

**An overview of  
Semantic Web Languages  
and Technologies**

# Semantic Web Technologies

- W3C “recommendations”
  - RDF, RDFS, RDFa, OWL, SPARQL, RIF, R2R, etc...
- Common tools and systems -- commercial, free and open sourced
  - Ontology editors, triple stores, reasoners, etc.
- Common ontologies and data sets
  - Foaf, DBpedia, SKOS, PROV, etc.
- Infrastructure systems
  - Search, ontology metadata, linking services
- Non W3C: schema.org, Freebase, ...

# Common KR languages

- [Knowledge representation and reasoning](#) (KR&R) is an important part of AI & other disciplines
- Many approaches have been developed, implemented, and evolved since the 1960s
- Most were one-offs, used only by their developers
- Starting in the 1990s, there was an interest in developing a common KR language to support knowledge reuse and distributed KB systems
- The Semantic Web languages (e.g., OWL) are a current generation of this idea

# Questions

- **Database (DB) vs. knowledge base (KB)?**
  - TL;DR: DBs have facts, KBs have general knowledge as well as facts
  - DBs typically have very simple schemas (knowledge) and lots of data (facts)
  - KBs have complex schemas (aka ontologies) and may or may not have a lot of instances (data)
  - Knowledge graphs are the new KBs
- **KBs support inference, e.g.,**
  - parent(?x,?y) => person(?x), person(?y), child(?y, ?x),  
older(?x, ?y), ?x≠?y.
  - parent(john, mary) => person(john), person(mary),  
child(mary,john), older(john, mary), john ≠ mary.

# Questions

What's the impact of using different structures to represent data or knowledge?

- Natural language
- Program code
- Relations vs. graphs vs. objects
- Logic vs. rules vs. procedures
- Neural networks
- Tensors

# Questions

What's our “semantic” model for facts and knowledge?

- Classical logic is a common choice
  - $\text{man}(\text{socrates}), \forall x \text{man}(x) \Rightarrow \text{mortal}(x)$
  - Classical logic has limitations: facts and relations and “rules” are either (always) True or False for all time
- May need to represent and reason with probabilistic or fuzzy facts and knowledge
- May need to handle dynamic facts or knowledge

# Semantic Web Technologies

- Basic approach uses classical logic for underlying semantics
  - + Simple, well understood, good reasoning algorithms
  - No probabilities, adding extensions (e.g., for time) adds complexity
- Knowledge represented as a graph
  - + Simple, good tool support
  - Sometimes may be too simple

# Two Semantic Web Notions

- **The semantic web**

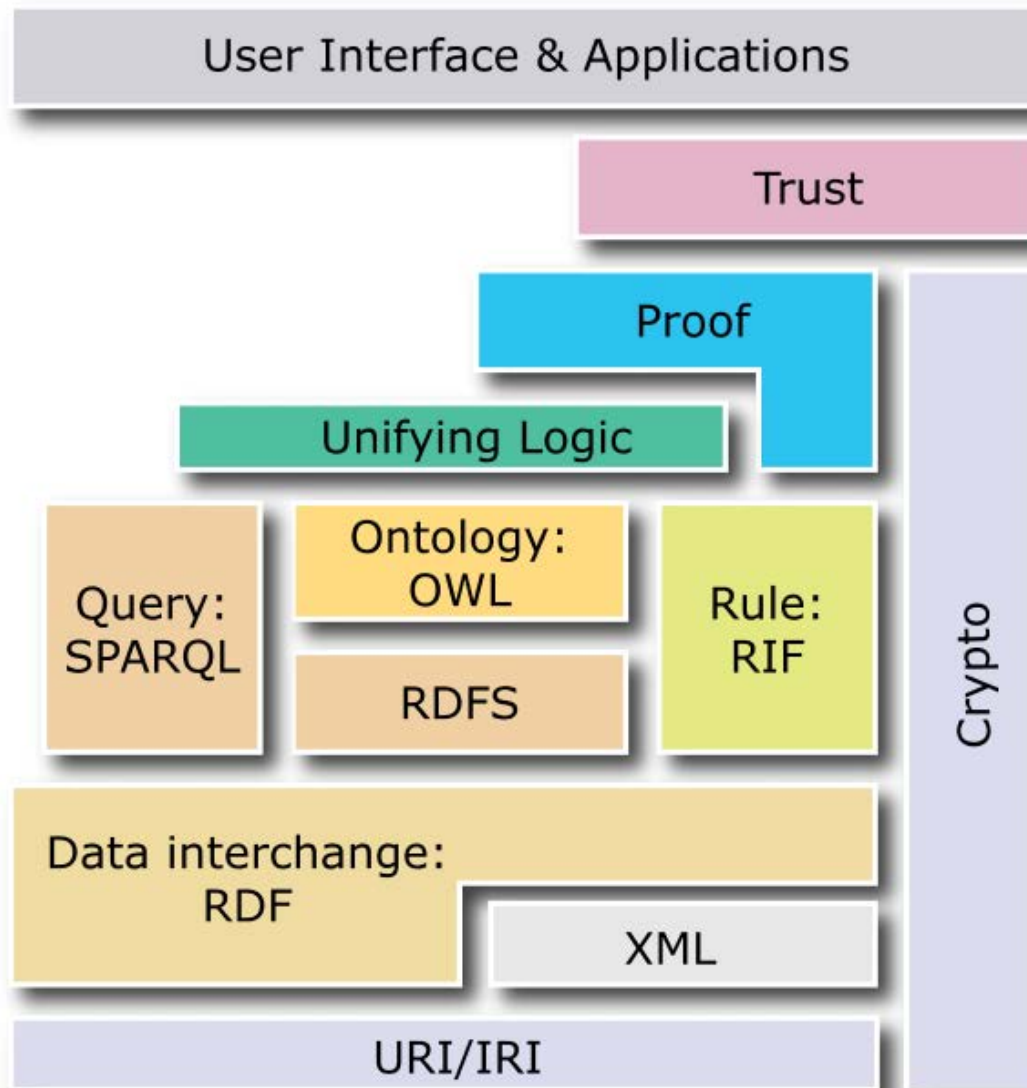
- Idea of a web of machine understandable information
- Agnostic about the technology used to support it
- May involve more AI (e.g., NLP, machine learning)
- Human end users in the center

- **The Semantic Web**

- Current vision of a semantic web as defined by the W3C community: a **Web of Data**
- Uses W3C supported standards, i.e., RDF, OWL, SPARQL, SHACL, XML, RIF, etc.
- Developing common reference KGs, i.e., DBpedia, Wikidata
- By machines for machines with human-oriented applications on top



# W3C Semantic Web Stack



# RDF is the first SW language

## XML Encoding

```
<rdf:RDF .....>
  <...>
  <...>
</rdf:RDF>
```

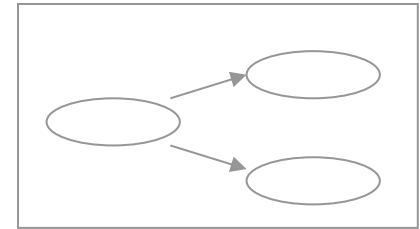
**Good for  
Machine  
Processing**

## JSON Encoding

```
{ "@context": {
  "name": "http://.../name",
  "Person": "http://.../Person"
}
"@type": "Person",
"name": "Markus Lanthaler"
}
```

**RDF  
Data Model**

## Graph



**Good for  
people, viz,  
graph DBMS**

## Triples

```
stmt(docInst, rdf_type, Document)
stmt(personInst, rdf_type, Person)
stmt(inroomInst, rdf_type, InRoom)
stmt(personInst, holding, docInst)
stmt(inroomInst, person, personInst)
```

