Published in final edited form as:

Trends Biotechnol. 2018 May ; 36(5): 473–476. doi:10.1016/j.tibtech.2017.11.009.

Augmenting research, education and outreach with client-side web programming

Luciano A. Abriata^{1,*}, João P.G.L.M. Rodrigues², Marcel Salathé³, and Luc Patiny⁴

¹Laboratory for Biomolecular Modeling, School of Life Sciences, École Polytechnique Fédérale de Lausanne and Swiss Institute of Bioinformatics, CH-1015 Lausanne, Switzerland ²Department of Structural Biology, Stanford University School of Medicine, Stanford, CA 94305, USA ³School of Life Sciences, School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland ⁴Institute of Chemical Sciences and Engineering, Swiss Federal Institute of Technology Lausanne (EPFL), Lausanne CH-1015, Switzerland

Abstract

The evolution of computing and web technologies over the last decade enabled the development of fully-fledged scientific applications that run directly on web browsers. Powered by JavaScript, the *lingua franca* of web programming, these “web apps” are starting to revolutionize and democratize scientific research, education and outreach.

Keywords

JavaScript; Visualization; Bioinformatics; Chemoinformatics; Open Science

It is hard to imagine modern science without the Internet, databases, web services, and high performance computing [1]. While much of this cyberinfrastructure relies on remote servers, client-side web applications, so-called “web apps”, are becoming increasingly common for querying, browsing, visualizing and analyzing scientific data. Web apps, the focus of this article, are built using web technologies around the JavaScript programming language. They are hosted at online servers and delivered as standard web pages, which confers them several advantages over traditional offline software (Box 1 and Figure S1). Whereas a recent article reviewed the advantages and applications of these technologies for science[2], here we focus on their potential to democratize science, provide technological support for open data policies, and improve user experience in resources for science education and outreach. A comprehensive example illustrating these ideas is the web app introduced for the latest Critical Assessment of Structure Prediction (a worldwide competition on protein structure prediction). This web app enabled the assessors to publish, for the first time in the history of the competition, the entire assessment results such that any reader can navigate all submitted models through multiple quality metrics and judge them by comparison to target structures. Importantly, this is done interactively, in 3D, directly from a web browser[3].

*Correspondence: luciano.abriata@epfl.ch.

The evolution of the scientific web

The early 2000s marked a paradigm shift in web design, moving from static content users could just browse to dynamic content they could interact with. Initial efforts to display interactive content relied on browser plugins, third-party software installed onto web browsers to extend their capabilities. Plugins posed security and usability issues, but the scientific community adopted them broadly because they allowed for much more engaging contents [4]. Over time, however, plugins lost support as JavaScript, a programming language created specifically for the web and built into every web browser, started being used to build interactive pages. Although initially of limited power, improvements in performance, direct coupling to HTML and other web technologies, and the establishment of industry-wide standards that ensure compatibility across browsers and devices, led to a burst in the number of JavaScript-based scientific web apps.

Web apps for scientific applications

JavaScript now allows the development of web apps with functionalities similar to those of standard computer programs, but delivered through web pages (Box 1 and [2]). There is a growing community creating JavaScript tools to process and display data in most fields of science. Contributions by and for the biological scientific community include fully-featured libraries and applications for structural biology, genomics and chemoinformatics, besides applications of machine learning in data classification and modeling, and image analysis algorithms (Box 2 and [2]).

Among the several aspects highlighted in Box 1, three are worth highlighting. First, most libraries are under permissive open-source licenses and code is inherently available to the user. Second, code is largely reusable; libraries can be combined to power fully-featured apps involving visualization and calculation, rivaling traditional offline software. Third, since web apps are just web pages, there is no technical barrier to installing, updating, or using them on a number of devices, from desktop computers to smartphones.

An important aspect to consider, especially relevant regarding data analysis and simulations, is the execution speed of web apps. Performance is getting closer to that of classical programs installed on local drives, thanks to technologies that enable porting computer-intensive programs to browsers (even molecular dynamics engines [5]), and APIs that enable browser access to GPUs, multiple cores, and even distributed calculation over multiple browsers (see [2,6] and Boxes). Hence, we expect that the number and complexity of scientific JavaScript libraries and web apps, especially those requiring intensive computing, will grow significantly in the near future.

Scientific data visualization

Visualization is an essential step on the road to discovery and learning, so it is not surprising that it has become the focus of several scientific web apps and libraries. In structural biology and chemistry, JavaScript and WebGL yielded versatile and widely accessible molecular viewers [2,7]. Jolecule (<https://jolecule.appspot.com/>) and JSmol (<https://sourceforge.net/projects/jsmol/>) were early adopter of web technologies, to display molecules and related 3D

objects such as molecular orbitals with rich annotations and graphical features. JSmol in particular is widely used by the wwPDB servers to display structures augmented with colors and labels that highlight specific features, as a support library for data extraction and submission in other web resources, and in several tutorials on molecular structure and reactivity in education and outreach websites like Proteopedia.

