Lab 8 JSON-LD

Quick overview of general terms

Before Starting the Lab

Get a an idea of how the structure of JSON-LD works. Look at the example graph provided, and try to understand what is happening.

Try to imagine how it would look like if serialized in turtle (the output from the first couple of labs).

We recommend that you look at the **keywords** used in JSON-LD and understand their use case. For instance, what is the purpose of @id and @value?

Also there are several types of JSON-LD forms, those being “expansion”, “compaction” and “flattening”, having a general understanding of the differences, could also be useful. This can be done on the JSON-LD playground.

Other Resources

There is another lab presentation under MittUIB's files, about JSON-LD which shows examples of **"embedding", "referencing"** (here is a good example showcasing it) and "explicit statement". Which are useful to know about.

If you are unsure about anything, the documentation provided by w3.org is really useful, and a good place to start.

You could also look at the videos provided by JSON-LD.org over basics of JSON.

JSON-LD Example

```
{
  "@context": {
    "@base": "http://example.org/",
    "edges": "http://example.org/triple",
      "start": "http://example.org/source",
      "rel": "http://exaxmple.org/predicate",
      "end": "http://example.org/object",
    "Person" : "http://example.org/Person",
    "birthday" : {
      "@id" : "http://example.org/birthday",
      "@type" : "xsd:date"
    },
    "nameEng" : {
      "@id" : "http://example.org/en/name",
      "@language" : "en"
  }
  },
  "@graph": [
    {
      "@id": "people/Jeremy",
      "@type": "Person",
      "birthday" : "1987.1.1",
      "nameEng" : "Jeremy"
    },
    {
      "@id": "people/Tom",
      "@type": "Person"
    },
    {"edges" : [
      {
      "start" : "people/Jeremy",
        "rel" : "knows",
        "end" : "people/Tom"

      }
    ]}
  ]
}
```

@Context

Provides a mapping of terms to IRIs. Simply put, a way to shortcut terms, like what we did previously in RDF with Namespace.

Just like with RDF, it's case sensitive. It also has a list of **reserved words** (that cannot be used as a term)

Terms to define:

1) **Object Type**
2) **Data Type**
3) **Type Coercion**
4) **Aliasing Keywords**
5) **Internationalization**

Object / Data Type

```
@context{
        "Person" :  "http://example.org/Person",
        },
        “@graph”: [
                {
                        “@type” : “Person”

                }
```

```
@context{}
        ...
                “birthday”: {
                        @value : “1987.1.1”,
                        @type : “xsd:date
                },
```

Type Coercion

```
@context{}
        ...
                "birthday": {
                        @value : "1987.1.1",
                        @type : "xsd:date
                },
```

Instead of defining the value within the graph itself, when using the terms, you can define it in the context!

Like this!

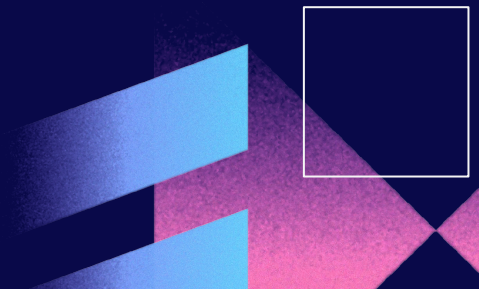
```
@context{
                "birthday": {
                        @id : "http://example.org/birthday",
                        @type : "xsd:date
                }} ,
                        "birthday" : "1987.1.1"
```

Aliasing Keywords

You can assign keywords from JSON-LD to terms, to shortcut them as well.

```
@context{
            "url": "@id",
            "a": "@type"
          },
           “url” : ...,
           “a” : ...,
```

Instead of using the “@” with the keyword in every use case.



Internationalization

When you want to assign a string value as the object, but determine what language it should include. For instance, if you want to represent someone's name in a different language.

```
@context{
    "nameEng" : {
      "@id" : "http://example.org/en/name",
      "@language" : "en"
   }},
        ...
        "nameEng" : "Jeremy"
```