

An Introduction to RDF



Acknowledgement

- This presentation is based on the excellent RDF primer by the W3C available at <http://www.w3.org/TR/rdf-primer/> and <http://www.w3.org/2007/02/turtle/primer/> .
- Much of the material in this presentation is verbatim from the above Web site.

Presentation Outline

- **Basic concepts of RDF**
 - **Basics: resources, properties, values, statements, triples**
 - URIs and URIrefs
 - RDF graphs
 - Literals
 - Shorthand notation: QNames
 - URIrefs as vocabularies
 - Other data modeling concepts: structured values and blank nodes
- **Serialization of RDF graphs: RDF/XML and Turtle**

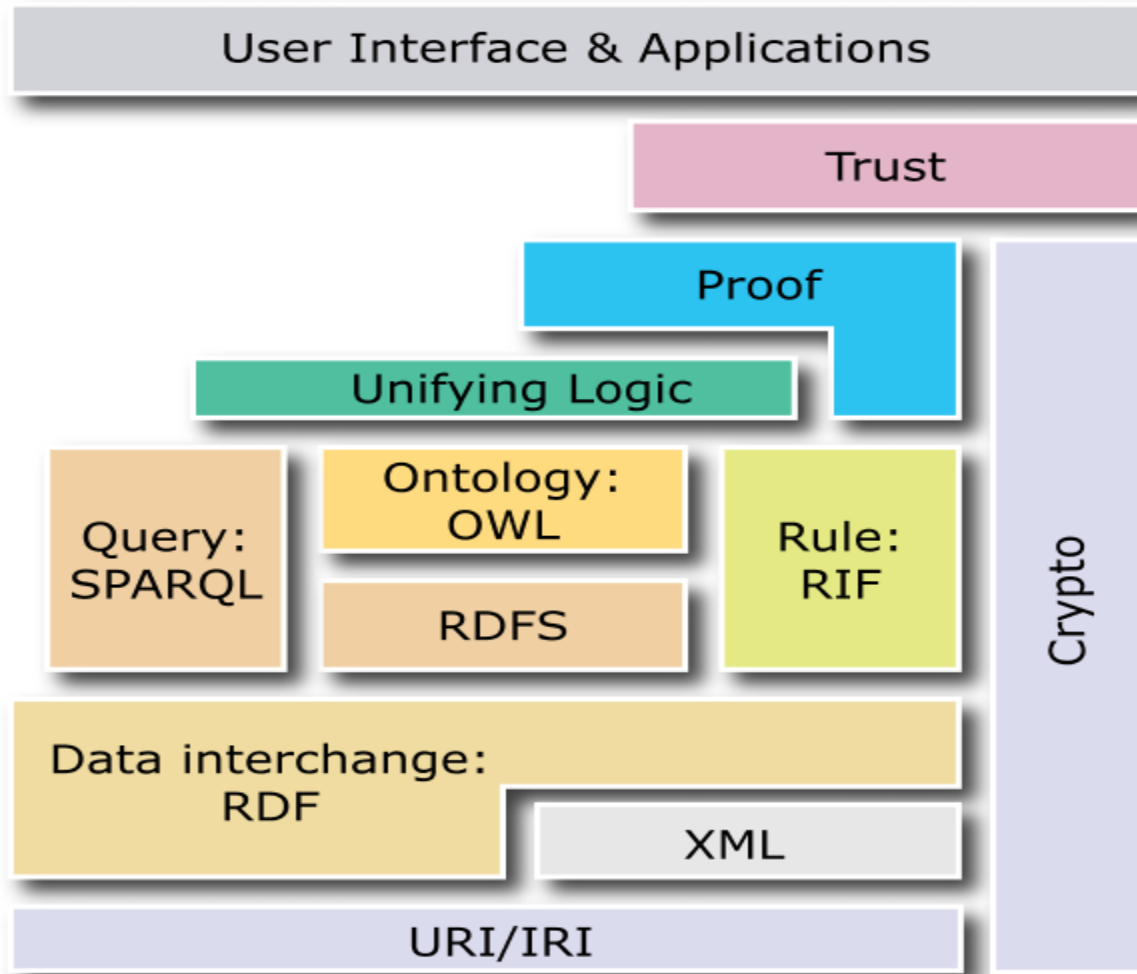
What is RDF?

- The **Resource Description Framework (RDF)** is a data model for representing information (especially **metadata**) about **resources** in the Web.
- RDF can also be used to represent information about things that can be **identified** on the Web, even when they cannot be directly **retrieved** on the Web (e.g., a book or a person).
- RDF is intended for situations in which information about Web resources needs to be **processed by applications**, rather than being only displayed to people.

Some History

- RDF draws upon ideas from knowledge representation, artificial intelligence, and data management, including:
 - Semantic networks
 - Frames
 - Conceptual graphs
 - Logic-based knowledge representation
 - Relational databases
- **Shameless self-promotion ☺** : The closest to RDF, pre-Web knowledge representation language is Telos:
John Mylopoulos, Alexander Borgida, Matthias Jarke, Manolis Koubarakis: Telos: Representing Knowledge About Information Systems. ACM Trans. Inf. Syst. 8(4): 325-362 (1990).

The Semantic Web “Layer Cake”



RDF Basics

- RDF is based on the idea of identifying resources using **Web identifiers** and describing resources in terms of simple **properties** and property **values**.
- To identify resources, RDF uses **Uniform Resource Identifiers (URIs)** and **URI references (URIrefs)**.
- **Definition:** A **resource** is anything that is identifiable by a URIref.

Example

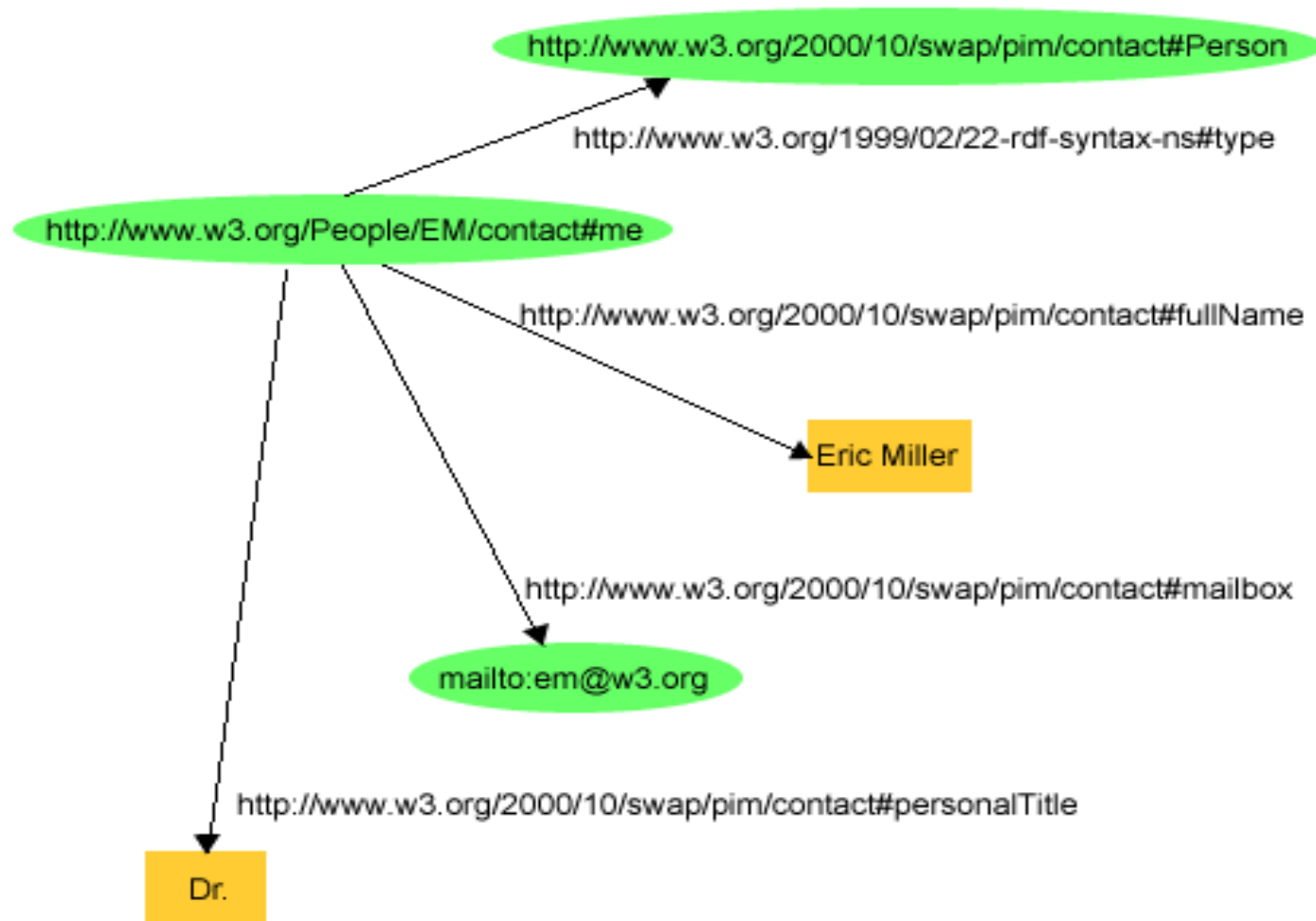
- Consider the following information:

“there is a Person identified by

`http://www.w3.org/People/EM/contact#me,`

whose name is Eric Miller, whose email address is `em@w3.org`, and whose title is Dr.”

Example (cont'd)



Basics (cont'd)

- Forget the long URIs for the moment!
- RDF is based on the idea that **the resources being described have properties which have values**, and that resources can be described by making **statements**, similar to the ones above, that specify those properties and values.
- **Terminology:**
 - The part that identifies the thing the statement is about is called the **subject**.
 - The part that identifies the property or characteristic of the subject that the statement specifies is called the **predicate**.
 - The part that identifies the value of that property is called the **object**.

Example

`http://www.example.org/index.html` has
a creator whose value is “John Smith”

- The **subject** is the URL

`http://www.example.org/index.html`

- The **predicate** is the word "creator"
- The **object** is the phrase “John Smith”

RDF Triples

- RDF statements can be written down using **triple notation**. In this notation, a statement is written as follows:

subject predicate object .

- **Example:**

```
<http://www.example.org/index.html>  
  <http://purl.org/dc/elements/1.1/creator>  
  <http://www.example.org/staffid/85740> .
```

```
<http://www.example.org/index.html>  
  <http://www.example.org/terms/creation-date> "August 16, 1999" .
```

```
<http://www.example.org/index.html>  
  <http://purl.org/dc/elements/1.1/language> "en" .
```

- **Note:** In this notation URIs are written out completely, in angle brackets.

Presentation Outline

- **Basic concepts of RDF**
 - Basics: resources, properties, values, statements, triples
 - **URIs and URIrefs**
 - RDF graphs
 - Literals
 - Shorthand notation: QNames
 - URIrefs as vocabularies
 - Other data modeling concepts: structured values and blank nodes
- Serialization of RDF graphs: RDF/XML and Turtle

Uniform Resource Identifiers

- The Web provides a general form of identifier, called the **Uniform Resource Identifier (URI)**, for identifying (naming) resources on the Web.
- Unlike URLs, URIs are not limited to identifying things that have network locations, or use other computer access mechanisms. A number of different **URI schemes (URI forms)** have been already been developed, and are being used, for various purposes.
- **Examples:**
 - `http:` (Hypertext Transfer Protocol, for Web pages)
 - `mailto:` (email addresses), e.g., `mailto:em@w3.org`
 - `ftp:` (File Transfer Protocol)
 - `urn:` (Uniform Resource Names, intended to be persistent location-independent resource identifiers), e.g., `urn:isbn:0-520-02356-0` (for a book)
- No one person or organization controls who makes URIs or how they can be used. While some URI schemes, such as URL's `http:`, depend on centralized systems such as DNS, other schemes, such as `freenet:`, are **completely decentralized**.

URIs (cont'd)

- A **URI reference** (or **URIref**) is a URI, together with an optional fragment identifier at the end.

Example: the URIref <http://www.example.org/index.html#section2> consists of the URI `http://www.example.org/index.html` and (separated by the "#" character) the fragment identifier `section2`.

- URIrefs may be either **absolute** or **relative**.
- An **absolute** URIref refers to a resource independently of the context in which the URIref appears, e.g., the URIref `http://www.example.org/index.html`.
- A **relative** URIref is a shorthand form of an absolute URIref, where some prefix of the URIref is missing, and information from the context in which the URIref appears is required to fill in the missing information.

Example: the relative URIref `otherpage.html`, when appearing in a resource `http://www.example.org/index.html`, would be filled out to the absolute URIref `http://www.example.org/otherpage.html`.

URIs (cont'd)

- A (relative) **URIref consisting of just a fragment identifier** is considered equivalent to the URIref of the document in which it appears, with the fragment identifier appended to it.

Example: Within the document

`http://www.example.org/index.html`, if `#section2` appeared as a URIref, it would be considered equivalent to the absolute URIref `http://www.example.org/index.html#section2`.

- A **URIref without a URI part** is considered a reference to the current document (the document in which it appears). So, an **empty URIref** within a document is considered equivalent to the URIref of the document itself.
- See the report from the Joint W3C/IETF URI Planning Interest Group at <http://tools.ietf.org/html/rfc3305> for more information.

URIrefs in RDF

- RDF uses URIrefs to identify and name the subjects, predicates, and objects in statements.
- RDF URIrefs can contain **Unicode** characters, allowing many languages to be reflected in URIrefs. So, more precisely, RDF uses **Internationalized Resource Identifiers (IRIs) and IRIrefs.**
- See <http://tools.ietf.org/html/rfc3987> which defines IRIs.

URIrefs in RDF (cont'd)

- Both RDF and Web browsers use URIrefs to **identify things**. However, RDF and browsers interpret URIrefs in slightly different ways:
 - RDF uses URIrefs **only** to identify things.
 - Browsers also use URIrefs to **retrieve** things.
- What is the difference?
 - When a URIref is used in a **browser**, there is the expectation that it identifies a resource that can actually be **retrieved**: that something is actually "at" the location identified by the URI.
 - In **RDF**, a URIref may be used to identify something, such as a person, that **cannot** be retrieved on the Web.
- But important uses of RDF, like **Linked Data**, insist that we use HTTP URIs so data identified by a URI can be retrieved (<https://www.w3.org/wiki/LinkedData>).

URIrefs in RDF (cont'd)

- Another difference is in the way URIrefs with **fragment identifiers** are handled. Consider the following URIrefs:
`http://www.example.org/index.html`
`http://www.example.org/index.html#Section2`
- In normal HTML usage, these **URIrefs are related** (they both refer to the same document, the second one identifying a location within the first one).
- RDF assumes **no particular relationship** between these two URIrefs. As far as RDF is concerned, they are syntactically different URI references, and hence may refer to unrelated things.

