# Fresnel - A Browser-Independent Presentation Vocabulary for RDF

Christian Bizer[1], Ryan Lee[2], and Emmanuel Pietriga[3]

[1] Freie Universität Berlin, Germany
chris@bizer.de
[2] W3C/MIT CSAIL, Cambridge, USA
ryanlee@w3.org
[3] INRIA & Laboratoire de Recherche en Informatique (LRI), Orsay, France
emmanuel.pietriga@inria.fr

**Abstract.** Semantic Web browsers and other tools aimed at displaying RDF data to end users are all concerned with the same problem: presenting content primarily intended for machine consumption in a human-readable way. Their solutions differ but in the end address the same two high-level issues, no matter the underlying representation paradigm: specifying (i) *what* information contained in RDF models should be presented (content selection) and (ii) *how* this information should be presented (content formatting and styling). However, each tool currently relies on its own *ad hoc* mechanisms and vocabulary for specifying RDF presentation knowledge, making it difficult to share and reuse such knowledge across applications. Recognizing the general need for presenting RDF content to users and wanting to promote the exchange of presentation knowledge, we developed Fresnel as a browser-independent extensible vocabulary of core RDF display concepts.

## 1 Introduction

Software agents are the primary consumers of Semantic Web content. RDF is thus designed to facilitate machine interpretability of information and does not define a visual presentation model since human readability is not one of its stated goals. However, RDF applications are not only about the semantic processing of information. Information coming from the Semantic Web, either directly from RDF repositories or as a result of complex processes, often must be presented to users. Displaying RDF data in a user-friendly manner is a problem addressed by various types of applications using different representation paradigms. Tools like IsaViz [1] and Welkin [2] represent RDF models as node-link diagrams, explicitly showing their graph structure. Other tools use nested box layouts (Longwell [3]) or table-like layouts (Brownsauce [4], Noadster [5], Swoop [6]) for displaying properties of RDF resources with varying levels of details. A third approach combines these paradigms and extends them with specialized user interface widgets designed for specific information items like calendar data, tree structures, or even DNA sequences, providing advanced navigation tools and other interaction capabilities (Haystack [7], mSpace[8]).

Such applications are confronted with the same two issues, independently of the underlying representation paradigm and interface capabilities: selecting what content to show and specifying how to format and style this content. Each application takes its own approach and defines its own vocabulary to specify how to present data to users. As with other kinds of knowledge, we believe that being able to share what we consider *presentation knowledge* makes sense in the context of the Semantic Web and that being able to exchange and reuse presentation knowledge between browsers and other visualization applications will benefit both programmers and end users. However, the current diversity of approaches and vocabularies for representing this knowledge makes such exchange and reuse difficult at best, if not impossible.

### 1.1 Current methods for specifying presentation knowledge

Early RDF visualization tools rendered RDF models in a predefined, non-customizable way [4]. Recent tools provide more flexible visualizations that can be customized by writing style sheets, transformations, or templates, following either a declarative or a procedural approach.

Procedural approaches consider the presentation process as a series of transformation steps. One such approach consists of using XSLT to transform RDF graphs encoded as RDF/XML trees in an environment such as Cocoon [9]. Authoring XSLT templates and XPath expressions to handle arbitrary RDF/XML is complex, if not impossible, considering the many potential serializations of a given RDF graph and the present lack of a commonly accepted RDF canonicalization in XML [10]. This problem has been partly addressed by Xenon [11], an RDF style sheet ontology that builds on the ideas of XSLT but combines a recursive template mechanism with SPARQL as an RDF-specific selector language. Xenon succeeds in addressing XSLT's RDF canonicalization problem but still has a drawback common to all procedural approaches, that transformation rules are tied to a specific display paradigm and output format preventing the reuse of presentation knowledge across applications.

Declarative approaches are based on formatting and styling rules applied to a generic representation of the content. They can be compared to XHTML+CSS, which has been successful for the classic Web. The Haystack Slide ontology [12], used to describe how Haystack display widgets are laid out, is one example. Another is IsaViz's Graph Style Sheets [13], which modifies the formatting, styling, and visibility of RDF graph elements represented as node-link diagrams. The main drawback of the declarative approaches developed so far is that they make strong assumptions about, and are thus tied to, the specific display paradigm for which they have been developed and are therefore unlikely to be meaningful across different representation paradigms.

