informatics

Review

Web Apps Come of Age for Molecular Sciences

Luciano A. Abriata

Laboratory for Biomolecular Modeling, École Polytechnique Fédérale de Lausanne and Swiss Institute of Bioinformatics, CH-1015 Lausanne, Switzerland; luciano.abriata@epfl.ch; Tel.: +41-021-69-30963

Academic Editor: Igor I. Baskin
Received: 11 August 2017; Accepted: 20 August 2017; Published: 24 August 2017

Abstract: Whereas server-side programs are essential to maintain databases and run data analysis pipelines and simulations, client-side web-based computing tools are also important as they allow users to access, visualize and analyze the content delivered to their devices on-the-fly and interactively. This article reviews the best-established tools for in-browser plugin-less programming, including JavaScript as used in HTML5 as well as related web technologies. Through examples based on JavaScript libraries, web applets, and even full web apps, either alone or coupled to each other, the article puts on the spotlight the potential of these technologies for carrying out numerical calculations, text processing and mining, retrieval and analysis of data through queries to online databases and web services, effective visualization of data including 3D visualization and even virtual and augmented reality; all of them in the browser at relatively low programming effort, with applications in cheminformatics, structural biology, biophysics, and genomics, among other molecular sciences.

Keywords: JavaScript; HTML; programming; data analysis; internet

1. Introduction

Slightly over 20 years ago, a small commentary by Prof. L. Stein in *Trends in Genetics* anticipated the impact that “web applets” were to have on how scientists browse, visualize, and analyze biological data available online [1]. Web applets are small programs that run inside clients, i.e., web browsers, as opposed to programs that run on servers. They therefore provide immediate feedback, allowing off-server computations and interactive display of online material, with the additional advantages that can be run offline, not overloading servers, and protecting data privacy as data never leaves the user’s computer. Prof. Stein’s discussion focused on the three most widespread web applet platforms available at that time, which were ActiveX controls, Java applets, and HTML-embedded scripts in the (totally unrelated to Java) JavaScript language [1]. Of these platforms, today JavaScript is by far the best established, most popular, and most developed tool, while the plugin-based alternatives (ActiveX, Java, Flash) increasingly loose support in web browsers. The purpose of this article is to put on the spotlight the powerful features of current JavaScript, both intrinsic and when coupled to HTML5, CSS, WebGL and related technologies, and to demonstrate its power for browsing, visualizing, and analyzing biological data. The reader will find out that JavaScript has evolved far beyond the tool for small, simple web applets it used to be 15–20 years ago, now being capable of complex operations and highly integrable with online resources and computer hardware, allowing even non-professional programmers to build complex applications that rely exclusively on client-side JavaScript, running on the client as “web applications”, or “web apps” for short, that are delivered entirely as web pages.

JavaScript is the trademarked but common name for a high-level programming language standardized as ECMAScript, which is delivered from servers to clients (personal computers, tablets, smartphones) together with HTML code, and executed at the client, i.e., at web browsers through built-in engines that do not require any plugins. JavaScript was born to control HTML objects and thus give interactivity and dynamism to web pages. However, it has evolved immensely in the last decade, linked to the development of technologies such as AJAX, asm.js/Emscripten/WebAssembly, CSS, HTML5, JQuery, JSON, Node.js, WebGL, and Web Workers, while plugin-based tools like ActiveX, Java, and Flash lost support. Moreover, the first JavaScript engines were slow interpreters, but recent years have seen much faster virtual machines and even just-in-time compilers like Google Chrome's engine, V8, where machine code is emitted dynamically and optimized to the kinds of data being handled. This and other developments that redound in increasing the speed and performance of modern JavaScript engines allow complex tasks to be carried out within web pages on the browser. Note that although new JavaScript engines are much faster than original interpreters, there is little agreement about how fast JavaScript actually is and about how it compares to other languages in terms of speed. It is clear that it has matured enough to be of practical use, as this article attempts to highlight; but moreover, many benchmarks show that under certain conditions it gets close to compiled languages like C++, especially when it can profit from optimizations that static compilation cannot foresee. This statement becomes more important in the light that bug-free C++ optimizations are hard to achieve, especially for non-experts, bringing JavaScript closer to compiled languages regarding practical utility by scientists.

Writing JavaScript programs is facilitated by the high-level nature of the language, and their use is facilitated by the lack of specific downloads or setups, because they are built into the webpage structure. The browsers' JavaScript engines connect directly to HTML DOM objects, greatly facilitating data input and output, even interactively and with graphics, granting simplified cross-platform access to multiple functions that would be hard to achieve in other languages, especially for non-expert programmers, including high-level graphics, (live) video and audio processing, connections to online databases and web services, etc., all accessible through relatively simple methods. Given all these advantages, plus the ready availability of JavaScript engines in all free modern web browsers, even in smartphones and tables, it was natural that scientists would try it out as the fabric for applying their ideas into programs.

2. Timeline of JavaScript in Science

Tracing the use of JavaScript in science through a search for "JavaScript" in PubMed abstracts reveals first uses by 1996, a rise to ~15 new articles per year between 2007 and 2012, and a burst afterwards (Figure 1). The most repeated words in the titles of these articles highlight interactivity, visualization, ease of use, and online availability (Figure 1, inset). Applications of JavaScript in these publications range from visualization to database and data processing in the domains of medicine (many providing interactive access to database systems and others simplifying calculations, visualization, and collaborations online), psychology (notable examples are online tests), and basic natural sciences most remarkably in biology and chemistry. The deeper focus of this article, biology and chemistry applications, spans tools for bioinformatics, genomics, molecular biology, structural biology, cheminformatics, and more.

