

**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) has always been 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
 - There are currently no other widely used KR languages

Questions

- Database (DB) vs. knowledge base (KB)?
 - TL;DR: DBs have facts, KBs have general knowledge and (maybe) facts
 - DBs typically have 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)
- 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), child(mary,john), ...

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
 - May be too simple

Two Semantic Web Notions

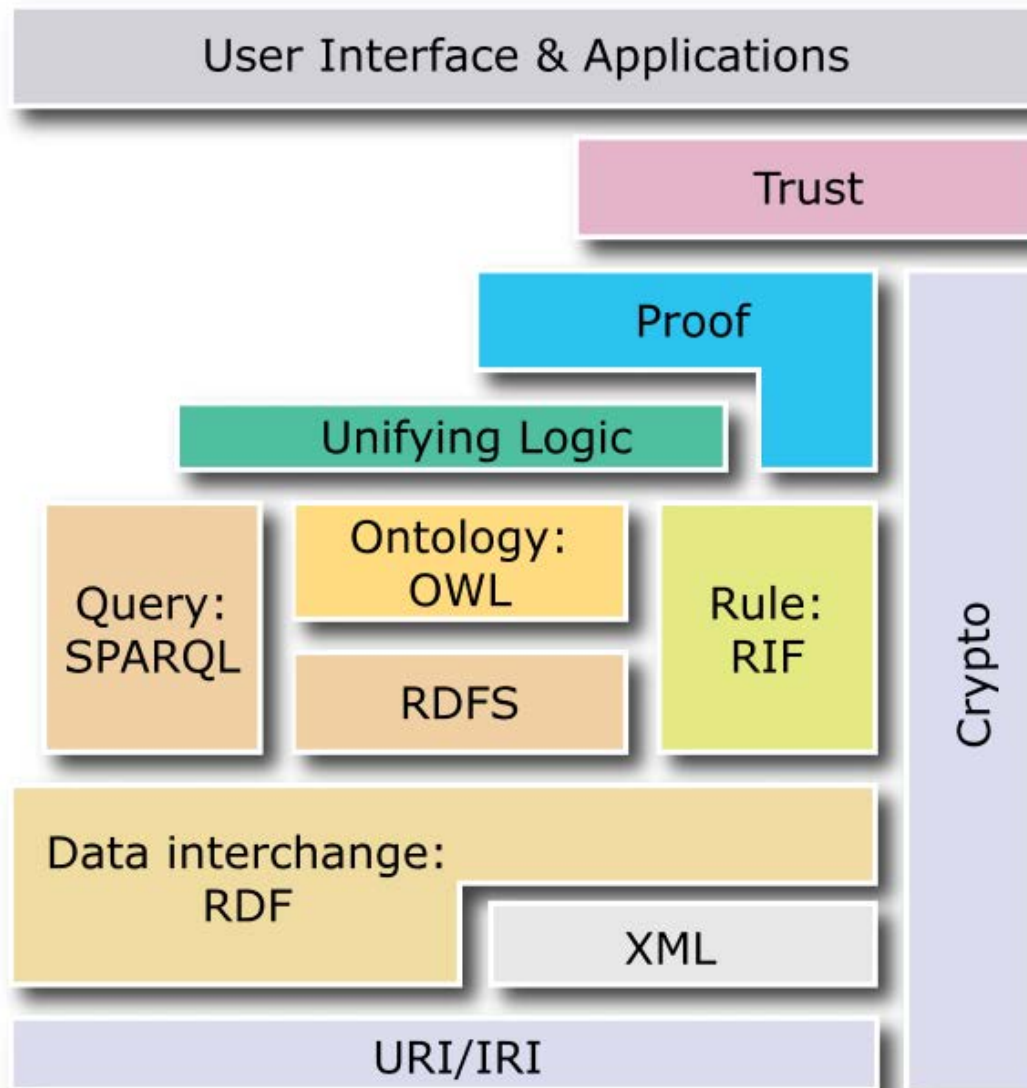
- **The semantic web**

- 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
- Using W3C supported standards, i.e., RDF, OWL, SPARQL, XML, RIF, etc.
- 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>
```

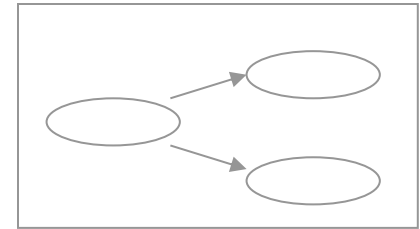
**Good for
Machine
Processing**

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)
```