Presentation Outline

- **Basic concepts of RDF**
 - Basics: resources, properties, values, statements, triples
 - URIs and URIrefs
 - **RDF graphs**
 - Literals
 - Shorthand notation: QNames
 - URIrefs as vocabularies
 - Other data modeling concepts: structured values and blank nodes
- Serialization of RDF graphs: RDF/XML and Turtle

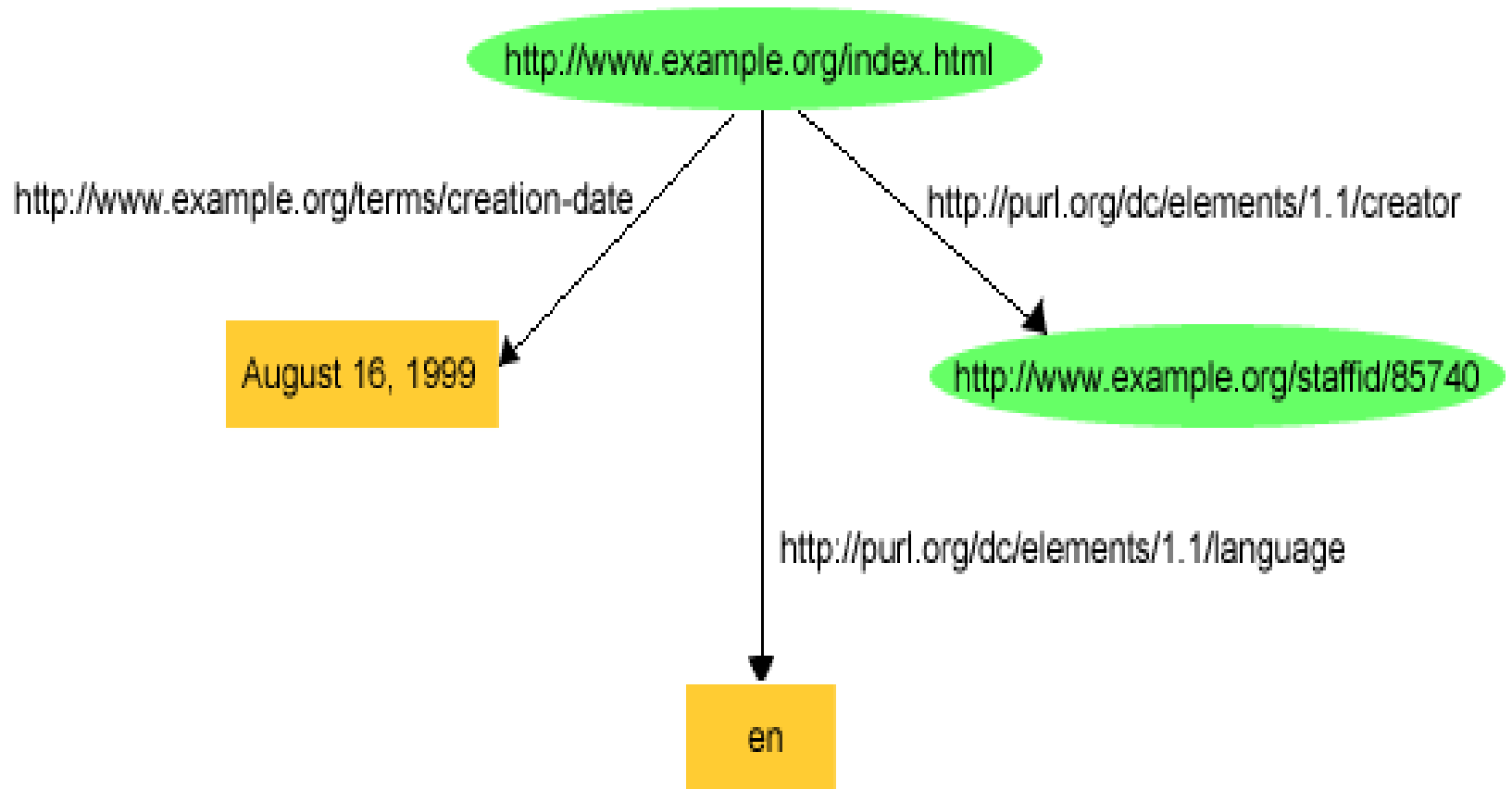
RDF Graphs

- Graphically, RDF models statements by **nodes** and **arcs** in a **graph**.
- In the RDF graph notation, a **statement** is represented by:
 - a node for the subject
 - a node for the object
 - an arc for the predicate, directed from the subject node to the object node.
- A **node** may be identified by a **URIref** or it can be a **literal** (it can also be a **blank node**; we will explain this later).
- An **arc** is identified by a **URIref**.
- **Note:** We will draw RDF graphs as **directed graphs**. But strictly speaking, directed graphs are not sufficient for capturing all of RDF (e.g., directed graphs assume that the sets of nodes and arcs are disjoint but RDF allows a property as a subject of a statement).

Example

- Consider the following statements:
 - `http://www.example.org/index.html` has a creation-date whose value is August 16, 1999.
 - `http://www.example.org/index.html` has a language whose value is English.

The RDF Graph of the Example



RDF and Related Data Models

- In terms of the **relational model**, an RDF statement is similar to a **tuple in a relation** called *Triples* or *Graph* with attributes *Subject*, *Predicate* and *Object*.
- In terms of **first-order logic**, an RDF statement is similar to an **atomic formula**

triple(subj,pred,obj)

where *triple* is a first-order logic predicate and *subj*, *pred* and *obj* are constants.

Presentation Outline

- **Basic concepts of RDF**
 - Basics: resources, properties, values, statements, triples
 - URIs and URIrefs
 - RDF graphs
 - **Literals**
 - Shorthand notation: QNames
 - URIrefs as vocabularies
 - Other data modeling concepts: structured values and blank nodes
- Serialization of RDF graphs: RDF/XML and Turtle

Literals

- There are two kinds of literals:
 - **Plain (untyped).**
 - **Typed.**

- Example:



- What is 27? Number or string?

Plain Literals

- Plain literals have a **lexical form** (their lexical value) and optionally a **language tag**.
- **Example:** `"27"` , `"Hello world"@en`

Typed Literals

- An **RDF typed literal** is formed by pairing a string with a URIref that identifies a particular **datatype**.
- **Example:**
`"27"^^http://www.w3.org/2001/XMLSchema#integer`

Typed Literals (cont'd)

- RDF has **no built-in set of datatypes** of its own. RDF typed literals simply provide a way to explicitly indicate, for a given literal, what datatype should be used to interpret it. The datatypes used in typed literals are defined **externally to RDF, and identified by their datatype URIs**.
- **Exception:** RDF defines a built-in datatype with the URIref `rdf:XMLLiteral` to represent XML content as a literal value. This is useful when we want to bring into an RDF sentence, content that has already been defined in XML (e.g., in the case of spatial data, a polygon defined in GML).
- RDF datatype concepts are based on a conceptual framework from **XML Schema datatypes**. This framework defines the **value space**, the **lexical space** and the **lexical-to-value** mapping for a datatype (see the RDF specifications for more details).

XML Schema Datatypes

- `xsd:integer`
- `xsd:boolean`
- `xsd:string`
- `xsd:date`
- ...

Presentation Outline

- **Basic concepts of RDF**
 - Basics: resources, properties, values, statements, triples
 - URIs and URIrefs
 - RDF graphs
 - Literals
 - **Shorthand notation: QNames**
 - URIrefs as vocabularies
 - Other data modeling concepts: structured values and blank nodes
- Serialization of RDF graphs: RDF/XML and Turtle

Triple Notation – QNames as Shorthands

- The full triple notation results in very long lines.
- **Shorthand:** We can use an **XML qualified name** (or **QName**) without angle brackets as an abbreviation for a full URI reference.
- A **QName** consists of a **prefix** that has been assigned to a **namespace URI**, followed by a **colon**, and then a **local name**. The full URIref is formed from the QName by appending the local name to the namespace URI assigned to the prefix.
- The concepts of **names** and **namespaces** used in RDF originate in XML.

