The Semantic Web

Damien Leprovost

Laboratoire LIMICS
Inserm – UPMC – Paris 13
http://www.damien-leprovost.fr


Lecture objectives

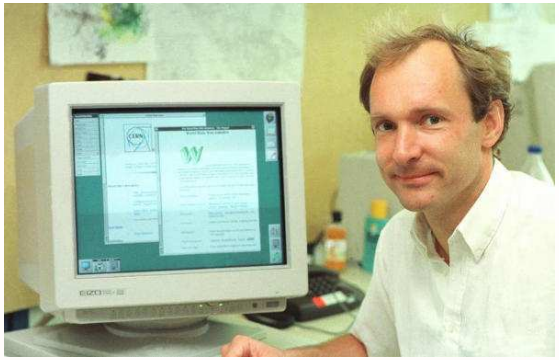
- **Understand what the Semantic Web is**
- **How to publish semantic data**
- **How to query semantic data**
- **. . . and explore the Semantic Web!**

Outline

Outline

A brief story of the World Wide Web

- The Web
  - A network of linked document
  - Proposed by Tim-Berners-Lee, 1989
  - Documents linked by hypertext (Ted Nelson, 1965)
  - Identified by their location: the URL scheme



`http://www.w3.org/Consortium/mission.html`

A brief story of the World Wide Web

- The Web 2.0
  - Interactions between users
  - A social and participative Web
  - New technologies
    - AJAX, Rich Internet Application, Wikis, ...
  - Popularized term by Tim O'Reilly, 2004
  - Exponential volume of published content!

A brief story of the World Wide Web

- Birth of the Semantic Web
  - A Web of Data
  - "A common framework that allows data to be shared and reused" (W3C, 2011)
  - Can be processed by machines
  - Formal semantics, reasoning, proof
  - Notion of resource

- Extension of the current Web
- Through W3C standards

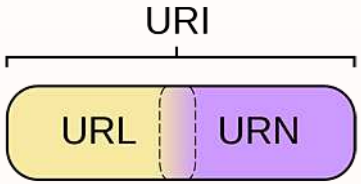
Outline

Notion of resource

- On the "early Web", the main element was the document
- An **URL** identifies where a document is on the Web
  - Uniform Resource Locator
- Nowadays, the Web references more than its own documents
  - Example: things in the real world, abstract concepts, ...

Notion of resource: URI/IRI

- **URI**: Uniform Resource Identifier
- Identify on the Web, any existing resource
- Two specializations known: URL and URN
  - Identifies a resource either by a location, or a name, or both



- **IRI** : Internationalized Resource Identifier
  - Using Unicode to allow any character

A Web of data

URI/IRI usage on the Semantic Web:

- Use HTTP URI (URL) to follow the resource
  ex: `http://dbpedia.org/resource/Eiffel_Tower`
- When URI is followed, send data about the resource
  format depending the query
- This data should contain link to other data
  (hypertext Web structure)

Outline

The World Wide Web Consortium

- 1994: creation of the The World Wide Web Consortium

- Consortium members are businesses, nonprofit organizations, universities, governmental entities, and individuals
- Purpose: Harmonize the Web
- By the development of standards, called Recommendations

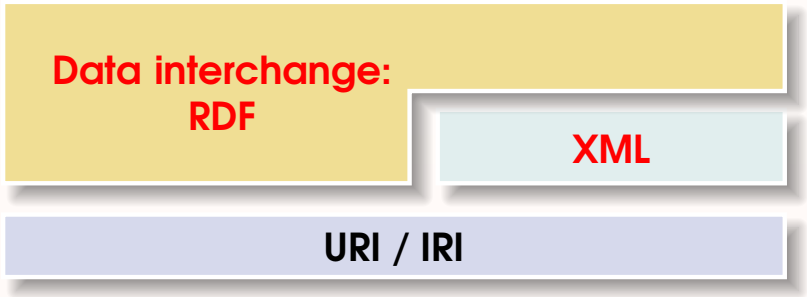
The Semantic Web Stack[1]

- Identification **URI / IRI**

[1] Tim Berners-Lee, 2008.

The Semantic Web Stack[1]

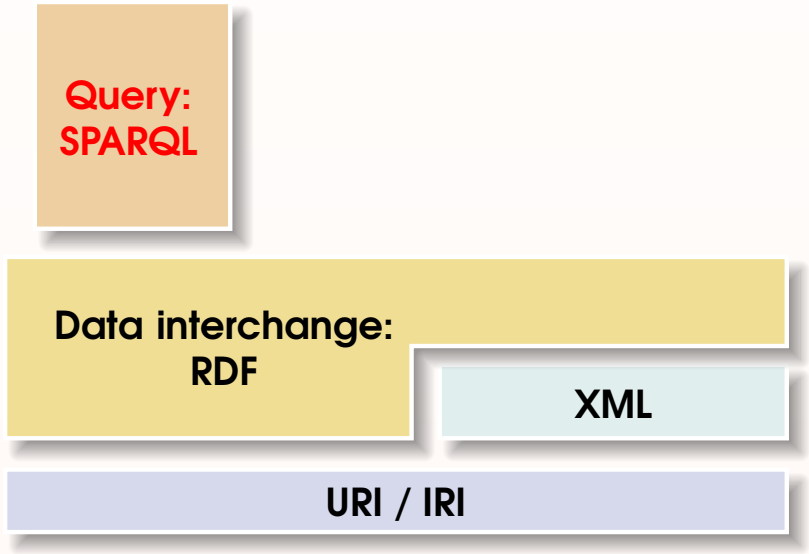
- Representation
- Identification



[1] Tim Berners-Lee, 2008.

The Semantic Web Stack[1]

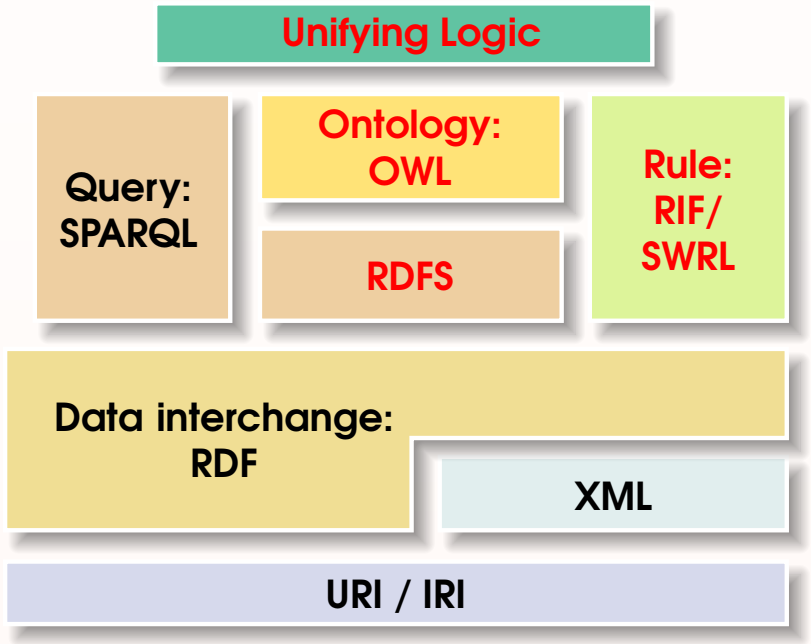
- Queries
- Representation
- Identification



[1] Tim Berners-Lee, 2008.