Tables 1–3 list JavaScript-based web apps and libraries associated to molecular biosciences from reviewed literature (Table 1), from unreviewed resources (Table 2), and other JavaScript tools not directly intended for molecular biosciences but with potential applications in it (Table 3). This article goes through some of these web apps and libraries and their combinations, exploring the implementation of advanced technologies exclusively in the web browser, highlighting simplicity for the programmer and open accessibility, and more broadly the utility of client-side JavaScript programming in scientific research, education, and outreach.

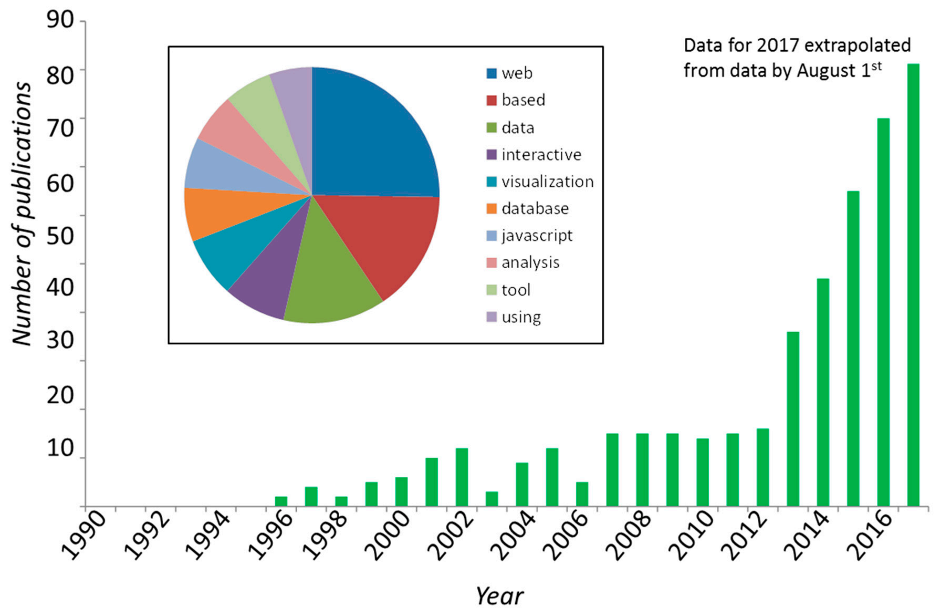


Figure 1. Tracking the use of JavaScript in science. Number of articles containing the word "JavaScript" in the title or abstract, yearly from 1990 to 2017. Inset: The 10 most frequent words in the titles of these articles. Data for 2017 was linearly extrapolated from counts by 1 August 2017.

Table 1. Web apps and libraries of use in molecular biosciences, that rely entirely or largely on JavaScript, described in scientific literature. See text for details on many of these tools.

Tool and Reference	Brief Description
Data visualization	
Protael [2]	Protein data visualization library for the web.
JSAV [3]	The first JavaScript-based interactive sequence alignment viewer.
MSAViewer [4]	Interactive JavaScript visualization of multiple sequence alignments.
jHeatmap [5]	Library to display HeatMaps in the browser.
BioJS [6]	Framework for biological data visualization, on which several libraries have been built (a collection featuring some of them is available [7]).
PhyD3 [8]	Library for phylogenetic tree visualization in the browser.
Molecular visualization and editing	
JSmol [9]	A full molecular viewer and editor program and library. See text for details.
NGL Viewer [10], 3Dmol.js [11], PV, Molmil [12], LiteMol	Various molecular viewers which profit from WebGL technologies better than JSmol providing nicer and smoother rendering, but which are much more limited in capabilities beyond visualization. Most work as embeddable libraries but are also available as full web apps.
JSME [13]	A library to create and edit small molecules graphically, often used as the starting point for subsequent calculations or database queries.
CH5M3D [14], Chemozart [15]	Libraries to edit molecules in 3D.
Full web applications	
PsychoProt [16]	A tool to analyze amino acid variation in proteins in terms of protein physical chemistry, to aid in the analysis of sequence alignments and deep-sequencing-based tolerance to substitutions maps.
Sequence Manipulation Suite [17]	A collection of tools to handle, build, and analyze DNA and protein sequences.
PDB-Explorer [18]	An interactive map to browse the Protein Data Bank.
Wikipedia Chemical Structure Explorer [19]	An interactive 2D structure and substructure search engine for small molecules inside Wikipedia.
ChemCalc [20]	Calculates molecular formulas, molecular weights, elemental compositions, and isotopic distribution. Works as full web app or as an API for JavaScript calls.
ChemInfo.org	A portal to a variety of web-based tools for cheminformatics and related fields, including the PDBExplorer and ChemCalc listed above.

Table 2. Some unreferenced web apps of direct use in molecular biosciences, that rely entirely or largely on JavaScript.