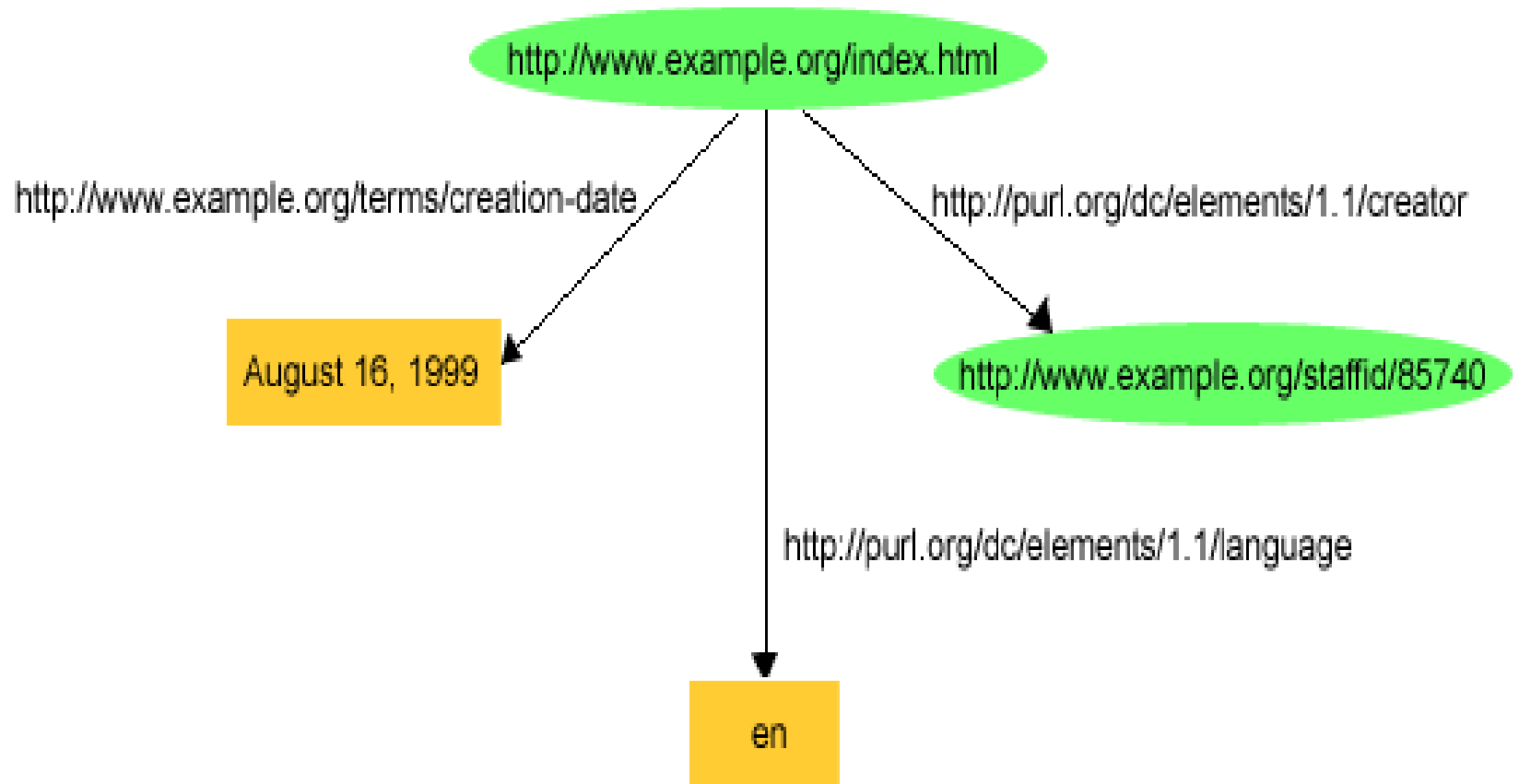
XML Namespaces

- A **namespace** is a way of identifying a subset of a set of names (e.g., the set of possible names of resources in the Web) which acts as a qualifier for the names in this subset.
- **XML namespaces** are used for providing uniquely named elements and attributes in an XML document.
- **XML namespaces help us eliminate ambiguity in an XML document.** For example, an XML document can use `id` to refer to both identifiers of customers and products if `id` is prefixed by an appropriate name space (e.g., <http://customers.org> and <http://products.com>).
- A **namespace** is created by creating a URI for it. By qualifying names with the URIs of their namespaces, **anyone can create their own names and properly distinguish them from names with identical spellings created by others.**
- See the **W3C Recommendation “Namespaces in XML 1.0”** available at <http://www.w3.org/TR/REC-xml-names/>.

QNames as Shorthands (cont'd)

- **Example:** If the QName prefix `foo` is assigned to the namespace URI `http://example.org/somewhere/`, then the QName `foo:bar` is shorthand for the URIref `http://example.org/somewhere/bar`.
- **More Examples:**
 - `prefix rdf: , namespace URI: http://www.w3.org/1999/02/22-rdf-syntax-ns#`
 - `prefix rdfs: , namespace URI: http://www.w3.org/2000/01/rdf-schema#`
 - `prefix dc: , namespace URI: http://purl.org/dc/elements/1.1/`
 - `prefix owl: , namespace URI: http://www.w3.org/2002/07/owl#`
 - `prefix ex: , namespace URI: http://www.example.org/ (or http://www.example.com/)`
 - `prefix xsd: , namespace URI: http://www.w3.org/2001/XMLSchema#`

Example



Example (cont'd)

- The previous graph can be written in triples notation using the shorthands we introduced earlier as follows:
`ex:index.html dc:creator exstaff:85740 .`
`ex:index.html exterms:creation-date "August 16, 1999" .`
`ex:index.html dc:language "en" .`
- We will later see **Turtle**, a textual syntax for RDF graphs that makes use of this shorthand notation based on QNames.

Presentation Outline

- **Basic concepts of RDF**
 - Basics: resources, properties, values, statements, triples
 - URIs and URIrefs
 - RDF graphs
 - Literals
 - QNames
 - **URIrefs as vocabularies**
 - Structured values and blank nodes
- **Serialization of RDF graphs: RDF/XML and Turtle**

URIs as Vocabulary

- Since RDF uses URIs instead of words to name things in statements, URIs define **vocabularies** in RDF.
- The URIs in RDF vocabularies are typically organized so that they can be represented as a **set of QNames with a common prefix**:
 - A common namespace URI is chosen for all terms in a vocabulary, typically a URI under the control of whoever is defining the vocabulary.
 - URIs that are contained in the vocabulary are formed by appending individual local names to the end of the common URI.

Example



- DBpedia (<http://wiki.dbpedia.org/About>) is a large data set which has been derived from Wikipedia by extracting various kinds of structured information from Wikipedia editions in 14 languages and combining this information into a huge, cross-domain knowledge base.
- In the DBpedia data set, each thing is identified by a URIref of the form `http://dbpedia.org/resource/Name`, where Name is taken from the URL of the source Wikipedia article, which has the form `http://en.wikipedia.org/wiki/Name`. Thus, each resource is tied directly to an English-language Wikipedia article.

DBpedia (cont'd)

- The URIref

`http://dbpedia.org/resource/Greece`
is the DBpedia resource about Greece.