The Semantic Web Stack[1]

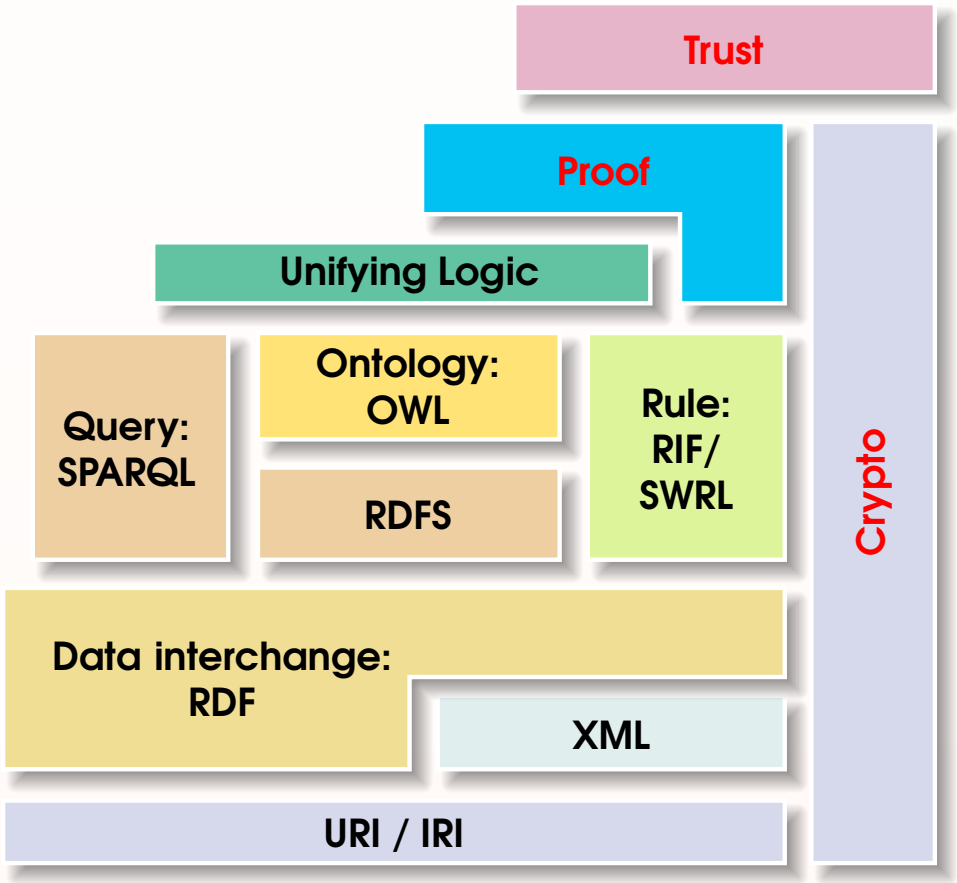
- Reasoning
- Queries
- Representation
- Identification



[1] Tim Berners-Lee, 2008.

The Semantic Web Stack[1]

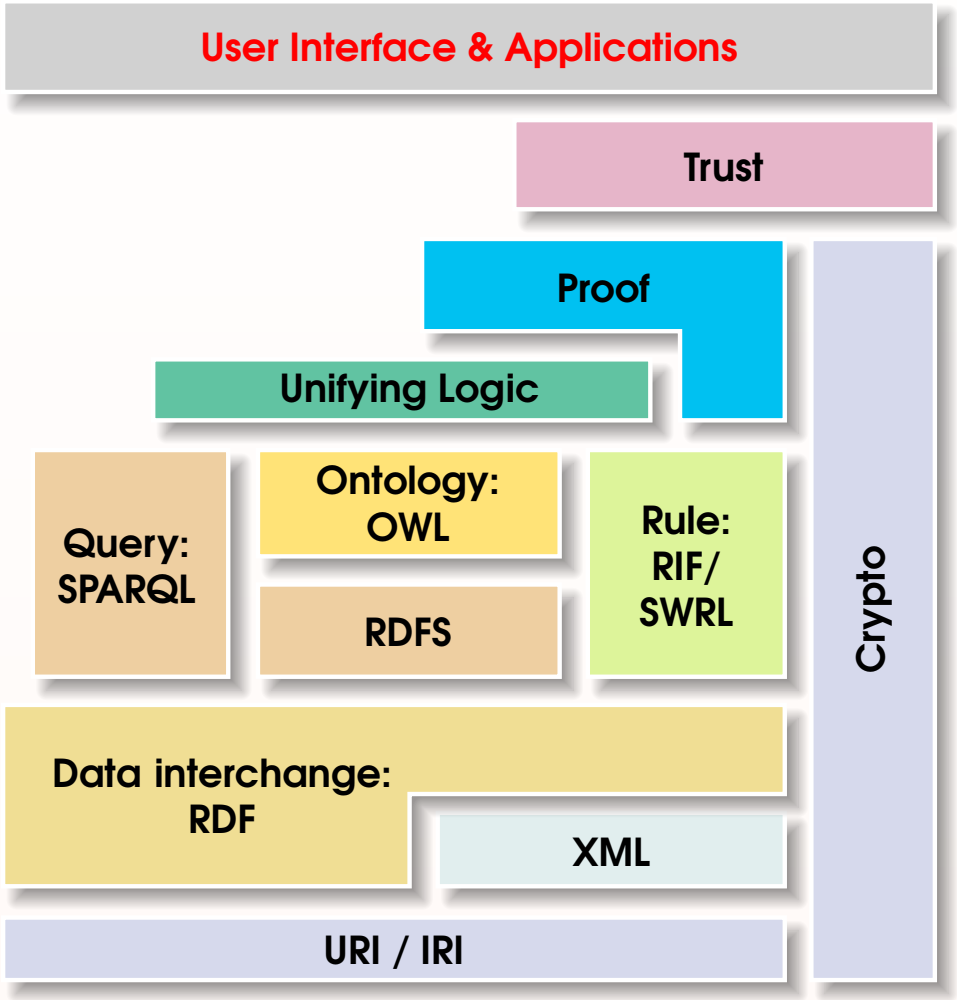
- Confidence
- Reasoning
- Queries
- Representation
- Identification



[1] Tim Berners-Lee, 2008.

The Semantic Web Stack[1]

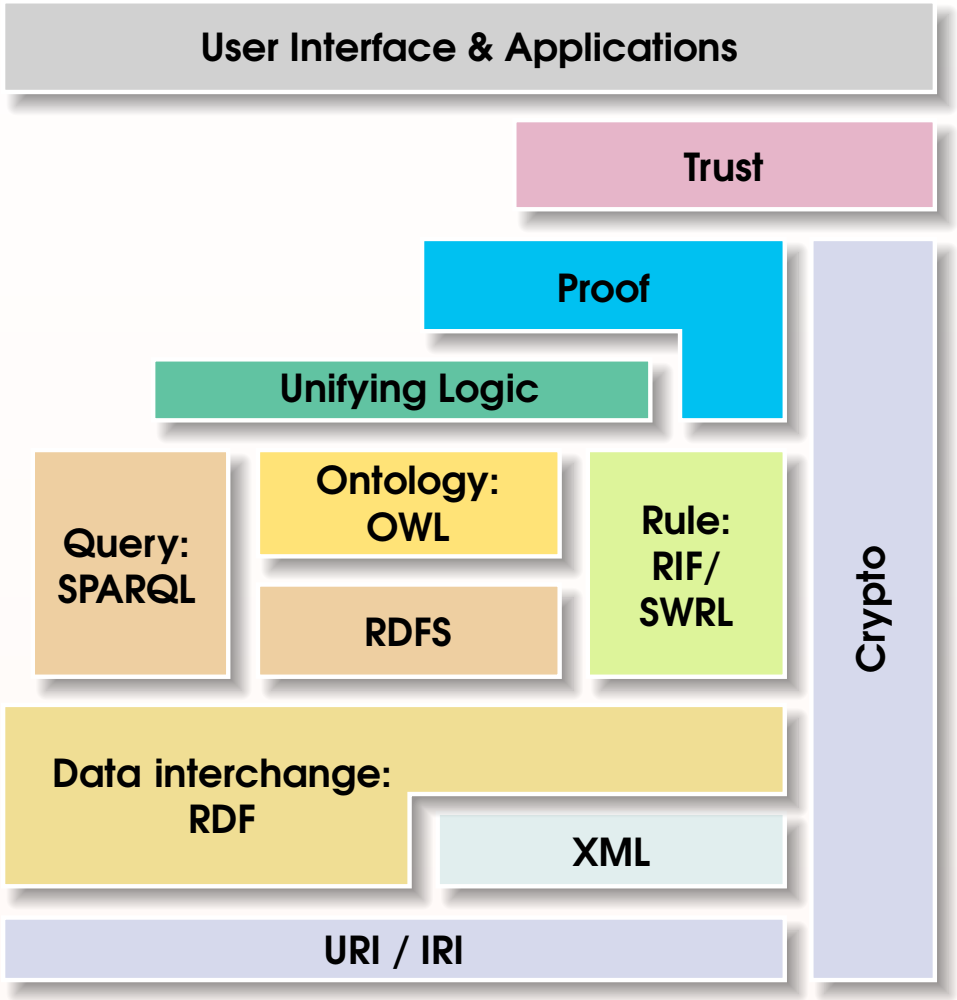
- Interaction
- Confidence
- Reasoning
- Queries
- Representation
- Identification



[1] Tim Berners-Lee, 2008.

The Semantic Web Stack[1]

- Interaction
- Confidence
- Reasoning
- Queries
- Representation
- Identification



[1] Tim Berners-Lee, 2008.