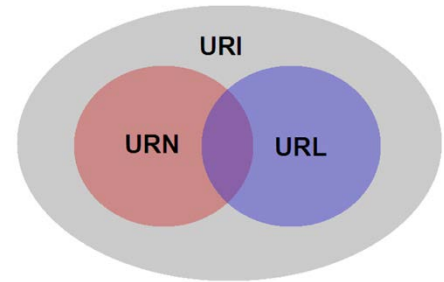
**Good For Reasoning and
Databases**

*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
- 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](#)) are a subset of URIs, used for resources that can be *accessed* on the web
- URIs look like URLs, often with fragment identifiers pointing to a document part:
 - `http://foo.com/bar/mumble.html#pitch`

URIs are a foundation

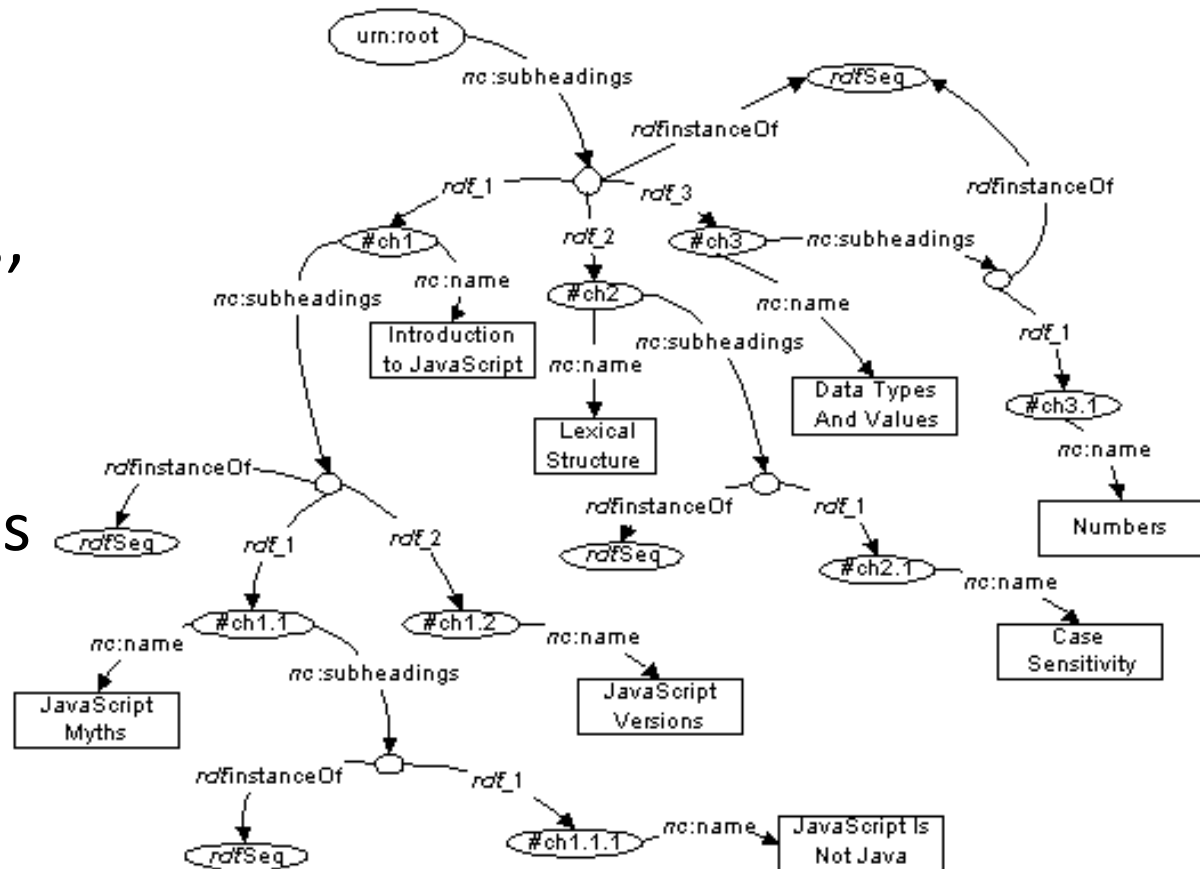
- URIs are unambiguous, unlike natural language terms -- the web provides a global **namespace**
- We can use a URI to **denote** something, e.g., a concept, entity, event or relation
- We usually assume references to the same URI are to the same thing

What does a URI mean?

