



DCMI Basic Syntaxes Tutorial

Mikael Nilsson <mikael@nilsson.name>

DC 2007, Singapore
Aug 27, 2007



About me...

- PhD student at Royal Inst. of Technology, Stockholm
 - “Metadata interoperability in the web environment”
- Co-chair
 - DCMI Architecture forum
 - IEEE LTSC/DCMI Task Force
- Co-author
 - DCMI Abstract Model
 - DC in RDF
 - DCMI Description Set Profile Model
- <http://kmr.nada.kth.se>



Overview

- Current specifications
- The Abstract Model and the role of syntaxes
- Dublin Core in RDF
- Dublin Core in XML
- Dublin Core in (X)HTML



Syntaxes for Dublin Core

- Dublin Core metadata can be expressed in many ways. Three ways are specified by the DCMI:
 - Resource Description Framework (RDF)



- XML

```
<dc:title>  
Frog maths  
</dc:title>
```

- (X)HTML <meta> and <link> elements

```
<meta name="DC.date" scheme="DCTERMS.W3CDTF" content="2001-07-18" />
```



Current syntax specifications

- RDF
 - Proposed recommendation:
Expressing Dublin Core Metadata using the Resource Description Framework (June 2007).
<http://dublincore.org/documents/dc-rdf/>
 - Deprecated:
 - *Expressing Qualified Dublin Core in RDF/XML* (May 2002)
 - To be deprecated:
 - *Expressing Simple Dublin Core in RDF/XML* (July 2002)



Current syntax specifications (cont.)

- XML

- *Guidelines for implementing Dublin Core in XML*
(April 2003)

<http://dublincore.org/documents/dc-xml-guidelines/>

- New version in working draft!

- HTML/XHTML

- *Expressing Dublin Core in HTML/XHTML meta and link elements* (Nov 2003)

<http://dublincore.org/documents/dcq-html/>

- New version in working draft!



The spec situation

(Proposed) recommendation, Working Draft,
Deprecated, Will be deprecated,

Pre-DCAM

- RDF
 - Simple DC in RDF
 - Qualified DC in RDF
- XML
 - DC in XML

(X)HTML

- DC in (X)HTML

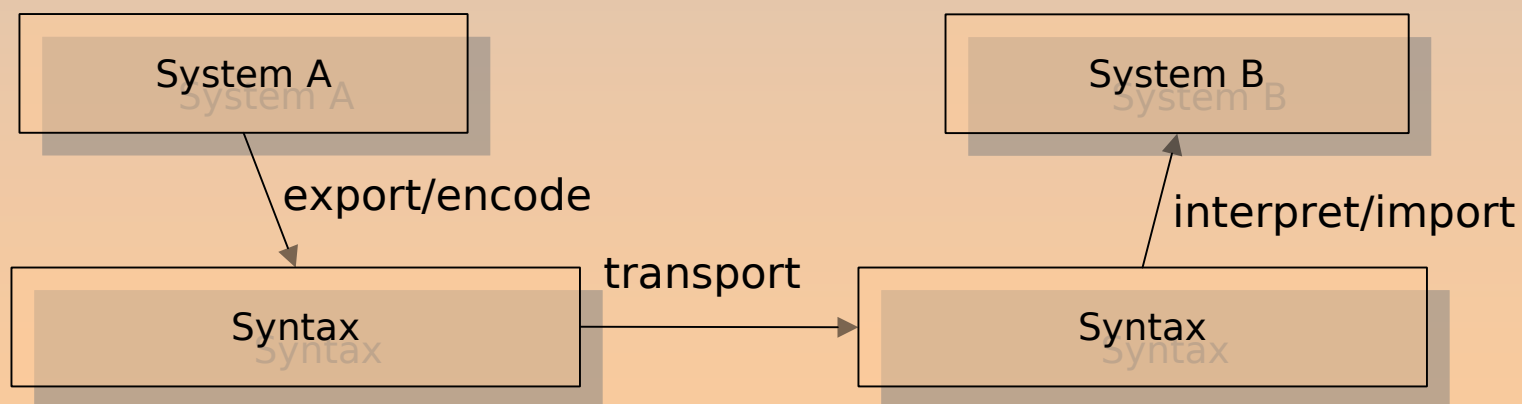
DCAM-based

- RDF
 - Expressing DC in RDF
- XML
 - DC-XML-Full
 - DC-XML-Min
- (X)HTML
 - DC in (X)HTML



What is the role of syntaxes?

- Transfer mechanism for metadata from system A to system B
- Archival format for metadata





