COMP3220: Document Processing and Semantic Technologies RDFa

Rolf Schwitter

Rolf.Schwitter@mq.edu.au

Today's Agenda

- RDFa
- Vocabularies
- Question Answering
- Benefits

"The search and social companies are serious about indexing RDFa content, which means that you may want to get serious about adding it into your pages before your competitors do".

Source: `http://rdfa.info`

RDFa

- RDFa is one of several ways of serialising RDF data.
- RDFa stands for RDF in attributes.
- RDFa empowers web authors to add semantic information to web documents.
- RDFa 1.1 is an extension to HTML5.
- Adding RDFa to a web page can improve
  - the ability to search (better indexing)
  - user experience.
- See: **`https://www.w3.org/TR/rdfa-primer/`**

HTML5

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Oswald Muster</title>
  </head>
  <body>
    <p>
       My name is Oswald Muster, you can send
       me an email to: Oswald.Muster@mq.edu.au.
    </p>
  </body>
</html>
```

CURIE

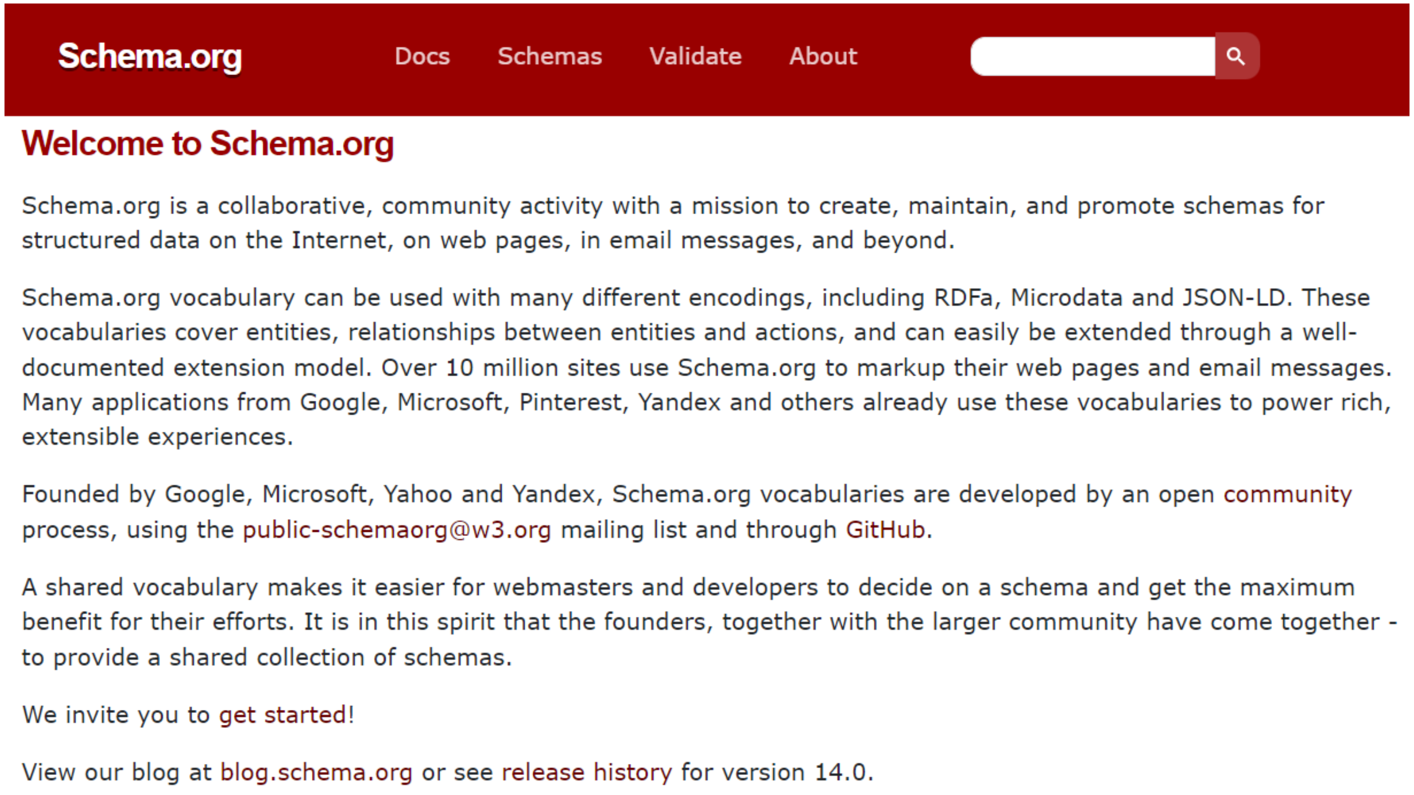
- RDFa allows us to abbreviate IRIs*.
- Compact IRI expressions are called CURIEs.
- A CURIE is comprised of a prefix and a reference.
- The prefix is separated from the reference by a colon (`:`).
- For example: `schema:familyName`
- *IRI (= Internationalised Resource Identifiers).

RDFa: Example

- When we want to talk about things on the Web, we need to specify at least one vocabulary.
- We do this by using the prefix attribute:

```
<!DOCTYPE html>
<html prefix="schema: http://schema.org/">
...
```

Schema.org



Welcome to Schema.org

Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond.

Schema.org vocabulary can be used with many different encodings, including RDFa, Microdata and JSON-LD. These vocabularies cover entities, relationships between entities and actions, and can easily be extended through a well-documented extension model. Over 10 million sites use Schema.org to markup their web pages and email messages. Many applications from Google, Microsoft, Pinterest, Yandex and others already use these vocabularies to power rich, extensible experiences.

Founded by Google, Microsoft, Yahoo and Yandex, Schema.org vocabularies are developed by an open community process, using the public-schemaorg@w3.org mailing list and through GitHub.

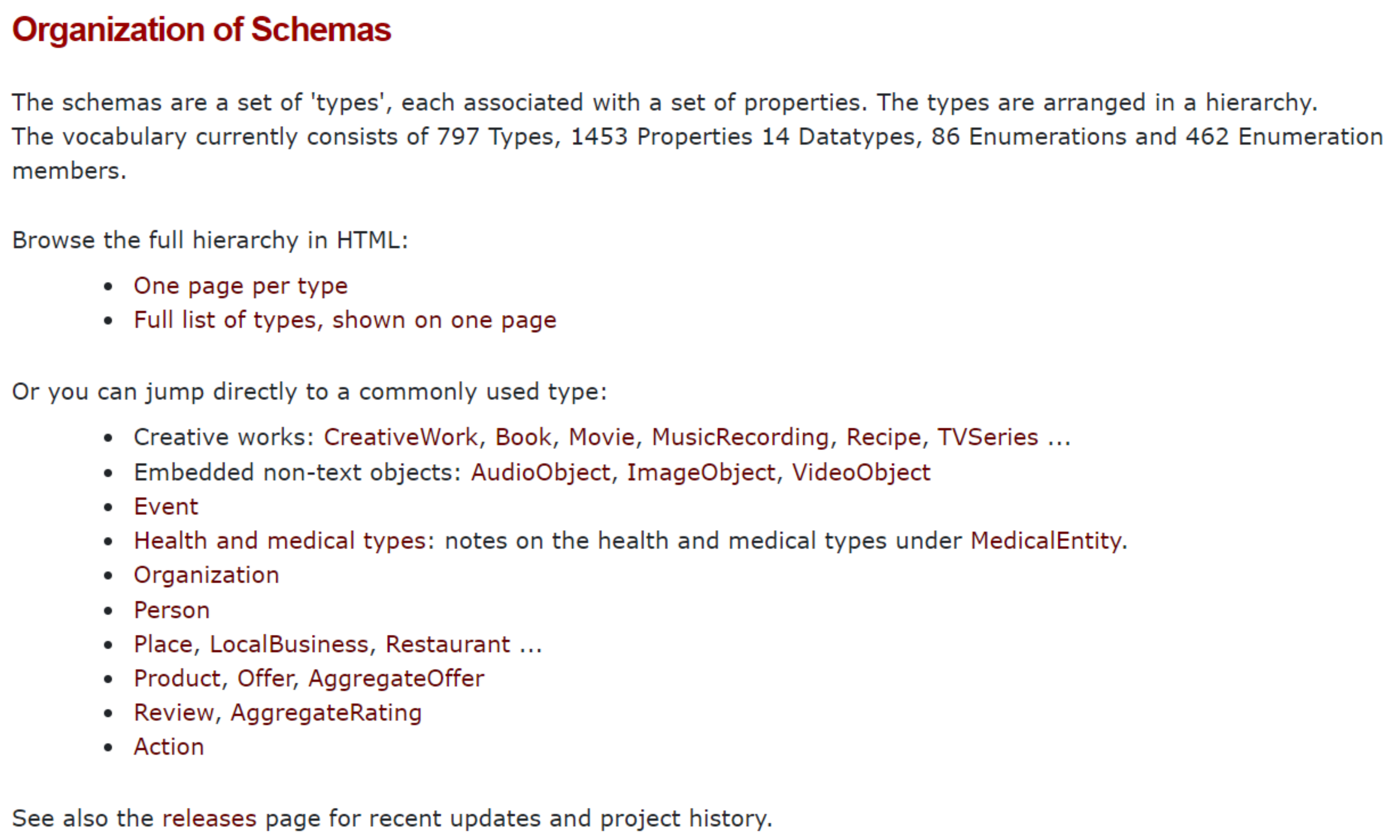
A shared vocabulary makes it easier for webmasters and developers to decide on a schema and get the maximum benefit for their efforts. It is in this spirit that the founders, together with the larger community have come together - to provide a shared collection of schemas.

We invite you to get started!

View our blog at blog.schema.org or see release history for version 14.0.

`http://schema.org/`

Organization of Schemas



Organization of Schemas

The schemas are a set of 'types', each associated with a set of properties. The types are arranged in a hierarchy.
The vocabulary currently consists of 797 Types, 1453 Properties 14 Datatypes, 86 Enumerations and 462 Enumeration members.

Browse the full hierarchy in HTML:

- One page per type
- Full list of types, shown on one page

Or you can jump directly to a commonly used type:

- Creative works: CreativeWork, Book, Movie, MusicRecording, Recipe, TVSeries ...
- Embedded non-text objects: AudioObject, ImageObject, VideoObject
- Event
- Health and medical types: notes on the health and medical types under MedicalEntity.
- Organization
- Person
- Place, LocalBusiness, Restaurant ...
- Product, Offer, AggregateOffer
- Review, AggregateRating
- Action

See also the releases page for recent updates and project history.

http://schema.org/docs/schemas.html

RDFa: Example

- We can create an identifier for things described on the Web page using the resource attribute:

