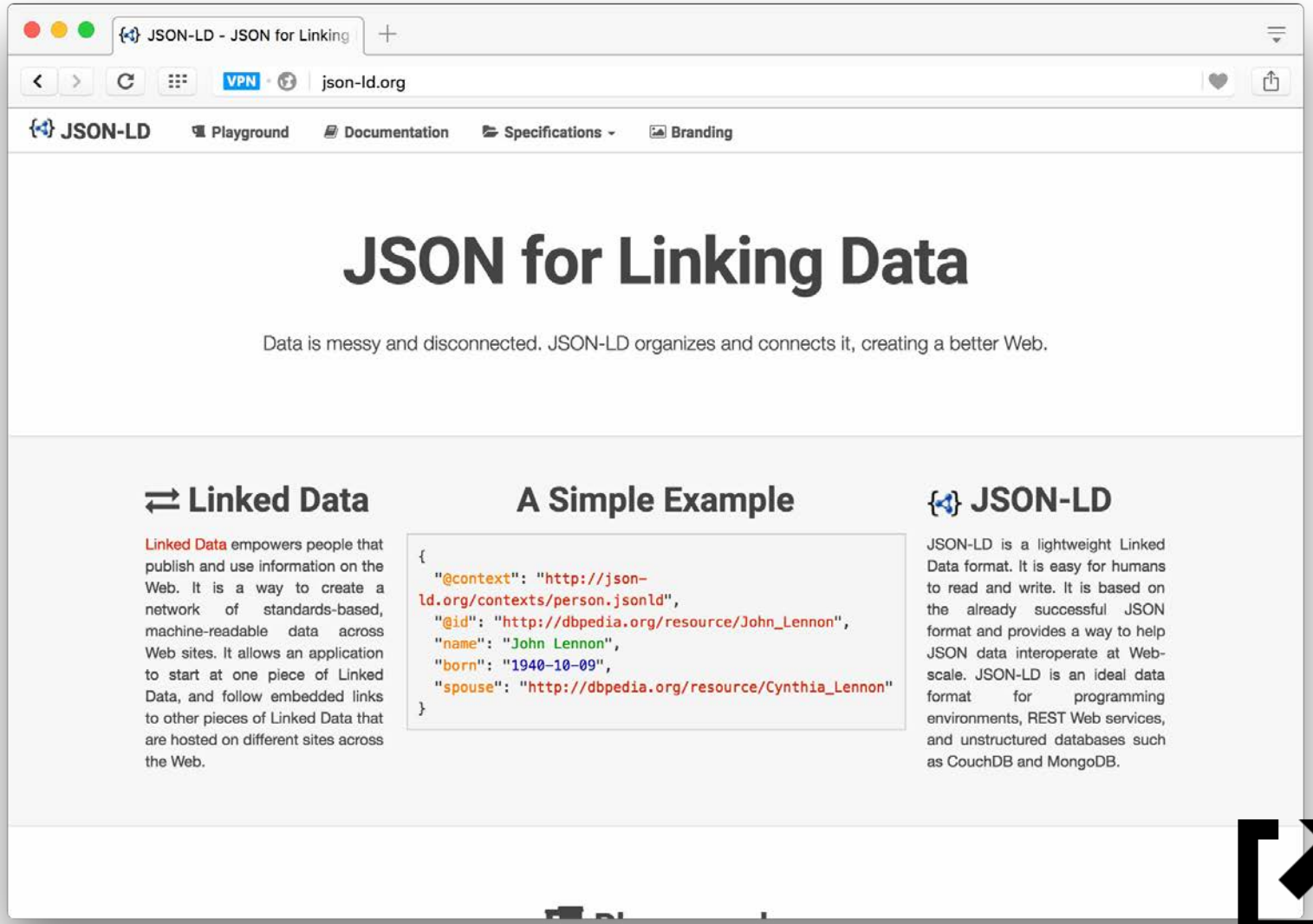
```
<script type="text/javascript" src="https://apis.google.com/js/plusone.js"></script>
  <script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "http://schema.org/EducationalOrganization",
  "http://schema.org/address": {
    "@type": "http://schema.org/PostalAddress",
    "http://schema.org/addressCountry": "USA",
    "http://schema.org/addressLocality": "Baltimore",
    "http://schema.org/addressRegion": "Maryland",
    "http://schema.org/postalCode": "21250",
    "http://schema.org/streetAddress": "University of Maryland, Baltimore County, 1000 Hilltop Circle"
    },
  "http://schema.org/description": "The UMBC Ebiquity Research Group consists of faculty and students from the Department of Computer Science and Electrical Engineering (CSEE) of the University of Maryland, Baltimore County (UMBC), located in Baltimore, Maryland.",
  "http://schema.org/name": "UMBC Ebiquity Research Lab",
  "http://schema.org/url": "http://ebiquity.umbc.edu",
  "email": "ebiquity@gmail.com",
  "faxNumber": "1-410-455-3969",
  "telephone": "1-410-455-3000",
  "logo": {"@id": "http://ebiquity.umbc.edu/img/ebq_logo.png" },
  "member": [
    {
      "@type": "Person",
      "email": "mailto:finin@umbc.edu",
      "image": "http://umbc.edu/~finin/images/headshot.jpg",
      "jobTitle": "Professor",
      "name": "Tim Finin",
      "telephone": "1-410-455-3522",
      "url": "http://umbc.edu/~finin/"
      }
  ]
}
</script>

 </head>
```