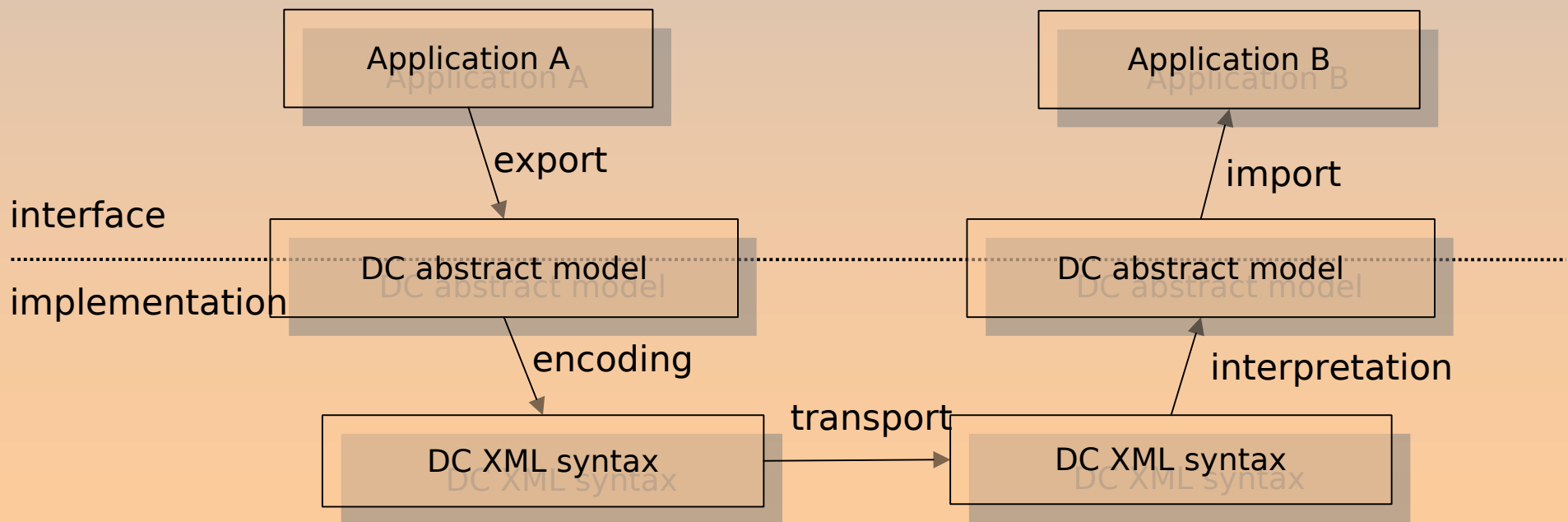
Why multiple syntaxes?

- Different syntaxes better in different contexts
 - Embedded in (X)HTML
 - Harvesting vs. federated search
 - Embedded in XML specifications/microformats
 - Semantic web applications
- Ask yourself...
 - what am I trying to achieve?
 - does using HTML, XML or RDF help me achieve it?
 - do software and services exist that will support the creation and use of my metadata?



How do the syntaxes interoperate?

- DCMI Abstract Model gives “interface”, or “conceptual model”
- syntaxes give realization/implementation



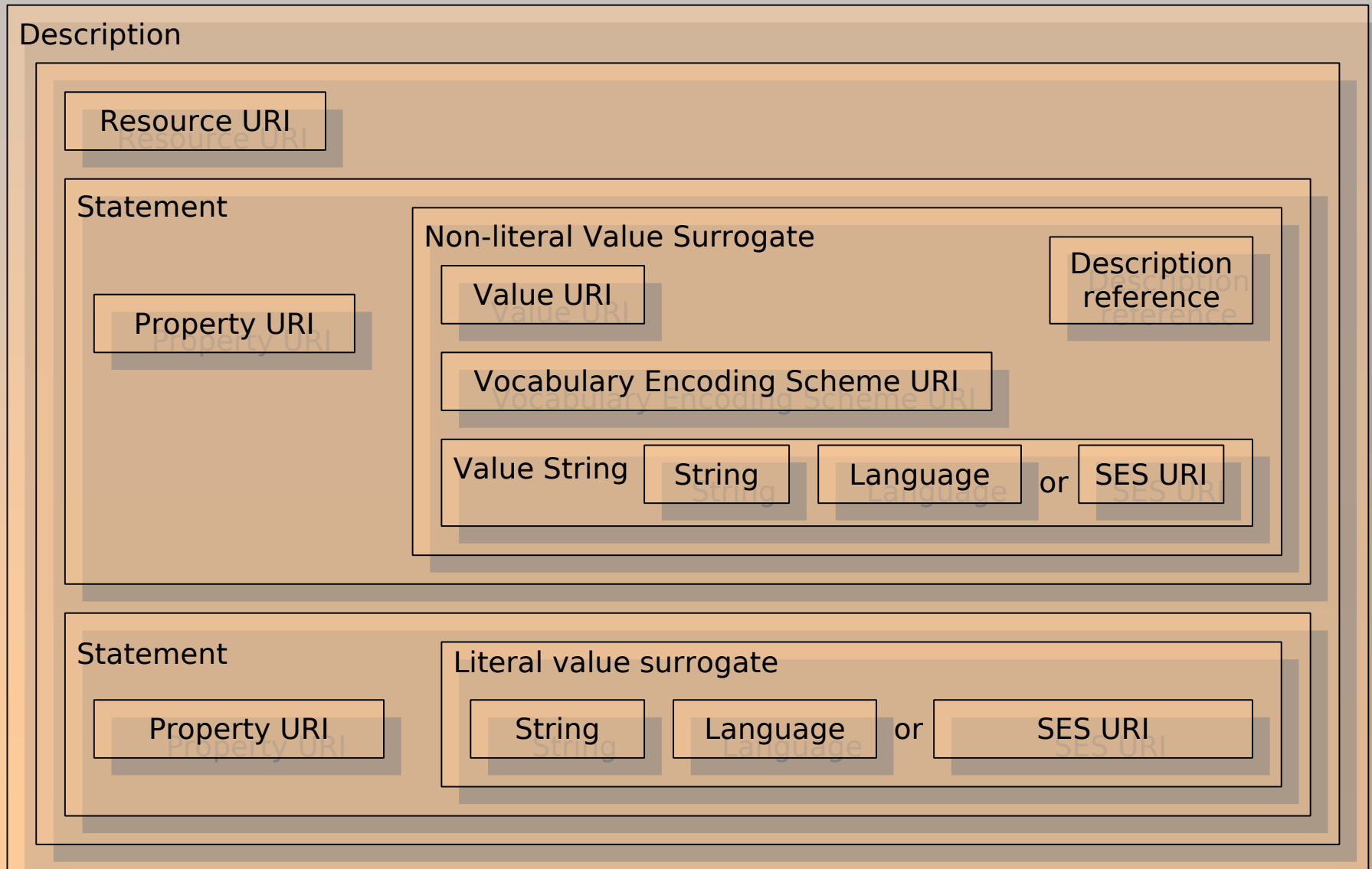


How is a syntax defined?