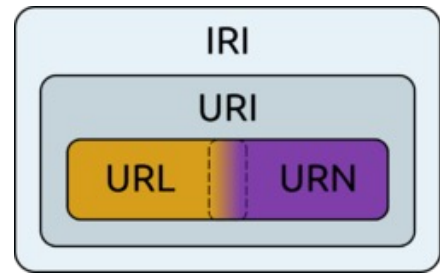
**Good For Reasoning and  
Databases**

*RDF is a simple  
language for building  
graph based  
representations*

# The RDF Data Model

- An RDF document is an unordered collection of statements, each with a **subject**, **predicate** and **object** (aka **triples**)
- A triple can be thought of as a labelled arc in a graph
- Statements describe properties of web **resources**
- Resources are objects that can be pointed to by a **URI**:
  - a document, a picture, a paragraph on the Web, ...
  - E.g., <http://umbc.edu/~finin/cv.html>
  - a book in the library, a real person (?)
  - [isbn://5031-4444-3333](http://isbn://5031-4444-3333)
- Properties themselves are also resources (URIs)

# URIs are a foundation



- **URI** = [Uniform Resource Identifier](#)
  - "The generic set of all names/addresses that are short strings that refer to resources"
  - **URLs** ([Uniform Resource Locators](#)) subset of URIs, used for resources that are *accessible* on web
- URIs look like URLs, often with fragment identifiers pointing to a document part:
  - <http://foo.com/bar/mumble.html#pitch>
- **IRIs** ([Internationalized Resource Identifier](#)) are URIs that allow Unicode characters

# IRIs are a foundation

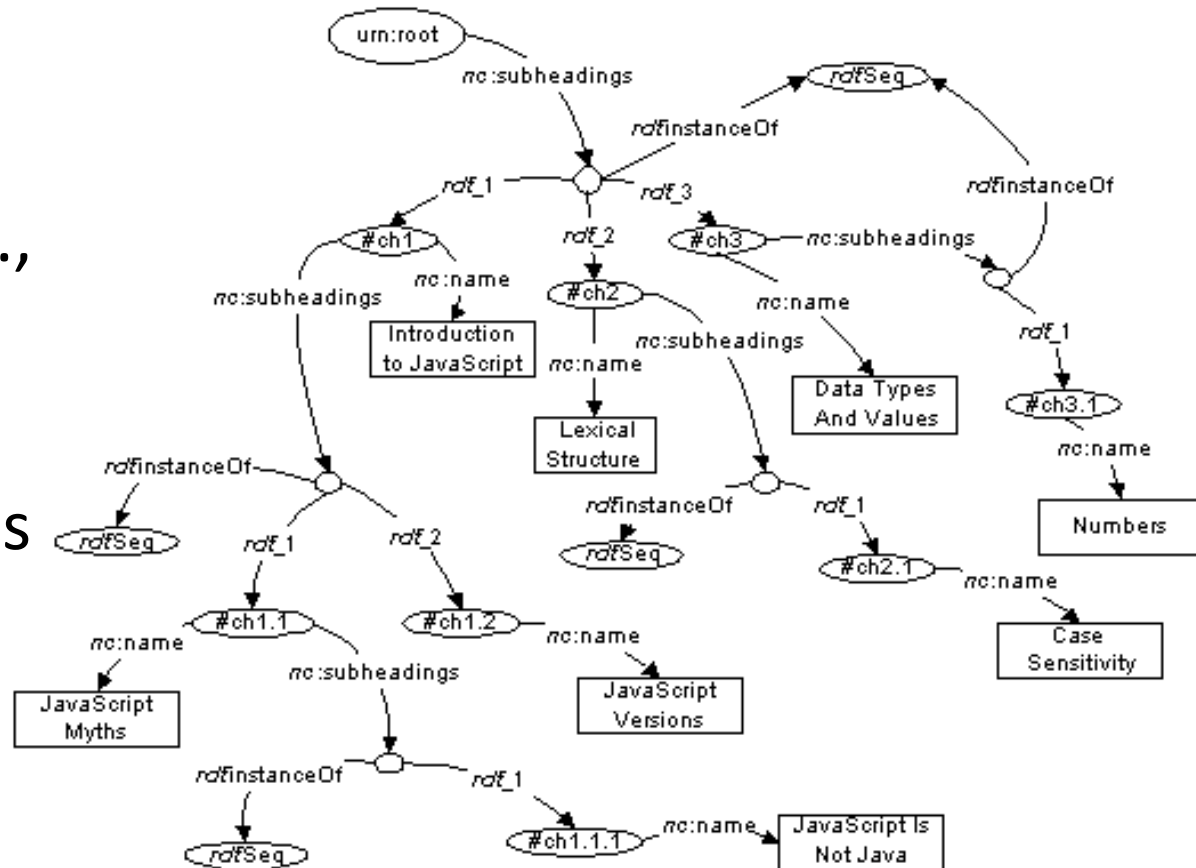
- IRIs and URIs are unambiguous, unlike natural language terms -- the web provides a **global namespace**
- We can use a URI to **denote** something, e.g., a concept, entity, event or relation
- We usually assume references to the same URI are to the same thing

# What does a IRI mean?

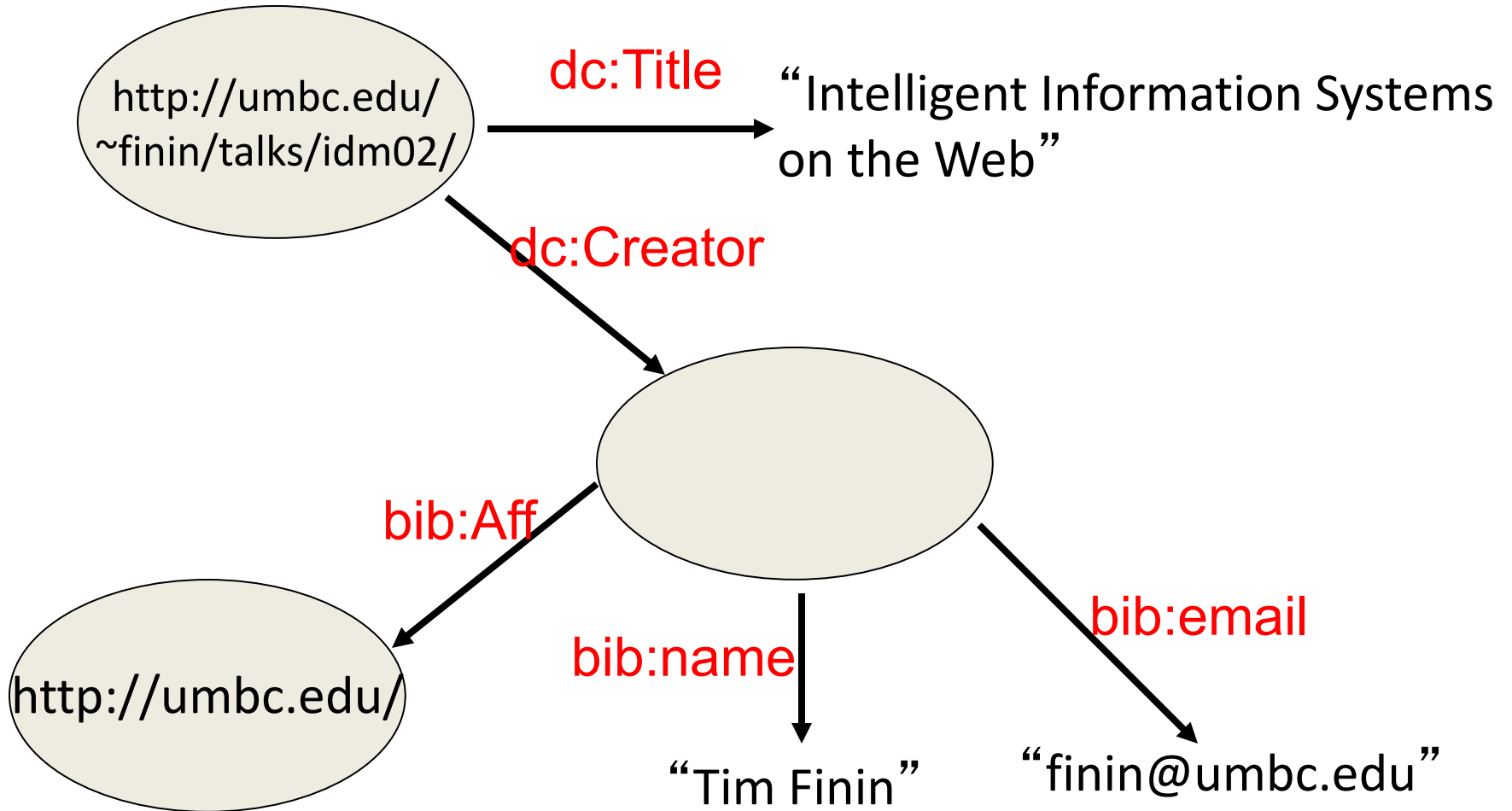
- An IRI can denote a **web resource**
  - <http://umbc.edu/~finin/finin.jpg> denotes a file
  - RDF can make assertions about it, e.g., it's an image and depicts a person with name Tim Finin, ...
- It can denote an **instance or concepts**
  - E.g., <http://umbc.edu/> denotes a particular university
- Resolving the ambiguity is done by context or social convention
- TBL: “Cool URIs don't change”
  - <http://www.w3.org/Provider/Style/URI>