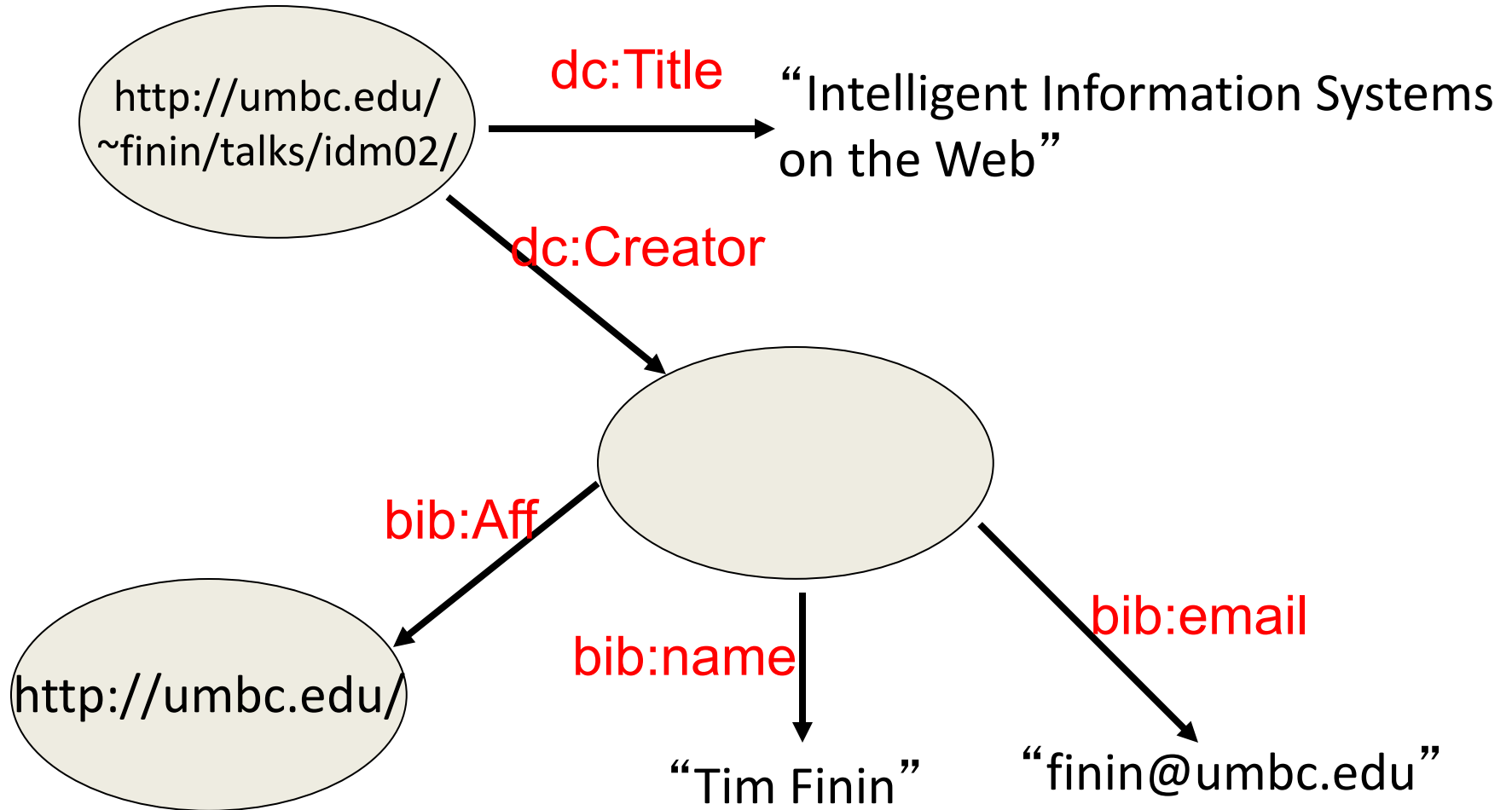
- Sometimes URIs denote a web resource
 - <http://umbc.edu/~finin/finin.jpg> denotes a file
 - We can use RDF to make assertions about the resource, e.g., it's an image and depicts a person with name Tim Finin, ...
- Sometimes concepts in the external world
 - E.g., <http://umbc.edu/> denotes a particular university located in Baltimore
 - This is done by social convention
- 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
- 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>
```

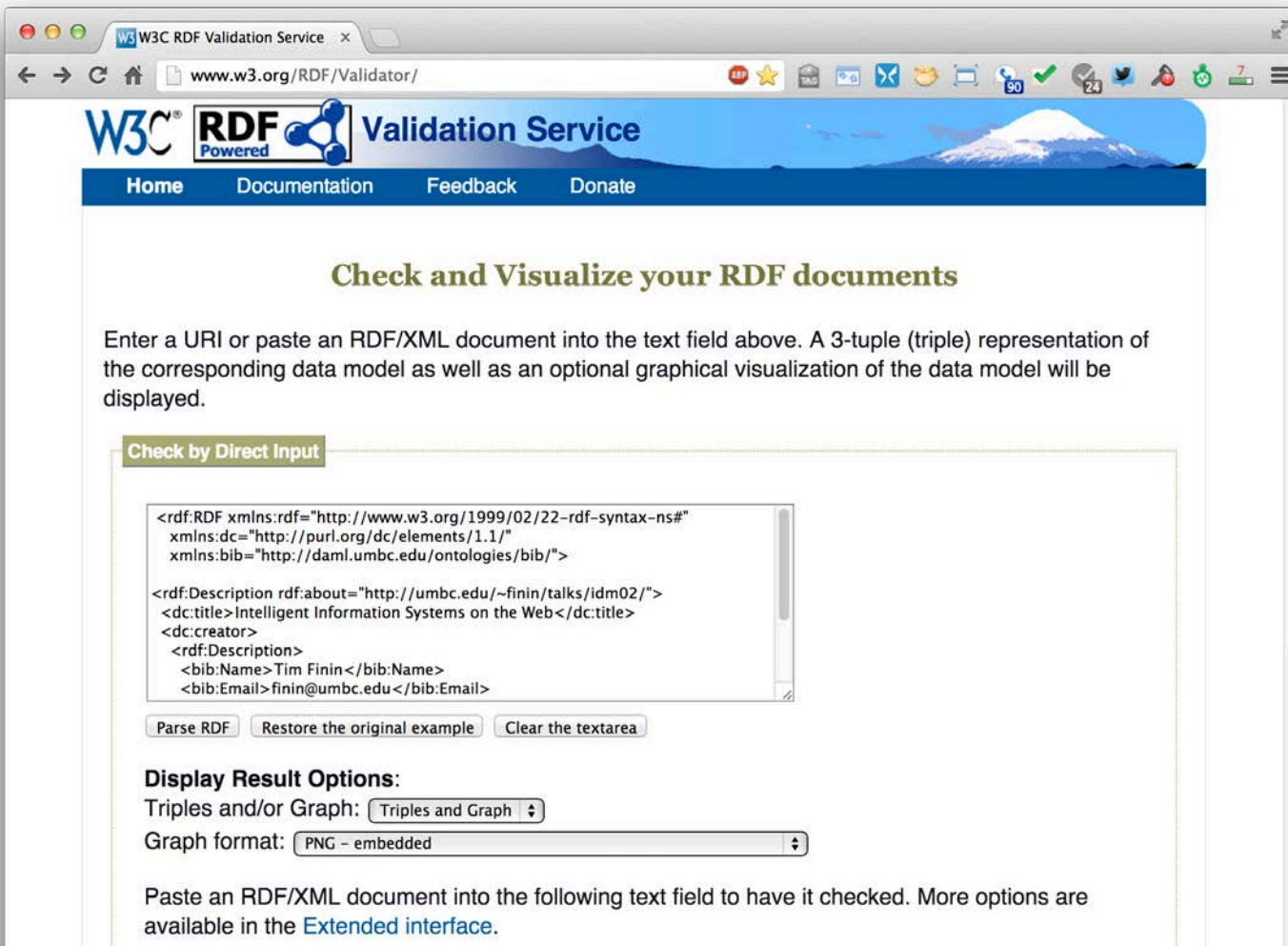
Note the prefix declarations