- The prefix `dbpedia` can be used instead of `http://dbpedia.org/resource/`
- For example: `dbpedia:Greece`

URIrefs as Vocabulary (cont'd)

- RDF uses this same approach to define its own vocabulary of terms with special meanings in RDF:
 - The URIrefs in the RDF vocabulary all begin with `http://www.w3.org/1999/02/22-rdf-syntax-ns#`, conventionally associated with the QName prefix `rdf:`.
 - The RDF Vocabulary Description Language defines an additional set of terms having URIrefs that begin with `http://www.w3.org/2000/01/rdf-schema#`, conventionally associated with the QName prefix `rdfs:`.
- Where a specific QName prefix is commonly used in connection with a given set of terms in this way, the QName prefix itself is sometimes used as the **name of the vocabulary**. For example, someone might refer to "**the `rdfs:` vocabulary**".

URIs as Vocabulary (cont'd)

- **Convention:** Organizations typically use a vocabulary's namespace URI as the URL of a Web resource that provides further information about that vocabulary.
- **Example:** the QName prefix `dc:` with the namespace URI `http://purl.org/dc/elements/1.1` refers to the **Dublin Core vocabulary**.
 - Accessing this namespace URI in a Web browser will retrieve additional information about the Dublin Core vocabulary (specifically, RDFS definitions of the Dublin core vocabulary).
 - **Reminder:** this is just a useful convention. RDF does not assume that a namespace URI identifies a retrievable Web resource.

URIs as Vocabulary (cont'd)

- Using URIs as subjects, predicates, and objects in RDF statements supports the **development and use of shared vocabularies** on the Web.
- People can **discover and begin using vocabularies** already used by others to describe things, reflecting a **shared understanding of those concepts**.

Example



- Consider the triple `ex:index.html dc:creator exstaff:85740` .

The predicate `dc:creator`, when fully expanded as a URIref, is an unambiguous reference to the "creator" attribute in the Dublin Core metadata attribute set, a widely-used set of attributes (properties) for describing a wide range of networked resources (see <https://www.dublincore.org/resources/userguide/>) .

The writer of this triple is effectively saying that the relationship between the Web page and the creator of the page is exactly the concept identified by `http://purl.org/dc/elements/1.1/creator`.

Another person familiar with the Dublin Core vocabulary, or who finds out what `dc:creator` means (say by looking up its definition on the Web) will know what is meant by this relationship. In addition, based on this understanding, people can **write programs to behave in accordance with that meaning** when processing triples containing the predicate `dc:creator`.

Another Example



- The Friend of a Friend (FOAF) vocabulary at <http://xmlns.com/foaf/spec/>.
- The FOAF project is creating a Web of machine-readable pages (written in RDF) describing people, the links between them and the things they create and do.

URIs as Vocabulary (cont'd)

- RDF gives meaning to the terms defined in the relevant RDF vocabularies `rdf:` and `rdfs:`.
- Others have defined the meaning of terms in other important vocabularies e.g., `dc:`

Presentation Outline

- **Basic concepts of RDF**
 - Basics: resources, properties, values, statements, triples
 - URIs and URIrefs
 - RDF graphs
 - Literals
 - QNames
 - URIrefs as vocabularies
 - **Other data modeling concepts: structured values and blank nodes**
- Serialization of RDF graphs: RDF/XML and Turtle

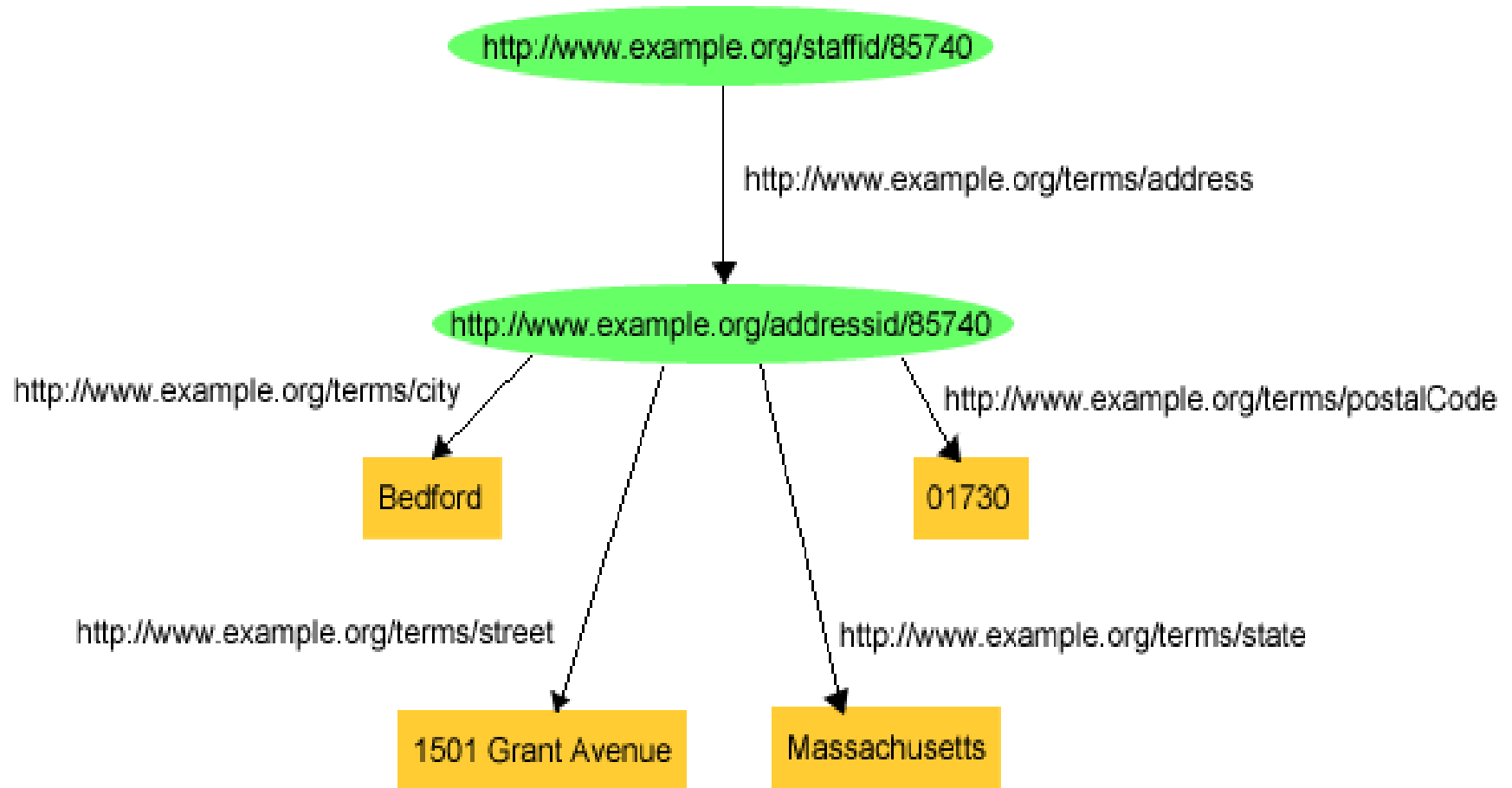
Structured Values in RDF

- Consider the triple:

```
exstaff:85740  exterm:address  "1501 Grant Avenue,  
Bedford, Massachusetts 01730" .
```

- What if the address needs to be represented as a **structure** consisting of separate street, city, state, and postal code values? How would this be done in RDF?
- Structured information is represented in RDF by considering **the aggregate thing to be described** (like John Smith's address) **as a resource**, and then making statements about that new resource.