### 1.2 Toward the specification of presentation knowledge

Providing a single global view of all the information contained in an RDF model is often not useful. The amount of data makes it difficult to extract information relevant to the current task and represents a significant cognitive overload for the user. From an abstract perspective, the first step of the presentation process thus consists in restricting the visualization to small but cohesive parts of the RDF graph, similarly to views in the database world or RMM slices [14]. Users can then select other points of interest by navigating in the model through hyperlinks and refine the selection with paradigms such as faceted browsing (e.g. Longwell [3]). But identifying what content to show is not sufficient for making a human-friendly presentation from the information. To achieve this goal, the selected content items must be laid out properly and rendered with graphical attributes that favor legibility in order to facilitate general understanding of the displayed information. Relying solely on the content's structure and exploiting knowledge contained in the schema associated with the data is insufficient for producing sophisticated visualizations. The second step thus consists in formatting and styling selected content items.

Fresnel's goal is to provide an RDF vocabulary to model information about how to present Semantic Web content to users (i.e., *what* content to show, and *how* to show it) as presentation knowledge that can be exchanged and reused between browsers and other visualization applications. However, we do not expect all applications, which do not necessarily rely on the same representation paradigms and formats, to exchange and reuse all formatting and styling instructions as they might not always be appropriate, depending on the underlying representation paradigm. We thus identified a set of core presentation concepts that are applicable across applications and which form the core modules of Fresnel. On top of these modules, we have also begun to define additional Fresnel vocabulary items which are grouped in extension modules. The remainder of this article mainly focuses on the core selection and formatting modules. More information about extension modules can be found in the Fresnel User Manual [15].

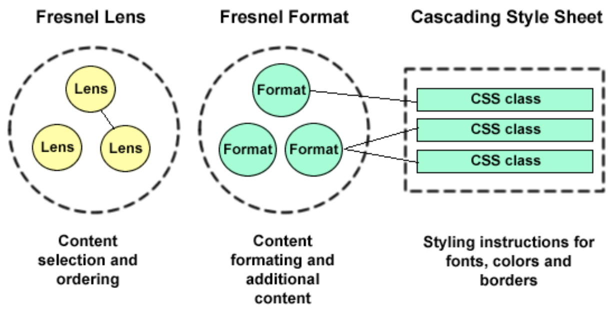

Fresnel Lens
Fresnel Format
Cascading Style Sheet
Lens
Lens
Lens
Format
Format
Format
CSS class
CSS class
CSS class
Content selection and ordering
Content formating and additional content
Styling instructions for fonts, colors and borders

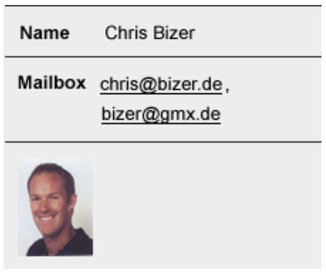
**Fig. 1.** Fresnel foundational concepts

## 2 Fresnel Core Vocabulary Overview

Fresnel is an RDF vocabulary, described by an OWL ontology [15]. Fresnel presentation knowledge is thus expressed declaratively in RDF and relies on two foundational concepts: *lenses* and *formats* (see Figure 1). Lenses specify which properties of RDF resources are shown and how these properties are ordered while formats indicate how to format content selected by lenses and optionally generate additional static content and hooks in the form of CSS class names that can be used to style the output through external CSS style sheets.

Figure 2 shows a simple lens and associated formats used to present information about a person described with the FOAF vocabulary [16]. This figure also shows a hypothetical rendering of such a resource, as an Horus [17] or Longwell-like browser could produce. Examples use the Notation 3 syntax [18].

```
(1)  :PersonLens a fresnel:Lens ;
(2)      fresnel:classLensDomain foaf:Person ;
(3)      fresnel:showProperties (
(4)          foaf:name
(5)          foaf:mbox
(6)          foaf:depiction
(7)      ).
(8)  :nameFormat a fresnel:Format ;
(9)         fresnel:label "Name" ;
(10)        fresnel:propertyFormatDomain foaf:name .
(11) :mboxFormat a fresnel:Format ;
(12)        fresnel:propertyFormatDomain foaf:mbox ;
(13)        fresnel:label "Mailbox" ;
(14)        fresnel:value fresnel:externalLink ;
(15)        fresnel:valueFormat [ fresnel:contentAfter "," ] .
(16) :depictFormat a fresnel:Format ;
(17)        fresnel:propertyFormatDomain foaf:depiction ;
(18)        fresnel:label fresnel:none ;
(19)        fresnel:value fresnel:image .
```



| | |
|---|---|
| **Name** | Chris Bizer |
| **Mailbox** | chris@bizer.de, bizer@gmx.de |