- 1) What subset of the DCAM is supported?
- 2) How is each feature of the DCAM encoded?
- 3) Conversely, how is each syntax construct interpreted in terms of the DCAM?



DCAM 2007 revisited



SES =
Syntax
Encoding
Scheme



Interrupt: what is this literal/nonliteral stuff?

- DCAM supports the distinction between
 - values that really are strings (literals)
 - titles (text)
 - counts (integers)
 - identifiers (string tokens)
 - etc.
 - values that are things, concepts or other non-string resources
 - Persons
 - Documents
 - Events
 - etc.



Using literals

- Literals:
 - Are self-contained
 - Can not be further described
 - can have
 - a language (plain), or
 - a syntax encoding scheme (typed)



Using Non-literals

- Non-literals
 - Refer to something “out there”
 - Can be described in a separate description
 - Can be described in a statement using
 - multiple value strings (“labels”)
 - A vocabulary encoding scheme (“from where is this taken?”)
 - A value URI (“unique identifier”)



When do I use literal values, and when not?

1) Does the property in question have a range?

- `dcterms:title` => **literal**
- `dcterms:creator` => **Agent (i.e. non-literal)**

2) If not:

1) Is the value "just a string"? => **literal**

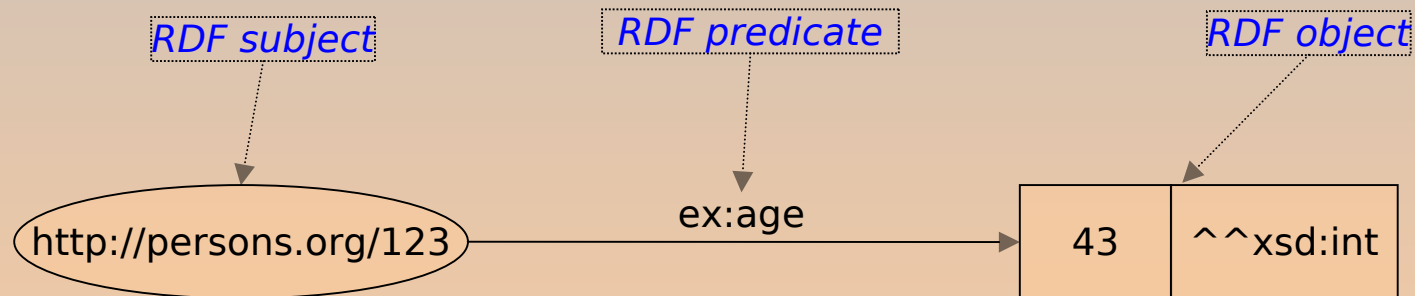
2) Can the value potentially be described elsewhere?
=> **non-literal**

3) Can I potentially want to use several value strings to describe the value? => **non-literal**

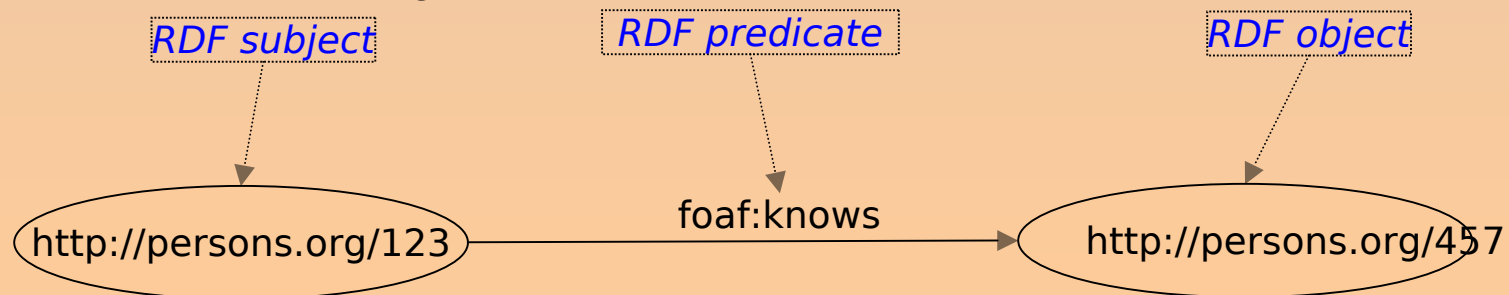


Two minutes of RDF

- Built on “triples”: Subject, Predicate, Object
- Literal objects:



- Non-literal objects:





Dublin Core and RDF

- Long history: Co-evolution since 1995
- Different, but overlapping, communities
- Compatible models: DCAM shares RDF semantics:

DCAM	RDF
Property/Element	Property
Resource	Resource
Class	Class
Syntax Encoding Scheme	Datatype
Domain/Range	Domain/Range
Subproperty	Subproperty
Subclass	Subclass
Plain value string	Plain literal
Typed value string	Typed literal



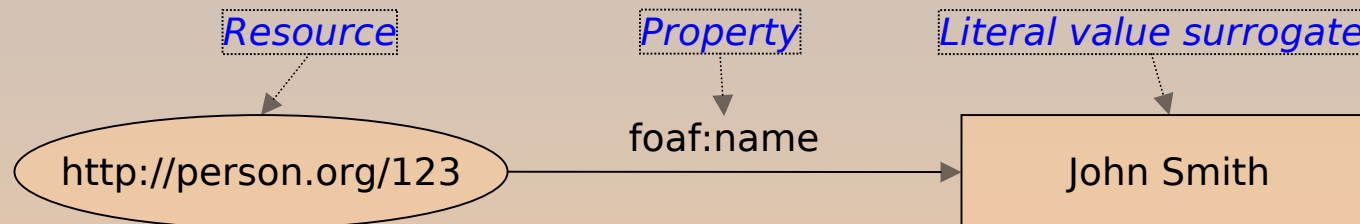
Dublin Core and RDF (cont.)