Structured Values in RDF (cont'd)



Structured Values in RDF (cont'd)

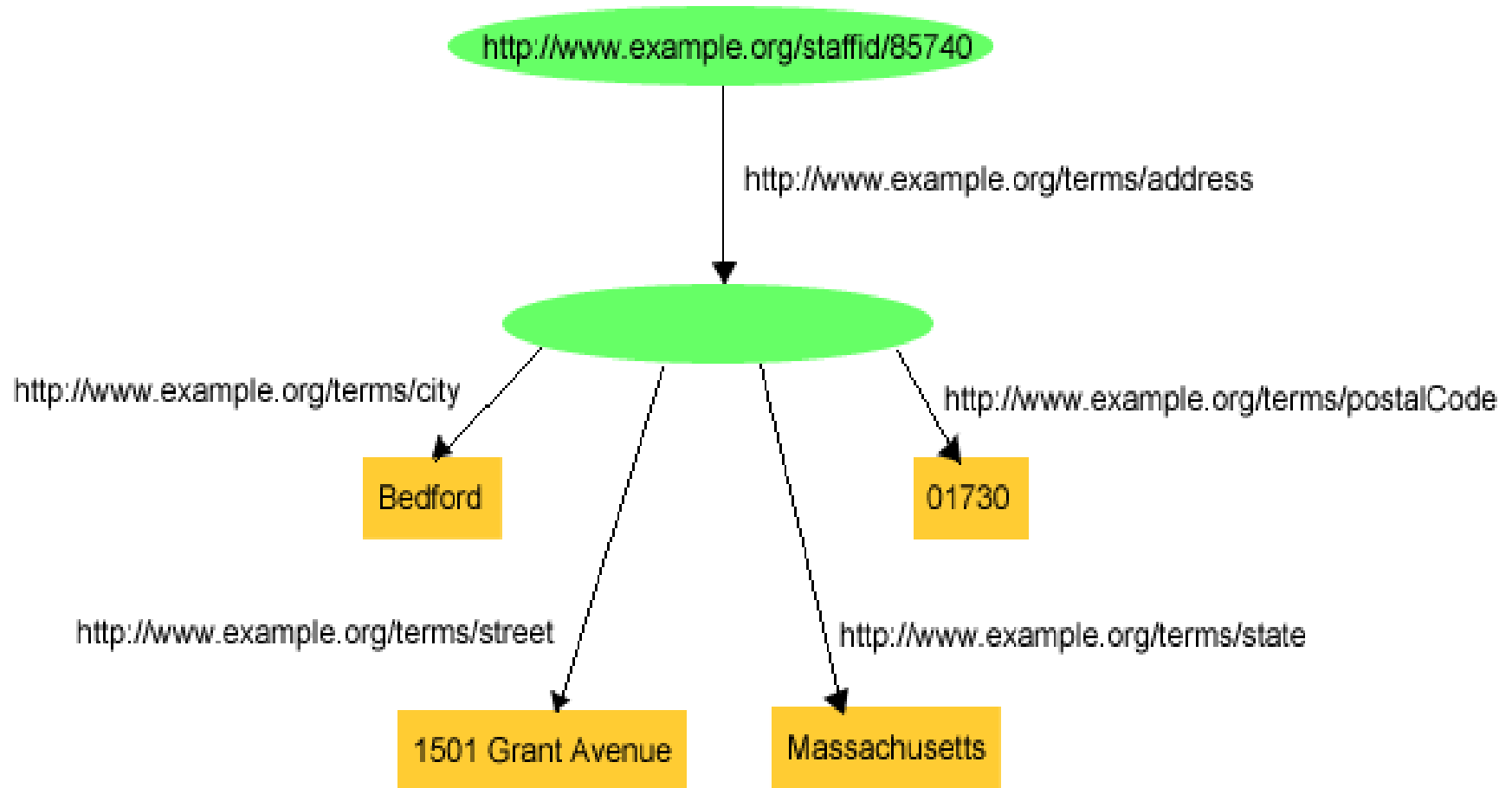
- Or in triples notation:

```
exstaff:85740 exterms:address exaddressid:85740 .  
exaddressid:85740 exterms:street "1501 Grant Avenue" .  
exaddressid:85740 exterms:city "Bedford" .  
exaddressid:85740 exterms:state "Massachusetts" .  
exaddressid:85740 exterms:postalCode "01730" .
```

Structured Values in RDF (cont'd)

- This way of representing structured information in RDF can involve generating **numerous “intermediate” URIs** such as `exaddressid:85740` to represent aggregate concepts such as John's address. Such concepts may never need to be referred to directly from outside a particular graph, and hence **may not require “universal” identifiers.**
- RDF allows us to use **blank nodes** and **blank node identifiers** to deal with this issue.

Blank Nodes



Blank Nodes Using Triples

```
exstaff:85740  exterms:address ??? .  
??? exterms:street "1501 Grant Avenue" .  
??? exterms:city "Bedford" .  
??? exterms:state "Massachusetts" .  
??? exterms:postalCode "01730" .
```

- But what about **different blank nodes**?
Can we handle them using this notation?

Blank Node Identifiers

```
exstaff:85740  exterms:address _:johnaddress .  
_:johnaddress exterms:street "1501 Grant Avenue" .  
_:johnaddress exterms:city "Bedford" .  
_:johnaddress exterms:state "Massachusetts" .  
_:johnaddress exterms:postalCode "01730" .
```

- In a triples representation of a graph, each distinct blank node must be given a **different** blank node identifier.
- Blank node identifiers have significance only within the triples representing a **single** graph.
- Blank node identifiers may only appear as subjects or objects in triples; blank node identifiers **may not be used as predicates** in triples.

Semantics of Blank Nodes

- In terms of **first-order logic**, a blank node corresponds to an existentially quantified variable (or a Skolem constant). Thus, a graph with blank nodes is similar to an **existentially quantified first order logic statement** or a **first-order database** in the sense of Reiter:

Raymond Reiter: Towards a Logical Reconstruction of Relational Database Theory. Appears in the collection: Brodie M. L., Mylopoulos J., and Schmidt J. W. (eds.), On Conceptual Modelling: Perspectives from Artificial Intelligence, Database and Programming Languages, pp. 191-233, Springer-Verlag.

- In terms of the **relational model**, a blank node corresponds to a **marked null value**. Thus, a statement with a blank node identifier is similar to a **tuple in a relation** in the marked nulls model of Imielinski and Lipski:

Tomasz Imielinski, Witold Lipski Jr.: Incomplete Information in Relational Databases. J. ACM 31(4): 761-791 (1984).

Semantics of Blank Nodes (cont'd)

- The RDF semantics in <http://www.w3.org/TR/rdf-mt/> takes this “**existential view**” of blank nodes and defines appropriate semantics for them.
- SPARQL, the standard query language for RDF, considers blank nodes **as constants scoped in the graph where they appear**.
- In practice, it varies how people use blank when they publish linked data.
- See the paper
Alejandro Mallea, Marcelo Arenas, Aidan Hogan, Axel Polleres.
On Blank Nodes. Proceedings of the International Semantic Web Conference 2011.
pages 421-437.
for a nice study of the relevant issues.

Blank Nodes (cont'd)

- Blank nodes are useful to represent **n-ary relationships** in RDF e.g., the relationship between John Smith and the street, city, state, and postal code components of his address.
- Blank nodes are also useful to more accurately make statements about **resources that may not have URIs**, but that are described in terms of relationships with other resources that do have URIs.

Example

- When making statements about a person, say Jane Smith, is it natural to use a URI based on that person's email address as her URI, e.g., `mailto:jane@example.org` ?
- Well, if we do so, how are we going to record information both about **Jane's mailbox** (e.g., the server it is on) as well as about **Jane herself** (e.g., her current physical address)? Similarly, if we use her Web page URI etc.