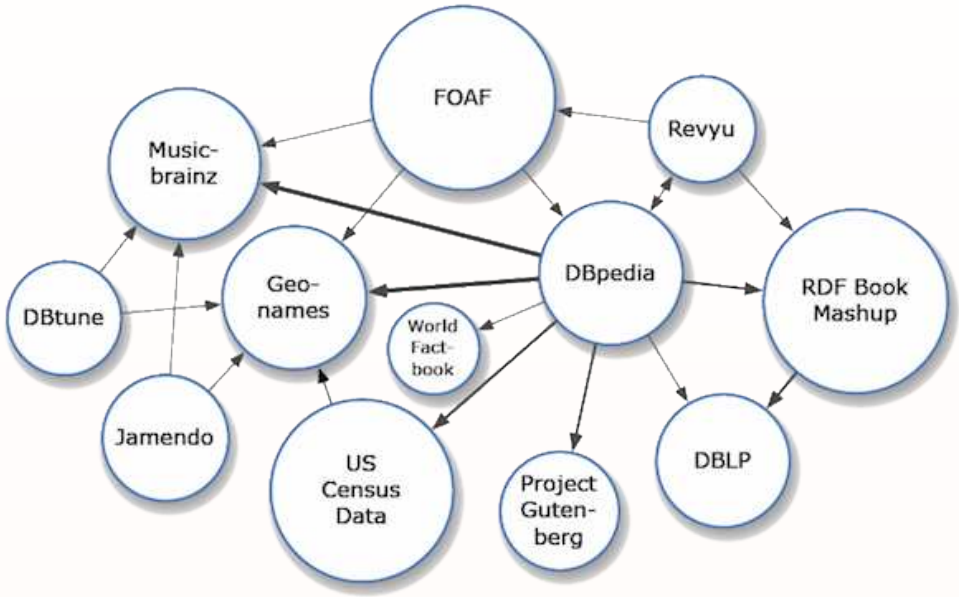
Outline

Linked Open Data

- Sets of open data are published on the Web (datasets)
- *Linking Open Data* aims to gather these data sources in one RDF graph
- Made possible by the notion of Linked Open Data

Linked Open Data[2]

May 2007:



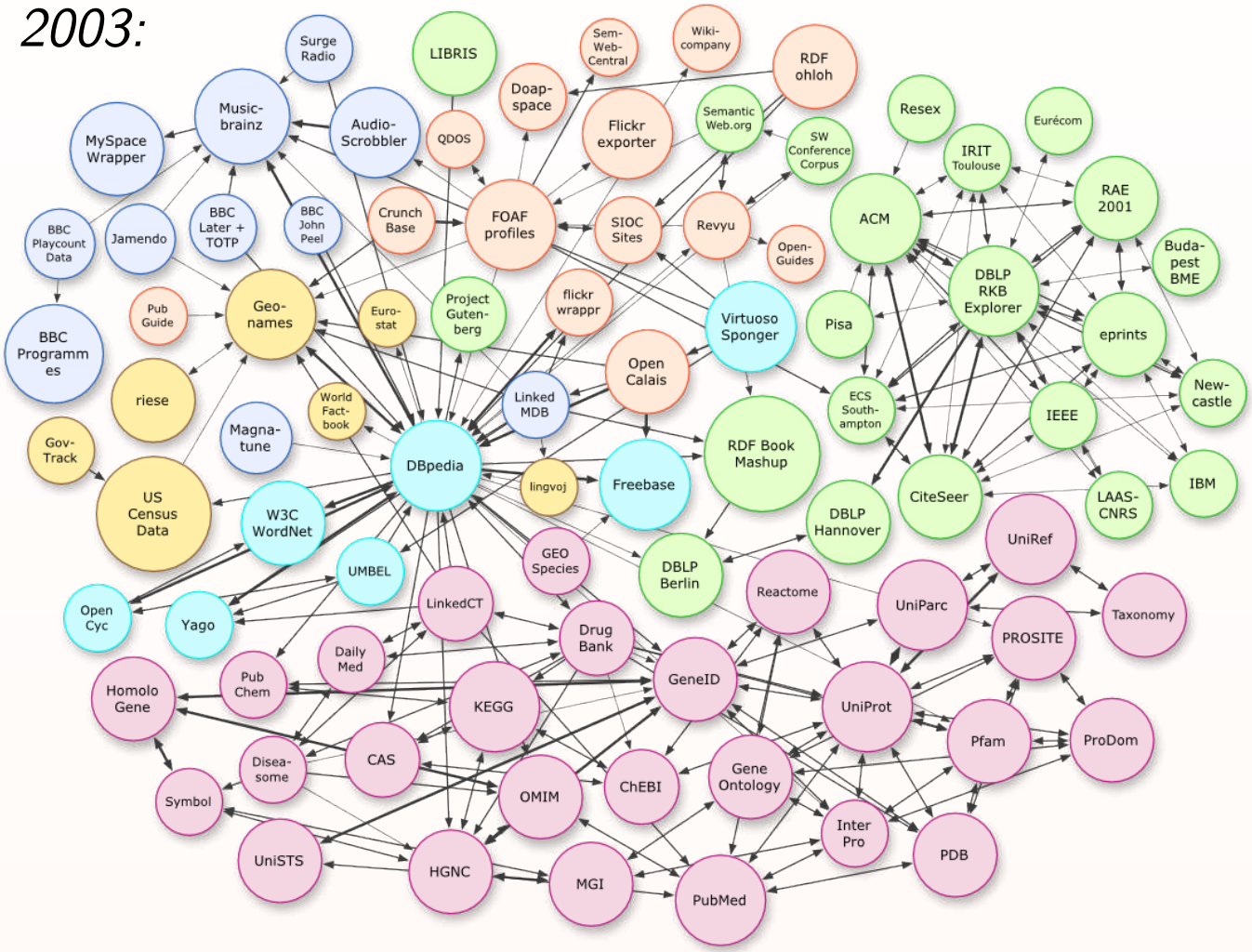
Any dataset in the graph has:

- More than 1000 triples
- More than 50 links to other datasets in the graph
- All their data accessible

[2] Linking Open Data cloud diagram: `http://lod-cloud.net`

Linked Open Data[2]

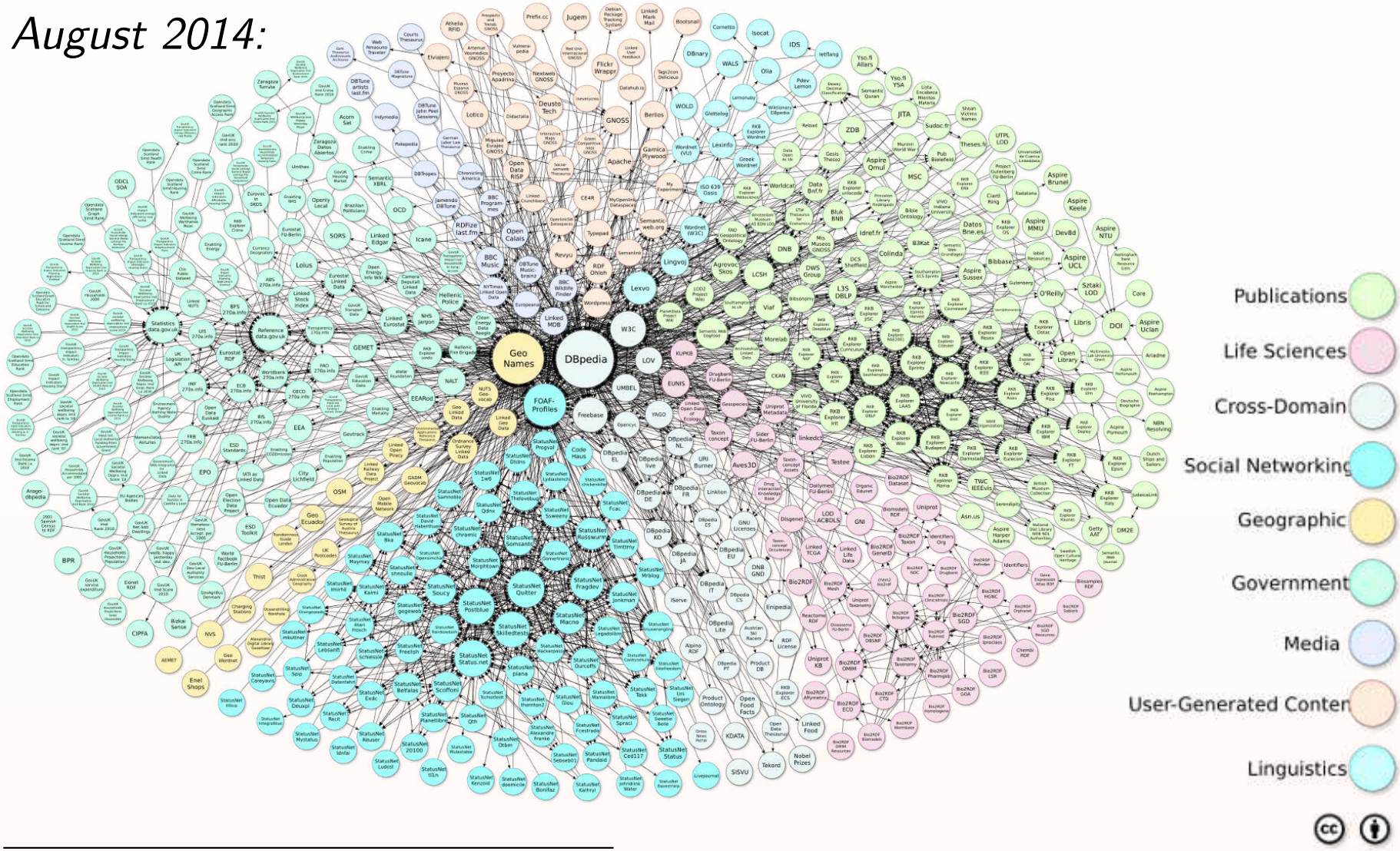
March 2003:



[2]Linking Open Data cloud diagram: `http://lod-cloud.net`

Linked Open Data[2]

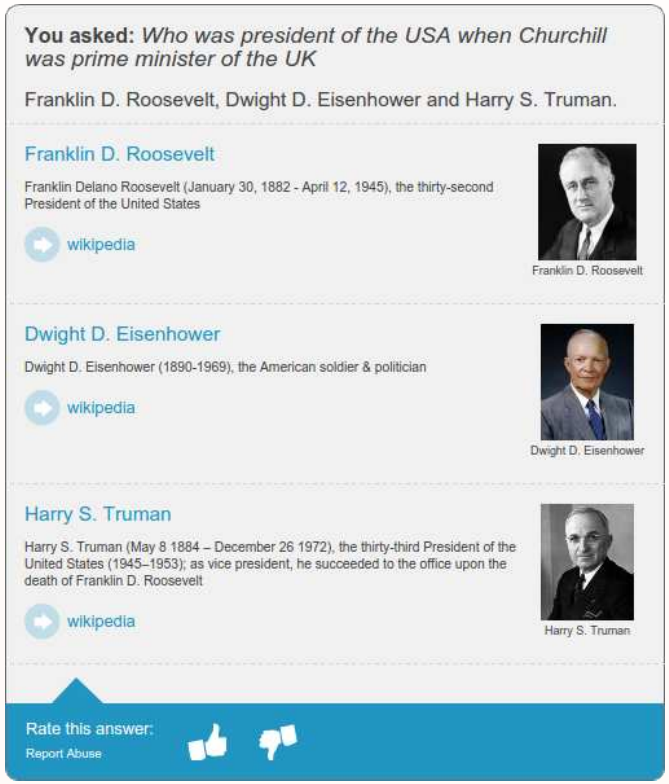
August 2014:



[2]Linking Open Data cloud diagram: `http://lod-cloud.net`

Examples

- Some popular datasets
  - DBpedia: RDF annotation of Wikipedia pages (4.58M things)
  - Geonames: geographical informations (8M placenames)
- Some applications
  - Evi: Semantic search engine
  - OpenCalais: Semantic annotation of content
  - Callimachus: Create website with RDF data



Outline

Presentation

Resource **D**escription **F**ramework

- Represent any knowledge, on the web
- Describe informations about resources
- Based on URI / IRI
- Expressed by statements

Outline

Data representation

- Descriptions are represented as triple

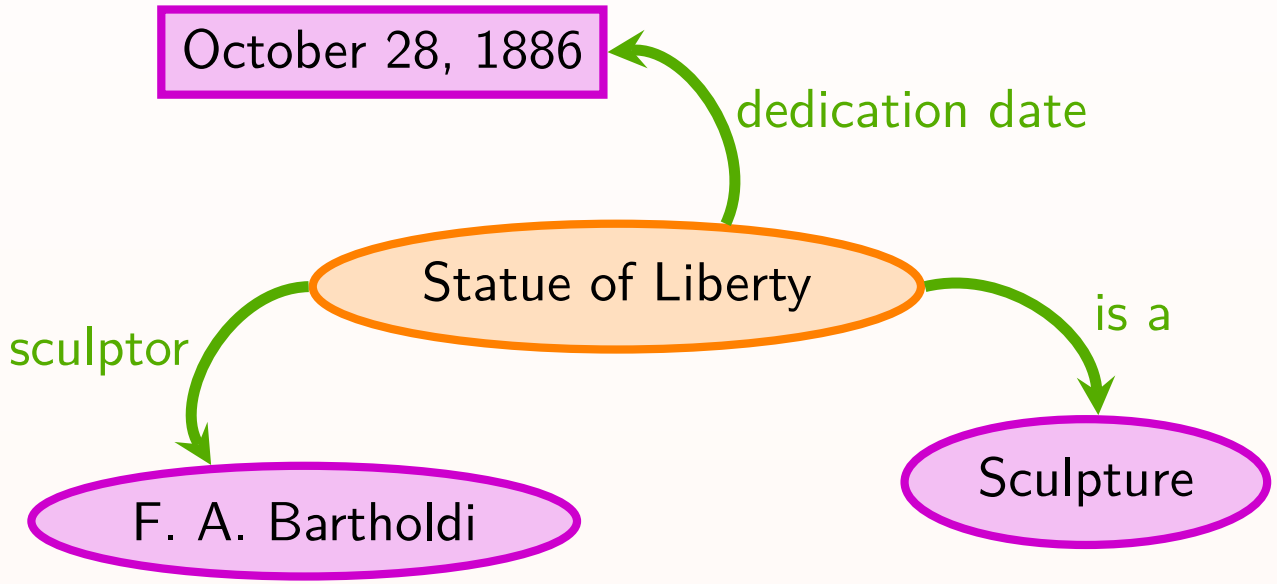
  (subject,predicate, object)

- RDF dataset can be represented as a graph

- The subject is a **resource**
- The predicate is a property of the resource
- The object is the value of the property for this resource; it can be another resource, or a literal

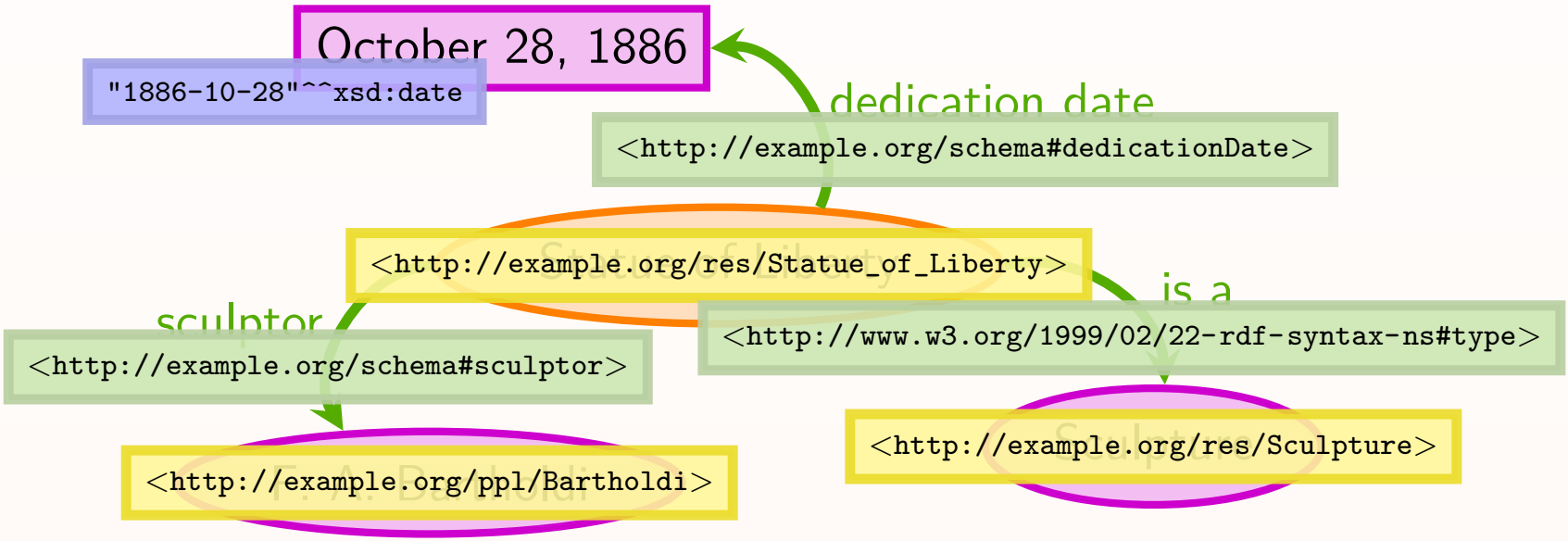
Data representation: Example

- The Statue of Liberty is a sculpture
- Bartholdi was its sculptor
- Dedicated on October 28, 1886



Data representation: Example

- Every graph node or edge is labeled by URI/IRI or literal
- URI/IRI identifies resources
- Literals can be integers, strings or dates



Outline

Vocabularies

- URI/IRI identifies everything
- Which one should I use for an (supposed) existing thing?
- Use RDF vocabularies
- Defining class entities and predicates between entities
- Often gathered around a theme
- Facility of use, not an obligation!