- What subset of the DCAM is supported?
 - Everything
- How is each feature of the DCAM encoded?
- Conversely, how is each syntax construct interpreted in terms of the DCAM?
 - See following slides

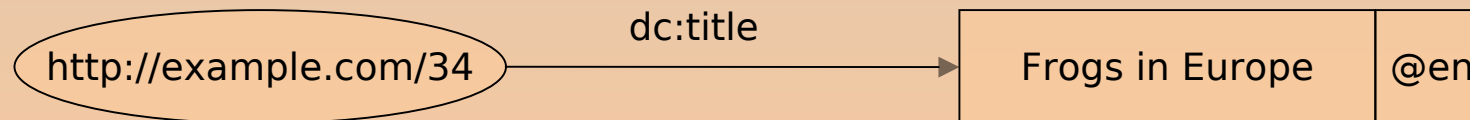


DCAM Statements with literal value surrogates

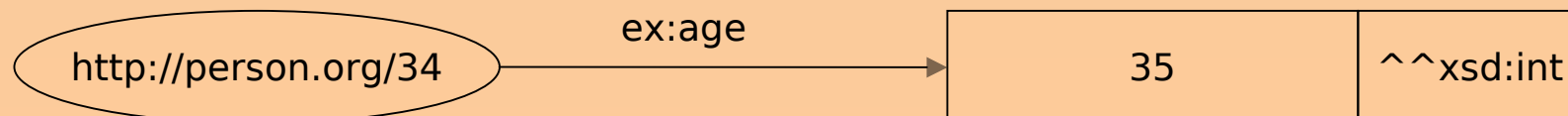
- No language, no syntax encoding schemes



