“State of RDF Javascript Libraries”

Software Requirements Specification

Version: 1.0
Date: 13.05.2016

Team:

Mehrdad Bozorg
Mohammad Tahaei
Todor Tsankov

Mentors:

Vinay Modi
Afshin Sadeghi

Table of Contents

1. Introduction

1.1 Purpose

This document brings an overview of the requirements specification for project “State of RDF JavaScript libraries”. Which is intended to facilitate the usage of RDF JS libraries, by investigating specific features and in this way providing the user with useful comparison between libraries to choose based on the demands of the project.

The final tool is to be used by a wide variety of users, starting from a developer to an enterprise willing to find the suitable solution based on the selected features of available RDF JS libraries.

1.2 System overview

Day to day growing usage of RDF requires projects which are developed to aim the lack of a flexible comparison and evaluation of RDF JS libraries. At the moment there is no confidential source and system to find the best suitable solution taking into account a set of specified features. Therefore, it`s necessary to define and develop a system to provide a well-defined comparison between RDF JS libraries.

1.3 Definitions, Acronyms and Abbreviations.

RDF	-	Resource description framework
JS	-	Javascript
GUI	-	Graphical user interface

2. Overall description

2.1 Product perspective

For securing users with a comparison system that targets the specific feature which are tested on a set of related libraries, using different tests cases with a wide amount of datasets in aspects of size and scope.
The applicability and usability of such a system is another important issue that should be targeted during the development process.

2.2 Product functions

The application of presenting RDF JS libraries, brings the list of available JS libraries which is linked to well-defined documentation of them. The user can browse through all libraries as well as see detailed information on a single library page. Based on a set of selected features from the search section, the application filters the libraries and displays the matching ones as a result. Figure 1 represents the Use Case diagram.

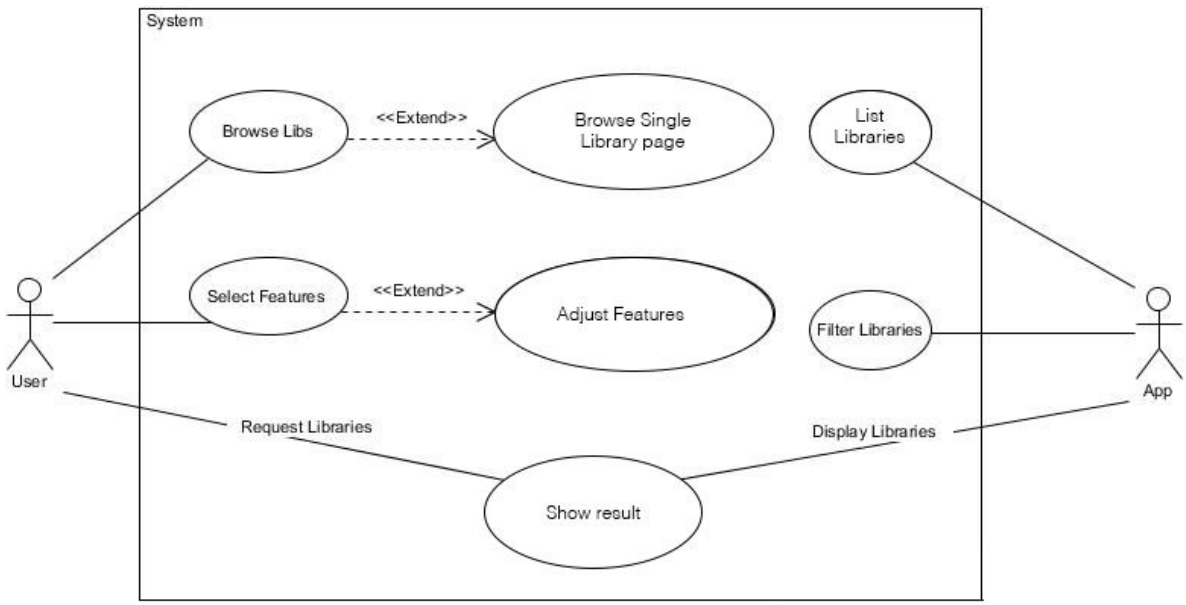

Figure 1. Use Case diagram

2.3 User characteristics

Target user of this system is a scholar in academia or a developer, interested in using a JavaScript library in the field related with RDF. The user shall posses basic knowledge and understanding of RDF and Semantic Web.

