BIOINFORMATICS APPLICATIONS NOTE Vol. 29 no. 8 2013, pages 1103–1104
doi:10.1093/bioinformatics/btt100

Bioimage informatics

Advance Access publication February 23, 2013

BioJS: an open source JavaScript framework for biological data visualization

John Gómez[1], Leyla J. García[1], Gustavo A. Salazar[2], Jose Villaveces[3], Swanand Gore[1], Alexander García[4], Maria J. Martín[1], Guillaume Launay[5], Rafael Alcántara[1], Noemi del-Toro[1], Marine Dumousseau[1], Sandra Orchard[1], Sameer Velankar[1], Henning Hermjakob[1], Chenggong Zong[6], Peipei Ping[6], Manuel Corpas[7] and Rafael C. Jiménez[1,*]

[1]European Bioinformatics Institute, Hinxton, Cambridge, CB10 1SD, UK, [2]Computational Biology Group, Department of Clinical Laboratory Sciences, University of Cape Town, Cape Town, 7925, South Africa, [3]Max Planck Institute for Biology of Ageing, Cologne, D-50931, Germany, [4]Institute for Digital Information and Scientific Communication, Florida State University, Tallahassee, FL, 32306-2100, USA, [5]Institut de Biologie et Chimie des Protéines, UMR 5086 CNRS-Université Lyon 1, IFR 128 Biosciences Gerland-Lyon Sud, 7 passage du Vercors 69367, Lyon Cedex 07, France, [6]Cardiovascular Research Laboratory, Division of Cardiology, Departments of Physiology and Medicine, UCLA David Geffen School of Medicine, Los Angeles, CA, 90095-1760, USA and [7]The Genome Analysis Centre, Norwich Research Park, Norwich, NR4 7UH, UK

Associate Editor: Jonathan Wren

ABSTRACT

Summary: BioJS is an open-source project whose main objective is the visualization of biological data in JavaScript. BioJS provides an easy-to-use consistent framework for bioinformatics application programmers. It follows a community-driven standard specification that includes a collection of components purposely designed to require a very simple configuration and installation. In addition to the programming framework, BioJS provides a centralized repository of components available for reutilization by the bioinformatics community.

Availability and implementation: http://code.google.com/p/biojs/.

Contact: rafael@ebi.ac.uk

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on October 25, 2012; revised on February 14, 2013; accepted on February 20, 2013

1 INTRODUCTION

Numerous web applications exist for the visualization and integration of biological data. Biological data can be complex and heterogeneous, making it challenging to integrate and visualize results in web applications. JavaScript is a popular language for developing Rich Internet Applications (RIA); most RIA rely on JavaScript libraries for deployment of functionality. Despite its widespread use in bioinformatics, JavaScript applications are usually implemented to satisfy a particular utility, ignoring its potential for reutilization by other applications with similar purposes.

Here, we present BioJS, an open source community-driven JavaScript framework of reusable graphical components. BioJS offers a set of standard guidelines on how to develop components for JavaScript modules in the Life Sciences. These guidelines make installation and addition of new code the same for all new components, minimizing effort and encouraging good programming practice.

BioJS provides a catalogue of open source modules in JavaScript for Life Sciences. These modules include many commonly used functionalities, available there for developers or scientists to download. BioJS embraces a similar philosophy to previously successful biologically oriented open source projects, such as BioPerl (Stajich *et al.*, 2002), BioJava (Prlić *et al.*, 2012), BioPython (Cock *et al.*, 2009) or BioRuby (Goto *et al.*, 2010), advocating that any functional component or module should be developed only once and reused whenever possible. Our approach offers a standard and scalable set of specifications to produce custom applications; it also facilitates the integration of components into more comprehensive web applications. To our knowledge, no community efforts have so far achieved a coordinated development of JavaScript applications for the visual representation of biological data.

2 BIOJS COMPONENTS

At the time of writing, there are 29 components available and 18 registered contributors to the project. Table 1 lists a few example components considered important and representative. BioJS components follow a standard specification facilitating their integration in third party web applications. The BioJS architecture defines common events that make components capable of communicating and exchanging data (Fig. 1). The project is serviced by a registry that enables users to interact and discover existing components. The registry periodically delivers documentation, providing search functionalities for component discovery. When a component is accessed from the registry (Fig. 2), its description includes a functional example, a sample code of how to install it and documentation on how to use dependencies, methods and events. The development of a BioJS component might initially take longer than developing the same module using an

*To whom correspondence should be addressed.