- With language

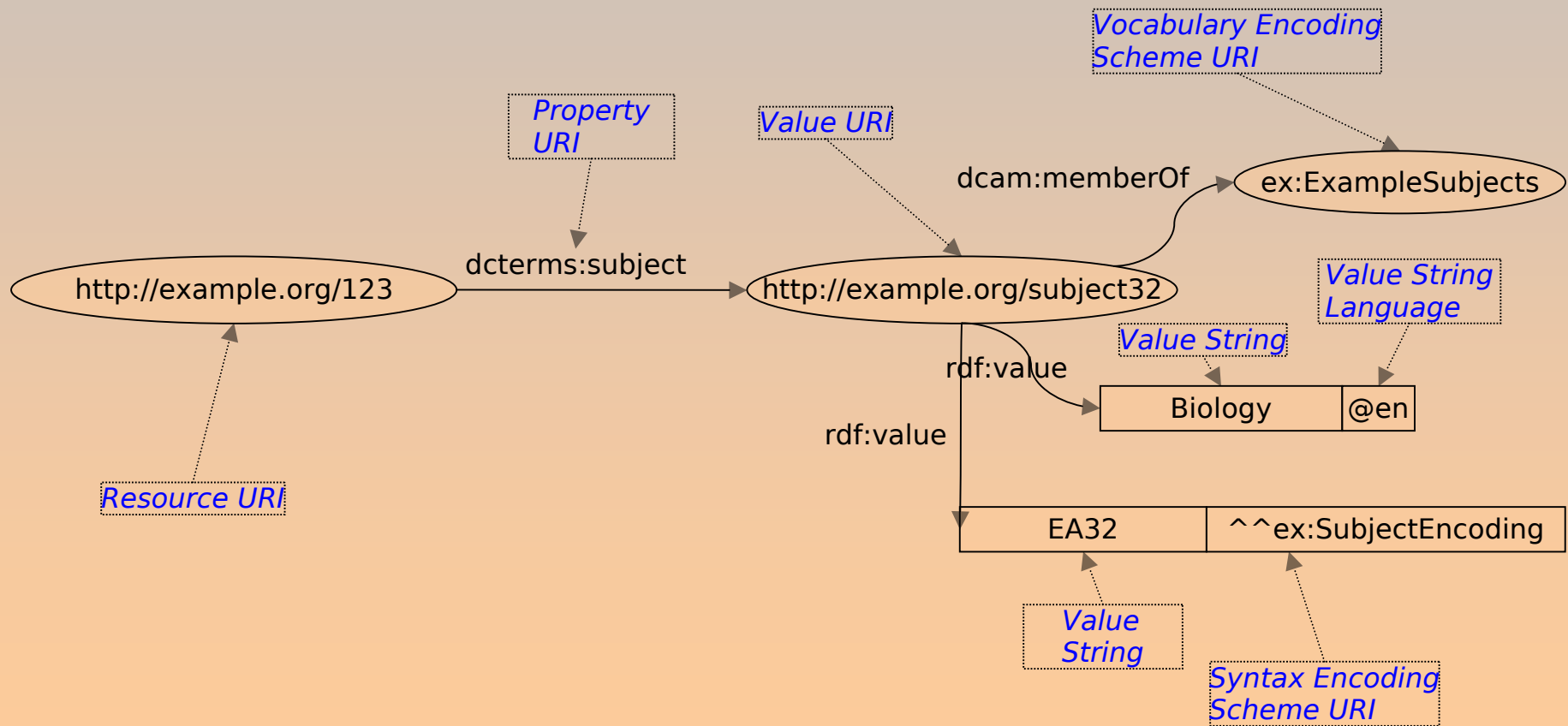


- With Syntax encoding scheme





DCAM Statements with nonliteral value surrogates



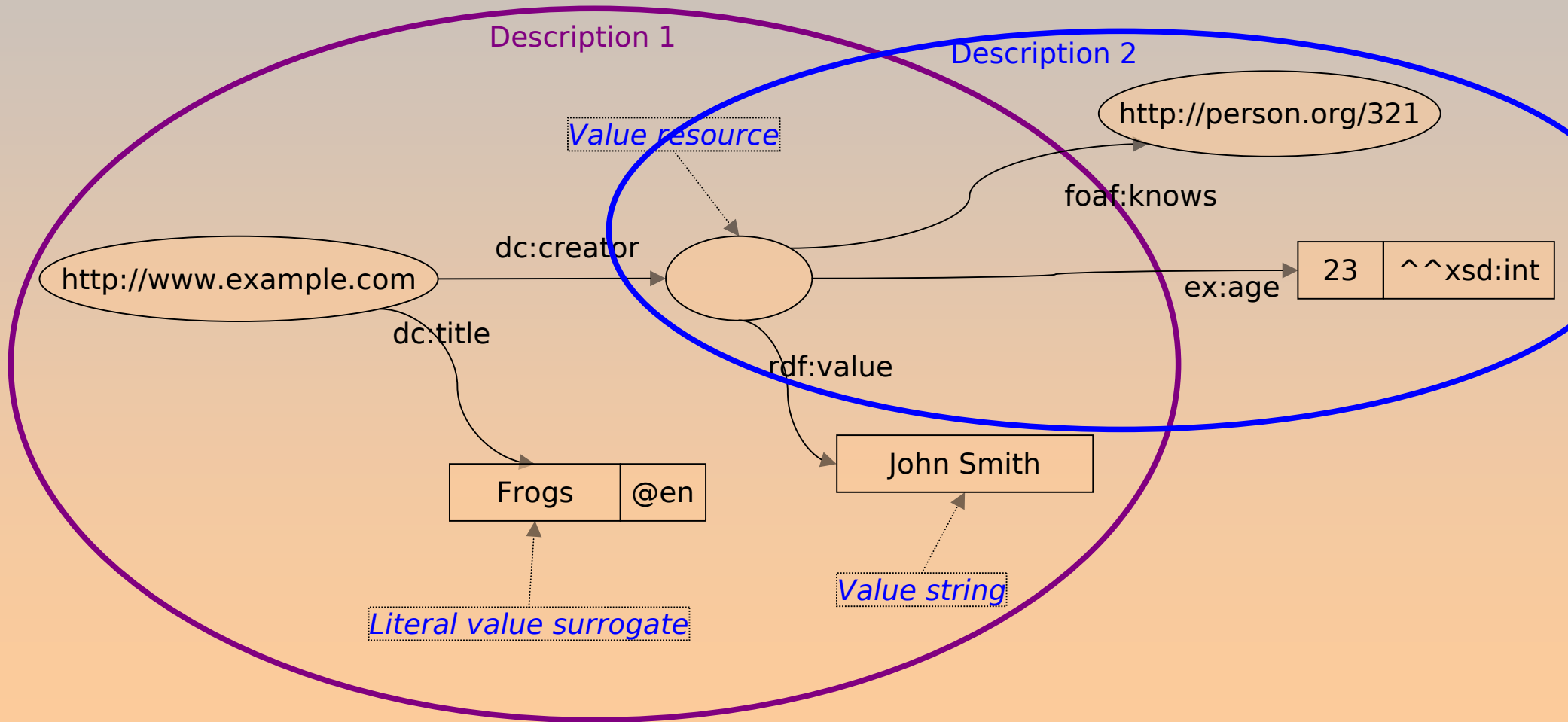


DCAM Description Sets in RDF

- One DCAM Description => an RDF graph centered on one subject
- One DCAM description set => one RDF Graph / Model
- RDF has built in mechanisms to link one nonliteral value surrogate to a separate description:
 - URI-based identification
 - blank nodes