# The RDF Graph

- An **RDF document** is an unordered **collection of triples**
- The subject of one triple can be the object of another
- The result is a **directed, labelled graph**
- A triple's object can also be a **literal**, e.g., a string or number
- Graphs are simpler than relational tables or objects
- This is both a plus and a minus



# Simple RDF Example





# Serialization

- A graph is an abstract model, we'll need to **serialize** it as text for many reasons, e.g., display, editing, exchange, ...
- Multiple standard RDF serializations
- Most important: XML, Turtle, ntriples, JSON-LD
- Most Semantic Web tools can read or write in any of these serializations

# XML encoding for RDF

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:bib="http://daml.umbc.edu/ontologies/bib/">
  <rdf:Description rdf:about="http://umbc.edu/~finin/talks/idm02/">
    <dc:title>Intelligent Information Systems on the Web</dc:title>
    <dc:creator>
      <rdf:Description>
        <bib:Name>Tim Finin</bib:Name>
        <bib:Email>finin@umbc.edu</bib:Email>
        <bib:Aff rdf:resource="http://umbc.edu/" />
      </rdf:Description>
    </dc:creator>
  </rdf:Description>
</rdf:RDF>
```

# Note the prefix declarations

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:bib=http://daml.umbc.edu/ontologies/bib/>
<rdf:Description rdf:about="http://umbc.edu/~finin/talks/idm02/">
  <dc:title>Intelligent Information Systems on the Web</dc:title>
  <dc:creator>
    <rdf:Description>
      <bib:Name>Tim Finin</bib:Name>
      <bib:Email>finin@umbc.edu</bib:Email>
      <bib:Aff rdf:resource="http://umbc.edu/" />
    </rdf:Description>
  </dc:creator>
</rdf:Description>
</rdf:RDF>
```

# Note the prefix declarations

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:bib=http://daml.umbc.edu/ontologies/bib/>
<rdf:Description rdf:about="http://umbc.edu/~finin/talks/idm02/">
  <dc:title>Intelligent Information Systems on the Web</dc:title>
  <dc:creator>
    <rdf:Description>
      <bib:Name>Tim Finin</bib:Name>
      <bib:Email>finin@umbc.edu</bib:Email>
      <bib:Aff rdf:resource="http://umbc.edu/">
    </rdf:Description>
  </dc:creator>
</rdf:Description>
</rdf:RDF>
```

Makes it easy to include terms from three different “vocabularies”:

- **rdf** for terms that are part of its representation language (e.g., `rdf:type`)
- **dc** for terms from the [Dublin Core](http://purl.org/dc/elements/1.1/) vocabulary developed by librarians
- **bib** for terms from a bibliography vocabulary developed at UMBC

# Easy to convert between serializations

Most RDF software tools can read and write different serializations

- [rdf2rdf](#) is a simple handy utility for converting from one RDF serialization to another
- [Any23](#) is another open-source library, web service and command line tool
- [riot](#) is yet another converter that's part of the open-source [Apache Jena](#) package

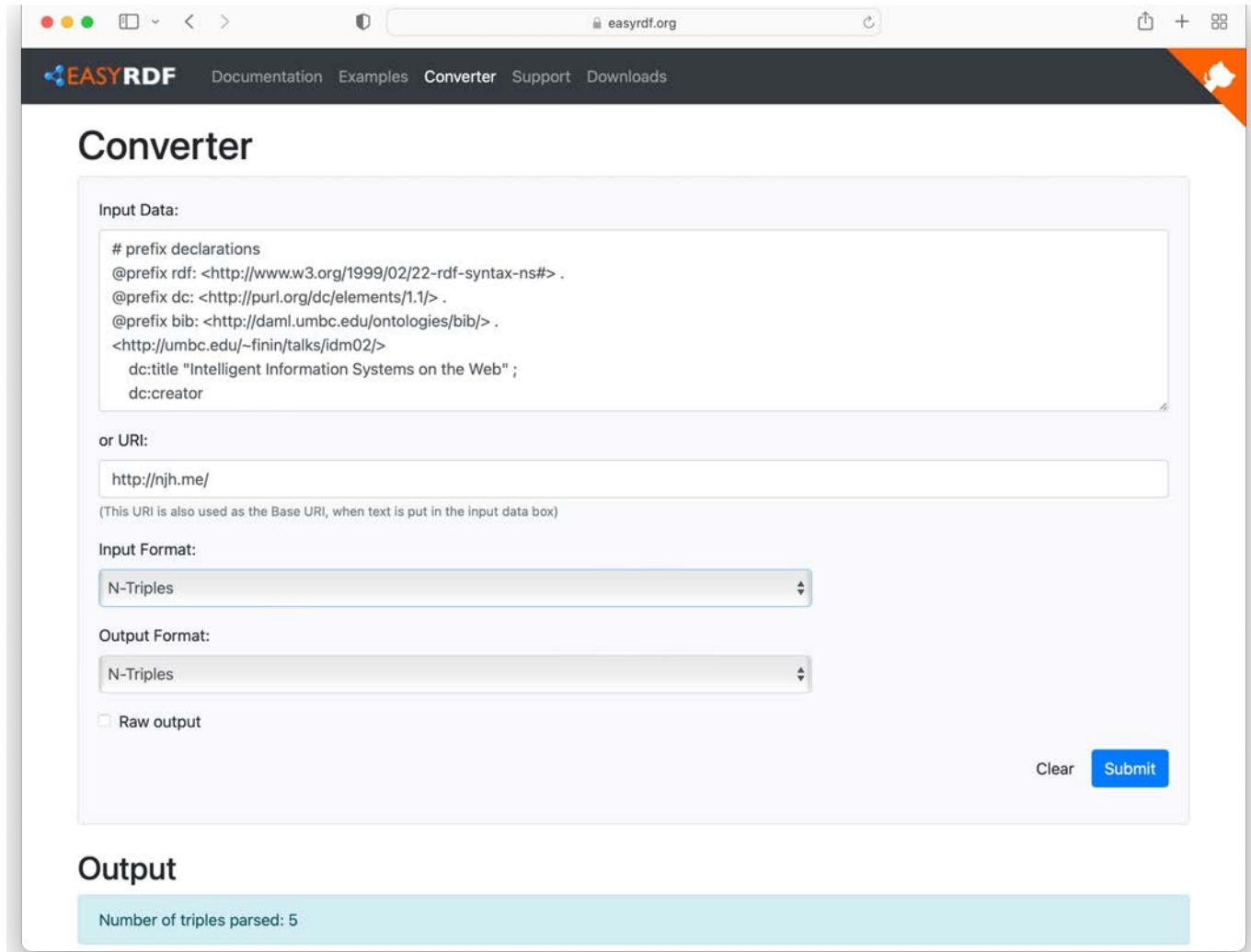
# Using riot to convert ttl to nt

- Guesses input and output formats
- The NT format has 5 lines, one for each triple

