The Function Hub: an implementation-independent read/write function description repository

Ben De Meester[0000−0003−0248−0987], Lander Noterman,
Ruben Verborgh[0000−0002−8596−222X], and Anastasia Dimou[0000−0003−2138−7972]

Ghent University – imec – IDLab,
Department of Electronics and Information Systems,
Technologiepark-Zwijnaarde 122, 9052 Ghent, Belgium
{firstname.lastname}@ugent.be

Abstract. Functions are essential building blocks of any (computer) information system. However, development efforts to implement these functions are fragmented: a function has multiple implementations, each within a specific development context. Manual effort is needed handling various search interfaces and access methods to find the desired function, its metadata (if any), and associated implementations. This laborious process inhibits discovery, and thus reuse. Uniform, implementation-independent access is needed. We demo the Function Hub, available online at `https://fno.io/hub`: a Web application using a semantic interoperable model to map function descriptions to (multiple) implementations. The Function Hub allows editing and discovering function description metadata, and add information about alternative implementations. This way, the Function Hub enables users to discover relevant functions independently of their implementation, and to link to original published implementations.

Keywords: Function · Linked Data · Repository

1 Introduction

Functions are processes that perform a specific task by associating one or more inputs to an output. However, the development, maintenance, and support efforts of *implementing* these functions are fragmented [1], across different *development contexts* (i.e., a combination of programming language, programming paradigm, architecture, etc.). It is unfeasible to consolidate these efforts by limiting all developers to the same development context [7]. On the one hand, implementations are tuned to meet different requirements, and, on the other hand, there might be prior investment [1]. The same function can thus have multiple implementations, each within a specific development context. For example, a function to normalize dates may have implementations available as JavaScript source code on GitHub, as part of a Java software package on Maven, and within a REST Web API.

As such, the exploration of function implementations is fragmented across different search engines. To find the desired implementation, users need a combination of, among others, general search engines, online code repositories, and package managers. To know how to invoke an implementation, additional effort is typically needed, e.g., going through the implementation's documentation. Such manual effort involves handling different search interfaces and access methods.

This laborious process inhibits discovery and reuse. Existing implementations are not found because they are built in a different development context. Hence, the same functions are (sometimes unnecessarily) re-implemented, and the fragmentation of development efforts increases. Uniform, implementation-independent access is needed to improve the discovery process.

In this demo paper, we present the Function Hub, which is available online at `https://fno.io/hub`. The Function Hub enables editing and exploring function description metadata, and linking them to implementations across development contexts. Users can look for a function, optionally using advanced filtering on input and output types. The result is a list of links to existing implementations in various repositories and across development contexts, and machine-readable descriptions of how to invoke them. Access is uniform and descriptions are linked, with the potential to improve implementation reuse. Using a common semantic model, the Function Hub allows interoperability of the function description metadata, and allows agents to unambiguously infer how to use the implementation.

2 The Function Hub

We now discuss several aspects of the Function Hub: the Function Ontology as semantic model to describe the metadata, the characteristics of the Function Hub application, and the read and write interfaces.

Function Ontology We use the Function Ontology (FnO) [3,4] to represent implementation-independent function description metadata in RDF, and link this metadata to existing and new implementations of varying development contexts. FnO distinguishes between the (*abstract*) function and the (*concrete*) implementation, similarly observed by Garijo et al. for workflows [5].

Functions, implementations, and the mappings between them are described separately. *Functions* are represented as a transformation of input data into output data. By providing an abstract description of a function, instead of a new (declarative) programming language to specify all imperative steps, existing implementations can be linked, they do not need to be re-coded in a new language. By describing the *mappings* separately, the same function can be linked to multiple *implementations*. For example: a date normalization function is linked to a Maven package, a JavaScript snippet, and a REST Web API.

Web application The Function Hub allows editing and exploring all relevant metadata with respect to functions, implementations, and mappings. All metadata is stored and published as Linked Data, exhibiting following advantages [6]:

(i) linking resources is a native feature of Linked Data; (ii) interoperability is achieved by design, using standardized HTTP operations and formats; and (iii) already available (Web) resources can be linked, for instance, by referring to existing implementations in package managers using their published URI.

The Function Hub is a Web application, responsible for managing and publishing the function descriptions and their implementation mappings. A (public) SPARQL endpoint of the data store is provided (`https://fno.io/hub/data/sparql`), together with developer-friendly (JSON) APIs (`https://fno.io/hub/api`), and a user interface (`https://fno.io/hub`). Its user interface is two-fold, enabling to read and write function description metadata.