```
<!DOCTYPE html>
<html prefix="schema: http://schema.org/">
<head>
  <meta charset="UTF-8">
  <title>Oswald Muster</title>
</head>
<body resource="#oswald">
...
```

RDFa: Example

- Once we have specified a vocabulary and a resource, we need to specify the type of things we are talking about:

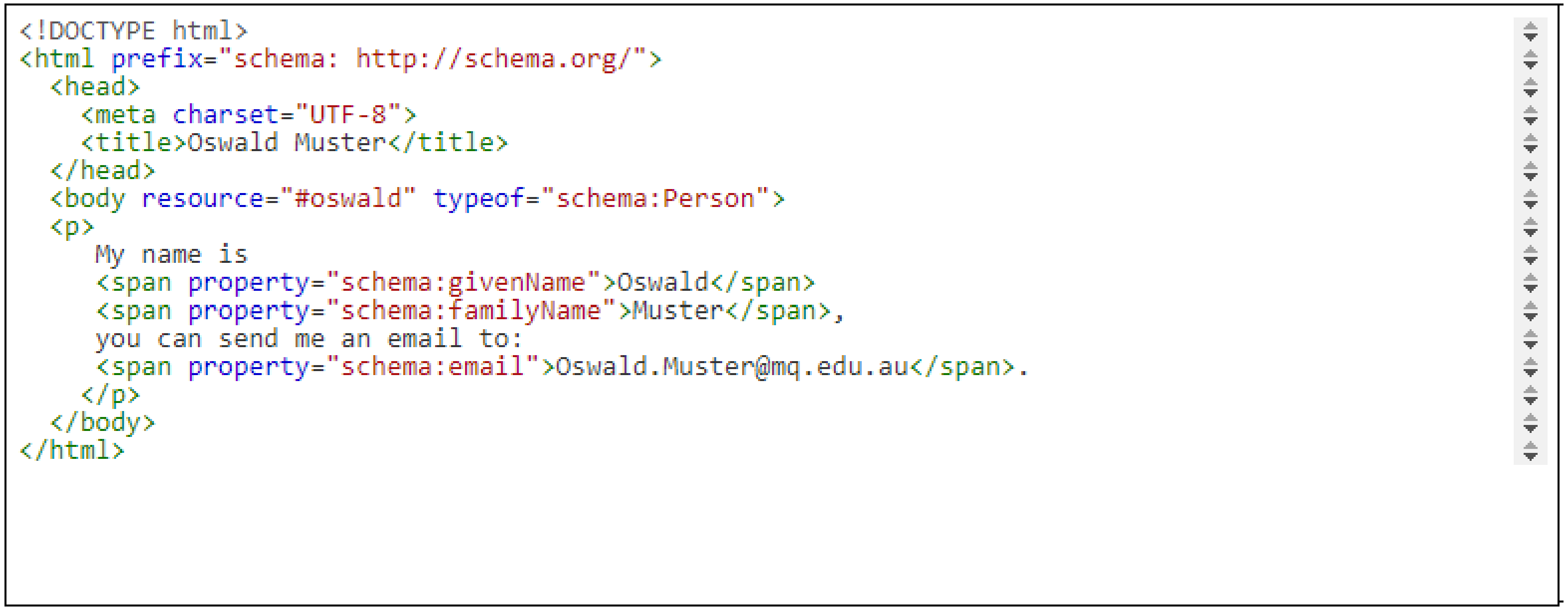
```
<!DOCTYPE html>
<html prefix="schema: http://schema.org/">
<head>
  <meta charset="UTF-8">
  <title>Oswald Muster</title>
</head>
<body resource="#oswald" typeof="schema:Person">
...
```

RDFa: Example

- Now, we can specify the properties of that person:

```
  ...
  <p>
   My name is
     <span property="schema:givenName">Oswald</span>
     <span property="schema:familyName">Muster</span>,
     you can send me an email to:
     <span property="schema:email">
       Oswald.Muster@mq.edu.au</span>.
  </p>
  </body>
</html>
```

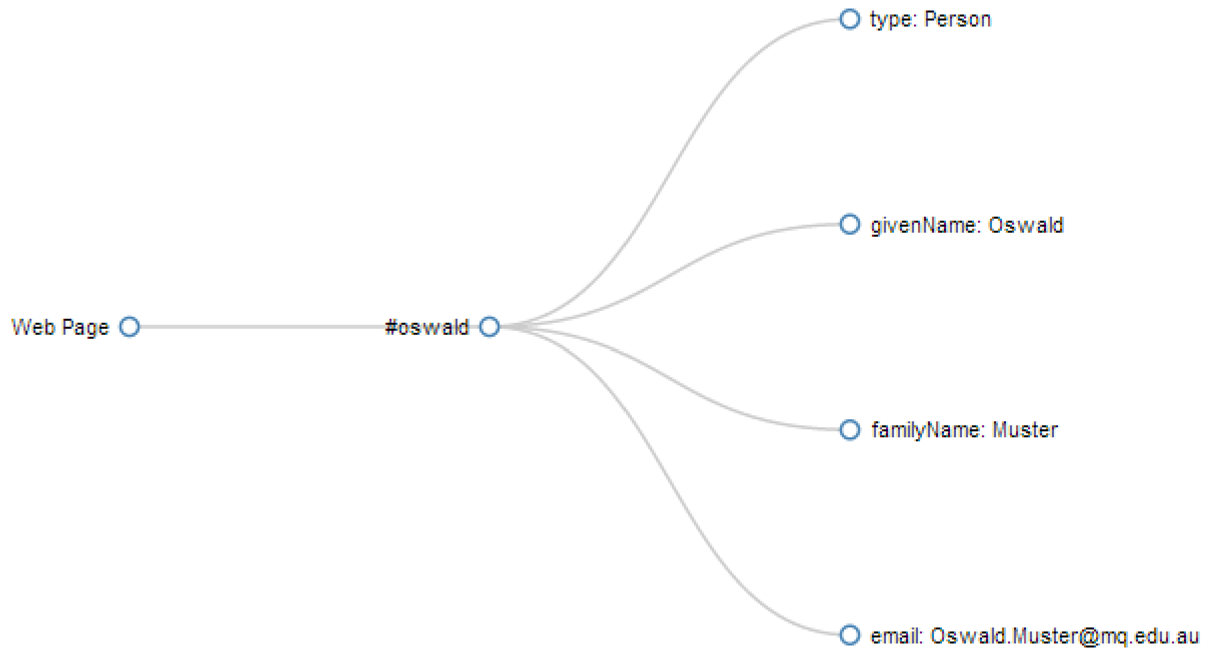
RDFa: Result