```
>>> riot ex1.ttl ex1.nt
```

```
<http://umbc.edu/~finin/talks/idm02/>  
<http://purl.org/dc/elements/1.1/title> "Intelligent  
Information Systems on the Web" .  
_:B55c4d1cd7d6ea94ecdf3cad94f462ee3  
<http://daml.umbc.edu/ontologies/bib/Name> "Tim Finin" .  
_:B55c4d1cd7d6ea94ecdf3cad94f462ee3  
<http://daml.umbc.edu/ontologies/bib/Email> "finin@umbc.edu"  
.  
_:B55c4d1cd7d6ea94ecdf3cad94f462ee3  
<http://daml.umbc.edu/ontologies/bib/Aff> <http://umbc.edu/>  
.  
<http://umbc.edu/~finin/talks/idm02/>  
<http://purl.org/dc/elements/1.1/creator>  
_:B55c4d1cd7d6ea94ecdf3cad94f462ee3 .  
>>>
```

# Online conversion services also exist



The screenshot shows the EasyRDF Converter web application. The browser address bar displays "easyrdf.org". The navigation menu includes "Documentation", "Examples", "Converter", "Support", and "Downloads". The main heading is "Converter".

**Input Data:**

```
# prefix declarations
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix bib: <http://daml.umbc.edu/ontologies/bib/> .
<http://umbc.edu/~finin/talks/idm02/>
  dc:title "Intelligent Information Systems on the Web" ;
  dc:creator
```

**or URI:**

(This URI is also used as the Base URI, when text is put in the input data box)

**Input Format:**

**Output Format:**

Raw output

**Output**

Number of triples parsed: 5

<https://www.easyrdf.org/converter>

# N-triple representation

- [N-triples](#) is a line-oriented serialization for RDF
- Also known as NT format
- URIs are wrapped in angle brackets, ended with a period  
<*subject*> <*predicate*> <*object*> .

```
<http://umbc.edu/~finin/talks/idm02/> <http://purl.org/dc/elements/1.1/title>  
  "Intelligent Information Systems on the Web" .  
<http://umbc.edu/~finin/talks/idm02/> <http://purl.org/dc/elements/1.1/creator>  
  _:node17i6ht38ux1 .  
  _:node17i6ht38ux1 <http://daml.umbc.edu/ontologies/bib/Name> "Tim Finin" .  
  _:node17i6ht38ux1 <http://daml.umbc.edu/ontologies/bib/Email> "finin@umbc.edu" .  
  _:node17i6ht38ux1 <http://daml.umbc.edu/ontologies/bib/Aff> <http://umbc.edu/> .
```

- NT is less readable for people, but loads into a triple store quickly because it's easier to parse



# Turtle Serialization

Turtle: a compact and readable serialization

```
# prefix declarations
```

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

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
```

```
@prefix bib: <http://daml.umbc.edu/ontologies/bib/> .
```

```
<http://umbc.edu/~finin/talks/idm02/>
```

```
  dc:title "Intelligent Information Systems on the Web" ;
```

```
  dc:creator
```

```
    [ bib:name "Tim Finin" ;
```

```
      bib:email "finin@umbc.edu" ;
```

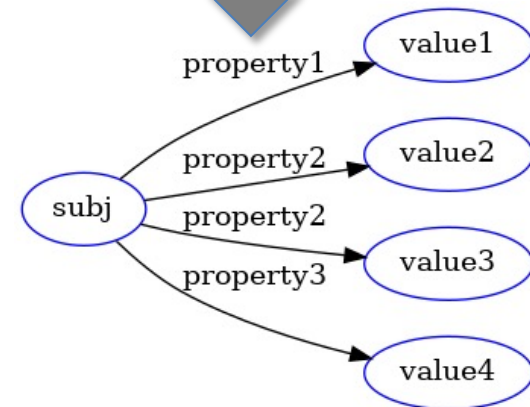
```
      bib:aff <http://umbc.edu/> ] .
```

# Basic Turtle Syntax

- Based on TBL's original [notation3](#) syntax
- Prefixes make URI references compact
- Note use of empty prefix for `http://ex.com/`
- Subject is followed by one or more property-values sets separated by **semicolons**
- Multiple values for the same property are separated by **commas**
- Blank nodes are enclosed in square brackets

```
@prefix : <http://ex.com/> .  
:subj  
  :property1 :value1;  
  :property2 :value2, value3;  
  :property3 value4.
```

```
:subj :property1 :value1 .  
:subj :property2 :value2 .  
:subj :property2 :value3 .  
:subj :property3 :value4 .
```



Namespaces:  
`http://ex.com/`

# More RDF Vocabulary

- RDF is a “pure” graph representation language
  - Graphs are unordered set of triples: node, edge, node
  - Nodes and edges are simple objects denoted by URIs
  - Object nodes can be URIs or RDF literals
- Literals can be strings ("dog"), [language-tagged strings](#) ("chien"@fr), or strings tagged with a [XML Schema datatype](#) ("3.14"^^xsd:decimal)
- For Turtle, [numbers](#) are automatically recognized as such by regexes, so both "3.14"^^xsd:decimal and 3.14 are recognized as a floating-point numbers, but "3.14" is seen as a string

# Making statements about statements

- Suppose we want to give an edge a probability  
(:flipper rdf:type :mammal) :probability 0.9
- Or assert that someone believes it to be true  
(:flipper rdf:type :fish) :believedBy :Bob  
((:flipper rdf:type :fish) :believedBy :Bob) :believedBy :Carol
- Or give a date for a population count  
(:arbutus :population 21655) :date 2020
- [Property graphs](#) let us attach properties with literal values to either nodes or edges, directly supporting these examples

# Property graphs?

- RDF is a “pure” graph model with only labeled nodes and edges
- Many popular graph databases implement property graphs (e.g., [Neo4j](#))
- Nodes & edges can have properties, whose values are *literals* or maybe *lists of literals*
- Results in a more compact graph
- But, as we’ll see, introduces some limitations

# RDF Reification (1)

- In an RDF graph each triple must be unique

```
@prefix : http://ex.com/  
:flipper rdf:type :mammal .  
:flipper rdf:type :mammal .
```