Table 1. A selection of BioJS components and their functionalities

Component	Functionality
Das Protein Feature Viewer	Uses a web service to retrieve and display protein features from DAS sources
Gene Expression Summary	Queries Gene Expression Atlas database to show published studies where a gene is over/under-expressed compared with its overall mean expression levels
Interactions Table	Shows binary molecular interactions in table format
Protein3D	Renders a Protein Data Bank (PDB) file in 3D using Jmol (http://jmol.sourceforge.net/)
Protein Portfolio	Shows protein description and PDB alignments if available.
Chromosome	Visualizes a chromosome and its bands, which can be recovered from a DAS source or a JavaScript model

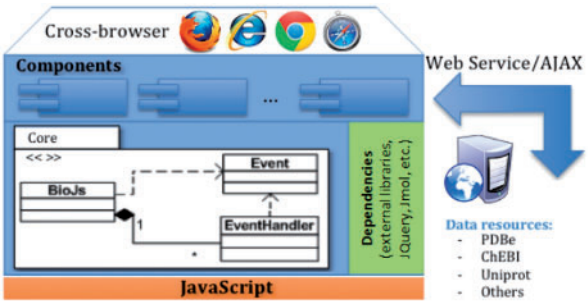

Fig. 1. Architectural representation of a typical BioJS application. BioJS core includes event and event handler classes

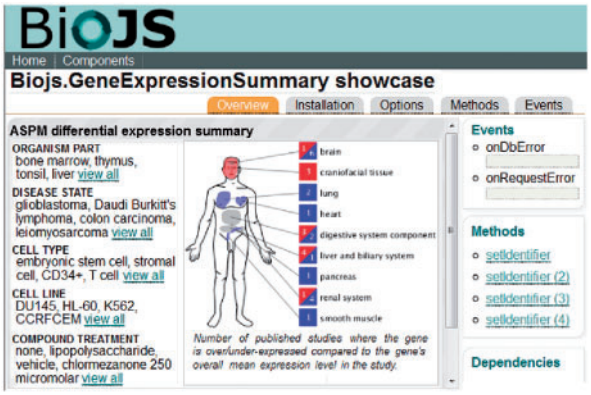

Fig. 2. Illustration of the BioJS registry displaying the GeneExpressionSummary component (http://goo.gl/pwtv3)

independent JavaScript application, but this is only for as long as the user familiarizes him/herself with the framework.

BioJS is framework agnostic, only requiring the code to be written in JavaScript. The developer of a new component is thus free to use any framework (e.g. JQuery, http://jquery.com/; YUI, http://yuilibrary.com/) and to include any other library (Raphael, http://raphaeljs.com/; D3, http://d3js.org/). The look and feel of a component can be easily configured using Cascade Style Sheet (CSS) to enable its integration via different web applications. Any component by definition extends the BioJS reference implementation incorporating the rules provided by the BioJS specification: (i) the component architecture, (ii) a protocol to handle events that allows communication between components, (iii) the component extension through Object-Oriented Inheritance, (iv) the code documentation format and (v) documentation on how to include examples to test the component functionality. We are aware of several bioinformatics projects already using BioJS to visualize data. For instance, the CATH (Cuff *et al.*, 2011) and the IntAct (Kerrien *et al.*, 2012) databases both use the Protein3D viewer, an example of the reutilization of an existing component. The Biotea Project (García-Castro *et al.*, 2012) reuses and extends BioJS components to visualize the biological data of PubMed Central articles.

3 CONCLUSIONS

BioJS provides a framework for development and sharing of graphical components, implemented in JavaScript. Availability of a central registry and a common set of guidelines facilitates component discovery while reducing development time. We have presented here some of BioJS's visualization capabilities for several biological applications, making use of existing components. We anticipate BioJS's available functionality to grow steadily in the near future. Our project will be successful in the measure in which it reduces overall development time for new web applications requiring visualization of biological data.

ACKNOWLEDGEMENTS

We are grateful to Lee Katz for his early contributions to this project.

Funding: NHLBI Proteomics Center Award (HHSN268201000035C); the UK's Biotechnology and Biological Sciences Research Council (BBSRC); European Commission grant PSIMEx (FP7-HEALTH-2007-223411).

Conflict of Interest: none declared.

REFERENCES

Cock,P.J. *et al.* (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, **25**, 1422–1423.

Cuff,A.L. *et al.* (2011) Extending CATH: increasing coverage of the protein structure universe and linking structure with function. *Nucleic Acids Res.*, **39**, D420–D426.

García-Castro,L.J. *et al.* (2012) *Conceptual Exploration of Documents and Digital Libraries in the Biomedical Domain*. International Workshop on Semantic Web Applications and Tools for Life Sciences. CEUR-WS.org., Paris, France.

Goto,N. *et al.* (2010) BioRuby: bioinformatics software for the Ruby programming language. *Bioinformatics*, **26**, 2617–2619.

Kerrien,S. *et al.* (2012) The IntAct molecular interaction database in 2012. *Nucleic Acids Res.*, **40**, D841–D846.

Prlić,A. *et al.* (2012) BioJava: an open-source framework for bioinformatics in 2012. *Bioinformatics*, **28**, 2693–2695.

Stajich,J.E. *et al.* (2002) The Bioperl toolkit: Perl modules for the life sciences. *Genome Res.*, **12**, 1611–1618.