```
<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 vocabulary developed by librarians
- **bib** for terms from a bibliography vocabulary developed at UMBC

An RDF validation service



The screenshot shows the W3C RDF Validation Service web page. The browser address bar displays "www.w3.org/RDF/Validator/". The page header includes the W3C logo, "RDF Powered" text, and the "Validation Service" title. A navigation menu contains "Home", "Documentation", "Feedback", and "Donate". The main heading is "Check and Visualize your RDF documents". Below this, a paragraph explains that users can enter a URI or paste an RDF/XML document to get a 3-tuple representation and an optional graphical visualization. A section titled "Check by Direct Input" contains a text area with the following RDF/XML code:

```
<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>
      </rdf:Description>
    </dc:creator>
  </rdf:Description>
</rdf:RDF>
```

Below the text area are three buttons: "Parse RDF", "Restore the original example", and "Clear the textarea". Underneath, the "Display Result Options:" section includes a dropdown menu for "Triples and/or Graph" (set to "Triples and Graph") and another dropdown for "Graph format" (set to "PNG - embedded"). At the bottom, a note says "Paste an RDF/XML document into the following text field to have it checked. More options are available in the [Extended interface](#)."

<http://www.w3.org/RDF/Validator/>

Easy to convert between serializations

- Most software tools can read and write different serializations
- [rdf2rdf](#) is a simple handy utility for converting from one RDF serialization to another
- [Any23](#) is an open source library, web service and command line tool that extracts structured data in RDF format from a variety of Web documents

N-triple representation

- [N-triples](#) is a line-oriented serialization for RDF
- 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/> .