- In an RDF graph there can be only one triple with a given subject, predicate and object
- Duplicate triples are ignored and not added to the knowledge graph
- This example specifies a graph with just one triple

# RDF Reification (2)

- RDF also can describe triples through reification
- Enabling statements about statements

```
@prefix : http://ex.com/  
:flipper rdf:type :mammal .  
_:s1 rdf:type rdf:Statement ;  
    rdf:subject :flipper;  
    rdf:predicate :type;  
    rdf:object :mammal;  
    :probability 0.9 .
```

- Reification is a term from philosophy for making concrete something that's abstract
- In CS, it has a similar meaning
- The underscore prefix is introduces a *blank node*
- More about this later, but for now, think of it as introducing “a new, nameless thing”

# RDF Reification (3)

- RDF also can describe triples through [reification](#)
- Enabling statements about statements

```
@prefix : http://ex.com/  
:flipper rdf:type :mammal .  
_:s1 rdf:type rdf:Statement ;  
    rdf:subject :flipper;  
    rdf:predicate :type;  
    rdf:object :mammal;  
    :probability 0.9 .
```

- Reification is a term from philosophy for making concrete something that's abstract
- In CS, it has a similar meaning
- The underscore prefix is introduces a *blank node*
- More about this later, but for now, think of it as introducing “a new, nameless thing”



# More RDF Vocabulary

- RDF ABILITY TO describe triples through reification enables statements about statements

:john bdi:believes \_:s.

\_:s rdf:type rdf:Statement.

\_:s rdf:subject <http://ex.com/catalog/widgetX>.

\_:s rdf:predicate cat:salePrice .

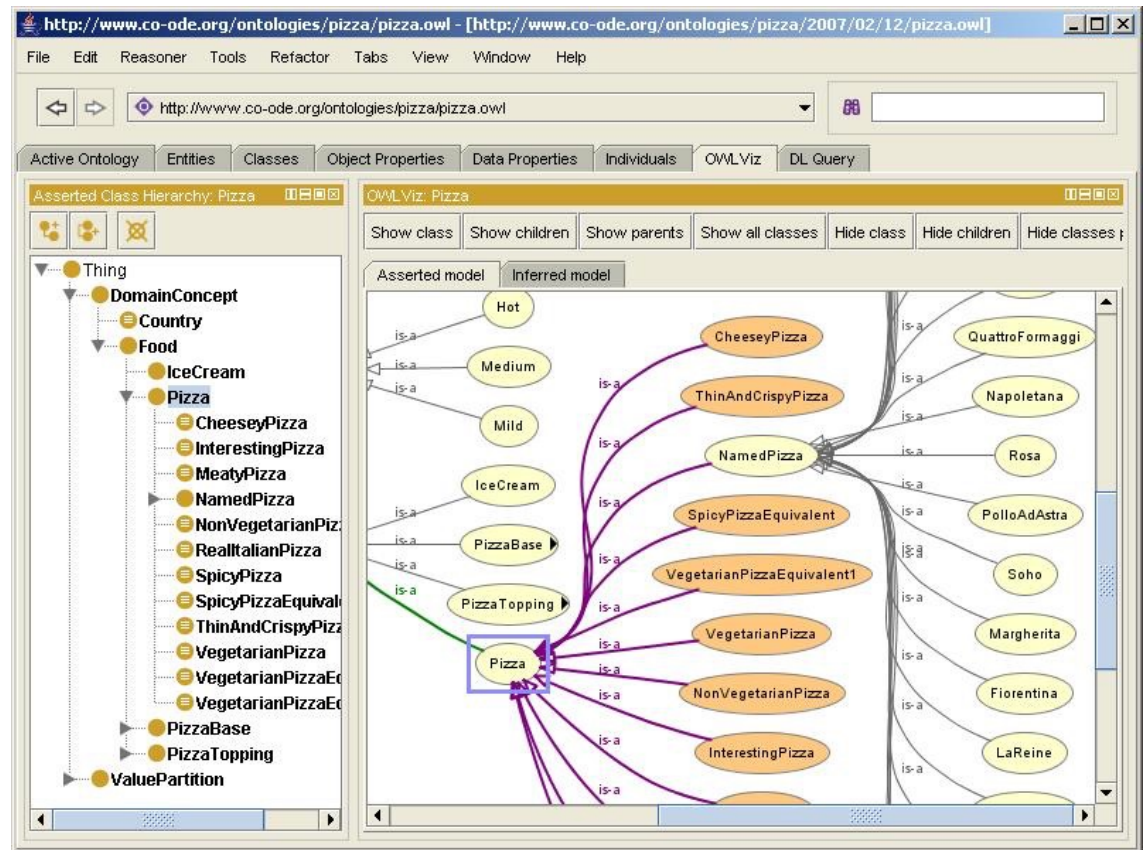
\_:s rdf:object "19.95" .

# RDF\* / RDF-star

- RDF\* (aka RDF-star) provides an easier syntax to make assertions about assertions
- This avoids having to use explicit reification
- It's been implemented by most popular RDF database systems (e.g., Jena, Stardog, RDFox)
- It's currently being considered for formal approval as a W3C standard

# RDF Schema (RDFS)

- [RDF Schema](#) adds taxonomies for classes & properties
  - **subClass** and **subProperty**
- and some **metadata**
  - **domain** and **range** constraints on properties
- Many widely used KG tools can import and export in RDFS



## Stanford [Protégé](#) KB editor

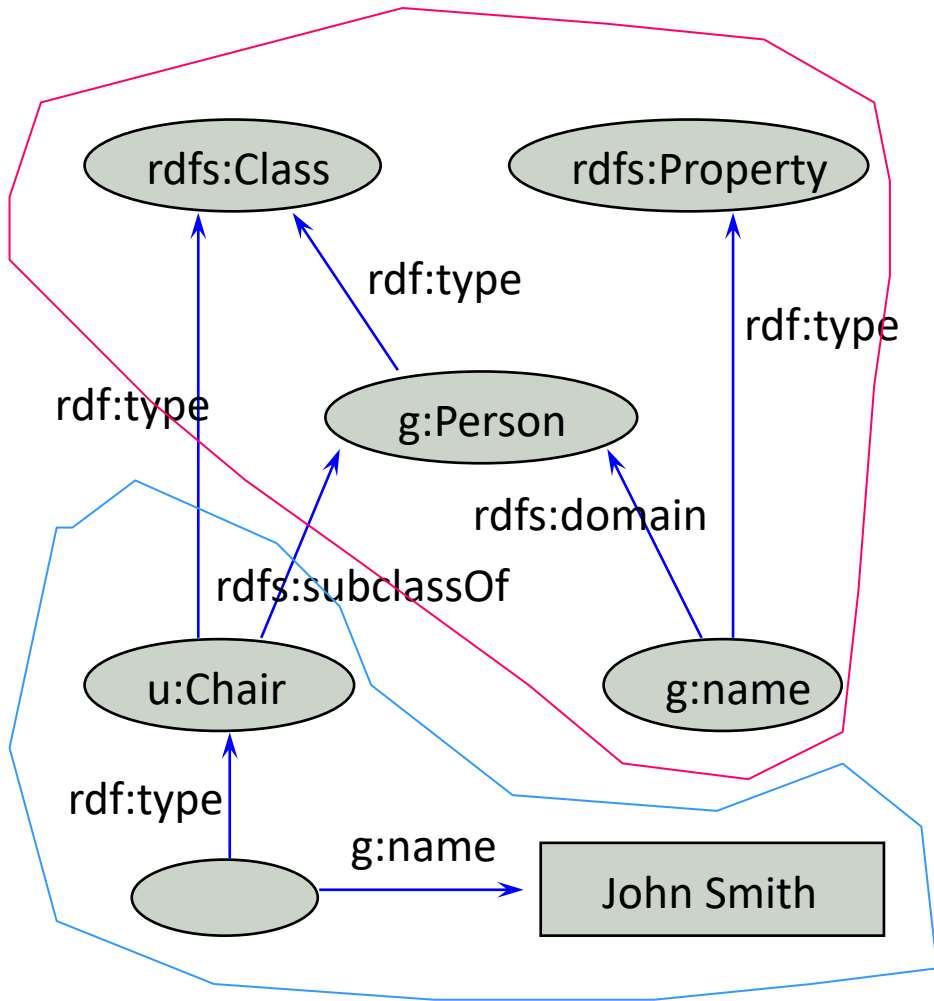
- Java, open sourced, free
- extensible, many plug-ins
- provides reasoning & server capabilities
- Local and web-based version