RDF example - description set



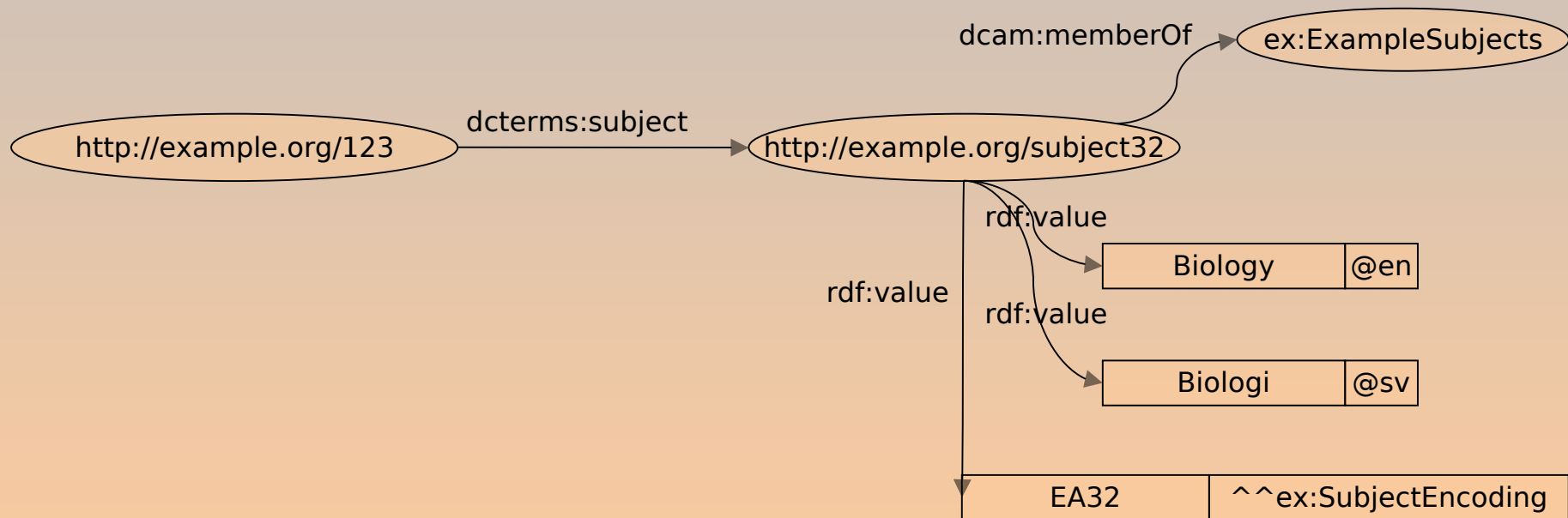


RDF syntaxes

- RDF is in itself a conceptual model, and several encodings exist:
 - RDF/XML
 - NOT to be confused with DC-XML
 - N-TRIPLE - simple listing of triples
 - N3, Turtle - compact RDF formats
 - RDFa - embedding RDF in HTML content
 - NOT to be confused with DC in HTML
- All these are valid options for using DC in RDF



Graph (informal)





RDF/XML

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:ex="http://example.org/taxonomy/"
  xmlns:dcam="http://purl.org/dc/rdf/">

  <rdf:Description rdf:about="http://example.org/123">
    <dcterms:subject>
      <rdf:Description rdf:about="http://example.org/subject32">
        <dcam:memberOf
          rdf:resource="http://example.org/taxonomy/ExampleSubjects"/>
        <rdf:value xml:lang="en">Biology</rdf:value>
        <rdf:value xml:lang="sv">Biologi</rdf:value>
        <rdf:value
          rdf:datatype="http://example.org/taxonomy/SubjectEncoding">
          EA32</rdf:value>
        </rdf:Description>
      </dcterms:subject>
    </rdf:Description>

  </rdf:RDF>
```



N-TRIPLE

```
<http://example.org/123> <http://purl.org/dc/terms/subject> <http://example.org/subject32> .  
<http://example.org/subject32> <http://purl.org/dc/rdf/memberOf> <http://example.org/taxonomy/ExampleSubjects> .  
<http://example.org/subject32> <http://www.w3.org/1999/02/22-rdf-syntax-ns#value> "Biology"@en .  
<http://example.org/subject32> <http://www.w3.org/1999/02/22-rdf-syntax-ns#value> "Biologi"@sv .  
<http://example.org/subject32> <http://www.w3.org/1999/02/22-rdf-syntax-ns#value>  
"EA32"^^http://example.org/taxonomy/SubjectEncoding .
```



DC in XML

- **NOTE:** The existing recommendation is not compatible with the DCMI Abstract Model
- **AND:** The updated version is still only “Working Draft”
- **SO:** This presentation will be sketchy, with examples taken from the Working Draft *subject to change!*
- The important aspect is that the format is a faithful representation of the constructs in the DCMI Abstract Model



DC in XML – two versions

- DC-XML-Full
 - covers 100% of the DCAM
 - at the cost of increased complexity
 - useful for generic DC processors with complex metadata requirements
- DC-XML-Min
 - covers a subset of the DCAM
 - more straightforward syntax
 - useful for simpler applications that have lower requirements
- [Other XML variants exist, such as OAI-PMH/oai_dc]



DC XML Full - literal example

```
<?xml version="1.0"?>
<dcxf:descriptionSet
  xmlns:dcxf="http://dublincore.org/xml/dc-xml-full/2007/06/19">
  <dcxf:description dcxf:resourceURI="http://dublincore.org/pages/home">
    <dcxf:statement dcxf:propertyURI="http://purl.org/dc/terms/title">
      <dcxf:literalValueString> DCMI Home Page </dcxf:literalValueString>
    </dcxf:statement>
  </dcxf:description>
</dcxf:descriptionSet>
```

Blue = instance data
Red = model constructs



DC-XML-Full Statements

- Statement with non-literal value surrogates

```
<dcxf:statement
  dcxf:propertyURI="http://purl.org/dc/terms/publisher"
  dcxf:valueURI="http://example.org/agents/DCMI">
  <dcxf:valueString> Dublin Core Metadata Initiative </dcxf:valueString>
</dcxf:statement>
```

```
<dcxf:statement
  dcxf:propertyURI="http://purl.org/dc/terms/subject"
  dcxf:vocabEncSchemeURI="http://purl.org/dc/terms/LCSH">
  <dcxf:valueString xml:lang="en"> Metadata </dcxf:valueString>
  <dcxf:valueString xml:lang="fr"> Métadonnées </dcxf:valueString>
</dcxf:statement>
```



DC-XML-Full notes

- Properties are given using their URIs
 - abbreviation mechanisms exist
- Multiple value strings allowed
 - But only a single string for literal value surrogates!
- Value string languages and literal languages are given using the **xml:lang** attribute
- Syntax encoding schemes are given using the **dcxf:syntaxEncSchemeURI** attribute



DC-XML-Min literal example

```
<?xml version="1.0"?>
```

```
<dcxm:descriptionSet
```

```
  xmlns:dcxm="http://dublincore.org/xml/dc-xml-min/2007/06/19"
```

```
  xmlns:dcterms="http://purl.org/dc/terms/">
```

```
  <dcxm:description dcxm:resourceURI="http://dublincore.org/pages/home">
```

```
    <dcterms:title>DCMI Home Page</dcterms:title>
```

```
  </dcxm:description>
```

```
</dcxm:descriptionSet>
```



DC-XML-Min Statements

- Statement with non-literal value surrogates

```
<dcterms:publisher dcxm:valueURI="http://example.org/agents/DCMI">
```

Dublin Core Metadata Initiative

```
</dcterms:publisher>
```

```
<dcterms:subject dcxm:vocabEncSchemeURI="http://purl.org/dc/terms/LCSH"  
xml:lang="en">
```

Metadata