Tool	Brief Description and URL
Data visualization	
Comparing contact predictions and contact maps	Web apps to compare protein contact maps from structures or models to residue-residue contact predictions from Gremlin [21], EVFold [22], and RaptorX [23] contact predictions at http://lucianoabriata.altervista.org/pdbms/
Examples of interactive open data from research articles by this author	Interactive visualization of data from articles: http://lucianoabriata.altervista.org/papersdata/ Interactive 3D visualization of published biomolecular models: http://lucianoabriata.altervista.org/modelshome.html
Molecular visualization and editing	
Hack-a-mol	Interconvert 2D and 3D molecular structures, SMILES and other codes, and coordinate files in several formats: https://chemapps.stolaf.edu/jmol/jsmol/hackamol.htm
Virtual Molecular Model Kit	A web app that provides easy access to JSmol's model editing ability, also using JSME and API calls to external webservices and databases to provide a full online molecular modeling experience: http://chemagic.org/molecules/mini.html
PDB manipulation suite	A growing suite of functions for manipulating PDB files online, for example shifting residue numbers, getting amino acid sequences, and mapping data to B-factors: http://lucianoabriata.altervista.org/pdbms/
Miscellaneous	
MultiProtScale	Client-side version of Expasy's ProtScale server, which handles multiple sequences simultaneously: http://lucianoabriata.altervista.org/multiprotscale/multiprotscale.html
CD fitter and simulator	Fit and simulate far-UV protein circular dichroism spectra: http://lucianoabriata.altervista.org/tests/cd2.html

Table 3. Free JavaScript tools not directly intended for, but with potential utility in, molecular biosciences. The listing is by no means extensive, and mainly tries to convey the reader a feel of the kinds of available tools and the potential of the technologies.

Tool	Brief Description, Possible Uses, and URL
Charts	
Google Charts, uvCharts, Chartist.js	They offer very complete sets of chart types, and are very customizable: https://developers.google.com/chart/ http://imaginea.github.io/uvCharts/ http://gionkunz.github.io/chartist-js/
ChartJS, Highcharts JS, Flot	Also very complete, but here grouped together because they all provide support for older web browsers, yet deliver high-quality graphics, interactivity, and animations in modern browsers. http://www.chartjs.org/
Smoothie Charts	Especially suited for plotting streamed data in real time. http://smoothiecharts.org/
D3.js, n3-charts, Ember charts	D3.js has very broad applications, which allows charting in multiple formats not covered by the libraries mentioned above. Libraries like n3-charts and Ember charts make plotting through D3.js easier. https://github.com/d3/d3/wiki/Gallery http://addepar.github.io/#/ember-charts/overview http://n3-charts.github.io/line-chart

Table 3. *Cont.*

Tool	Brief Description, Possible Uses, and URL
Mathematics and statistics	
NumCalc.com	A very complete scientific calculator at http://numcalc.com/
LALOLib and Mlweb	Very complete libraries for mathematics in the browser, with tools for advanced matrix algebra, statistics, optimization, and machine learning. Can be used as libraries for web apps or through a complete environment (LALOLab) similar to that of stand-alone math programs, with its own plotting capabilities. See http://mlweb.loria.fr/
Numeric.js	Another powerful library for mathematics in the browser, also usable through an environment similar to that of stand-alone math programs. http://www.numericjs.com/
ConvNetJS, synaptic.js, brain, mind, DN2A	Libraries for neural networks and deep learning, covering from simpler utilities like data and function approximation and regression, to self-organizing maps, image regression and object identification, deep learning, and linguistics. Most can be trained and used online, or trained offline and used online. http://cs.stanford.edu/people/karpathy/convnetjs/ http://caza.la/synaptic/#/ https://github.com/harthur/brain https://github.com/stevenmiller888/mind https://github.com/dn2a/dn2a-javascript
jsfft, FFT.js, FFT in many languages, DSP.js, timbre.js	Libraries for Fast Fourier Transforms; together cover direct and inverse transforms, real and complex convolutions, discrete transforms, and other functions (plus, some are actually full packages for signal processing): https://github.com/dntj/jsfft https://www.nayuki.io/page/free-small-fft-in-multiple-languages https://github.com/corbanbrook/dsp.js http://mohayonao.github.io/timbre.js/
Math.js, Sushi, numbers.js, jstat, rift, science, glMatrix	Other math libraries, some with unique features or tailored for specific applications (for example, rift is intended for games, glMatrix, and Sushi for efficient matrix calculations)
Strings	
String.js	Functions that extend those of standard JavaScript strings http://stringjs.com/
textmining and text-miner	Two similar packages for text mining: https://www.npmjs.com/package/textmining https://www.npmjs.com/package/text-miner/ Potentially useful for mining contents in large corpuses of scientific text; Figure 1B of this article was built using such techniques. Text-miner includes lists of stopwords in four languages.
RiTa.js and nlp-compromise	Two libraries for natural language processing: https://rednoise.org/rita/index.php https://nlp-expo.firebaseapp.com/
User interfaces	
WebGazer [24]	Eye tracking library that uses common webcams to infer eye-gaze locations of the user on the web page in real time. Could be used to facilitate molecular visualization as shown in the proof-of-concept example at https://lucianoabriata.altervista.org/jsinscience/jsmolwebgazer/jsmolwebgazer.html
Tangle	Library to create reactive documents, i.e., which interact with the user in a contextual manner and virtually immediate response.
MathJax	Library to display formatted maths in the browser.