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/"] .

Turtle Syntax

:subj

:property1 :value1;

:property2 :value2, value3;

:property3 :value4.

More RDF Vocabulary

- RDF has terms for describing lists, bags, sequences, simple datatypes, etc.
- RDF is a “pure” graph representation language
 - Nodes and edges are simple objects
 - Both have identifiers that are URIs
- Suppose we want to associate a probability with an edge, e.g.,
 - (:flipper rdf:type :mammal) :probability 0.9
 - (:flipper rdf:type :fish) :probability 0.1

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

More RDF Vocabulary

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

:flipper rdf:type :mammal .

- All non-literals have to be URIs
- RDF uses prefixes for readability
- We can specify what a null prefix means
- If we don't it means "in this file"
- <https://prefix.cc/> is one service for finding standard prefixes

More RDF Vocabulary

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

```
:flipper rdf:type :mammal .
_:s1 rdf:type rdf:Statement .
_:s1 rdf:subject :flipper .
_:s1 rdf:predicate :type .
_:s1 rdf:object :mammal .
_:s1 :probability 0.9
```

- The underscore prefix is special
- It introduces *blank nodes*
- We'll talk about this in more detail later
- For now, think of it as introducing “a new, nameless thing”

More RDF Vocabulary

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

```
:flipper rdf:type :mammal .  
_:s1 a rdf:Statement;  
    rdf:subject :flipper;  
    rdf:predicate :type;  
    rdf:object :mammal;  
    :probability 0.9 .
```

- Lookup the rdf namespace via <https://prefix.cc/>
- Visit it on the web
- You'll see it defines 18 terms that have special meaning in RDF

More RDF Vocabulary

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

:john bdi:believes _:s.

_:s rdf:type rdf:Statement.

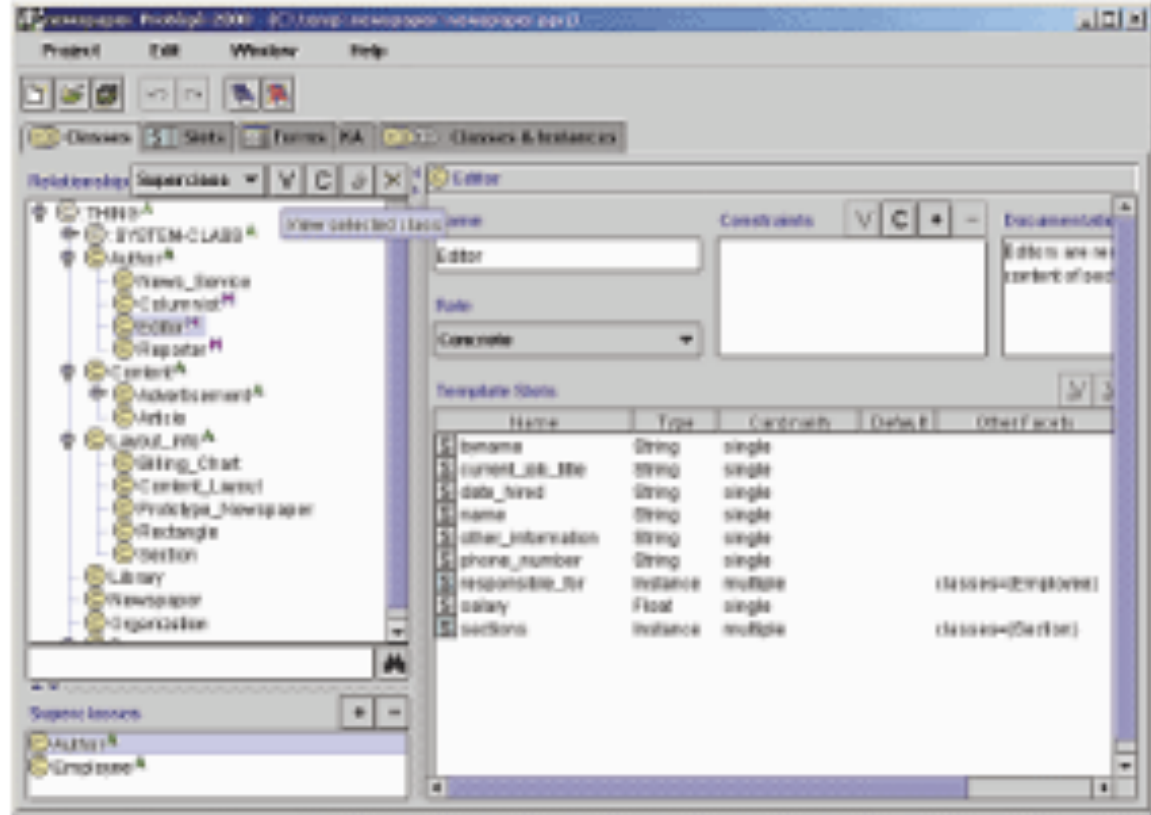
_:s rdf:subject <http://ex.com/catalog/widgetX>.