# RDFS Vocabulary

RDFS introduces the following terms and gives each a meaning w.r.t. the rdf data model

- Terms for classes
  - [rdfs:Class](#)
  - [rdfs:subClassOf](#)
- Terms for properties
  - [rdfs:domain](#)
  - [rdfs:range](#)
  - [rdfs:subPropertyOf](#)
- Special classes
  - [rdfs:Resource](#)
  - [rdfs:Literal](#)
  - [rdfs:Datatype](#)
- Terms for collections
  - [rdfs:member](#)
  - [rdfs:Container](#)
  - [rdfs:ContainerMembershipProperty](#)
- Special properties
  - [rdfs:comment](#)
  - [rdfs:seeAlso](#)
  - [rdfs:isDefinedBy](#)
  - [rdfs:label](#)

# RDF and RDF Schema



## Schema-level information

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

```
@prefix rdfs:
```

```
<http://www.w3.org/2000/01/rdf-schema#> .
```

```
@prefix g: <http://schema.org/gen> .
```

```
@prefix u: <http://schema.org/univ> .
```

```
g:name rdf:type rdfs:Property;  
rdfs:domain g:Person .
```

```
u:Chair rdfs:subClassOf g:Person .
```

## Instance-level information

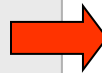
```
_:john rdf:type u:Chair;  
g:name "John Smith" .
```

# RDFS supports simple inferences



- An RDF ontology plus some RDF statements may imply additional RDF statements
- Not true of XML data
- Note that this is **part of the data model** and not of the accessing or processing code

```
@prefix rdfs: <http://www...>.
@prefix : <...genesis.n3>.
:parent rdfs:domain :Person;
        rdfs:range :Person.
:mother
  rdfs:subProperty :parent;
  rdfs:domain :Woman.
:eve :mother :cain.
```



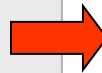
- The subject of a :person relation must be a :Person
- The object of a :person relation must be a :Person
- A :mother relation is a special kind of :parent relation
- The subject of a :mother relation must be a :Woman

# RDFS supports simple inferences



- An RDF ontology plus some RDF statements may imply additional RDF statements
- Not true of XML data
- Note that this is **part of the data model** and not of the accessing or processing code

```
@prefix rdfs: <http://www...>.
@prefix : <...genesis.n3>.
:parent rdfs:domain :Person;
        rdfs:range :Person.
:mother
  rdfs:subProperty parent;
  rdfs:domain :Woman.
:eve :mother :cain.
```



## New triples inferred from the RDFS semantics

```
:parent a rdf:Property.
:Person a rdf:Class.
:Woman rdfs:subClassOf Person.
:mother a rdf:Property.
:eve a :Person;
      a :Woman;
      :parent :cain.
:cain a :Person.
```

# RDFS Terms

- Information on the RDFS vocabulary is given by the file its prefix resolves to
- <https://www.w3.org/2000/01/rdf-schema>
- It provides some insight, e.g., rdfs: domain goes from a rdfs:Property to a rdfs:Resource
- Not a formal definition though; that's given in logic



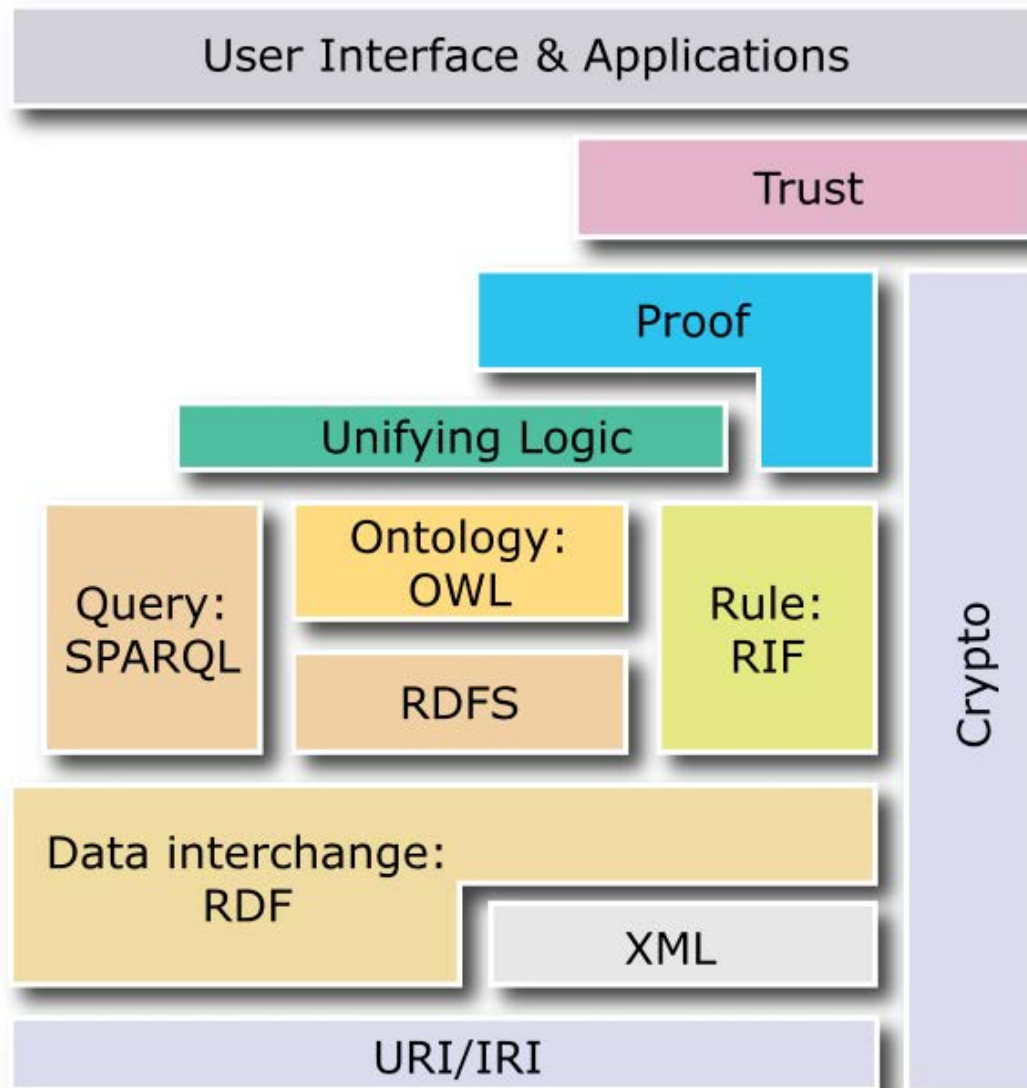
# Is RDF(S) better than XML?