2.4 Constraints and assumptions

The development of such a system requires well-defined data sets with different scales as well as a variety of scopes. In addition defining the set features which filtering is based on is a high priority task this project.
The final web application requires active JavaScript in the internet browser.

2.5 Test and evaluation rules

- The benchmark must run on standard hardware and should run released software, in order to make it possible to replicate results.
- Generic configurations for each test case
- Different size and complexity of data sets
- The results of the test should be reported as a document

3 System features

Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases [https://en.wikipedia.org/wiki/Functional_requirement].

Tag	Feature	Priority
RDFJS-FR-1	Listing	1
RDFJS-FR-2	Filtering	1
RDFJS-FR-3	Testing	2

3.1 Listing

The first functionality of the system will be to list all the available RDF JS libraries. The user will be able to see the current state and general features of all the available RDF JS libraries. Listing will be based on metrics from categorisations listed below.

W3c Comparison

[https://www.w3.org/community/rdfjs/wiki/Comparison_of_RDFJS_libraries]:

3.1.1 Parsing libraries

List any libraries capable of parsing triples/statements out of another format:

- Media types: RDF/JSON, JSON-LD, N3, Turtle, RDF/XML, ...
- Environment: Browser, Node.js, AMD, CommonJS, ...
- Interfaces (which APIs/interfaces does it implement): RDF Interfaces, Streams API, Node.js Streams, ...

3.1.2 SPARQL/Query libraries

- Query language (SPARQL/1.0, SPARQL/1.1, SPARUL, SPIN)
- Query API

3.1.3 Data storage libraries

- Storage method (memory, disk-backed, disk-indexed)
- I/O style (in-line, synchronous, asynchronous)
- Referenced standards (RDF Interfaces)

3.1.4 UI data binding libraries

- Integrations with other libraries and frameworks
- Data binding approaches
- Referenced standards (RDF Interfaces)

3.1.5 Reasoners

- Vocabulary (RDFS, OWL-Lite, OWL-DL, OWL-Full, OWL2-EL, OWL2-QL, OWL2-RL, etc)
- Reasoner type (Forward-chaining, backward-chaining)

Berlin SPARQL Benchmark

[http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/]

3.1.6 Metrics for Single Queries

- Average Query Execution Time (aQET)
- Queries per Second (QpS)
- Min/Max Query Execution Time (minQET, maxQET)

3.1.7 Metrics for Query Mixes

- Queries Mixes per Hour (QMpH)
- Overall Runtime (oaRT)
- Composite Query Execution Time (cQET)
- Average Query Execution Time over all Queries (aQEToA)

3.1.8 Price/Performance Metric for the Complete System under Test

- The Price/Performance Metric defined as $ / QMpH.

3.2 Filtering

User would be able to set and adjust the features in the search list for a specific library. This requirement will deal with user preferences, the user will be able to search or filter the current list and apply his preferences.
Main filtering features will be as follows (W3):

- Name
- URL
- Source
- Documentation
- License
- First Release
- Latest Stable Release
- Platform support (Browser-side, Node.js, AMD, etc)

Additionally, worth mentioning:

- Library size
- Community size
- Covered test cases
- Query time

3.3 Testing

The system will provide information regarding the libraries default information, additionally some testing and evaluation data would be presented. This enables the user to have an insight into the RDF JS libraries, and also assures a suitable choice among RDF JS libs.

4 Non-functional requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions [https://en.wikipedia.org/wiki/Non-functional_requirement].

Tag	Feature	Priority
RDFJS-NFR-1	Performance	1
RDFJS-NFR-2	Usability	1
RDFJS-NFR-3	Documentation	2

4.1 Performance Requirements

System will be responsive in a reasonable time. The user will have interactive communication with the system. Solid experience relies on a decent internet connection and latest internet browser.

4.2 Usability Requirements

The main focus of the system is web browsers in normal screen sizes, and also the system will be able to be viewed in mobile devices as a user friendly GUI.

4.3 Documentation Requirements

A well documented user guide will help the users to get the most out of the system. Documentation would be available in paper and online forms.