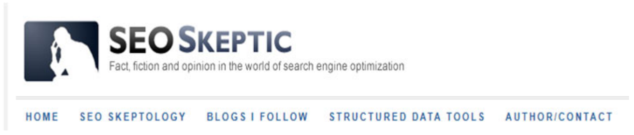
JSON-LD & SEO

Search Engine Optimization

From: www.seoskeptic.com


SEO SKEPTIC
Fact, fiction and opinion in the world of search engine optimization

HOME SEO SKEPTOLOGY BLOGS I FOLLOW STRUCTURED DATA TOOLS AUTHOR/CONTACT

- Semantic SEO is the nascent art of optimizing web sites and other web-based resources for semantic search. But, strictly speaking, it's unnecessary to speak of "semantic SEO" or "semantic search" because the reality of contemporary search engines have made the qualifier redundant.
- Semantic web technologies are now intrinsic to the way modern search engines work, and organic search marketing strategies need to address this reality.

Three (competing) technologies for Adding Semantic Markup to web pages

W3C RDFa

SEO MICRODATA

Semantic Web

JSON-LD

JSON-LD (encodes subj-pred-obj in a json format)

Google provides guidelines to what it expects from web sites marked up with JSON-LD.

Simplicity

No extra processors or software libraries are necessary to use JSON-LD. The language provides developers with a very easy learning curve. Developers only need to know JSON and two keywords : **@context** and **@id** to use the basic functionality in JSON-LD.

Compatibility

A JSON-LD document is always a valid JSON document. This ensures that all of the standard JSON libraries work with JSON-LD .

Expressiveness

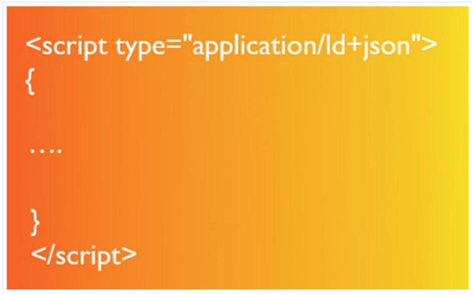
The syntax serializes directed graphs. This ensures that almost every real world data model can be expressed.

Terseness

The JSON-LD syntax is very terse and human readable, requiring as little effort as possible from the developer. Knowledge of RDF/Ne not needed – can transform into RDF/N3

JSON-LD vs RDFa

- No <div> or <span> or any of that stuff
- JSON-LD lives in a <script> tag within the <head> tag of HTML
- It's just a Javascript JSON Object inside <script>
- The JSON objects holds semantic information about the web page just the way RDFa does.
- The <body> of the web page is not modified.



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

….