An important feature of molecular viewers is the possibility to embed them in websites with minimal effort. Anyone can easily create a web page that displays structures and models to augment static 2D figures from publications (examples in Box 2). Moreover, JSmol supports the inclusion of ready PyMOL sessions, and even multiple synchronized sessions to facilitate structural comparisons.

Other fields of biology and chemistry are permeated by similar interactive visualization technologies for the web. Major efforts include Bionode and BioJS [8], which implement modules for genomic and phylogenetic data processing and visualization. There are also tools to display small molecules and chemical reactions [2], and several developments for anatomical visualizations sharable in real time through the internet, with enormous utility in the clinic [9].

Exciting upcoming possibilities regarding visualization algorithms (Box 2 and [2]) include concurrent interactive visualization from multiple devices, using webcam feeds to control visualizations with gestures and eye-gazing, achieving virtual and augmented reality applications through simple coding and setups, on-browser molecular dynamics visualization/analysis, and exploiting advanced formats such as MMTF and binary CIF to optimize information transfer for visualization of large systems, as in NGL[10] and Litemol[11].

Opening data and democratizing knowledge through web apps

Visualization tools pave the way for transparency and open science by offering a highly interactive medium to display data from web services or provided by publishers alongside the online manuscript. This interactivity facilitates exploring results from database queries, server calculations, and publications (so far only limitedly explored by journals), helping democratize access to scientific data to experts and the general public, and constituting an important resource for teachers and students.

Augmenting online publications with interactive plots from raw data would help both the communication of the findings and the detection of scientific fraud since readers could quickly judge the quality of the data and its interpretation by its authors. Embedding interactive 3D views helps immensely to explain structural features, dynamic processes such as conformational changes and chemical reactions, and so on. Simple website examples today work as online supporting information hyperlinked from publications (Box 2); however, the ultimate goal should be to integrate such interactive content directly into the publications themselves.

Naturally, the success of these proposals hinges on the free availability of the original data used to produce figures and tables; and on the creation of a set of standards to improve

reusability of the same software across multiple publishers and websites. Encouragingly, the journal *Magnetic Resonance in Chemistry* is preparing such a platform to open up availability of raw NMR data and assignments [12].

Learning programming through JavaScript and web apps

Finally, web apps have an inherent didactic potential since they are freely available to anybody with access to a device with a web browser. Endorsing the role of JavaScript as a first language for biologists learning programming, the digital book Nature, In Code[13] teaches the fundamentals of programming by implementing key concepts of evolutionary biology and epidemiology into programs that learners can run directly in the browser. We further argue that, while JavaScript is not an ideal language to learn proper programming, its versatility and streamlined connectivity to HTML and WebGL serve as a powerful motivator for first-time programmers, since they can very easily translate raw data to interactive and engaging visualizations. Accordingly, we believe JavaScript can be introduced in the early years of STEM education curricula to expose students to basic programming concepts, which are increasingly considered an asset in the scientific community [14]. Moreover, since JavaScript source code is inherently open and human-readable, anyone can check how algorithms are implemented and improve or adapt them to their needs (for examples see resources in Box 2). This applies to the hundreds of highly-optimized modules implementing complex functionalities already available under open licenses.

Yet another role for web apps in education comes from emulators of Linux environments, most notably jor1k (Box 2), which allow training experimental scientists in basic command-line interfaces and simple scripting, skills in demand by employers and for all science disciplines [15].

Conclusion

Web apps are revolutionizing the way we interact with information online. Thanks to the massive investment from highly-influential industry actors, web technologies have become ubiquitous, easy-to-use, and capable of performing complex calculations in real time. Scientists were quick to join the web app revolution and have been actively (re)building feature-rich visualization and analysis tools in multiple areas of research. The growing number of open-source scientific JavaScript modules facilitates writing new tools, opening a plethora of possibilities which, without a doubt, will continue to shape the future of scientific research, education and outreach.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

References

1. Stein LD (2008) Towards a cyberinfrastructure for the biological sciences: progress, visions and challenges. *Nat. Rev. Genet* 9, 678–688 [PubMed: 18714290]