```
</dcterms:subject>
```

```
<dcterms:creator dcxm:valueType="NonLiteral">
```

Jon Smith

```
</dcterms:creator>
```



DC-XML-Min notes

- Properties are given as XML elements
- Only one value string allowed!
- Value string languages and literal languages are given using the `xml:lang` attribute
- Syntax encoding schemes are given using the `dcxm:syntaxEncSchemeURI` attribute
- Nonliteral if one of the following:
 - Value URI given, or
 - Vocabulary encoding scheme given, or
 - `dcxm:valueType="NonLiteral"`



Multiple descriptions

```
<?xml version="1.0"?>
<dcxm:descriptionSet
  xmlns:dcxm="http://dublincore.org/xml/dc-xml-min/2007/06/19"
  xmlns:dcterms="http://purl.org/dc/terms/">

  <dcxm:description>
    <dcterms:title>DCMI Home Page</dcterms:title>
  </dcxm:description>

  <dcxm:description>
    <dcterms:title>UKOLN Home Page</dcterms:title>
  </dcxm:description>

</dcxm:descriptionSet>
```



Linking descriptions

```
<?xml version="1.0"?>
<dcxm:descriptionSet
  xmlns:my="http://my.example.org/terms/"
  xmlns:dcxm="http://dublincore.org/xml/dc-xml-min/2007/06/19"
  xmlns:dcterms="http://purl.org/dc/terms/">

  <dcxm:description
    dcxm:resourceURI="http://dublincore.org/pages/home">

    <dcterms:title>DCMI Home Page</dcterms:title>

    <dcterms:publisher
      dcxm:descriptionRef="DCMI"/>

  </dcxm:description>

  <dcxm:description dcxm:descriptionId="DCMI">

    <my:name>Dublin Core Metadata Initiative</my:name>

  </dcxm:description>
</dcxm:descriptionSet>
```



DC in (X)HTML <meta> and <link>

- a DC description embedded in an (X)HTML document describes **that document**
- if you want to describe something else using DC-in-HTML, **don't embed it in the (X)HTML document!**
- Don't confuse DC-in-HTML with RDFa!
 - RDFa provides a general method to encode RDF metadata in HTML content
 - DC metadata can be encoded using RDFa (using the DC-in-RDF expression), but that is a separate discussion
 - Hopefully, the two specifications can coexist



DC in (X)HTML

- **NOTE:** The existing recommendation is not compatible with the DCMI Abstract Model
- **AND:** The updated version is still only “Working Draft”
- **SO:** This presentation will be sketchy, with examples taken from the Working Draft *subject to change!*
- The important aspect is that the format is a faithful representation of the constructs in the DCMI Abstract Model



DC-in-(X)HTML: supported features

- A description set containing *a single description* (of the current document)
- In a non-literal value surrogate:
 - a maximum of *one value string*
 - a value string must be a plain value string, i.e. *no Syntax Encoding Schemes*
 - *a value URI must be provided*
 - the provision of a *vocabulary encoding scheme URI is not supported*



DC-in-(X)HTML: Basic notions

- the DC description is embedded into the <head> section of the (X)HTML document

```
<html>  
  <head>  
    ...DC description goes here...  
  </head>  
  <body>...
```

The examples in the following will use **XHTML** syntax, beware of small differences when using HTML



DC-in-(X)HTML: Use the profile!

- Declare the DC XHTML profile using the “profile” attribute!
- Declare the used namespaces using <link>

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head profile="http://dublincore.org/documents/2007/07/27/dc-html/">

  <title>Services to Government</title>

  <link rel="schema.DCTERMS" href="http://purl.org/dc/terms/" />
  <link rel="schema.MARCREL" href="http://www.loc.gov/loc.terms/relators/" />
```

WARNING: will change!



DC-in-(X)HTML: Literals

- Literal value surrogates are encoded using <meta>

```
<meta name="DCTERMS.title" content="Services to Government" />
```

- Language can be specified using “lang” in HTML and “xml:lang” in XHTML:

```
<meta name="DCTERMS.title" xml:lang="en" content="Services to Government" />
```

- Syntax encoding scheme can be given using the “scheme” attribute (don't forget the namespace!)

```
<link rel="schema.XSD" href="http://www.w3.org/2001/XMLSchema#" />  
<meta name="DCTERMS.modified" scheme="XSD.date" content="2007-07-22" />
```



DC-in-(X)HTML: Non-literals

- Non-literal value surrogates are encoded using `<link>`. The “href” attribute gives the value URI, “title” a single value string

```
<link rel="DCTERMS.subject" href="http://example.org/topics/archives" title="Archives" />
```

- Language for the single value string can be given using “lang” in HTML and “xml:lang” in XHTML:

```
<link rel="DCTERMS.subject" href="http://example.org/topics/archives" xml:lang="en" title="Archives" />
```

- No other features are supported



DC-in-(X)HTML: Casing etc

- Use property URIs as specified by DCMI
 - [DCTERMS.title](#) instead of DCTERMS.Title
- Don't "double-dot" for sub-properties,
 - [DCTERMS.abstract](#) instead of DCTERMS.description.abstract
- Don't use other constructs to indicate sub-properties
 - `<meta name="DCTERMS.abstract ...`
 - NOT `<meta name="DCTERMS.description" type="abstract" ...`



Some misc notes

Looking at other aspects of syntaxes



DCMI XML formats and GRDDL

- GRDDL, a W3C development
 - Gleaning Resource Descriptions from Dialects of Languages
- In short: adds an attribute for use in XML languages (DC-XML-Full/Min and XHTML)
- The attribute gives a link to an XSLT transform from the XML language to RDF
- This way, any XML language can be processed by an RDF+GRDDL processor
- DCMI syntaxes (DC-XML-Full/Min and DC-in-(X)HTML) will support GRDDL