Some Vocabularies

- RDF Schema: the basis

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

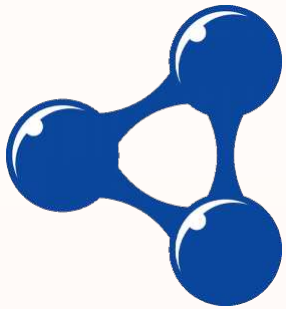
  `rdfs:<http://www.w3.org/2000/01/rdf-schema#>`
  - *Ex. of classes*: Class, Property, Datatype, List
  - *Ex. of properties*: type, label, value
- Friend of a Friend: social relations

  `foaf:<http://xmlns.com/foaf/0.1/>`
  - *Ex. of classes*: Person, Organization, Project
  - *Ex. of properties*: firstName, knows, homepage
- Dublin Core: document specifications

  `dc:<http://purl.org/dc/elements/1.1>`
  - *Ex. of classes*: BibliographicResource, PhysicalResource, Policy
  - *Ex. of properties*: creator, references, title
- SIOC: Semantically-Interlinked Online Communities
- SKOS: Simple Knowledge Organization System
- OWL: Web Ontology Language

Interlinking

- These vocabularies provide properties to interlink resources and datasets
  - `rdfs:seeAlso` for additional informations
  - `owl:sameAs` when two URI actually refer to the same thing
  - `rdfs:isDefinedBy` when a resource defining the resource subject
- Inheritance provides unlimited classes and properties
- `seeAlso`: Linked Open Vocabularies (LOV) [3]

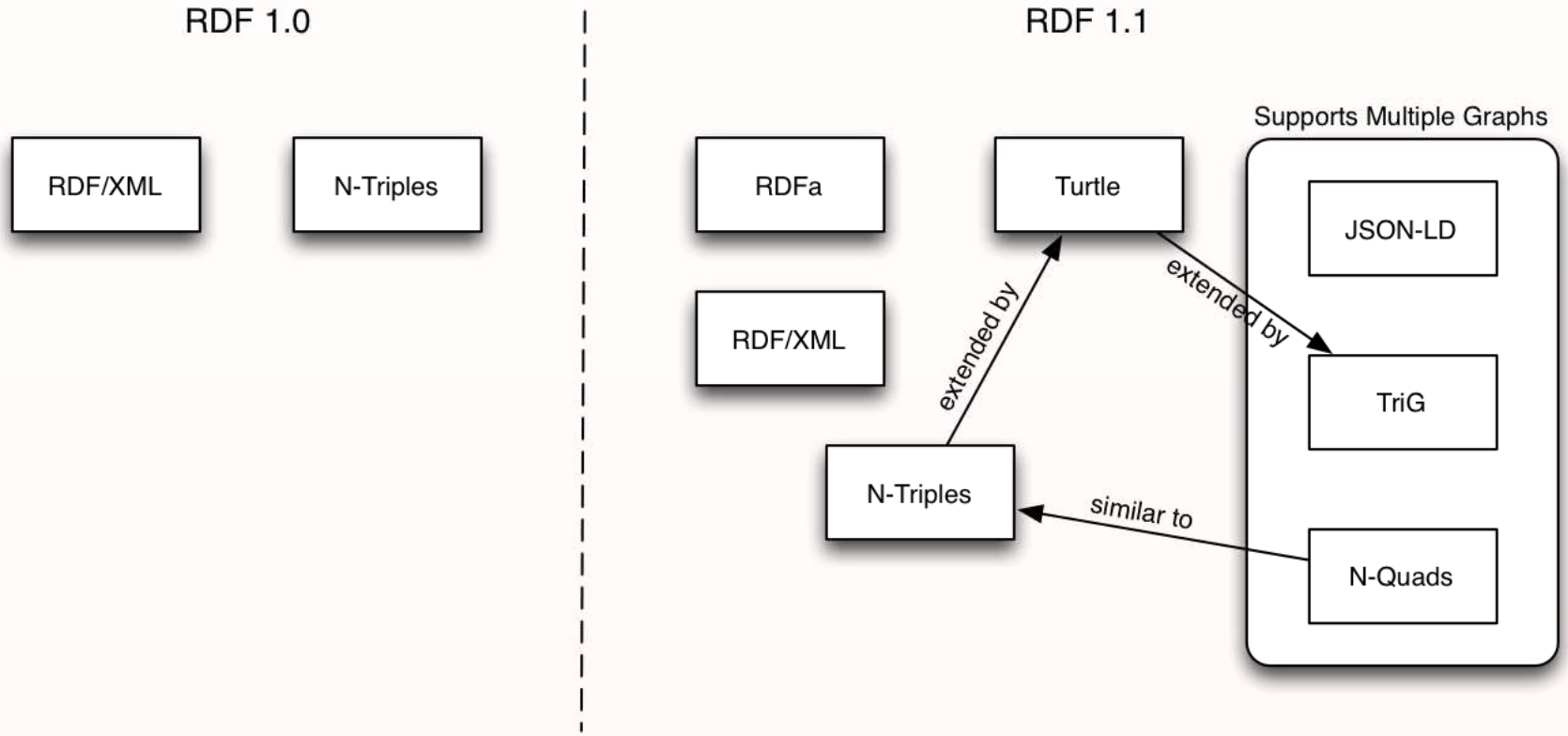


[3] `http://lov.okfn.org`

Outline

Serialization

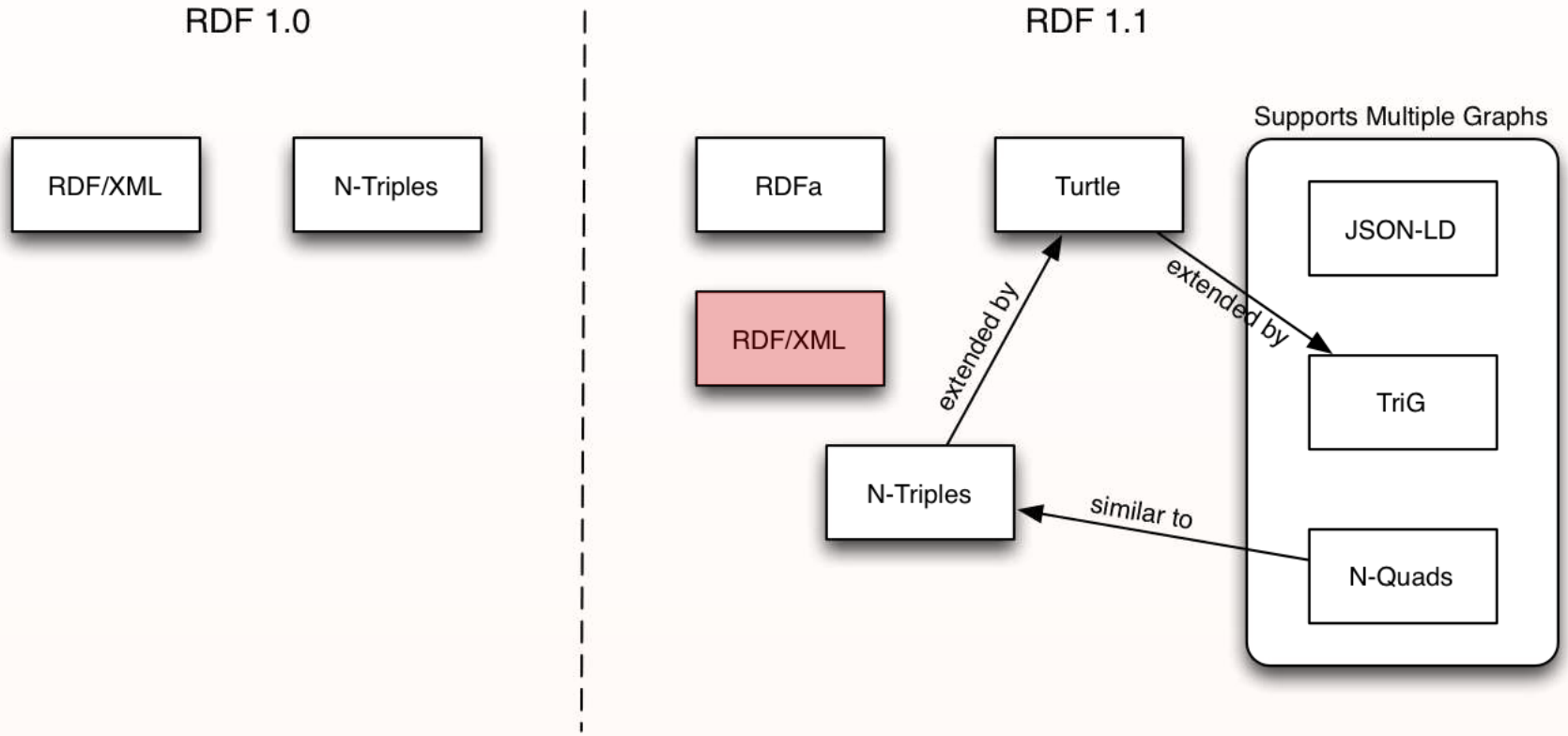
- In which format write RDF?
- Seven recommended serialization formats by W3C[4].