Example (cont'd)

- **Blank nodes to the rescue:** When Jane herself does not have a URI, a blank node provides a more accurate way of modeling this situation.

```
_:jane exterm:mailbox <mailto:jane@example.org> .  
_:jane rdf:type exterm:Person .  
_:jane exterm:name "Jane Smith" .  
_:jane exterm:empID "23748" .  
_:jane exterm:age "26" .
```

Presentation Outline

- Basic concepts of RDF
 - Basics: resources, properties, values, statements, triples
 - URIs and URIrefs
 - RDF graphs
 - Literals
 - QNames
 - URIrefs as vocabularies
 - Other data modeling concepts: structured values and blank nodes
- **Serialization of RDF graphs: RDF/XML and Turtle**

Machine Readable Formats for RDF

RDF has a number of machine readable formats:

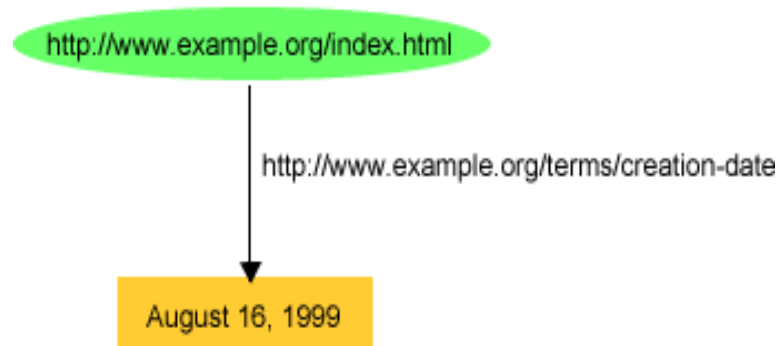
- RDF/XML
- Turtle (Terse RDF Triple Language)
- N3
- ...

An XML Syntax for RDF: RDF/XML

- The conceptual model for RDF is a graph.
- RDF provides an XML syntax for **writing down and exchanging RDF graphs**, called **RDF/XML**.
- RDF/XML is the **normative syntax** for writing RDF.

Example

<http://www.example.org/index.html> has a creation-date whose value is August 16, 1999.



Example in RDF/XML

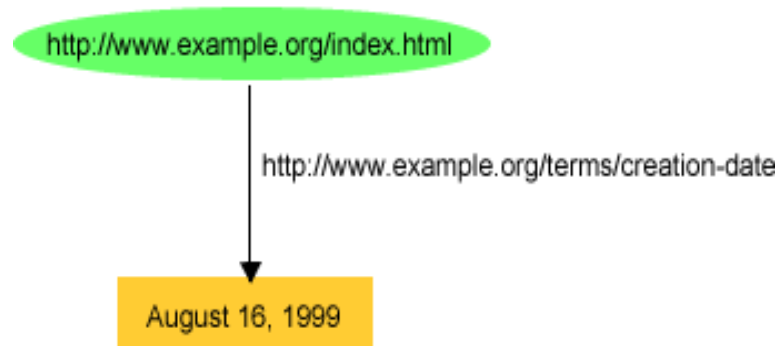
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:exterms="http://www.example.org/terms/">
  <rdf:Description rdf:about="http://www.example.org/index.html">
    <exterms:creation-date>August 16, 1999</exterms:creation-date>
  </rdf:Description>
</rdf:RDF>
```

Turtle

- **Turtle** provides another textual syntax for writing down and exchanging RDF graphs.
- **Turtle** is based on the **triple notation** we used so far, but also includes some additional syntactic means to **simplify the specification** of RDF graphs.

Example I

<http://www.example.org/index.html> has a creation-date whose value is August 16, 1999.



Example 1 in Turtle

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

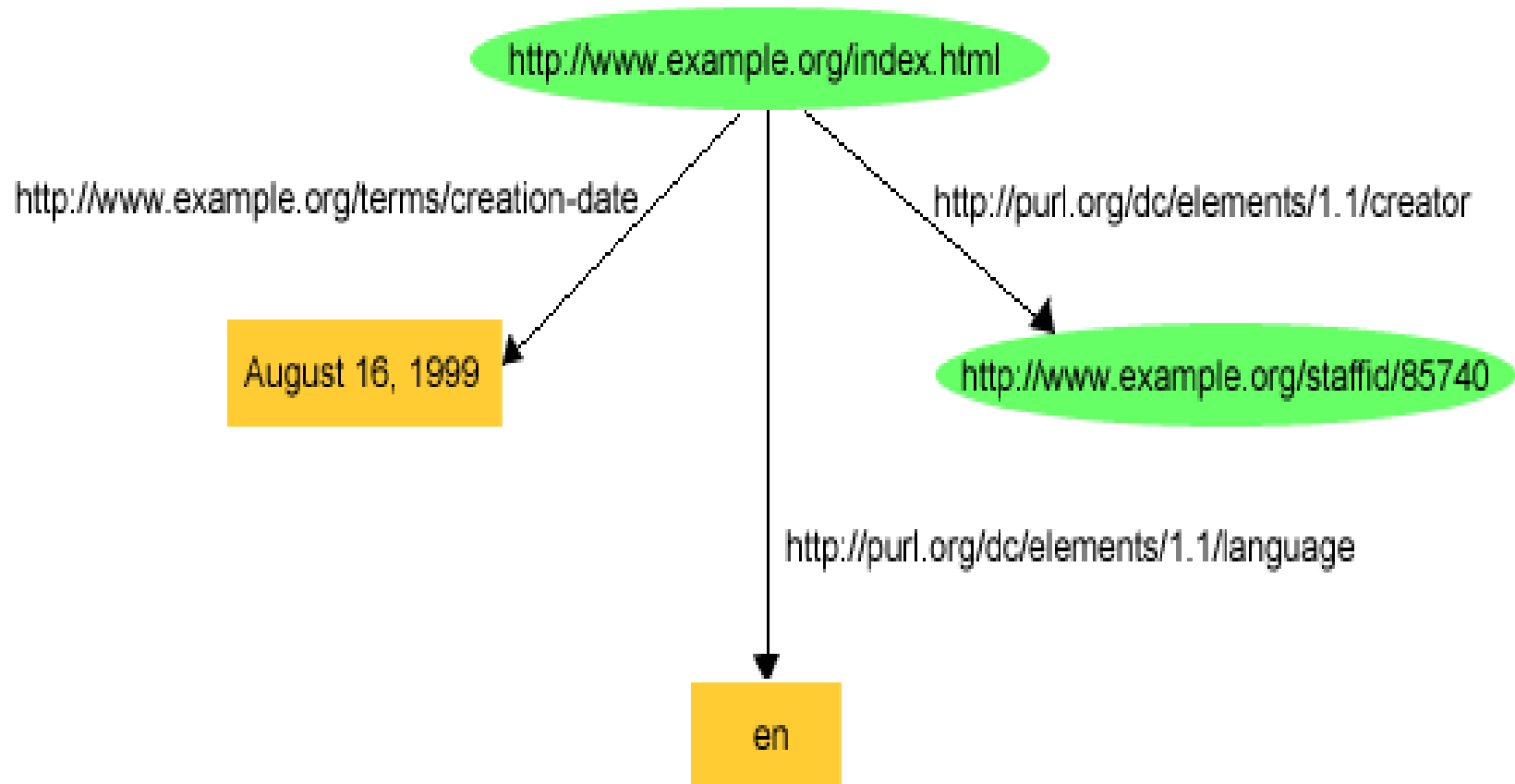
```
@prefix exterms: <http://www.example.org/terms/>.
```

```
<http://www.example.org/index.html>
```

```
    exterms:creation-date "August 16, 1999" .
```

- **Notation:** The `@prefix` keyword in the first two lines declares namespaces `rdf:` and `exterms:`

Example II



Example II Using Triples