Read interface The Function Hub's user interface helps users explore (i.e., browse and search) function descriptions without needing to execute any queries on the public querying endpoint. When *browsing*, users can navigate the different function descriptions, and their implementation mappings, making it easy to discover the available implementations of a function, and what type of parameters are used. For every function, the Function Hub shows the name, description, parameters, outputs, and a list of implementations. The declarative mappings unambiguously describe how to invoke the found implementation. Each resource is a link that can be resolved in the browser for more information.

When *searching*, users can perform *common keyword search* over function names and descriptions, and *advanced search* which allows filtering on specific semantic constraints for parameters and outputs, e.g., by parameter or return type (Fig. 1). This allows more detailed search, returning more accurate results. The results are displayed as a list of functions, which can be further inspected via the browse functionality.

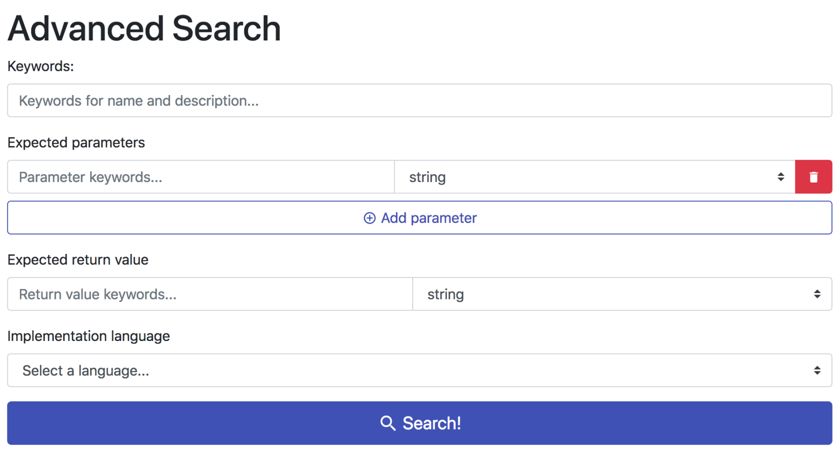
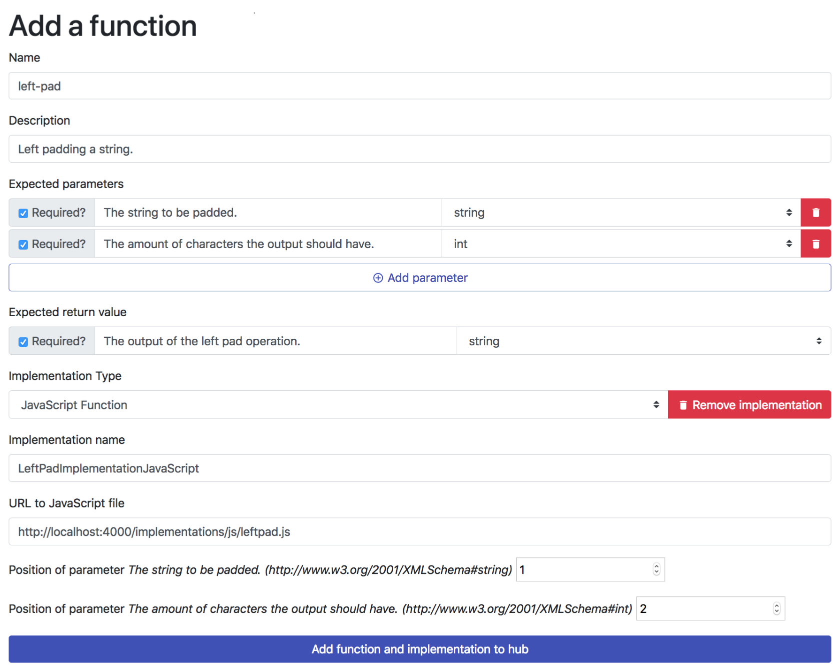


Fig. 1: Detailed, unambiguous search of functions (across implementations).



Name: left-pad

Description: Left padding a string.