Table 3. *Cont.*

Tool	Brief Description, Possible Uses, and URL
Emulators and programming	
Linux emulators	Two complete emulators of a linux system, the second including internet connectivity and a graphical interface: http://bellard.org/jslinux/, http://s-macke.github.io/jor1k/ Could be used for learning linux in a safe and simple environment for example at the beginning of a tutorial course that requires linux knowledge; also for executing shell commands and scripts as well as programs written for Perl, Python, or even compiling C code, etc. on non-linux computers without the need for complex installations, as exemplified with the NESmapper Perl script.
Online programming environments	The two linux emulators listed above include C compilers and allow running Perl, Python, etc. Moreover, there are JavaScript-based web applications specifically tailored for writing, compiling, and running programs. Some notable links: site for learning and running C at http://cs-education.github.io/sys/#VM; a Perl interpreter at https://gfx.github.io/perl.js/; see also Emscripten at http://emscripten.org/
Vi emulator	A working online version of the linux vi editor, could be used to quickly process text files in non-linux computers: http://gpl.internetconnection.net/vi/
3D graphics, computer vision, and augmented and virtual reality	
Three.js, A-Frame	Three.js is probably the most used library for animated 3D graphics, using WebGL. Applications are in molecular visualization and in more generic data visualization, but beyond these uses, it includes multiple numerical algorithms that could be recycled for other purposes: https://threejs.org/ A-Frame is an entity component system framework for Three.js, that makes it much easier to use in virtual and augmented reality applications, the latter especially through AR.js: https://aframe.io/
Jsfeat, tracking.js, js-aruco, AR.js, awe.js, and argon.js	Computer vision and image processing libraries which can identify and track objects or markers. Just like Three.js, they include numerical algorithms that could be used for other purposes. Put together with graphical libraries like Three.js leads directly to augmented reality web apps. https://inspirit.github.io/jsfeat/ https://trackingjs.com/ https://github.com/jcmellado/js-aruco/ https://github.com/jeromeetienne/AR.js/ https://github.com/buildar/awe.js/ https://www.argonjs.io/
Tesseract.js, Ocrad.js	Libraries for optical character recognition from images, useful for retrieving information from article figures. http://antimatter15.com/ocrad.js/demo.html http://tesseract.projectnaptha.com/
Hardware feeds and communication	
WebRTC	Not a library but a collection of standards, protocols, and APIs for real-time server-less, plugin-free communication and data exchange directly between web browsers: https://webrtc.org/ See also the example at https://apprtc.appspot.com/ which implements a web app for server-less video conferencing.
HTML5 guitar tuner	A web app that guides guitar tuning; exemplifies how to read microphone data and analyze the signal through Fourier transforms to decompose the frequency spectrum: https://jbergknoff.github.io/guitar-tuner/

Table 3. *Cont.*

Tool	Brief Description, Possible Uses, and URL
Hardware feeds and communication	
gyro.js	Simplified access to gyroscope and accelerometer information, expanding the information provided by the built-in Geolocation API (which only locates the user's position based on network and/or GPS data but gives no orientation information): http://tomg.co/gyrojs/
GPS.js	Access to primary GPS data: https://github.com/infusion/GPS.js

3. JavaScript Libraries for Online, Interactive Data Display

One powerful application of JavaScript in science is simply that of elegantly and effortlessly presenting data in interactive formats online. There are many JavaScript libraries that facilitate interactive display of numerical data online, either of general format by libraries for generic plotting and 3D graphics, or of specific forms as, for example, libraries that display sequences and sequence alignments, molecular structures in 2D or 3D, and much more as reviewed here. Thanks to these tools, online data display with JavaScript is so simple that it gives strong technological backup to the case for open data availability [25–27]. It is in fact very easy to build JavaScript-based web content to display zoomable versions of data plots and interactive 3D molecular models from publications, as in this author's website (links in Table 2). In the extreme of simplicity, RStudio's shinny app (https://shiny.rstudio.com/) can be used to easily create online content from R analyses, without any programming required as it automatically builds the required HTML, CSS, and JavaScript codes.

There are several good open JavaScript libraries for plotting numerical data (Table 3). Google Charts is especially simple and allows many types of numerical plots such as scatter, lines, bar, and pie charts (example in Figure 2A); moreover, its GeoCharts allow mapping of numerical data on geographical representations, as offered also by only a few other libraries.

Some tools for numerical display actually focus on specific kinds of data. SpeckTackle implements charts for 1D and 2D NMR, UV/visible and infrared spectroscopies, charts for mass spectrometry, and other continuous variables like those recorded in a chromatographic run [28]. More documented and with more examples available, JSpecView [29] displays and converts JCAMP-DX and XML spectra formats, thus handling many kinds of spectral and chromatographic data. JSpecView was originally written in Java and used JavaScript only for interfacing with HTML, but since 2012 there is a full version running entirely on JavaScript, based on the Jmol JavaScript Object which was originally developed to port the Java-based molecular viewer Jmol into JavaScript (JSmol, see next section). Another tool, jsNMR, is dedicated exclusively to 1D and 2D NMR spectral data [30].