**Fig. 2.** A lens and some formats for presenting instances of class `foaf:Person`

### 2.1 Content selection

The domain of a lens indicates the set of resources to which a lens applies (line 2). Property `showProperties` is used to specify what properties of these resources to show and in what order (lines 3-7). In this example, the values of both `classLensDomain` and `showProperties` are basic selectors, which take the form of plain URIs (represented here as qualified names), respectively identifying the class of resources and property types to select. More complex selection expressions can be written using either FSL or SPARQL (see section 3), making it possible to associate lenses with untyped RDF resources, which do occur in real-world models since `rdf:type` properties are not mandatory. Format domains can also be specified with one of these three solutions.

Fresnel Core provides additional constructs for specifying what properties of resources to display. The special value `fresnel:allProperties` can be used to avoid having to explicitly name each property that should be displayed. This value is also useful when the list of properties that can potentially be associated with resources handled by a lens is unknown to the lens' author but should nevertheless be displayed. When it appears as a member of the list of properties to be shown by a lens, `fresnel:allProperties` designates the set of properties that are not explicitly designated by other property URI references in the list, except for properties that appear in the list of properties to hide (`fresnel:hideProperties`). The extended lens vocabulary defines two other constructs to handle the potential irregularity of RDF data stemming from the fact that different authors might use similar terms coming from different vocabularies to make equivalent statements. Sets of such similar properties can be said to be `fresnel:alternateProperties`. For instance, `foaf:name`, `dc:title`, and `rdfs:label` could be considered by a lens as giving the same information about resources. A browser using this lens would then try to display the resources' `foaf:name`. If the latter did not exist, the browser would look for `dc:title` or `rdfs:label`. The second construct, `fresnel:mergeProperties`, is used to merge the values of related properties (e.g. `foaf:homepage` and `foaf:workHomepage`) into one single set of values that can later be formatted as a whole.

The presentation of property values is not limited to a single level, and recursive calls to lenses can be made to display details about the value of a property. Inserting the following lines in Figure 2 between lines 6 and 7:

```
(6a)    [rdf:type fresnel:PropertyDetails ;
(6b)     fresnel:property "foaf:knows[foaf:Person]"^^fresnel:fslSelector;
(6c)     fresnel:sublens :PersonLabelLens]
```

tells the browser to render values of the property `foaf:knows`, which must be instances of `foaf:Person`, using another lens (`PersonLabelLens`). Infinite loops caused by recursive calls to the same lens are prevented by a closure mechanism.

## 2.2 Content formatting

The representation of selected information items mainly depends on the browser's representation paradigm (e.g. nested box layout, node-link diagrams, etc.) which defines the rendering method. The final rendering can be further customized by associating formatting instructions with elements of the representation.

Formats apply to resources, or to properties and their values, depending on the specified domain. The three example formats of Figure 2 apply respectively to the properties `foaf:name`, `foaf:mbox` and `foaf:depiction` (lines 10,12,17). Formats can be used to set properties' labels (lines 9, 13, 18) and to indicate how to render values. For instance, line 14 indicates that `foaf:mbox` values should be rendered as clickable links (email addresses). Values of `foaf:depiction` should be fetched from the Web and rendered as bitmaps (line 19). Fresnel also defines default display methods in case such indications are not given by a format.

Property values can be grouped, and additional content such as commas (line 15) and an ending period can be specified to present multi-valued properties. CSS class names can also be associated with the various elements being formatted. These names appear in the output document and can be used to style the output by authoring and referencing CSS style sheets that use rules with the same class names as selectors.