```
<!DOCTYPE html>
<html prefix="schema: http://schema.org/">
  <head>
    <meta charset="UTF-8">
    <title>Oswald Muster</title>
  </head>
  <body resource="#oswald" typeof="schema:Person">
  <p>
     My name is
     <span property="schema:givenName">Oswald</span>
     <span property="schema:familyName">Muster</span>,
     you can send me an email to:
     <span property="schema:email">Oswald.Muster@mq.edu.au</span>.
    </p>
  </body>
</html>
```

http://rdfa.info/play/

Visualisation



http://rdfa.info/play/

Turtle Notation

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix schema: <http://schema.org/> .

<http://rdfa.info/play/#oswald>
    rdf:type schema:Person;
    schema:givenName "Oswald";
    schema:familyName "Muster";
    schema:email "Oswald.Muster@mq.edu.au".
```

Turtle Notation

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix schema: <http://schema.org/> .

<http://rdfa.info/play/#oswald>
    rdf:type schema:Person;
    schema:givenName "Oswald";
    schema:familyName "Muster";
    schema:email "Oswald.Muster@mq.edu.au".
```

- Remember RDF: subject, predicate, object.

Turtle Notation

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix schema: <http://schema.org/> .

<http://rdfa.info/play/#oswald>
    rdf:type schema:Person;
    schema:givenName "Oswald";
    schema:familyName "Muster";
    schema:email "Oswald.Muster@mq.edu.au".
```

- Remember RDF: subject, predicate, object.

Turtle Notation

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix schema: <http://schema.org/> .

<http://rdfa.info/play/#oswald>
    rdf:type schema:Person;
    schema:givenName "Oswald";
    schema:familyName "Muster";
    schema:email "Oswald.Muster@mq.edu.au".
```

- Remember RDF: subject, predicate, object.

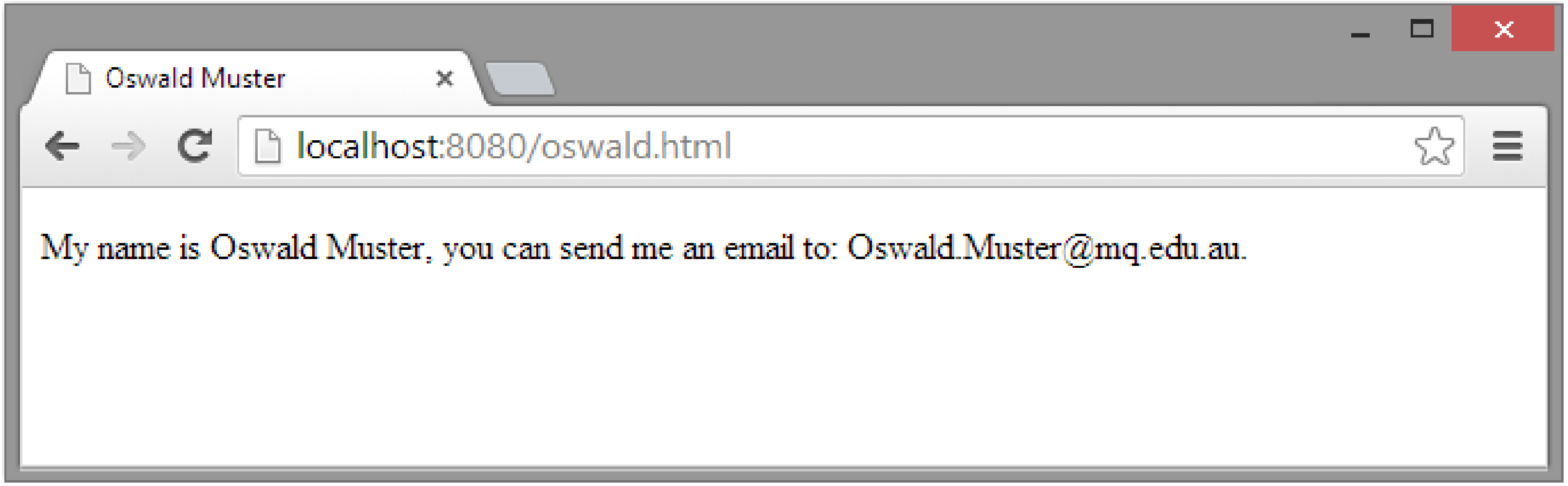
RDFa: Example (oswald.html)