With several handy functions, the first JavaScript tool for online visualization of multiple sequence alignments was JSAV [3] (Figure 2C) followed by MSAViewer [4]. There are also tools for visualization of protein features and annotations, such as pViz.js [31] and Protael [2]. A related tool, MultiProtScale (Table 2), computes and displays physicochemical properties of amino acids over sequences and alignments, acting as an interactive extension of Expasy's ProtScale tool. Moving on to genomics, SnipViz [32] and BioCircos.js [33] help visualize and disseminate gene and protein sequences and their annotations, while GenomeD3Plot [34] and pileup.js [35] allow for interactive in-browser visualization of genomic data.

Taking browser-based data visualization to its current state of the art in scientific applications, there are a series of JavaScript tools for molecular visualization (next section). Another noteworthy, recent work describes a JavaScript tool for efficient real-time collaborative neuroimage visualization [36].

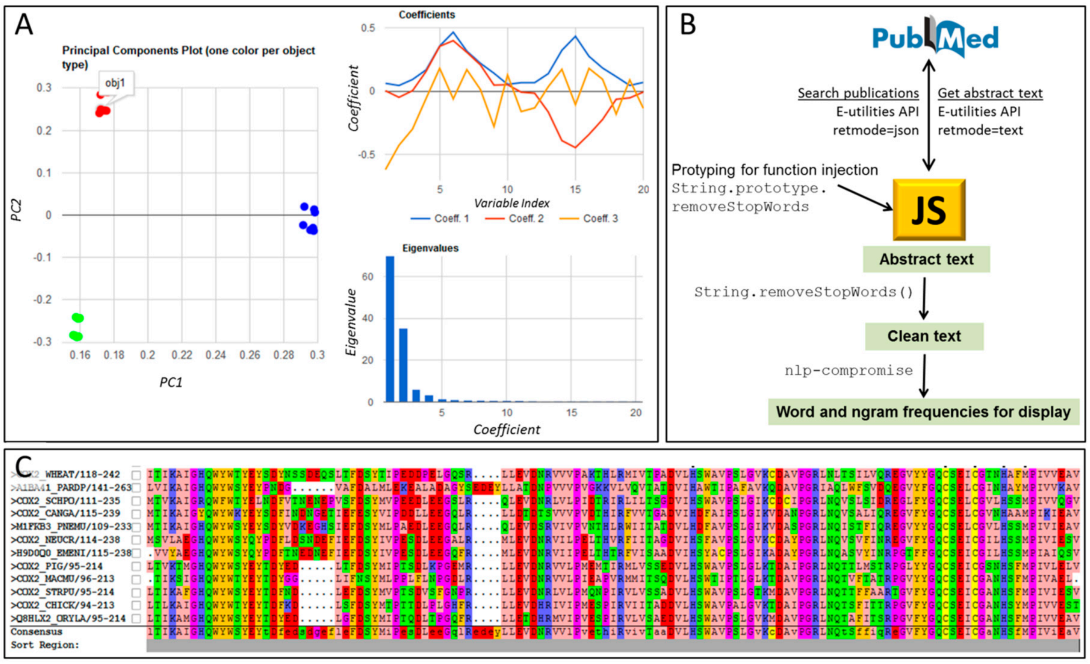
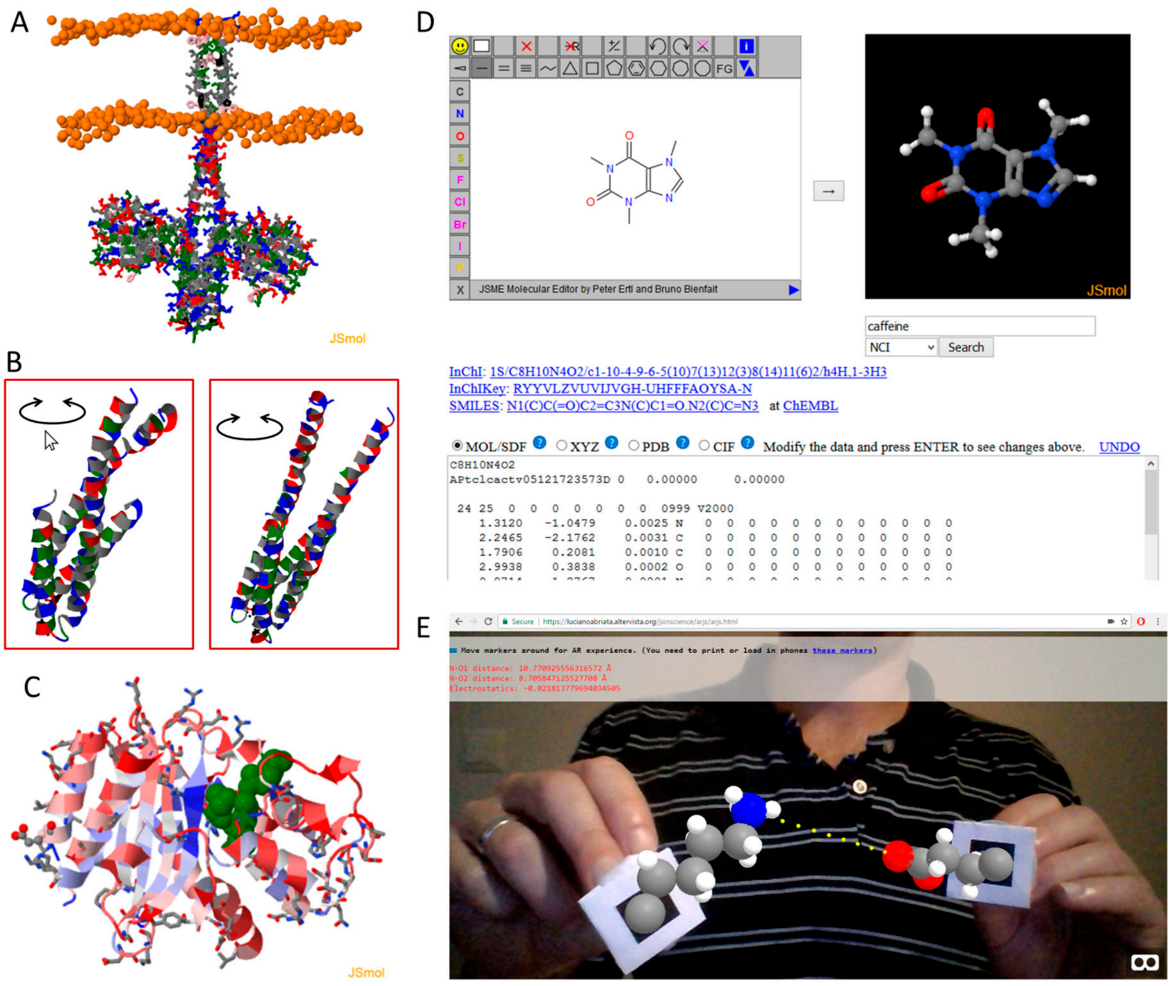