[4]Image from http://www.w3.org/TR/rdf11-new/

Serialization

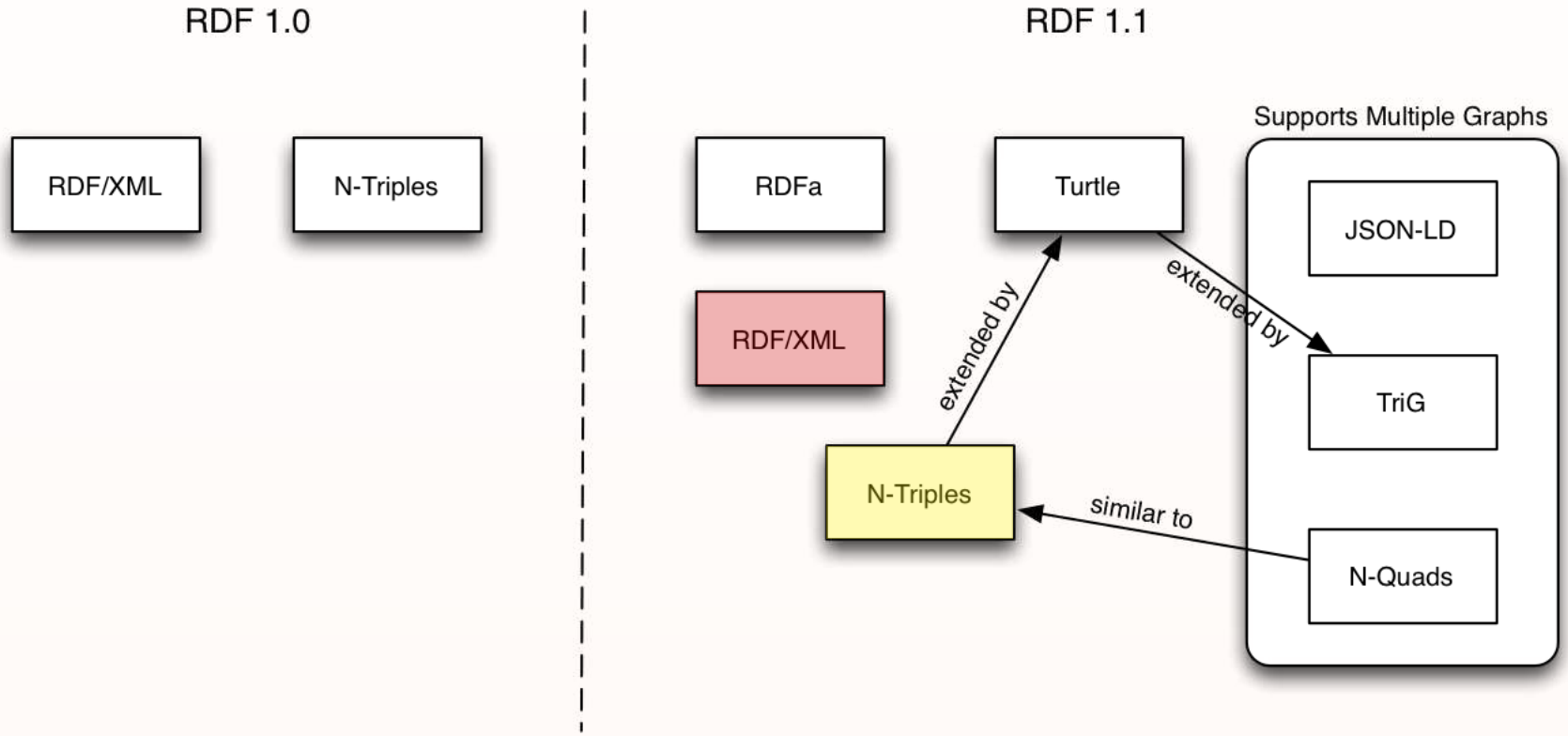
- In which format write RDF?
- Seven recommended serialization formats by W3C[4].



[4]Image from `http://www.w3.org/TR/rdf11-new/`

Serialization

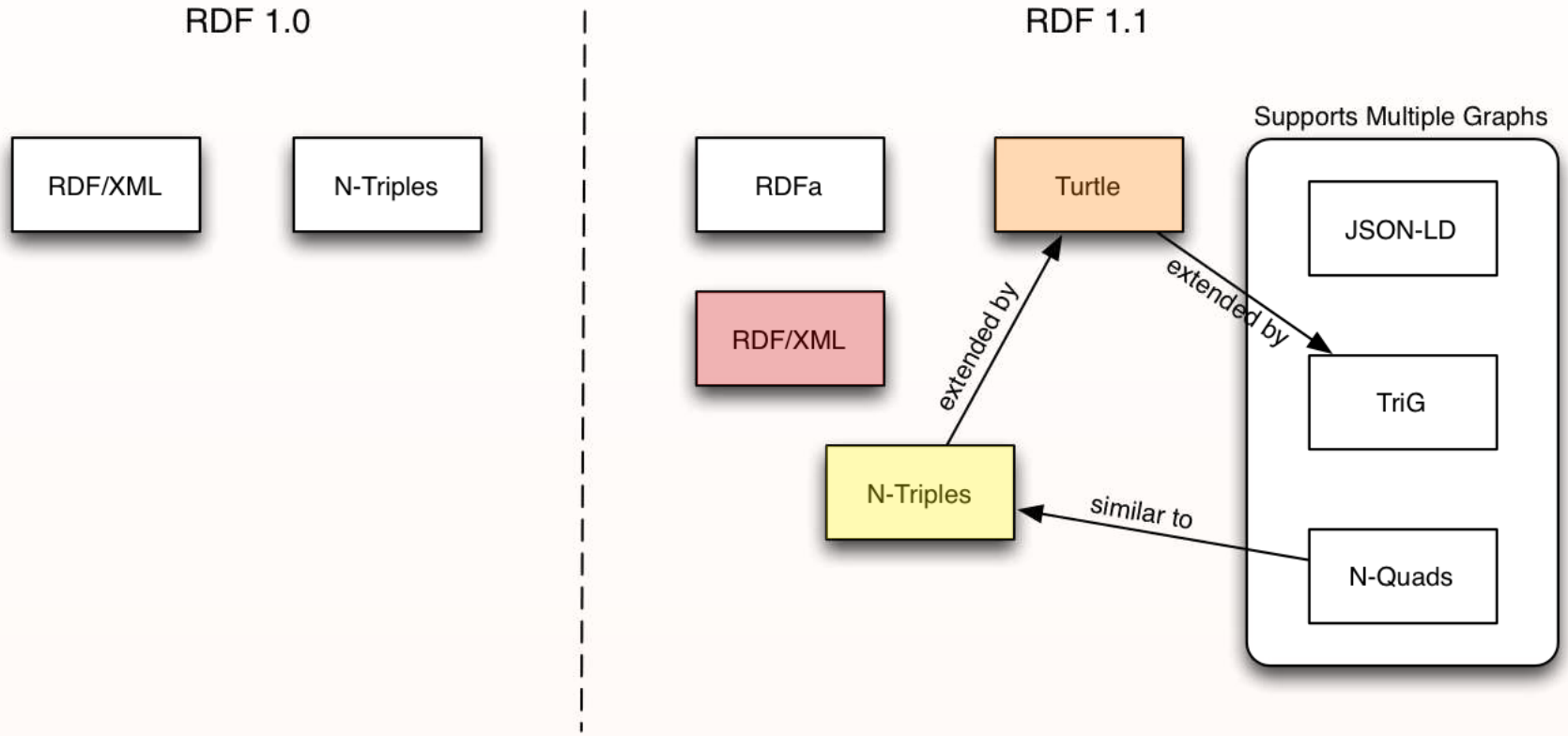
- In which format write RDF?
- Seven recommended serialization formats by W3C[4].