Expected parameters

✓ Required?	The string to be padded.	string
✓ Required?	The amount of characters the output should have.	int

Expected return value

✓ Required?	The output of the left pad operation.	string

Implementation Type: JavaScript Function

Implementation name: LeftPadImplementationJavaScript

URL to JavaScript file: http://localhost:4000/implementations/js/leftpad.js

Position of parameter *The string to be padded. (http://www.w3.org/2001/XMLSchema#string)* 1

Position of parameter *The amount of characters the output should have. (http://www.w3.org/2001/XMLSchema#int)* 2

Fig. 2: Editing a function description metadata, and adding/editing links to implementations.

Write interface The editing environment supports creating new and editing existing function descriptions (Fig. 2). More specifically, all metadata of the function and its associated resources (inputs, outputs, problems) can be edited, and mappings with implementations can be added. Updates are directly reflected in the queryable dataset.

3 Conclusion

This demo shows how abstract function descriptions, with mappings to concrete implementations as Linked Data resources on the Web, are made available by the Function Hub. A uniform search interface is provided, usable across development contexts with more advanced search operators.

The Function Hub is interoperable and scalable. W3C recommendations are used whenever possible to improve *interoperability* with other systems, whether it be for the format (RDF, JSON) or the query interface (SPARQL). The Function Hub provides merely links to implementations: the amount of stored data is limited to metadata, leading to better *scalability* with respect to the amount of available descriptions, and allows for federated querying.

The Function Hub can be used as a development context-independent search engine that returns sufficient metadata to manually integrate an implementation of a requested function. The Function Hub can, thus, already be used in existing software projects, and its usage can be compared to typical search. However, the detailed mapping description also allows for automatic invocation of these found implementations. This future work is being pursued by implementing Function Handlers for different development contexts[1], further specified in [4].

The Function Hub's data is currently manually curated. To scale up, automatic metadata extraction approaches can be used. We can use existing frameworks to automatically generate meaningful function descriptions, based on data of package managers, such as npm [8], and based on programming code source files in Java code [2]. Further, additional non-functional metadata can also be taken into account to, e.g., represent a notion of expected quality (similar to showing the amount of stars and forks within a GitHub repository). Once this metadata is ingested, the Function Hub represents – and allows to search for – the contents of these software packages in a uniform, development context-independent way.

Acknowledgements The described research activities were funded by Ghent University, imec, Flanders Innovation & Entrepreneurship (VLAIO), and the European Union. Ruben Verborgh is a postdoctoral fellow of the Research Foundation – Flanders (FWO).

References

1. Atkinson, M., Gesing, S., Montagnat, J., Taylor, I.: Scientific workflows: Past, present and future. Future Generation Computer Systems **75**, 216–227 (Oct 2017)
2. Atzeni, M., Atzori, M.: CodeOntology: RDF-ization of Source Code. In: The Semantic Web – ISWC 2017. vol. 10588, pp. 20–28. Springer, Cham (2017)
3. De Meester, B., Dimou, A.: The Function Ontology. Unofficial Draft, Ghent University – imec – IDLab (2016), `https://w3id.org/function/spec`
4. De Meester, B., Dimou, A., Verborgh, R.: Implementation-independent Function Reuse. Future Generation Computer Systems (2019), under review
5. Garijo, D., Gil, Y.: A New Approach for Publishing Workflows: Abstractions, Standards, and Linked Data . In: Proceedings of the 6th workshop on Workflows in support of large-scale science - WORKS '11. ACM Press (2011)
6. Garijo, D., Gil, Y., Corcho, O.: Abstract, link, publish, exploit: An end to end framework for workflow sharing. Future Generation Computer Systems **75**, 271–283 (Oct 2017)
7. Liew, C.S., Atkinson, M.P., Galea, M., Ang, T.F., Martin, P., Hemert, J.I.V.: Scientific workflows: moving across paradigms. ACM Computing Surveys **49**(4), 66:1–66:39 (Dec 2017)
8. Van Herwegen, J., Taelman, R., Capadisli, S., Verborgh, R.: Describing configurations of software experiments as linked data. In: Garijo, D., van Hage, W.R., Kauppinen, T., Kuhn, T., Zhao, J. (eds.) Proceedings of the First Workshop on Enabling Open Semantic Science (SemSci). pp. 23–30. No. 1931 in CEUR Workshop Proceedings, Aachen (2017)

[1] `https://github.com/FnOio/function-handler-js` and `https://github.com/FnOio/function-handler-java` are Function Handler proof-of-concept implementations for JavaScript and Java, respectively.