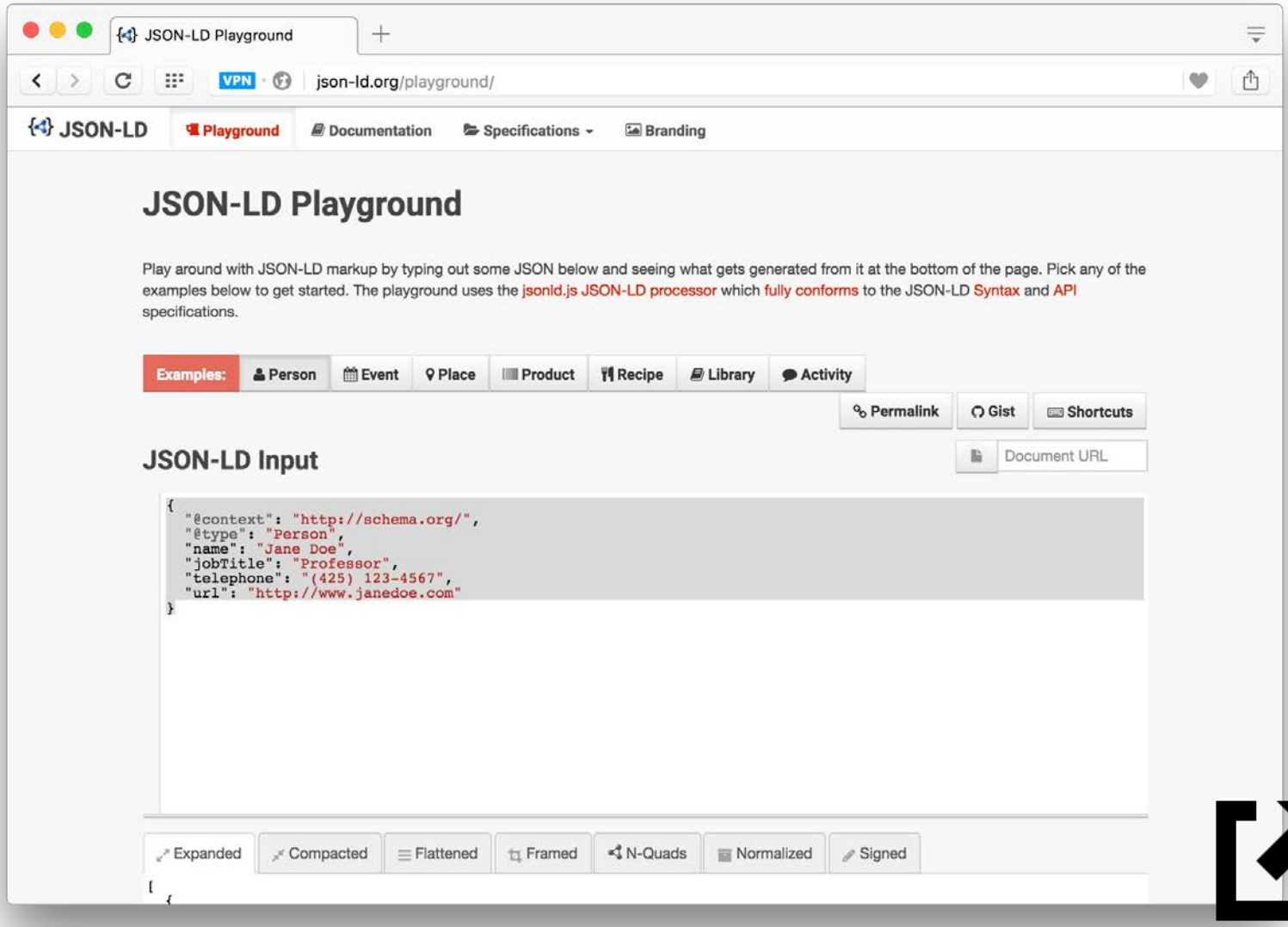
Google's Schema.org Validator



http://json-ld.org/



JSON-LD Playground



Conclusion

- JSON-LD is a good solution to putting blocks of semantic data on web pages
- Aimed at publishing linked data, not ontologies, i.e., ABOX not TBOX
- Tools available for extracting RDF triples
- Search engines look for and use JSON-LD that use vocabularies they understand (i.e., schema.org)