```
<!DOCTYPE html>
<html prefix="schema: http://schema.org/">
  <head>
    <meta charset="UTF-8">
    <title>Oswald Muster</title>
  </head>
  <body resource="#oswald" typeof="schema:Person">
  <p>
     My name is
     <span property="schema:givenName">Oswald</span>
     <span property="schema:familyName">Muster</span>,
     you can send me an email to:
     <span property="schema:email">Oswald.Muster@mq.edu.au</span>.
  </p>
  </body>
</html>
```

Question Answering

- Python server:

```
c:>python -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 …
```



My name is Oswald Muster, you can send me an email to: Oswald.Muster@mq.edu.au.

Question Answering with Python/SPARQL

```
# pip install rdflib
# pip install pyRdfa3

import rdflib

g = rdflib.Graph()

g.parse('http://localhost:8080/oswald.html', format='rdfa')
```

Question Answering with Python/SPARQL

```
res = g.query(
      """ PREFIX schema: <http://schema.org/>
          SELECT ?subj ?obj1 ?obj2
          WHERE { ?subj rdf:type schema:Person .
                  ?subj schema:givenName ?obj1 .
                  ?subj schema:familyName ?obj2 . } """)
```

Question Answering with Python/SPARQL

```
print("Person found:")

for row in res:
    print("%s has given name %s"%(row.subj, row.obj1))
    print("%s has family name %s"%(row.subj, row.obj2))
```

Result

```
Person found:
http://localhost:8080/oswald.html#oswald has given name Oswald
http://localhost:8080/oswald.html#oswald has family name Muster
```

RDFa: Example

- We can specify more than one vocabulary using the prefix attribute:

```
<!DOCTYPE html>
<html prefix="schema: http://schema.org/
              ex: http://example.org/terms/">
<head>
  <meta charset="UTF-8">
  <title>Oswald Muster</title>
</head>
...
```

RDFa: Example

```
...
</p>
  My name is
  <span property="schema:givenName">Oswald</span>
  <span property="schema:familyName">Muster</span>,
  you can send me an email to:
  <span property="schema:email">
    Oswald.Muster@mq.edu.au</span>.
  My favourite animal is the
  <span property="ex:preferredAnimal">platypus</span>.
</p>
...
```

Vocabularies

- Schema.org: `http://schema.org/`
- Dublin Core: `https://www.dublincore.org/specifications/dublin-core/dcmi-terms/`
- FOAF: `http://xmlns.com/foaf/spec/`
- SKOS: `http://www.w3.org/2004/02/skos/vocabs`
- SIOC: `http://rdfs.org/sioc/spec/`

- **FOAF = Friend of a Friend**
- **SKOS = Simple Knowledge Organization System**
- **SIOC = Semantically-Interlinked Online Communities**

Custom Vocabularies

- For some data there might be no existing vocabulary.
- The creation of a custom vocabulary in RDFa involves:
  - selecting an IRI where the vocabulary will reside;
  - defining classes and properties that make up the vocabulary;
  - publishing the vocabulary at the specified IRI.

Benefits of RDFa

- Publisher independence: each site can use its preferred vocabulary.
- Data reuse: data is not duplicated.
- Self-containment: HTML and RDFa are separated.
- Schema modularity: attributes are reusable.
- Evolvability: additional attributes can be added and semantics of data can be extracted.
- Accessibility: better usability of web sites.

Take-Home Messages

- RDF can be serialised in may different ways.
- Using RDFa, web authors can turn their existing human-visible text into machine-readable data.
- RDFa adds an attribute-level extension to web documents for embedding rich metadata.
- RDFa uses the same subject-predicate-object structure as RDF.
- SPARQL can be used as a query language to extract information from RDFa triples.