_:s rdf:predicate cat:salePrice .

_:s rdf:object "19.95" .

RDF 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

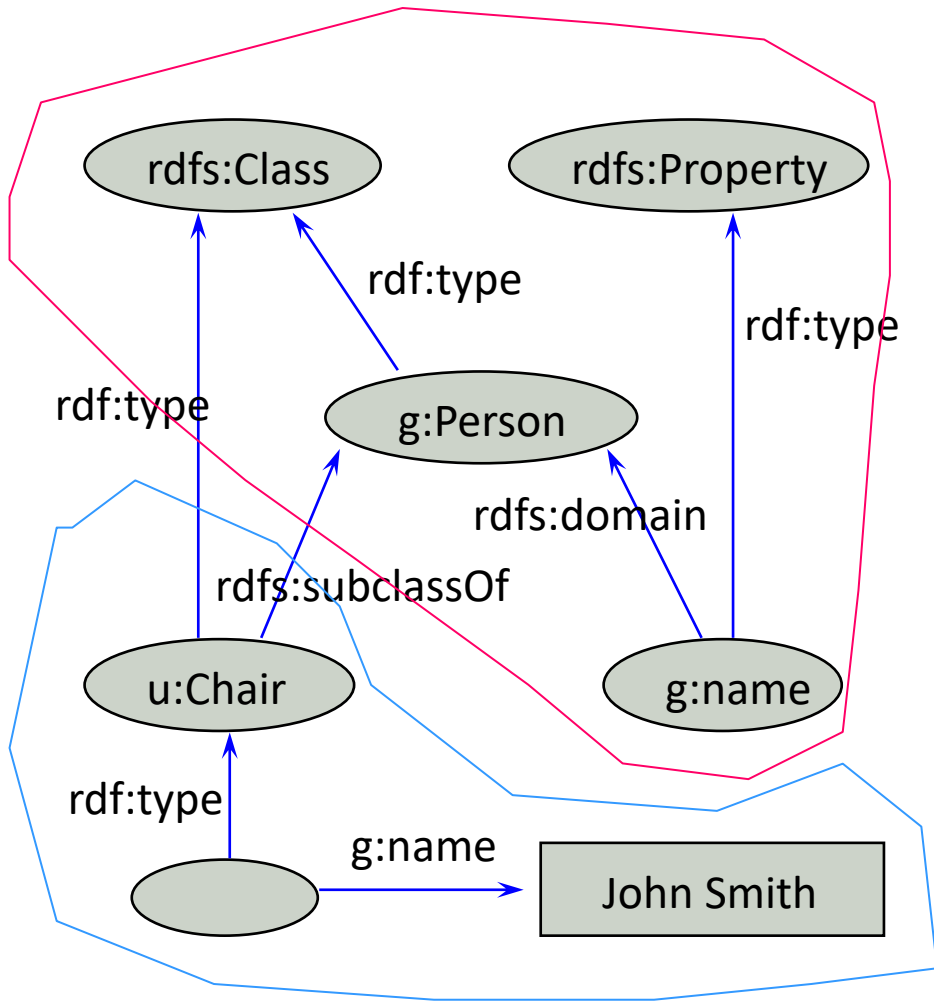
- Java, open sourced
- extensible, lots of plug-ins
- provides reasoning & server capabilities

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 rdfs:type rdfs:Property;
rdfs:domain g:Person .

u:Chair rdfs:subClassOf g:Person .

Instance-level information

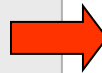
_:john rdfs:type u:Chair;
g:name "John Smith" .