Q: For a specific application, should I use XML or RDF?

A: It depends...

- XML's model is
  - a tree, i.e., a strong hierarchy
  - applications may rely on hierarchy position
  - relatively simple syntax and structure
  - not easy to *combine* trees
- RDF's model is
  - a *loose* collections of relations
  - applications may do database-like search
  - not easy to recover hierarchy
  - easy to combine relations in one big collection
  - great for the integration of heterogeneous information

# W3C Semantic Web Stack



# Problems with RDFS

- RDFS **too weak** to describe resources in detail, e.g.
  - No *localised range and domain* constraints  
Can't say that the range of hasChild is person when applied to persons and dog when applied to dogs
  - No *existence/cardinality* constraints  
Can't say that all *instances* of person have a mother that is also a person, or that persons have exactly two parents
  - No *transitive, inverse or symmetrical* properties  
Can't say isPartOf is a transitive property, hasPart is the inverse of isPartOf or touches is symmetrical
- We need RDF terms providing these and other features.

# W3C's Web Ontology Language (OWL)

- DARPA project, DAML+OIL, begat OWL
- OWL released as W3C recommendation 2/10/04
- See the [W3C OWL pages](#) for overview, guide, specification, test cases, etc.
- Three layers of OWL are defined of decreasing levels of complexity and expressiveness
  - **OWL Full** is the whole thing
  - **OWL DL** (Description Logic) introduces restrictions
  - **OWL RL** is a subset of OWL that can be efficiently implemented with rules (e.g., [horn clauses](#))
- Owl 2 became a W3C recommendation in 2009, updated in 2012

# OWL ↔ RDF

- An OWL document is a set of RDF statements
  - OWL defines semantics for certain statements
  - Does **NOT** restrict what can be said; documents can include arbitrary RDF
  - But no OWL semantics for non-OWL statements
- Adds capabilities common to [description logics](#), e.g., cardinality constraints, defined classes, equivalence, disjoint classes, etc.
- Supports ontologies as objects (e.g., importing, versioning, ...)
- A complete OWL reasoning is significantly more complex than a complete RDFS reasoner

# OWL ↔ RDF

- RDF allows us to define instance-level data
- RDFS adds the ability to add some schema-level data
- OWL extends this to allow much more schema-level information
- We typically use RDFS and OWL to define domain **ontologies** (i.e., schemas)
- And then use those ontologies to state information about **instances**

# Embedding Semantic Data in HTML

- Embedding semantic data in HTML allows documents to be understood by people and machines
  - RDFa is a ‘standard’ for embedding RDF in HTML as tag attributes
  - JSON-LD is a ‘standard’ for embedding RDF in a simple json-compatible serialization
- Facebook looks for embedded RDFa statements using its opengraph (og) vocabulary
- Bestbuy embeds produce info in RDFa

# Detecting semantic data via a browser

The screenshot displays the Allrecipes website interface for the recipe "Apple Pie by Grandma Ople". The browser's address bar shows the URL: [allrecipes.com/recipe/12682/apple-pie-by-grandma-ople/](https://www.allrecipes.com/recipe/12682/apple-pie-by-grandma-ople/). The page features a navigation bar with "allrecipes" logo, a search bar, and a "Create a profile" button. The main content area includes the recipe title, a star rating of five stars, and statistics: "22k made it | 10k reviews | 3k photos". The recipe is attributed to "MOSHASMAMA" with a user profile picture and a count of 94. A quote from the user reads: "This was my grandmother's apple pie recipe. I have never seen another one quite like it. It will always be my favorite and has won me several first place prizes in local competitions. I hope it becomes one of your favorites as well!". A large image of the pie is shown with a "Watch" button overlaid. Below the image is a "Featured in Allrecipes Magazine" banner. The recipe card includes buttons for "Save", "I Made It", "Rate it", and "Print". The "Ingredients" section lists "1 recipe pastry for a 9 inch double crust pie" and "1/2 cup unsalted butter". A "On Sale" section for AmazonFresh is visible, indicating "What's on sale near you." for "AmazonFresh Groceries delivered to your door" in "BALTIMORE, MD 21250". A "Recommended" section on the right shows "Apple Butter Spice Cake" and "Cinnamon Bread I". The footer of the page contains a long URL: [https://www.allrecipes.com/recipe/13801/apple-butter-spice-cake/?clickId=right\\_rail0&internalSource=rr\\_feed\\_recipe\\_sb&referrerId=12682&referrerContentId=recipe](https://www.allrecipes.com/recipe/13801/apple-butter-spice-cake/?clickId=right_rail0&internalSource=rr_feed_recipe_sb&referrerId=12682&referrerContentId=recipe)

<https://www.allrecipes.com/recipe/12682/apple-pie-by-grandma-ople/>



# Detecting semantic data via a browser

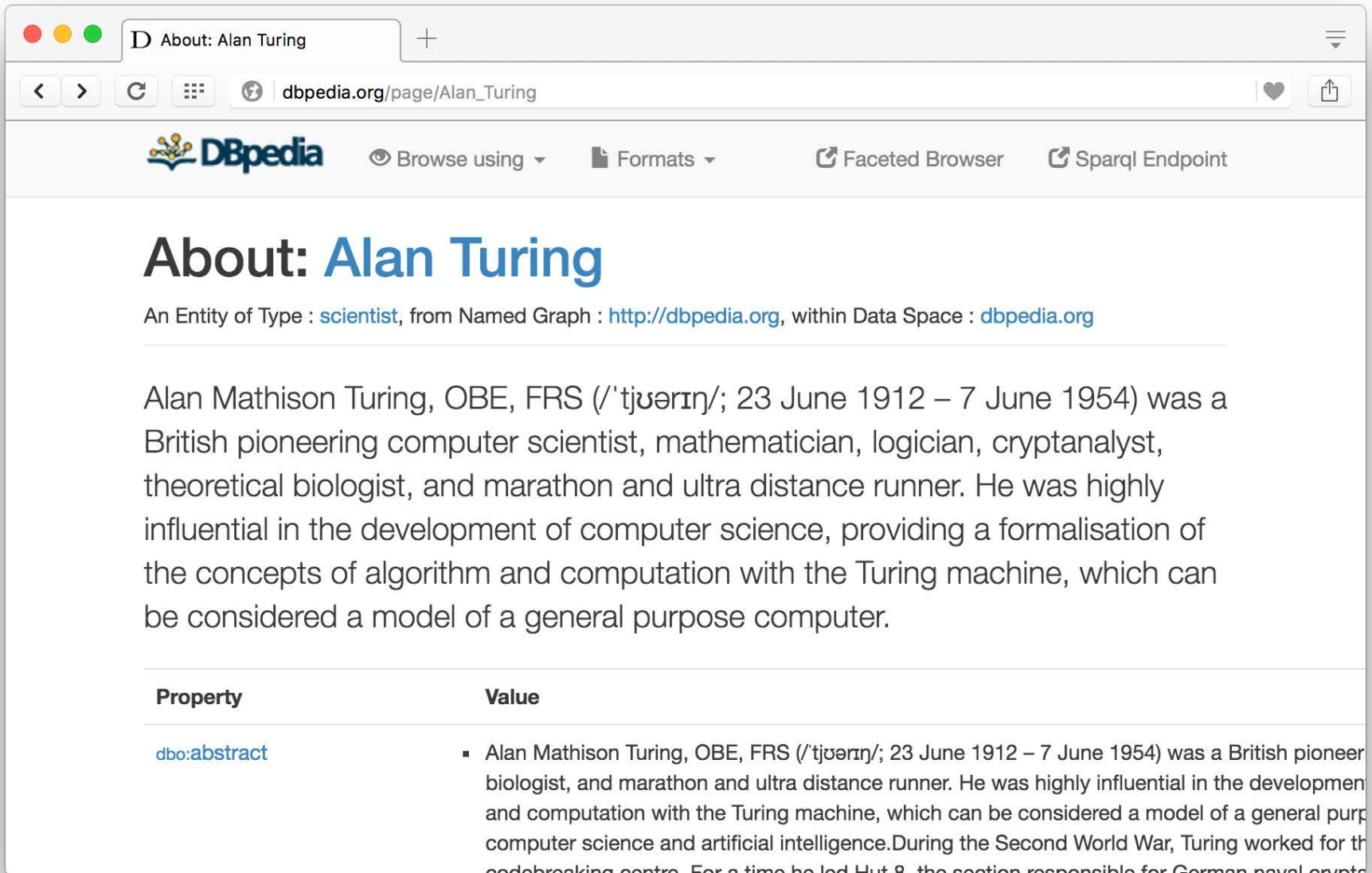
Structured data sniffer  
Shows many kinds of embedded data