GRDDL example

- [Link to examples](#)



Other syntaxes

- Sometimes a context requires a tailored format
 - such as OAI_PMH
 - These are exceptions - the DCMI syntaxes cover most situations (hopefully!)
- Sometimes a different standard defines a mapping to Dublin Core element
 - Atom syndication format
 - IEEE LOM
 - These reuse/map to the DC semantics in idiosyncratic ways



OAI_PMH

```
<header>
<identifier>oai:arXiv:cs/0112017</identifier>
<timestamp>2002-02-28</timestamp>
<setSpec>cs</setSpec>
<setSpec>math</setSpec>
</header>
<metadata>
<oai_dc:dc
  xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
<dc:title>Using Structural Metadata to Localize Experience of Digital
Content</dc:title>
<dc:creator>Dushay, Naomi</dc:creator>
<dc:subject>Digital Libraries</dc:subject>
<dc:description>With the increasing technical sophistication of both
informers and providers, there is increasing demand for
more meaningful experiences of digital information. We present a
framework that separates digital object experience, or rendering,
from digital object storage and manipulation, so the
rendering can be tailored to particular communities of users.
</dc:description>
<dc:description>Comment: 23 pages including 2 appendices,
8 figures</dc:description>
<dc:date>2001-12-14</dc:date>
<dc:type>e-print</dc:type>
<dc:identifier>http://arXiv.org/abs/cs/0112017</dc:identifier>
</oai_dc:dc>
</metadata>
<about>
<provenance
  xmlns="http://www.openarchives.org/OAI/2.0/provenance"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/provenance
http://www.openarchives.org/OAI/2.0/provenance.xsd">
<originDescription harvestDate="2002-02-02T14:10:02Z" altered="true">
<baseURL>http://the.oa.org</baseURL>
<identifier>oai:r2:klik001</identifier>
<timestamp>2002-01-01</timestamp>
<metadataNamespace>http://www.openarchives.org/OAI/2.0/oai_dc</metadataNamespace>
</originDescription>
</provenance>
</about>
```



Atom

atom:title - conveys a human-readable title for the entry

dc:title - A name given to the resource.

atom:author - construct that indicates the default author of the feed

dc:creator - An entity primarily responsible for making the content of the resource.

atom:contributor - construct that indicates a person or other entity who contributes to the feed.

dc:contributor - An entity responsible for making contributions to the content of the resource.

atom:tagline - conveys a human-readable description or tagline for the feed

dc:description - An account of the content of the resource.

atom:id - conveys a permanent, globally unique identifier for the feed.

dc:identifier - An unambiguous reference to the resource within a given context.

etc.



IEEE LOM

Nr	Name	Explanation	Size	Order
1.4	Description	<p>A textual description of the content of this learning object.</p> <p>NOTE— This description need not be in language and terms appropriate for the users of the learning object being described. The description should be in language and terms appropriate for those that decide whether or not the learning object being described is appropriate and relevant for the users.</p>	smallest permitted maximum: 10 items	unordere
1.5	Keyword	<p>A keyword or phrase describing the topic of this learning object.</p> <p>This data element should not be used for characteristics that can be described by other data elements.</p>	smallest permitted maximum: 10 items	unordere
1.6	Coverage	<p>The time, culture, geography or region to which this learning object applies.</p> <p>The extent or scope of the content of the learning object. Coverage will typically include spatial location (a place name or geographic coordinates), temporal period (a period label, date, or date range) or jurisdiction (such as a named administrative entity). Recommended best practice is to select a value from a controlled vocabulary (for example, the Thesaurus of Geographic Names [TGN]) and that, where appropriate, named places or time periods be used in preference to numeric identifiers such as sets of coordinates or date ranges.</p> <p>NOTE — This is the definition from the Dublin Core Metadata Element Set, version 1.1 [B1]. (http://www.dublincore.org/documents/dces/)</p>	smallest permitted maximum: 10 items	unordere



Can we imagine other syntaxes?

- “syntax” can be generalized to “binding” to non-textual systems, such as
 - Relational database schemas
 - Programming APIs (e.g. Java interfaces)
 - etc.
- This requires careful consideration of interoperability
 - What defines a syntax as a “Dublin Core” syntax?
 - Compatibility with the DCMI Abstract Model!



A few words about DC-Text

- A simple format for specification writing and email exchanges.
- Direct representation of the constructs in DCAM
- Similar in scope to N-TRIPLE for RDF
- Please don't use in applications!

```
@prefix dcterms: <http://purl.org/dc/terms/> .
DescriptionSet (
  Description (
    Statement (
      PropertyURI ( dcterms:title )
      LiteralValueString ( "DCMI Home Page" )
    )
  )
)
```



Simple DC / Qualified DC?

- DCMI has made use of the terms Simple/Qualified DC
- These terms are in the process of being recast
 - not as syntax options
 - but as Application Profiles
- Currently, these terms are outdated, and should not be used in formal contexts.



Multiple values

- Three options:
 - 1) Repeat the property!
 - 2) Multiple value strings (*only if the value is the same*)
 - 3) Structured value strings (comma-separated, etc.)
Please avoid!



Terms names vs. URIs

- In DC metadata, properties are *always* referred to using URIs
 - DCTERMS.title => <http://purl.org/dc/terms/title>
 - <dcterms:title> => <http://purl.org/dc/terms/title>
 - etc.
- So: base element names etc on the URIs, not on the labels
 - i.e. DCTERMS.title, not DCTERMS.Title, etc



dc: versus dcterms:

- Proposed to copy all 15 dc: properties to the dcterms: namespace
 - <http://purl.org/dc/elements/1.1/title> => <http://purl.org/dc/terms/title>
 - Result: All DCMI terms in one namespace.
 - PLUS: addition of domains and ranges to all terms in dcterms: namespace



Questions?