Figure 2. Tools for data analysis and visualization. (**A**) Outputs from the tool for principal component analysis (https://lucianoabriata.altervista.org/jsinscience/pca/pca3.html) run on a collection of 21 objects of three types, each described by 20 variables. Singular value decomposition of the 21 × 20 matrix is performed with LALOLib. Google Charts is used to display a scatter plot of the 21 objects mapped on the two first principal components, line plots of the contribution of each variable to each principal component, and bar plots of the eigenvalues to measure the fraction of variability explained by each principal component; (**B**) Diagram depicting how to search, retrieve and analyze PubMed abstracts with JavaScript (https://lucianoabriata.altervista.org/jsinscience/texts/textworks.html). A query through Eutils API retrieves articles in JSON format; then, further calls retrieve their texts. Abstract texts are processed with a function prototyped into the Script class, which removes stop words (these are irrelevant words that would confound further analysis). The cleaned text is then processed with nlp-compromise to calculate frequencies of words and n-grams (word pairs, triplets, etc.); (**C**) Example interactive visualization of an alignment of PFAM seed members for COX2 (PF00116) with JSAV [3], achieved through the demonstration webpage at http://www.bioinf.org.uk/software/jsav/.

JavaScript Molecular Viewers, Editors, and Calculators, from Chemistry to Cheminformatics and Structural Biology

Integration of 3D visualizations of biological macromolecules directly inside HTML without plugins is now possible with little effort thanks to tools based on HTML5, CSS, WebGL, and canvases. The full web apps and libraries JSmol [9], 3dmol.js [11], and NGLviewer [10] are among the most widely used, and roughly balance technical features with graphic quality. Some other online visualizers explore advanced features such as gesture interaction [37] and capabilities for very large macromolecules [38]. With such easy-to-use resources, researchers publishing biomolecular models shall, and should, make their models available to the scientific community in 3D, as this author does (Figure 3A and links in Table 2). Likewise, webservers using plugin-based molecular viewers should replace them with some of the JavaScript alternatives, as plugins are becoming deprecated and unsupported by web browsers. A recent article discusses online tools for molecular visualizations [39]; meanwhile, this section of the present article focuses on capabilities other than visualization, for example for molecular editing, calculations, and integration with other JavaScript libraries for molecules.



InChI: 1S/C8H10N4O2/c1-10-4-9-6-5(10)7(13)12(3)8(14)11(6)2/h4H,1-3H3
InChIKey: RYYVLZVUVIJVGH-UHFFFAOYSA-N
SMILES: N1(C)C(=O)C2=C3N(C)C1=O.N2(C)C=N3 at ChEMBL

[✓ MOL/SDF] XYZ PDB CIF Modify the data and press ENTER to see changes above. UNDO

```
C8H10N4O2
APtclcactv05121723573D 0   0.00000     0.00000

 24 25  0  0  0  0  0  0  0  0999 V2000
    1.3120   -1.0479    0.0025 N   0  0  0  0  0  0  0  0  0  0  0  0
    2.2465   -2.1762    0.0031 C   0  0  0  0  0  0  0  0  0  0  0  0
    1.7906    0.2081    0.0010 C   0  0  0  0  0  0  0  0  0  0  0  0
    2.9938    0.3838    0.0002 O   0  0  0  0  0  0  0  0  0  0  0  0
```

https://lucianoabriata.altervista.org/jsinscience/arjs/arjs.html

```
Move markers around for AR experience. (You need to print or load in phones these markers)
N-O1 distance: 10.770925556316572 Å
N-O2 distance: 8.705847125527708 Å
Electrostatics: -0.021813779694034505
```