[4]Image from http://www.w3.org/TR/rdf11-new/

Serialization

- In which format write RDF?
- Seven recommended serialization formats by W3C[4].



[4]Image from `http://www.w3.org/TR/rdf11-new/`

RDF/XML

- RDF/XML: First RDF Serialization in time
- RDF described as an XML document

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:exs="http://example.org/schema#">
<rdf:Description
  rdf:about="http://example.org/res/Statue_of_Liberty">
  <rdf:type rdf:resource="http://example.org/res/Sculpture"/>
  <exs:sculptor rdf:resource="http://example.org/ppl/Bartholdi"/>
  <exs:dedicationDate rdf:datatype="xsd:date">
    1886-10-28
  </exs:dedicationDate>
</rdf:Description>
```

N-Triples

- RDF described as triples
- `<subject IRI> <predicate IRI> <object IRI or literal>`

```
<http://example.org/res/Statue_of_Liberty>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://example.org/res/Sculpture> .

<http://example.org/res/Statue_of_Liberty>
<http://example.org/schema#sculptor>
<http://example.org/ppl/Bartholdi> .

<http://example.org/res/Statue_of_Liberty>
<http://example.org/schema#dedicationDate>
"1886-10-28"^^<http://www.w3.org/2001/XMLSchema#date> .
```

Turtle

- Extends N-Triple
- Adding writing shortcuts to make the code more readable

```
BASE <http://example.org/res/>
  PREFIX xsd:<http://www.w3.org/2001/XMLSchema#">
  PREFIX exs:<http://example.org/schema#">
  PREFIX ppl:<http://example.org/ppl/">

Statue_of_Liberty
  a Sculpture ;
  exs:sculptor ppl:Bartholdi ;
  exs:dedicationDate "1886-10-28"^^xsd:date .
```

- **a** shortcuts `rdf:type`
- Semicolons group multiple statements on a same subject, comas group values

Outline

Literal values

- Literals use datatypes of *XML schema datatypes*
- Most commons are string, integer and date

```
BASE <http://example.org/res/>
  PREFIX xsd:<http://www.w3.org/2001/XMLSchema#">
  PREFIX exs:<http://example.org/schema#">

<Statue_of_Liberty>
  exs:dedicationDate "1886-10-28"^^xsd:date ;
  exs:visitation "3.2e6"^^xsd:double ;
  exs:visitationYear "2009"^^xsd:integer ;
  exs:name "Statue of Liberty"^^xsd:string ;
  exs:publicOpened "true"^^xsd:boolean .
```

- In Turtle: numbers, strings and booleans may be written directly (avoiding datatype)

Literal values

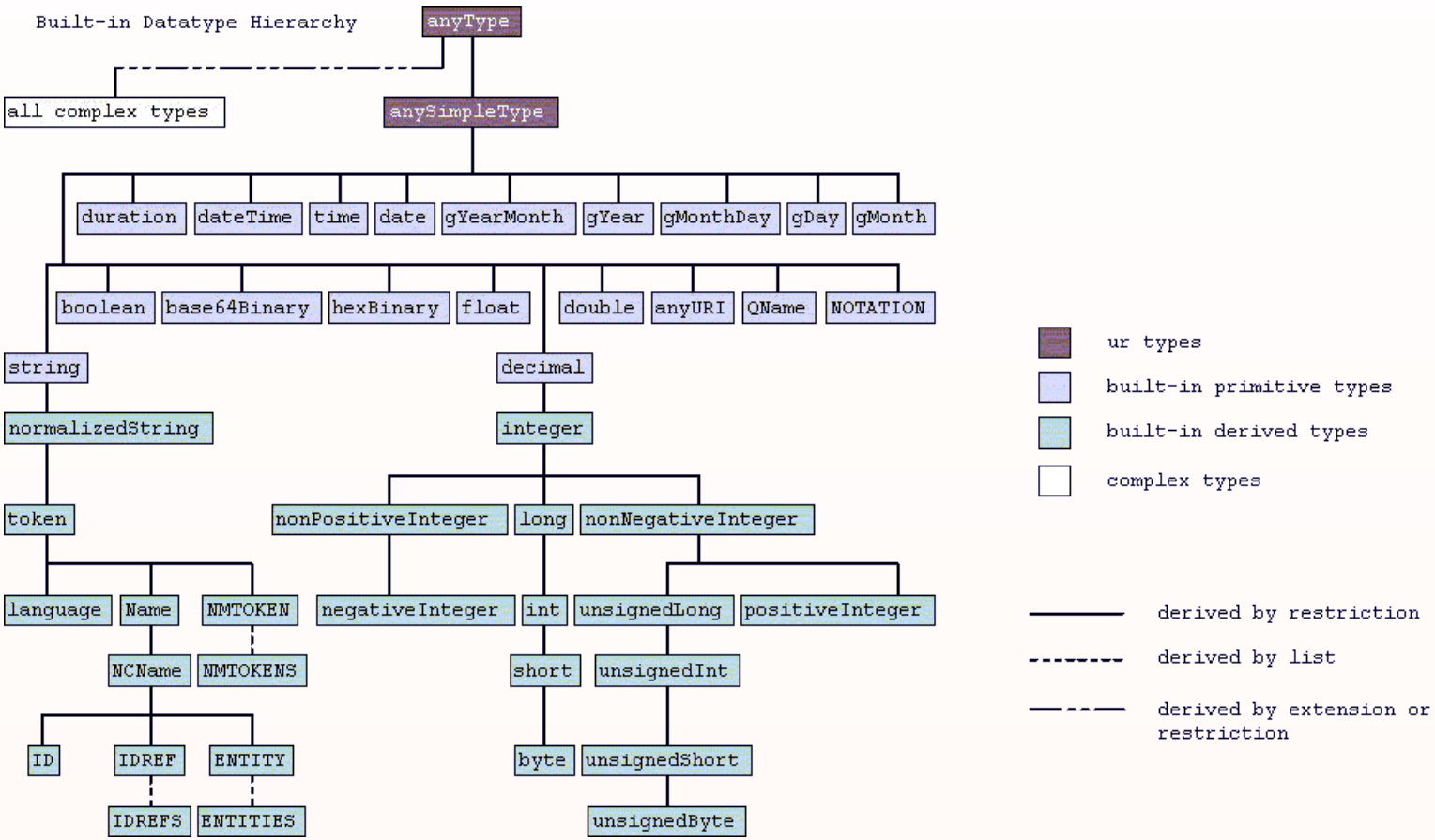
- String can be localized
- `xml:lang` property in RDF/XML

```
BASE <http://example.org/res/>
  PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#">

<Statue_of_Liberty>
  rdfs:label "Statue of Liberty"@en ;
  rdfs:label "Statue de la Liberté"@fr ;
  rdfs:label "Freiheitsstatue"@de .
```

Literal values

- Many datatype available[5]



[5] Image from http://www.w3.org/TR/xmlschema-2/

Blank nodes

- Used to describe informations about anonymous resources
- Only as subject or object of a triple
- Example: *Tobias knows somebody named David.*

```
<Tobias> foaf:knows [ foaf:name "David" ] .
```

- Blank node can be named to reuse them

```
<Tobias> foaf:knows _:a1 .
_:a1 foaf:name "David" ;
  foaf:age 40 .
```

Containers

Describe groups of things

- `rdf:Bag` , a set of values without specific order

```
<Statue_of_Liberty> exs:contains [ a rdf:Bag ;
  rdf:li <Torch> ;
  rdf:li <Statue> ;
  rdf:li <Pedestal> .  ]  .
```

- `rdf:Seq` , an ordered list of values