}
</script>
```

FIRST – You need to understand the rules of JSON

JSON – a standard

JSON is built on two structures:

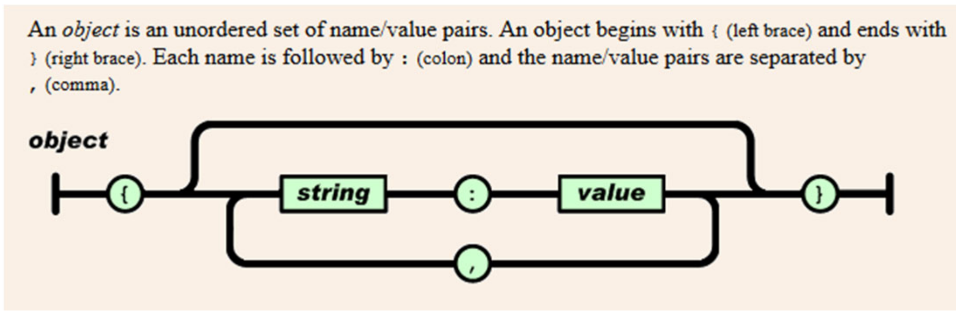
OBJECT: A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array. {"name" : "fred", "age": 25}

JSON object

ARRAY: An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.
[{"name" : "fred", "age": 25}, 22, "foo"]

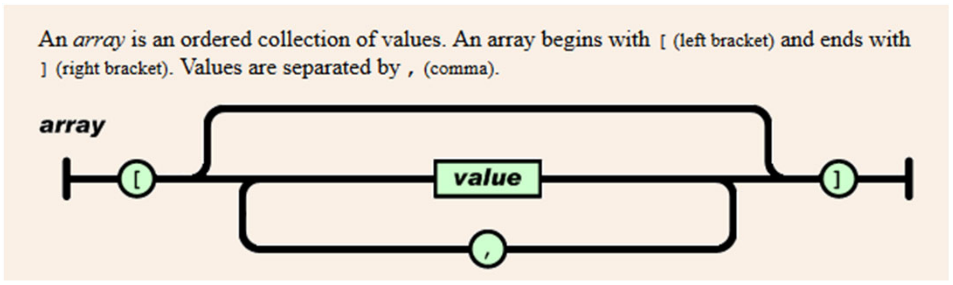
JSON Array

JSON object


An *object* is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).

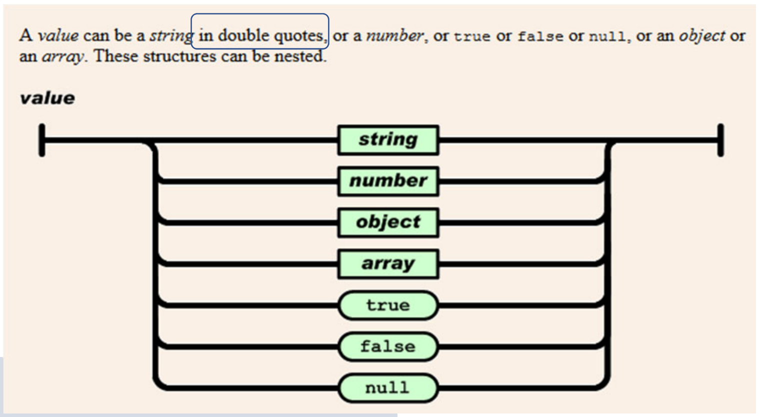
{"name": "rollo",   "age": 22 }

JSON Array


An *array* is an ordered collection of values. An array begins with `[` (left bracket) and ends with `]` (right bracket). Values are separated by `,` (comma).

```
[
{"name": "rollo", "age": 22 },
{"name": "zippy", "age": 21}
]
```

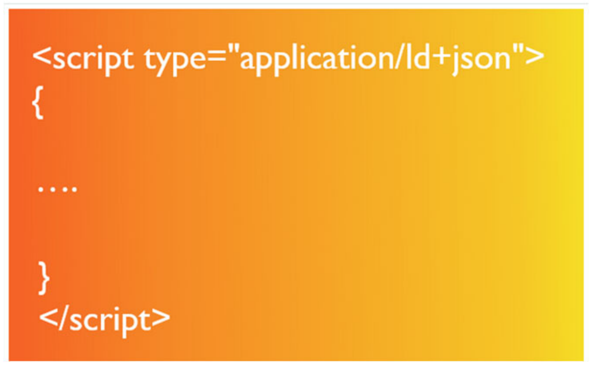
Value


A *value* can be a *string* in double quotes, or a *number*, or `true` or `false` or `null`, or an *object* or an *array*. These structures can be nested.

```
{"name": "rollo",
"age": 22,
"address"  {"city": "memphis", "zip": 21203},
"friends": ["pinky", "foobar", 12, true, {"a":1} ] ,
"homebase": null ,
"rulerOfWorld": false,
"@@@" : 199
}
```

Why JSON-LD ??

- It's a pain to do semantic markup of text with RDFa where you (the developer) OR a program, scans the text for keywords and wraps good ones in <div> or <span> using property=



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

 ….

 }
 </script>
```

Jump in to JSON-LD

```
{
 "name": "Barack Obama",
 "givenName": "Barack",
 "familyName": "Obama",
 "jobTitle": "44th President of the United States"
}
```

JSON-LD sees this as regular JSON – no semantic properties.

Add a @context property to specify a namespace.