## 3 Fresnel Selectors

Selection in Fresnel occurs when specifying the domain of a lens or format and when specifying what properties of a resource a lens should show. Such selection expressions identify elements of the RDF model to be presented; in other words, specific nodes and arcs in the graph. As we expect selection conditions to be of varying complexity, we allow them to be expressed using three different languages in an attempt to balance expressive power against ease of use.

The simplest selectors, called basic selectors, take the form of plain URI references as shown in section 2. Basic selectors simply name the type of resources or properties that should be selected. They are easy to use but have very limited expressive power. For instance, they cannot be used to specify that a lens should apply to all instances of class `foaf:Person` that are the subject of at least five `foaf:knows` statements. More powerful languages are required to express such selection constraints.

The Fresnel Selector Language (FSL) is a language for modeling traversal paths in RDF graphs, designed to address the specific requirements of a selector language for Fresnel. It does not pretend to be a full so-called RDFPath language (contrary to XPR [19], an extension of FSL) but tries to be as simple as possible, both from usability and implementation perspectives. FSL is strongly inspired by XPath, reusing many of its concepts and syntactic constructs while adapting them to RDF's graph-based data model. RDF models are considered directed labeled graphs according to RDF Concepts and Abstract Syntax [20]. FSL is therefore fully independent from any serialization.

An FSL expression represents a path from a node or arc to another node or arc, passing by an arbitrary number of other nodes and arcs. FSL paths explicitly represent both nodes and arcs as steps on the path, as it is desirable to be able to constrain the type of arcs a path should traverse (something that is not relevant in XPath as the only relation between the nodes of an XML tree is the parent-child relation which bears no explicit semantics).

A full description of FSL is outside the scope of this paper and will be the subject of future publications. In the meantime, more information about the language, including its grammar, data model and semantics is available in the FSL specification [21]. A lens definition using two FSL expressions follows:

```
# A lens for foaf:Person resources that know at least five other resources
:PersonLens a fresnel:Lens ;
       fresnel:instanceLensDomain "foaf:Person[count(foaf:knows) >= 5]" ;
# and which shows the foaf:name property of all foaf:Person
# instances known by the current resource.
       fresnel:showProperties ("foaf:knows/foaf:Person/foaf:name") .
```

The SPARQL RDF query language [22] is the last alternative. SPARQL queries must always return exactly one result set, meaning that only one variable is allowed in the query's SELECT clause.

```
# A lens for John Doe's mailboxes
:PersonLens a fresnel:Lens ;
fresnel:instanceLensDomain "SELECT ?mbox WHERE ( ?x foaf:name 'John Doe' )
                                                ( ?x foaf:mbox ?mbox )" .
```

FSL was designed as a compact and simpler alternative to SPARQL, but the two languages will probably show a significant overlap in expressive power. For instance, the above SPARQL lens domain could have been written in FSL as `*[in::foaf:mbox/*[foaf:name/text() = 'John Doe']]`. In such cases, one language might be more appropriate than the other with respect to the conciseness and legibility of expressions, and the choice of what language to use to write each selection expression should be left to Fresnel users.

Applications implementing Fresnel are required to support basic selectors, and we expect a reasonable share of them to support the two other languages: SPARQL is likely to gain momentum quickly as a W3C recommendation, and two open-source Java implementations of FSL, for IsaViz and HP's Jena, are already available[4], with a third being written for the Sesame RDF database (http://openrdf.org).

## 4 Conclusion

We have given an overview of Fresnel, a browser-independent, extensible vocabulary for modeling Semantic Web content presentation knowledge. Fresnel has been designed as a modularized, declarative language manipulating selection, formatting, and styling concepts that are applicable across representation paradigms, layout methods, and output formats. Fresnel core modules can be used to model presentation knowledge that is compatible and reusable between browsers and other types of Semantic Web information visualization tools.

Although core modules have been frozen for the time being, the Fresnel vocabulary remains a work in progress as new extension modules meeting special needs are being developed (e.g., for describing the *purpose* of lenses or adding new formatting capabilities). Extension modules are not necessarily aimed at being application- and paradigm-independent, as they might not be relevant in all cases; but their inclusion in Fresnel provides users with a unified framework for modeling presentation knowledge.