Figure 3. Molecular viewers and editors online. (**A**) JSmol displaying a 3D model of a histidine kinase [40] augmented with interactive decorations to highlight specific features of the protein, from this author's gallery of biomolecular models (http://lucianoabriata.altervista.org/modelshome.html). Since computational models are not available in standard databases like the Protein Data Bank for experimental structures, they remain as no more than mere static pictures, hence the importance of resources for easily sharing 3D views, such as JavaScript molecular viewers; (**B**) Two JSmol applets embedded side to side in the same webpage and synchronized such that rotating any of them produces the same rotation on the other, thus facilitating comparison of two related structures, in this case two X-ray structures of a histidine kinase in different conformational states [41] (https://lucianoabriata.altervista.org/papersdata/accounts2017.html). See also the works by Mwalongo et al. about more complex setups for remote and concurrent visualizations [42,43]; (**C**) Example of JSmol being used to display results from online analyses, in this case data about amino acid variability in proteins by the PsychoProt server [16,44]; (**D**) JSME and JSmol coupled inside the Hack-a-mol web app together with JavaScript queries to external web services to achieve a complete molecular editor, molecular data search facility, and molecular format converter (https://chemapps.stolaf.edu/jmol/jsmol/hackamol.htm); (**E**) Proof-of-concept augmented reality web app that displays lysine and glutamate side chains which the user can move in space with two physical markers. The example uses atomic coordinates for real-time computations of the distance and electrostatic potential between lysine's side chain N atom and glutamate's side chain O atoms. This example is based on A-Frame which gives high-level access to Three.js, thus requiring very simple and short pieces of code (https://lucianoabriata.altervista.org/jsinscience/arjs/arjs.html).

Possibly the most complete JavaScript-based molecular viewer and editor is JSmol (Figure 3A–D), born as the Java-free version of Jmol (currently actively developed by Prof. R. Hanson) and likely destined to replace it as Java becomes less compatible with modern browsers. JSmol includes all of Jmol's capabilities, which means not only potential to display molecules of different formats but also molecular orbitals and any 3D object, useful to draw vectorial information in 3D such as distances or electric field lines. JSmol recently incorporated the capability to display DSSR-derived structural features from RNA structures [45]. Furthermore, it supports rendering of premade PyMOL [46] sessions, synchronization of multiple applets inside a web page (as in Figure 3B, an example of enhanced content [41]), and editing of the visualizations through selection and scripting codes embedded in JavaScript. All these features are important for simplified dissemination and online presentation of structural data and models, allowing online inspection without the need to download files or install specialized programs (Figure 3A–C).

JSmol's capabilities further include modification of atomic coordinates through rotations, translations and even force field-based minimizations, recalculation of hydrogen bonds, aromaticity, secondary structures in proteins, analysis of chiral centers, calculations about molecular electrostatics, and much more. Using these and other JSmol capabilities in conjunction with JSME (see below) and API calls (treated later on) to public chemical data services, Prof. O. Rothenberger built a fully working web-based Virtual Molecular Model Kit [47] that provides easy access to the potential of all these tools (Table 2).

JSmol's selection and scripting languages make integration to other JavaScript tools relatively simple. Thus, it is widely used for visualization in multiple websites for displaying structures retrieved from databases [48], as well as for calculations and for extracting information from files with molecular data in web applications such as protein sequences from PDB files (Figure 3C,D) [16]. Also, JSmol is often used for building molecules and submitting them to servers for calculations through third-party programs, as in the MolCalc [49] web app for fast quantum mechanics-based estimation of molecular properties using GAMESS [50], whose results are displayed back through JSmol—a very interesting tool for education, outreach, and simple research questions. Further interesting resources built from JSmol are a Crystal Symmetry Explorer, including a tool to create files for 3D printing directly from JSmol [51], web interfaces that connect structures to spectral data display with JSpecView described above, and a powerful molecular editor dubbed "Hack-a-mol", described next.

Some other JavaScript molecular viewers specialize in small molecules, allowing users to build and edit molecules online [13,14,52,53]. Probably the most widely used, free tool for small molecules is JSME [13], the JavaScript version of the Java-based molecular builder and editor JME. JSME works on 2D, but a particularly interesting resource called Hack-a-mol (Figure 3D and Table 2) integrates JSME with JSmol and calls to PubChem, NCI, InCHI, and ChEMBL, to interconvert 2D sketches, 3D structures, SMILES, InChI, and InChIKey codes, and MOL/SDF, XYZ, PDB and CIF file formats of any molecule. When any of Hack-a-mol's containers are modified (for example, the file contents in text format are manually changed, or a new atom is drawn in JSME, or the SMILES string is modified) the other containers react to update their contents accordingly. Such a tool, originally devised by Prof. R. Hanson for students of cheminformatics, is actually powerful for research in this field as well as in molecular modeling, for example to produce inputs for parameterization of small molecules for molecular dynamics simulations—for which there also exist web-based tools [54,55] that would be handy to have streamlined. Other tools like Chemozart [15] and CH5M3D [14] are designed to build molecules interactively and directly in 3D.