All the properties are in ONE namespace: http://schema.org

```
{
 "@context": "http://schema.org",
 "name": "Barack Obama",
 "givenName": "Barack",
 "familyName": "Obama",
 "jobTitle": "44th President of the United States"
}
```

```
_:b0 schema:familyName "Obama" .
_:b0 schema:givenName "Barack" .
_:b0 schema:jobTitle "44th President of the United States" .
_:b0 schema:name "Barack Obama" .
```

MULTIPLE Namespaces with prefixes

```
{
 "@context": { "@vocab" : "http://schema.org/",
           "foaf"  : "http://xmlns.foaf/0.1/",
           "db"    : "http://dbpedia.org/page/"
         },
 "name": "Barack Obama",
 "givenName": "Barack",
 "familyName": "Obama",
 "jobTitle": "44th President of the United States",
 "foaf:knows" : "Zippy"
}
```

Can define a default namespace with "@vocab" and other namespaces as key-value pairs.
Note: the value of @context is now a JSON object

```
_:b0 <http://schema.orgfamily/Name> "Obama" .
_:b0 schema:givenName "Barack" .
_:b0 schema:jobTitle "44th President of the United States"
_:b0 schema:name "Barack Obama" .
_:b0 foaf:knows "Zippy" .
```

BUT, how can we turn the string "Barack Obama" into the dbpedia URL for him??

Use the '@id' property to indicate that a string is really a URL

```
{
 "@context": { "@vocab" : "http://schema.org/",
            "foaf"   : "http://xmlns.foaf/0.1/",
            "db"     : "http://dbpedia.org/page/"
           },
 "@id" : "db:Barack_Obama",
 "name": "Barack Obama",
 "givenName": "Barack",
 "familyName": "Obama",
 "jobTitle": "44th President of the United States",
 "foaf:knows" : "Zippy"
}
```

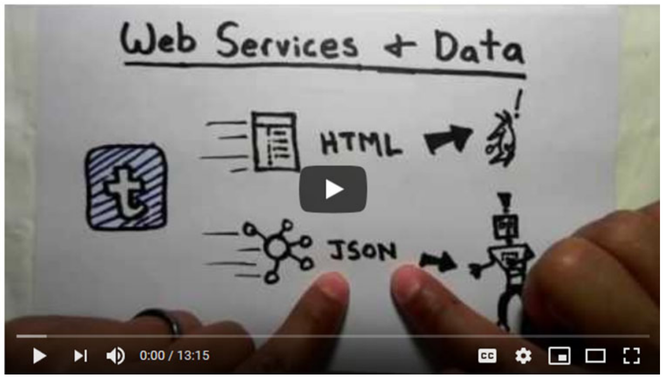
Interpret the string as a URL in the db: namespaces

```
db:Barack_Obama schema:familyName "Obama" .
db:Barack_Obama schema:givenName "Barack" .
db:Barack_Obama   schema:jobTitle "44th President of the United States" .
db:Barack_Obama schema:name "Barack Obama" .
db:Barack_Obama foaf:knows "Zippy" .
```

No longer an anonymous node

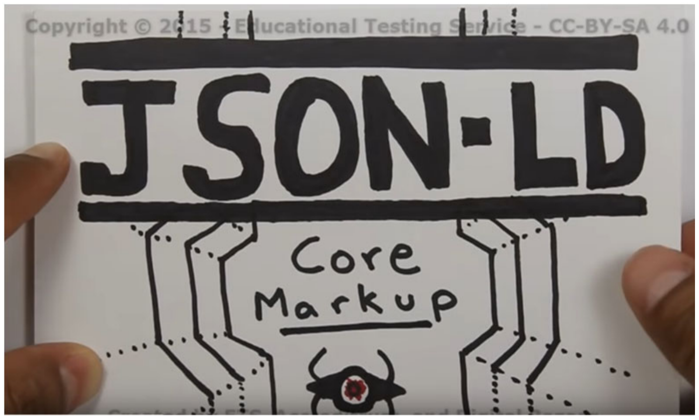
See the handout on JSON-LD Q&A

Check out the Website: https://www.youtube.com/watch?v=vioCbTo3C-4



Great site to learn about JSON-LD : Playground : https://json-ld.org/playground/

Check out the Website: JSON-LD Core Concepts https://www.youtube.com/watch?v=UmvWk_TQ30A



Note that in this video, values and properties are not always surrounded by double quotes.
When you do JSON-LD, you must surround properties and values with double quotes.