```
<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/creator>  
  <http://www.example.org/staffid/85740> .
```

```
<http://www.example.org/index.html>  
  <http://www.example.org/terms/creation-date> "August 16, 1999" .
```

```
<http://www.example.org/index.html>  
  <http://purl.org/dc/elements/1.1/language> "en" .
```

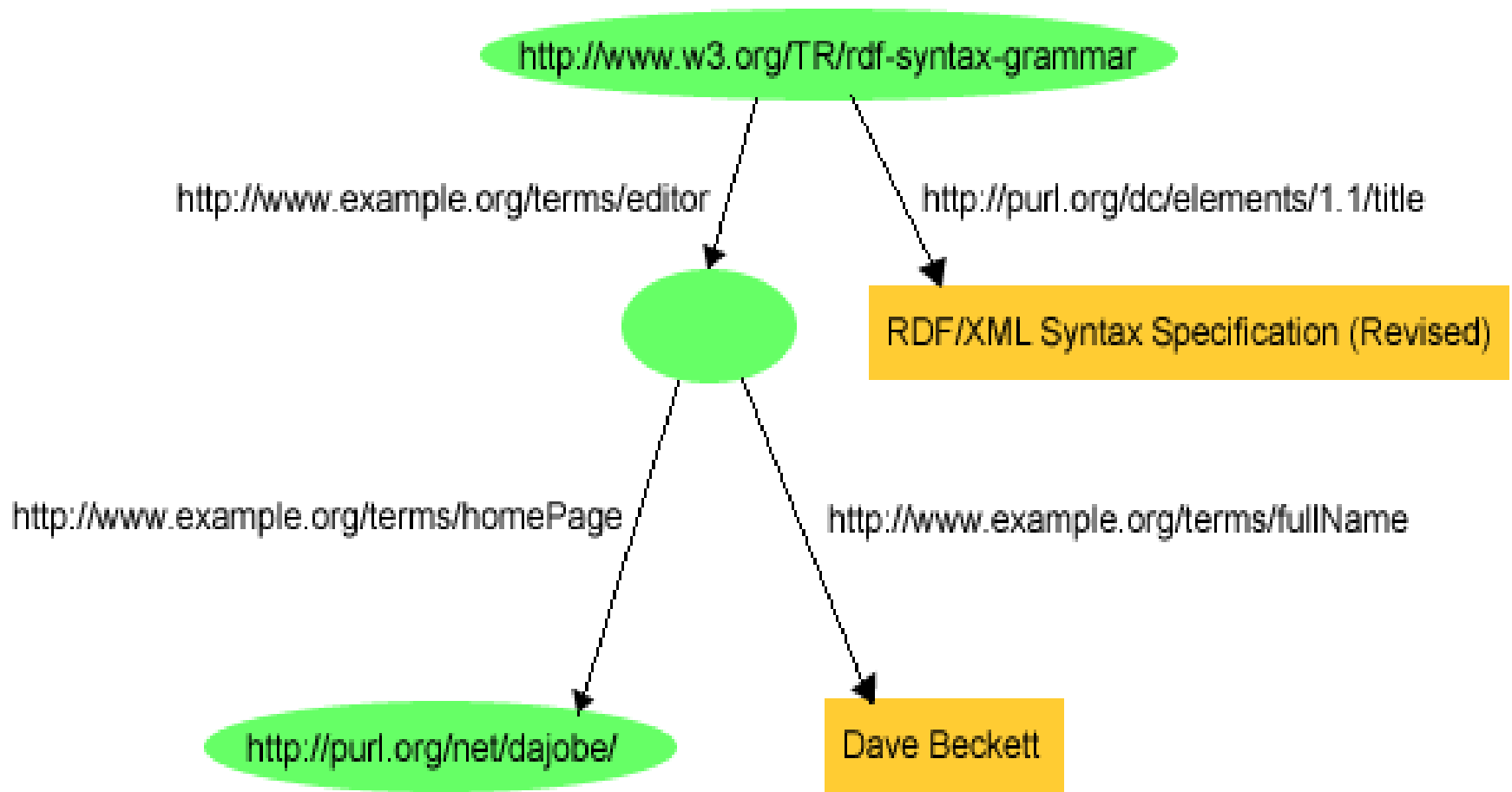
Example II Using Turtle

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dc: <http://purl.org/dc/elements/1.1/#>.
@prefix exterm: <http://www.example.org/terms/>.

<http://www.example.org/index.html>
  exterm:creation-date "August 16, 1999";
  dc:language "en";
  dc:creator <http://www.example.org/staffid/85740>.
```

- **Notation:** In this case Turtle allows the semi-colon to separate **predicate-object pairs** for the same subject. A list of such pairs is terminated with a period.

Blank Nodes in Turtle



Blank Nodes in Turtle (cont'd)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dc: <http://purl.org/dc/elements/1.1/#>.
@prefix exterm: <http://www.example.org/terms/>.
```

```
<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax Specification (Revised)";
  exterm:editor _:abc.
```

```
_:abc
  exterm:fullName "Dave Beckett";
  exterm:homePage <http://purl.org/net/dajobe/>.
```

- **Notation:** Blank node identifiers are used.

Anonymous Blank Nodes

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dc: <http://purl.org/dc/elements/1.1/#>.
@prefix exterm: <http://www.example.org/terms/>.
```

```
<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax Specification (Revised)";
  exterm:editor [
    exterm:fullName "Dave Beckett";
    exterm:homePage <http://purl.org/net/dajobe/>.
  ].
```

- **Notation:** This is reminiscent of notations for complex objects from the area of database systems.

Summary

- In RDF, **statements** about a domain are encoded by **triples**.
- **Triples** are of the form
subject predicate object .
- The **subject** of a triple must be a **URIref** or a **blank node**.
- The **predicate** of a triple must be a **URIref**.
- The **object** of a triple must be a **URIref**, a **literal** or a **blank node**.
- Literals are not allowed as subjects or predicates. Blank nodes are not allowed as predicates.
- **Definition:** An **RDF graph** is a set of RDF triples.

RDF 1.1

- RDF, as we presented it, became a W3C standard in 2004. This is now referred as **RDF 1.0**.
- In 2014, **RDF1.1** was defined. It is a simple extension of RDF1.0.

What's New in RDF 1.1

- Identifiers are IRIs.
- All literals have a data type.
- Literals with a language tag have data type `rdf:langString`.
- The concept of RDF dataset was defined. An **RDF dataset** is a collection of RDF graphs. We will see again this concept when we discuss the query language SPARQL.
- Compatible XSD data types were specified.
- The datatype `rdf:HTML` was introduced so that we can have objects of triples to be HTML code.
- More serialization formats: RDF/XML, RDFa, N-Triples, N-Quads, Turtle, TriG and JSON-LD.
- You can see more details in <https://www.w3.org/TR/rdf11-new/> .

Readings

- Chapter 2 of the book “Foundations of Semantic Web Technologies” or Chapters 2 and 3 of the Semantic Web Primer available from <http://www.csd.uoc.gr/~hy566/SWbook.pdf> .
- The following material from the Semantic Web Activity Web page on RDF <http://www.w3.org/RDF/> :
 - RDF Primer (<https://www.w3.org/TR/rdf11-primer/>)
 - Resource Description Framework (RDF): Concepts and Abstract Syntax (<https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>)
 - The Turtle language (<https://www.w3.org/TR/2014/REC-turtle-20140225/>)
- Check out the content published at the RDF namespace URI:
 - <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
where you will find an RDF Schema description of the RDF vocabulary given in RDF/XML!
- The DBpedia project (<https://www.dbpedia.org/>) a nice application of RDF and Linked Data.