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.
```



```
: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

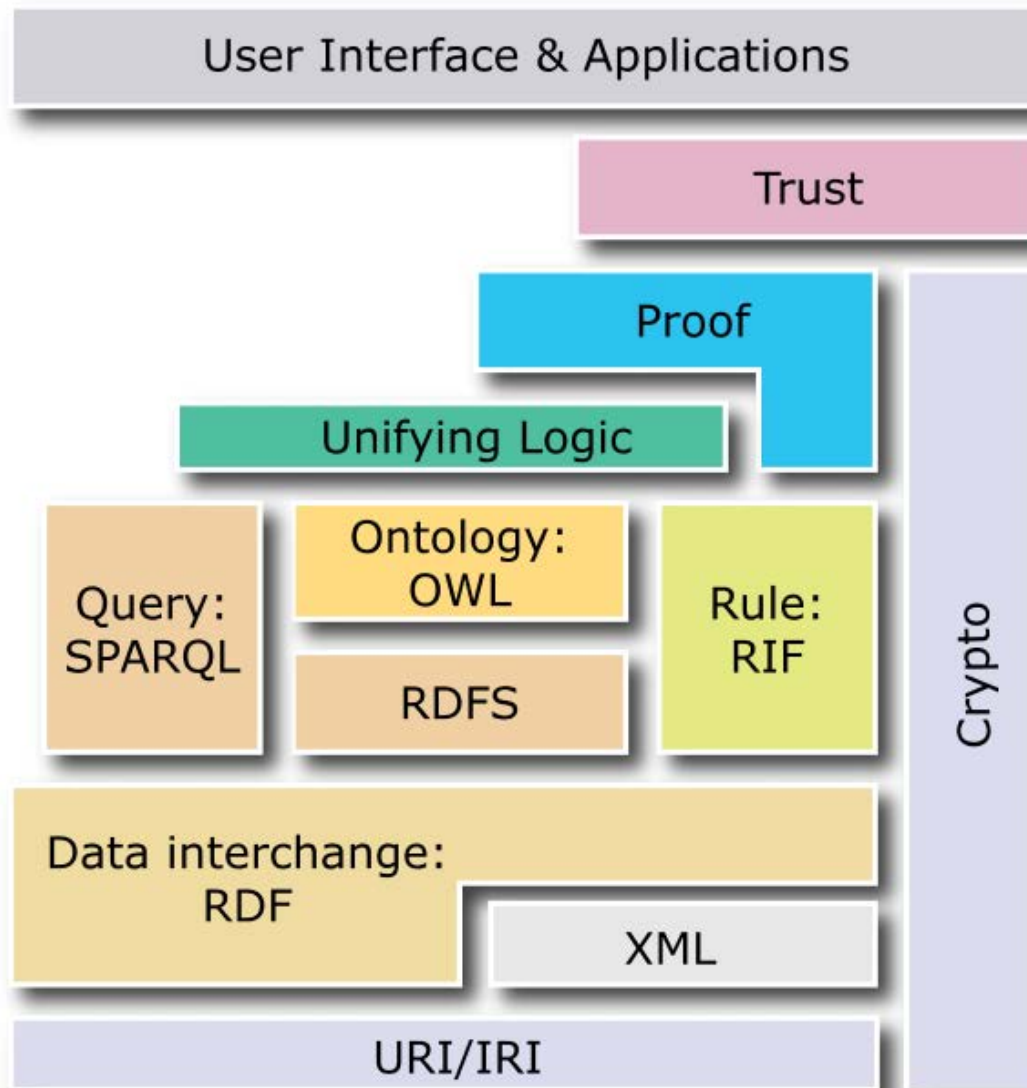
Is RDF(S) better than XML?

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 Lite** is an entry level language intended to be easy to understand and implement
- 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**
- **Aside:** I typically reserve the word *ontology* to refer to schema definitions. These can include individuals, of course, but often do not.

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 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 user options like "Create a profile". The main content area includes the recipe title, a 5-star rating, and statistics: "22k made it | 10k reviews | 3k photos". The recipe is attributed to "MOSHAMAMA" with a user profile picture and a "94" follower count. A testimonial 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. Below the image is a "Featured in Allrecipes Magazine" banner. The bottom section contains interactive buttons: "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 Amazon Fresh is also visible, indicating "What's on sale near you." in Baltimore, MD.