Structured data testing  
Shows/debugs schema.org

Apple Pie by Grandma Ople  
★★★★★  
22k made it | 10k reviews | 3k photos  
Recipe by: MOSHASMAMA  
94  
"This was my grandmother's apple pie recipe. I have never seen another one quite like it. It will always be my favorite and has won me several first place prizes in local competitions. I hope it becomes one of your favorites as well!"  
Featured in Allrecipes Magazine — Subscribe!  
Save I Made It Rate it Print  
Ingredients 1 h 30 m 8 servings 512 cals  
+ 1 recipe pastry for a 9 inch double crust pie  
+ 1/2 cup unsalted butter  
On Sale AmazonFresh Groceries delivered to your door BALTIMORE, MD 21250  
Apple Butter Spice Cake 133  
Cinnamon Bread I 897  
By Shirley Fit... By Carol

<https://www.allrecipes.com/recipe/12682/apple-pie-by-grandma-ople/>

# Semantic Data Browser/Query



The screenshot shows a web browser window with the address bar containing "dbpedia.org/page/Alan\_Turing". The page title is "About: Alan Turing". The DBpedia logo is visible in the top left. The main content area displays the title "About: Alan Turing" in large blue text, followed by a subtitle: "An Entity of Type : [scientist](#), from Named Graph : <http://dbpedia.org>, within Data Space : [dbpedia.org](#)". Below this is a paragraph of text describing Alan Turing. At the bottom, there is a table with two columns: "Property" and "Value". The first row shows the property "dbo:abstract" and its corresponding value, which is a truncated version of the paragraph above.

DBpedia

Browse using Formats Faceted Browser Sparql Endpoint

## About: Alan Turing

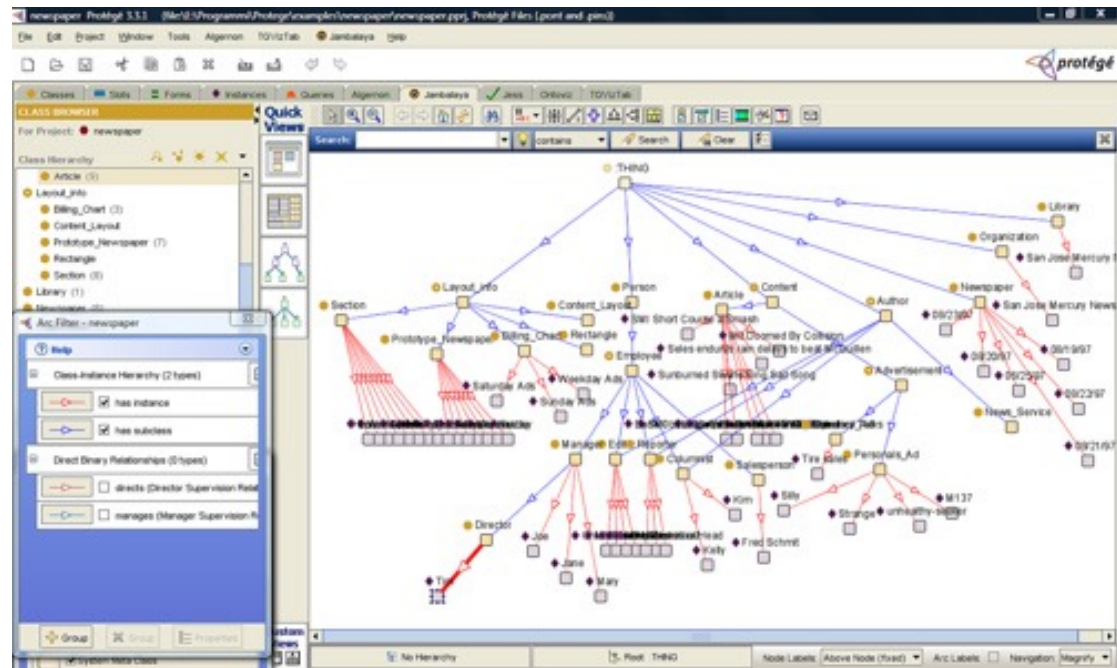
An Entity of Type : [scientist](#), from Named Graph : <http://dbpedia.org>, within Data Space : [dbpedia.org](#)

Alan Mathison Turing, OBE, FRS (/ˈtʃʊərɪŋ/; 23 June 1912 – 7 June 1954) was a British pioneering computer scientist, mathematician, logician, cryptanalyst, theoretical biologist, and marathon and ultra distance runner. He was highly influential in the development of computer science, providing a formalisation of the concepts of algorithm and computation with the Turing machine, which can be considered a model of a general purpose computer.

| Property                     | Value   |
|------------------------------|---|
| <a href="#">dbo:abstract</a> | <ul style="list-style-type: none"><li>Alan Mathison Turing, OBE, FRS (/ˈtʃʊərɪŋ/; 23 June 1912 – 7 June 1954) was a British pioneer biologist, and marathon and ultra distance runner. He was highly influential in the development and computation with the Turing machine, which can be considered a model of a general purpose computer science and artificial intelligence. During the Second World War, Turing worked for the codebreaking centre. For a time he led Hut 8, the section responsible for German naval crypt</li></ul> |

# Ontology Editor

- There are a number of editors available for creating and editing ontologies and data
- We recommend using [Protégé](#), a java-based free system developed at Stanford
  - Good support for reasoning
  - Lots of plugins



# RDF Triple Stores

- A triple store is a database for RDF triples
- It usually has a native API and often accepts SPARQL queries
- It might do reasoning, either in an *eager* manner (as triples are loaded) or *on demand* (to answer queries), etc.
- Some stores focus on scalability and others on flexibility and features
- We'll look at several, including [Apache Jena](#), [Stardog](#), [rdf4j](#), [Amazon Neptune](#), and [RDFox](#).

# Frameworks and Libraries

- There are frameworks, libraries and packages for most programming languages
- [RdfLib](#) is an excellent Python package for RDF
  - We'll try this via Jupyter notebooks and scripts
- [Jena](#) is a very comprehensive Java framework originally developed by HP and now Apache
  - Triple store, SPARQL engine, Reasoners, and more

# Conclusion

- There's quite a bit of technology needed to support the Semantic Web
- This has been a brief tour
- We'll cycle back on these and explore them in more detail
- And give you a chance to use and experiment with them