Core modules are currently being implemented in various types of applications: SIMILE's Longwell [3] faceted browser, IsaViz [1] which represents RDF graphs as node-link diagrams, and Horus [17], a PHP-based RDF browser. More information about Fresnel can be found on its web site[5]. Its development is an open, community-based effort and new contributors are welcome to participate in it.

[4] http://dev.w3.org/cvsweb/java/classes/org/w3c/IsaViz/fresnel/
[5] http://www.w3.org/2005/04/fresnel-info/

## Acknowledgments

We would like to thank members of the SIMILE and Haystack projects at MIT for their valuable input, especially Stefano Mazzocchi, David Karger, Stephen Garland, David Huynh, Karun Bakshi, and the others who contributed to the discussion about Fresnel: Hannes Gassert, Jacco van Ossenbruggen, Dennis Quan, and Lloyd Rutledge.

## References

1. W3C: IsaViz: A Visual Authoring Tool for RDF (2001-2005) http://www.w3.org/2001/11/IsaViz/.
2. SIMILE: Welkin (2004-2005) http://simile.mit.edu/welkin/.
3. SIMILE: Longwell RDF Browser (2003-2005) http://simile.mit.edu/longwell/.
4. Steer, D.: BrownSauce: An RDF Browser. XML.com (2003) http://www.xml.com/pub/a/2003/02/05/brownsauce.html.
5. Rutledge, L., van Ossenbruggen, J., Hardman, L.: Making RDF Presentable: Selection, Structure and Surfability for the Semantic Web. In: Proceedings of the 14th international conference on World Wide Web. (2005)
6. Kalyanpur, A., Parsia, B., Hendler, J.: A Tool for Working with Web Ontologies. In: Proceedings of Extreme Markup Languages. (2004)
7. Quan, D., Karger, D.: How to Make a Semantic Web Browser. In: Proceedings of the 13th international conference on World Wide Web. (2004) 255–265
8. mc schraefel, Smith, D., Owens, A., Russell, A., Harris, C.: The evolving mSpace platform: leveraging the Semantic Web on the Trail of the Memex. In: 16th ACM Conference on Hypertext and Hypermedia. (2005)
9. ASF: The Apache Cocoon Project (2005) http://cocoon.apache.org.
10. Carroll, J.J., Stickler, P.: Trix : Rdf triples in Xml. In: In the International Journal on Semantic Web and Information Systems, Vol.1, No.1, Jan-Mar 2005. (2005)
11. Quan, D., Karger, D.: Xenon: An RDF Stylesheet Ontology (2005) Submitted to the World Wide Web Conference.
12. Huynh, D.: Haystack's user interface framework: Tutorial and reference (2003) http://haystack.lcs.mit.edu/documentation/ui.pdf.
13. Pietriga, E.: Styling RDF Graphs with GSS. XML.com (2003) http://www.xml.com/pub/a/2003/12/03/gss.html.
14. Isakowitz, T., Stohr, E.A., Balasubramanian, P.: RMM: A Methodology for Structured Hypermedia Design. Communications of the ACM **38** (1995) 34–44
15. Bizer, C., Lee, R., Pietriga, E.: Fresnel - Display Vocabulary for RDF (2005) http://www.w3.org/2005/04/fresnel-info/manual-20050726/.
16. FOAFers: Friend-of-a-Friend (FOAF) (2001) http://www.foaf-project.org/.
17. Erdmann, T.I.: Horus RDF Browser (2005) http://www.wiwiss.fu-berlin.de/suhl/bizer/rdfapi/tutorial/horus/.
18. Berners-Lee, T.: Primer: Getting into RDF & Semantic Web using N3 (2005) http://www.w3.org/2000/10/swap/Primer.html.
19. Cohen-Boulakia, S., Froidevaux, C., Pietriga, E.: Selecting Biological Data Sources and Tools with XPR, a Path Language for RDF. In: Pacific Symposium on Biocomputing (PSB), Maui, Hawaii. (2006)
20. W3C: Resource description framework (RDF): Concepts and Abstract Syntax (2004) http://www.w3.org/TR/rdf-concepts/.
21. Pietriga, E.: Fresnel Selector Language for RDF (2005) http://www.w3.org/2005/04/fresnel-info/fsl-20050726/.
22. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF (2005) http://www.w3.org/TR/rdf-sparql-query/.