```
<Statue_of_Liberty> exs:restored [ a rdf:Seq ;
  rdf:li 1938 ;
  rdf:li 1986 ;
  rdf:li 2012 .  ]  .
```

- `rdf:Alt` , a list of alternative values

```
<Statue_of_Liberty> rdfs:label [ a rdf:Alt ;
  rdf:li "Statue of Liberty"@en ;
  rdf:li "Statue de la Liberté"@fr ;
  rdf:li "Freiheitsstatue"@de .  ]  .
```

Outline

Presentation

- Recursive acronym for
  SPARQL **P**rotocol and **R**DF **Q**uery **L**anguage

- Query language for RDF and RDFS
- Use pattern matching over the graph
- Turtle syntax

Presentation

- Four types of query
  - **SELECT** Return all the values matching the query pattern
  - **CONSTRUCT** Build a RDF graph from returned data
  - **DESCRIBE** Request information about something
  - **ASK** Return a boolean for a question query
- Used on a SPARQL endpoint
- Each call usually takes a WHERE clause to restrict the query

Presentation

- The query contains pattern of triples
- They can use:
  - IRIs
  - Literals
  - Variables, prefixed by **?** or **$**
  - Blank nodes
- Using a Turtle syntax and prefixes

Outline

The SELECT Query

- Global form:

[PREFIX <*prefixes*>]

SELECT <*return vars*>

[FROM <*sources*>]

WHERE { <*triple patterns*> }

The SELECT Query

Examples of SELECT query

- Return all statements of the graph:

```
SELECT ?subject ?property ?value
WHERE { ?subject ?property ?value }
```

- Return the first and last names of all sculptors of the graph:

```
SELECT ?firstName ?lastName WHERE { _:x
 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://example.org/res/Sculptor> .
 _:x <http://xmlns.com/foaf/0.1/firstName>
    ?firstName .
 _:x <http://xmlns.com/foaf/0.1/lastName>
    ?lastName .  }
```

- Return all properties having 2015 as value:

```
SELECT DISTINCT ?prop
WHERE { [] ?prop "2015" }
```

SPARQL is in Turtle!

- Turtle reminder:
  - `BASE` and `PREFIX` are available for vocabularies
  - `a` for `rdf:type`
  - Factoring statements with `;` and values with `,`
- Better write of the second query of previous slide:

```
BASE <http://example.org/res/>
PREFIX foaf:<http://xmlns.com/foaf/0.1/">
SELECT ?firstName ?lastName WHERE
{ [] a Sculptor ;
    foaf:firstName ?firstName ;
    foaf:lastName ?lastName .  }
```

Optional pattern: OPTIONAL

- A part of the pattern is not mandatory:

```
BASE <http://example.org/res/>
PREFIX foaf:<http://xmlns.com/foaf/0.1/">
SELECT ?firstName ?lastName ?homepage WHERE {
_:x a Sculptor ;
    foaf:firstName ?firstName ;
    foaf:lastName ?lastName .
  OPTIONAL { _:x foaf:homepage ?homepage .}
}
```

- ?homepage can be unbound in some results

Union of patterns: UNION

- Union of pattern, similar to a **OR** clause:

```
BASE <http://example.org/res/>
PREFIX foaf:<http://xmlns.com/foaf/0.1/">
SELECT ?firstName ?lastName WHERE {
  {
    _:x a Sculptor ;
  }
  UNION
  {
    _:x a Singer .
  }
  _:x foaf:firstName ?firstName ;
    foaf:lastName ?lastName .
}
```

Exclusion of patterns: MINUS

- Remove patterns from the results:

```
BASE <http://example.org/res/>
PREFIX foaf:<http://xmlns.com/foaf/0.1/">
SELECT ?x WHERE {
  ?x a Sculptor .
  MINUS { ?x a Jazz_trumpeter }
}
```

- This request excludes Jeff Nuttall from the results!

Path patterns

- Some regular expression about path between resources:

```
BASE <http://example.org/res/>
PREFIX foaf:<http://xmlns.com/foaf/0.1/">
SELECT ?relation WHERE {
  [] foaf:name "John Doe" ;
    foaf:knows+ ?relation .  }
```

- Available patterns:
  - / sequence
  - + one or several
  - ? optional
  - ! negation
  - | alternative
  - * zero or several
  - ^ reverse

Order and paginate

- These clauses works like in SQL
  - `ORDER` orders results with `ASC` and `DESC`
  - `GROUP BY` groups results given a variable
  - `LIMIT` limits the results size
  - `OFFSET` start results at given offset

Outline

Filters

- Apply filters in the `WHERE` clause
- `FILTER` clause is written in XPath

```
BASE <http://example.org/res/>
PREFIX ex:<http://example.org/schema#>
PREFIX foaf:<http://xmlns.com/foaf/0.1/">
SELECT ?x ?ach WHERE {
  ?x a Sculptor ;
     ex:achievements ?ach .
  FILTER (xsd:integer(?ach) >= 50)
}
```

- Can also use external filters

Filters

- Tests on values:
  - Comparators: $<$, $>$, $=$, $<=$, $>=$, $!=$
  - Regular Expressions: `regex(?x, ".*rdo")`
  - Type checking: `isURI(?x)`, `isBlank(?x)`, `isLiteral(?x)`, `isNumeric(?x)`
  - String checking: `contains(?x,"foobar")`, `strstarts(?x,"foo")`, `strends(?x,"bar")`
- Combinations:
  - Boolean operators: `&&`, `||`, `!`, `()`

Filters

- Conditional statements:
  - `if - then - else` construct:

```
BASE <http://example.org/res/>
PREFIX ex:<http://example.org/schema#>
PREFIX foaf:<http://xmlns.com/foaf/0.1/">
SELECT ?adults WHERE {
  ?adults foaf:age ?age ;
     ex:country ?c .
  FILTER (?c="USA",?age>=21,?age>=18)
}
```

  - Any filter can be denied: `FILTER NOT EXISTS (...)`

Outline

Functions

- Many functions can be used in `FILTER` or `SELECT`
- Types: `lang()`, `datatype()`, `str()`, `xsd:integer(?x)`, `strdt(v, t)`, ...
- Strings: `strlang(v, l)`, `concat(?x,?y)`, `ucase(?x)`, `lcase(?x)`, `substr(?x,s[,l])`, `strlen(?x)`, ...
- Numerics: `abs(?x)`, `ceil(?x)`, `floor(?x)`, `round(?x)`, `rand()`, ...
- Dates: `year(?x)`, `month(?x)`, `day(?x)`, `hours(?x)`, `timezone(?x)`, `now()`, ...
- Many others available[6], external calls possible

[6] Have a look at: `http://www.w3.org/TR/sparql11-query/#expressions`

Outline

The others queries

- **ASK** Return a boolean for a question query
- Return `true` if any data can match the query
- Return `false` otherwise
- **ASK** is a `WHERE` clause

```
BASE <http://example.org/res/>
ASK {?x a Actor, Politician }
```

The others queries

- **CONSTRUCT** Build a RDF graph from returned data
- Result graph is a sub-graph of the data source

```
PREFIX ppl:<http://example.org/ppl/>
CONSTRUCT { ppl:Bartholdi ?predicate ?object }
WHERE { ppl:Bartholdi ?predicate ?object }
```

The others queries

- **DESCRIBE** Request information about something
- What will actually be returned depends of server configuration

```
DESCRIBE <http://example.org/ppl/Bartholdi>
```

Outline

SPARQL UPDATE

- It's out of the scope of this lecture, but you had to know it:

SPARQL can alter data

- `LOAD` loads external datasets
- `INSERT DATA` puts triples into the graph
- `DELETE DATA` removes triples from the graph
- An update can be done by the combination of `DELETE DATA` and `INSERT DATA`.

SPARQL UPDATE

```
BASE <http://example.org/res/>
  PREFIX exs:<http://example.org/schema#">
  PREFIX ppl:<http://example.org/ppl/">

DELETE DATA {Statue_of_Liberty exs:sculptor ppl:Bartholdi }
INSERT DATA {Statue_of_Liberty exs:sculptor ppl:Matt_Groening }
```

Conclusion

- This is only a very beginning of "What the Semantic Web is"
- Share knowledge, make it machine readable
- Query and use this knowledge
- **Be open, be linked!**

The Semantic Web

Damien Leprovost

April 3, 2015

permalink: http://www.damien-leprovost.fr/enseignements/semweb.2015.pdf