JSON-LD
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/](http://json.org/) has more information
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 if you compress 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
JSON-LD: **2014 W3C recommendation** for representing RDF data as JSON objects

```json
{
  "@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/
}
```
In the beginning

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

```json
{
  "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'
Define a context

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

[1] This means that 'name' is shorthand for 'http://schema.org/name'
[2] This means that 'image' is shorthand for 'http://schema.org/image'
[3] This means that a string value associated with 'image' should be interpreted as an identifier that is an IRI
[4] This means that 'homepage' is shorthand for 'http://schema.org/url'
[5] This means that a string value associated with 'homepage' should be interpreted as an identifier that is an IRI
Reference an external context

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

• Fetch `http://json-ld.org/contexts/person.jsonld`
  ```
curl http://json-ld.org/contexts/person.jsonld
  ```
  Returns a JSON object defining the context

• Fetch `http://schema.org/`
  ```
curl –L http://schema.org/
  ```
  Returns HTML page, since its default application/type is HTML

• Fetch
  ```
  ```
Making assertions about things

{
    "@context": {

        ... 

        "Restaurant": "http://schema.org/Restaurant",
        "Brewery": "http://schema.org/Brewery"

    }

    "@id": "http://example.org/places#BrewEats",
    "@type": [ "Restaurant", "Brewery" ],
    ... 

}
Adding a default 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/markuslanthaler"
}
Embedding other objects

{

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

    
}

...  

}
Google looks for and uses some JSON-LD markup (e.g., for organizations)

- Put a JSON-LD object in head or body of web page wrapped with script tags:

```html
<script type="application/ld+json">
{
...
}
</script>
```
{  "@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",
  "email": "ebiquity@gmail.com",
  "faxNumber": "1-410-455-3969",
  "telephone": "1-410-455-3000",
  "logo": {
    "@id": "http://ebiquity.umbc.edu/img/ebg_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/"
    }
  ]
}
http://json-ld.org/

**JSON for Linking Data**

Data is messy and disconnected. JSON-LD organizes and connects it, creating a better Web.

Linked Data empowers people that publish and use information on the Web. It is a way to create a network of standards-based, machine-readable data across Web sites. It allows an application to start at one piece of Linked Data, and follow embedded links to other pieces of Linked Data that are hosted on different sites across the Web.

```json
{
  "@context": "http://json-ld.org/contexts/person.jsonld",
  "@id": "http://dbpedia.org/resource/John_Lennon",
  "name": "John Lennon",
  "born": "1940-10-09",
  "spouse": "http://dbpedia.org/resource/Cynthia_Lennon"
}
```

JSON-LD is a lightweight Linked Data format. It is easy for humans to read and write. It is based on the already successful JSON format and provides a way to help JSON data interoperate at Web-scale. JSON-LD is an ideal data format for programming environments, REST Web services, and unstructured databases such as CouchDB and MongoDB.
JSON-LD Playground

Play around with JSON-LD markup by typing out some JSON below and seeing what gets generated from it at the bottom of the page. Pick any of the examples below to get started. The playground uses the jsonld.js JSON-LD processor which fully conforms to the JSON-LD Syntax and API specifications.

JSON-LD Input

```json
{
  "@context": "http://schema.org/",
  "@type": "Person",
  "name": "Jane Doe",
  "jobTitle": "Professor",
  "telephone": "(425) 123-4567",
  "url": "http://www.janedoe.com"
}
```
JSON-LD is a good solution to putting blocks of semantic data on web pages.

Aimed at publish linked data, not ontologies, i.e., ABOX not TBOX.

Tools available for extracting RDF triples.

Search companies look for and JSON-LD that use vocabularies they understand (i.e., schema.org).