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

Detecting semantic data via a browser

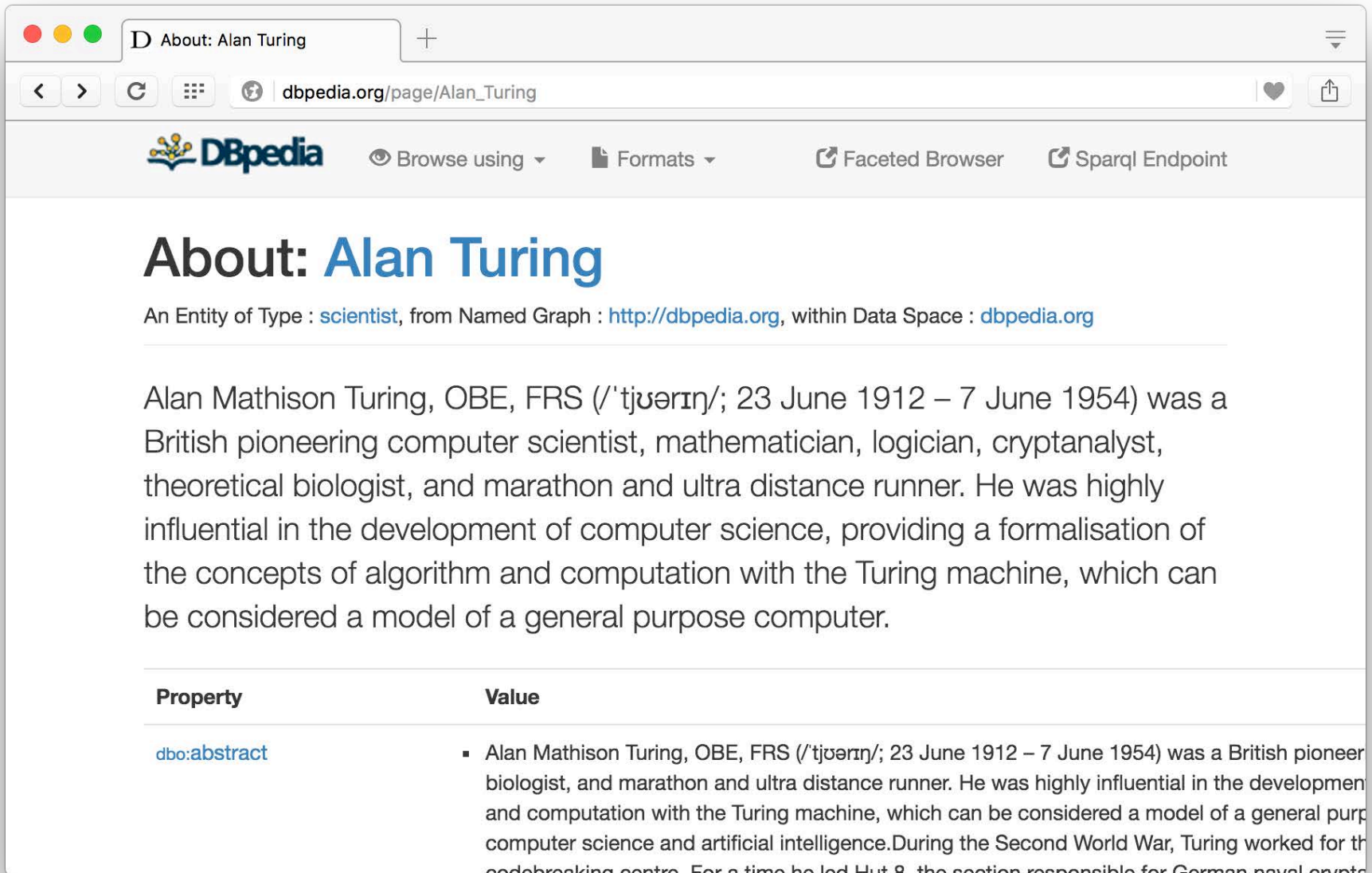
The image shows a browser window displaying the Allrecipes website. The URL in the address bar is <https://www.allrecipes.com/recipe/12682/apple-pie-by-grandma-ople/>. The page features a recipe for "Apple Pie by Grandma Ople" with a 5-star rating, 22k "made it" count, 10k reviews, and 3k photos. The recipe is by MOSHASHAMA and includes a video player with a "Watch" button. Below the recipe, there are sections for "Ingredients" (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), and "Featured in Allrecipes Magazine".

Two blue callout boxes with arrows pointing to the browser's developer tools are overlaid on the image:

- Structured data sniffer**
Shows many kinds of embedded data
- Structured data testing**
Shows/debugs schema.org

<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 displaying 'dbpedia.org/page/Alan_Turing'. The page title is 'About: Alan Turing'. The DBpedia logo is visible in the top left, and navigation options like 'Browse using', 'Formats', 'Faceted Browser', and 'Sparql Endpoint' are in the top right. The main content area features a large blue heading 'About: Alan Turing' followed by a line of metadata: '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 as a British pioneering computer scientist, mathematician, logician, cryptanalyst, theoretical biologist, and marathon and ultra distance runner. At the bottom, a table lists properties and values, with the first entry being 'dbo:abstract' and a detailed description of Alan Turing's life and work.

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 (/ˈtjʊərɪŋ/; 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
dbo:abstract	<ul style="list-style-type: none">Alan Mathison Turing, OBE, FRS (/ˈtjʊərɪŋ/; 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

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 [Sesame](#), Apache [Jena](#) and [stardog](#)

Frameworks and Libraries

- There are frameworks, libraries and packages for most programming languages
- Jena is a very comprehensive Java framework originally developed by HP and now Apache
 - Triple store, SPARQL engine, Reasoners, and more
- Others are available for Python, Ruby, C#, Perl, PHP, Prolog, Lisp, etc.

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