2. Abriata LA (2017) Web Apps Come of Age for Molecular Sciences. *Informatics* 4,
3. Abriata LA et al. (2017) Assessment of hard target modeling in CASP12 reveals an emerging role of alignment-based contact prediction methods. *Proteins Struct. Funct. Bioinforma* in press,
4. Stein L (1996) Web applets: Java, JavaScript and ActiveX. *Trends Genet. TIG* 12, 484–485 [PubMed: 8973160]
5. Jiang C and Jin X (2017) Quick Way to Port Existing C/C++ Chemoinformatics Toolkits to the Web Using Emscripten. *J. Chem. Inf. Model* DOI: 10.1021/acs.jcim.7b00434
6. Wilkinson SR and Almeida JS (2014) QMachine: commodity supercomputing in web browsers. *BMC Bioinformatics* 15, 176 [PubMed: 24913605]
7. Yuan S et al. (2017) Implementing WebGL and HTML5 in Macromolecular Visualization and Modern Computer-Aided Drug Design. *Trends Biotechnol* DOI: 10.1016/j.tibtech.2017.03.009
8. Yachdav G et al. (2015) Anatomy of BioJS, an open source community for the life sciences. *eLife* 4,
9. Bernal-Rusiel JL et al. (2017) Reusable Client-Side JavaScript Modules for Immersive Web-Based Real-Time Collaborative Neuroimage Visualization. *Front. Neuroinformatics* 11, 32
10. Rose AS and Hildebrand PW (2015) NGL Viewer: a web application for molecular visualization. *Nucleic Acids Res* 43, W576–579 [PubMed: 25925569]
11. Sehnal D et al. (2017) LiteMol suite: interactive web-based visualization of large-scale macromolecular structure data. *Nat. Methods* in press,
12. Gil R (2017) MRC's proposal to change the submission procedure for the future submission of assignment papers. *Magn. Reson. Chem* DOI: 10.1002/mrc.4631
13. Salathé M (2015) *Nature, In Code*, Leanpub.
14. Resnick M et al. (2009) Scratch: Programming for All. *Commun ACM* 52, 60–67
15. Mangul S et al. (2017) Addressing the Digital Divide in Contemporary Biology: Lessons from Teaching UNIX. *Trends Biotechnol* DOI: 10.1016/j.tibtech.2017.06.007

Box 1:**Web apps – how and why**

Web apps provide functionalities similar to those of standard programs, but directly coded inside web pages using HTML, JavaScript, CSS, WebGL and other web technologies (Figure S1). They require no browser plugins, installation, maintenance or updates; they often offer flat learning curves; they have no server requirements so can be used even in free webhosts; they are inherently compatible with all devices; they run on the client, protecting data privacy; and they are inherently open-source because code is delivered in text format.

At the core of web app programming is JavaScript, a high-level language among the most popular ones. Several JavaScript libraries (Figure I) are available for numerical calculations, data plotting, string analysis, 3D display including augmented and virtual reality, collection and analysis of hardware feeds from device peripherals, queries to web services, etc. Related technologies include trans-compilation of programs into JavaScript (often reaching near-native performance), just-in-time compilation adapted to data types under use, and server-side JavaScript (Node.js). The latter enables large-scale data analysis, a bottleneck of browser-only solutions. Towards improving speed, mainstream developments focus on optimized JavaScript subsets (asm.js), a JavaScript-compatible, assembly-like, low-level language (WebAssembly), and tools to run JavaScript on GPUs and multiple cores, and even on parallelization through multiple web browsers[6].

Box 2:**Examples of web apps in research, education and outreach**

A hyperlinked list is maintained at <http://lucianoabriata.altervista.org/jsinscience/>. Examples are available there, at GitHub accounts <https://labriata.github.io/jsinscience/>, <https://github.com/JoaoRodrigues>, <https://github.com/marcelsalathe> and <https://github.com/lpatiny>, and at EPFL's chemical information service website (<https://isic.epfl.ch/cheminformatics>).

*Examples of full scientific web apps: Sequence Manipulation Suite (<http://www.bioinformatics.org/sms2/>, collection of scripts to operate on DNA and protein sequences); PsychoProt (analysis of amino acid variability in proteins, <http://psychoprot.epfl.ch/>); Wikipedia Structure Explorer, PDB Explorer and Cheminfo.org from EPFL's chemical information service; and web app for protein tertiary structure prediction assessment in CASP12 (http://predictioncenter.org/casp12/FM_assessors_app.html [3]).

*Libraries by EPFL's chemical information service: C++-to-JavaScript port of the RDKit cheminformatics toolkit, C++-to-WebAssembly port of the LIBSVM support vector machine library, libraries for image processing and machine learning in JavaScript.

*Proof-of-concept applications for augmented/virtual reality in molecular visualization, webcam-based eye gazing for hands-free molecular visualization, text analysis, and integration of libraries for data analysis (<http://lucianoabriata.altervista.org/jsinscience/>).

*Selected resources for education and outreach: Proteopedia (<http://www.proteopedia.org/>), quantum chemistry explorer (<http://www.bcbp.gu.se/~orjan/qc/>), chemagic.org, Hack-a-mol (<https://chemapps.stolaf.edu/jmol/jsmol/hackamol.htm>)

*Code from the book *Nature, In Code*: http://www.natureincode.com/code_examples.html

*Opening data and molecular visualizations: <https://lucianoabriata.altervista.org/papersdata/>, <http://lucianoabriata.altervista.org/modelshome.html> and [3].

*A fully-fledged Linux emulator (<http://jor1k.com/jor1k/>) including Perl, Python, Bash interpreters and a C compiler, ideal for quick operations, command-line learning and utilization in tutorials.

Figure I.
Scheme of the multiple features described in Box 1.