There are also JavaScript libraries designed for browser-based cheminformatics. Notable examples are kekule.js [52] for tasks such as (sub)structure searching and comparison, functional group detection, 2D-to-3D conversion and format conversions; and ChemDoodle Web Components [53] for building molecules and editing them both in 2D and 3D, computation and display of Lewis structures, simulating and displaying NMR and MS spectra, getting IUPAC names from structures, counting numbers of atoms or bonds, and more. Yet other libraries like MarvinJS facilitate 2D drawing of molecules, reactions, and reaction mechanisms, further usable for reshuffling substituents and querying databases and web services; while JSdraw is a complete resource to design graphical web content for chemistry and biology, including chemical structures, spectra, simulated electrophoretic runs, etc.

4. JavaScript Tools for Numerical Calculations in Simulations, Data Processing, and Analysis

Serious libraries for mathematics and statistics in JavaScript have been around for some time. Some are rather general while others focus on specific subjects such as matrix algebra, statistics, machine learning, neural networks, Fourier transforms, signal processing, or image analysis (Table 3). Different libraries implementing the same algorithms vary in performance, but all of them perform reasonably for small to medium sized datasets. For intensive workloads, some of these libraries support parallelization through multiple workers, a relatively new technology.

A few of the most complete math libraries, such as numeric.js and most notably LALOLib, provide online interfaces that look much like standard desktop programs for mathematics (besides being utilizable as libraries), allowing the user to work online directly on numerical problems without any need to know HTML nor JavaScript, in a graphical environment. Currently, the most up-to-date and comprehensive math library seems to be LALOLib, part of a project aimed at achieving efficient machine learning on the web (MLweb). LALOLib and ML.js include functions for linear algebra, statistics, optimization, and several machine-learning algorithms like principal components analysis, support vector machines, and neural networks. LALOLib even includes tools for high-level graphics within the same environment, and allows loading and saving data and sessions to local drives.

The example in Figure 2A is an entirely client-side web app that maps complex multidimensional objects on two principal components and analyzes how these components reflect the variability in the dataset. It uses LALOLib to run singular value decomposition, and Google Charts to display the results. More advanced ways to carry out principal components analysis are available directly from LALOLib.

As a last remark, it is important to bear in mind that many JavaScript libraries not specifically thought for mathematics actually do include mathematical algorithms coded into them. This holds especially for libraries for 3D graphics, virtual and augmented reality, object detection and tracking, computer vision, and image and sound analysis. For example, Three.js (Table 3 and commented later on when discussing augmented reality applications) includes functions for handling quaternions, otherwise accessible through specialized libraries but which are probably not as thoroughly tested as Three.js which is so widely used.

5. JavaScript Tools for Handling Strings, Text Mining, and Linguistics

Basic built-in JavaScript functions for strings include commands to search or extract characters, search or replace substrings even using regular expressions, split strings to vectors, and do upper/lower case conversions, among others. External libraries provide additional powerful functions for string handling and also for text mining and linguistic analyses (Table 3). Applications of string-handling tools beyond assisting data entry, parsing, and formatting for display include the growing field of scientific text mining [56]. Probably the most remarkable example of the power of string handling in science is the widely used (and cited) chilibot.net bot for mining PubMed abstracts [57]. This bot, introduced in year 2004, recently went offline due to incompatibilities with the latest changes in the Entrez Utils API; it is now back but would likely profit from a JavaScript rewrite.

The String.js library provides functions to capitalize first letters, extract left or right substrings, convert between URL-compatible and human-readable strings, handle HTML tags and codes, spaces, accents, and punctuation, detect different sets of characters, read and write tables in CSV format, and more. Libraries like textmining and text-miner contain utilities for exploring texts, including functions for cleanup and text-mining. Functions for cleanup include word removal from predefined but customizable dictionaries, line removal, number cleanup and handling, case conversion, contraction expansion, etc. Text-mining functions include commands to compute the frequencies of terms and ngrams, and filter out infrequent words and weighting.

At the most advanced end, libraries for linguistic analysis such as RiTa.js and nlp-compromise allow for natural language processing. Together, they cover part-of-speech tagging (i.e., detection of nouns, verbs, adverbs, etc.), recognition and processing of proper names including persons and places, acronyms, numbers, etc., verb conjugation detection and change, negation, single/plural detection and conversion, stemming and normalization procedures, etc. Both come with prebuilt lexicons, i.e., collections of words "understood" by the mining algorithm, which are rather generic but can be extended as required for specific applications by using lexicons built for specific disciplines [58–60]. For applications in chemistry and biology, lexicons could in turn be extended by using tools for conversion between molecular formulas, structures and names, annotations and ontologies from databases, possibly even on-the-fly through JavaScript API calls to external web services or by mining knowledge databases such as DBpedia (the structured form mirror of Wikipedia) or Wordnik (a meta-dictionary).

Besides the tools for text cleanup and analysis, and databases of lexicons and other information, the other main ingredient for text mining is the text to be analyzed. Probably the most important free, open resource of scientific text are PubMed abstracts, which can be easily searched and retrieved directly from NCBI through JavaScript using Entrez Util's API as covered in the next section and in an online example (Figure 2B). Full texts of open access articles are also available from PubMed Central's OAI-PMH Service, but calls to this resource obviously yield much larger volumes of text to be analyzed. Last, there are JavaScript libraries for optical character recognition, which could in principle handle access to text-format data contained within article figures (Table 3).

6. On-the-Fly Data Retrieval and Utilization within JavaScript Web Apps

JavaScript provides very simple ways to retrieve data on the fly from various servers, using asynchronous communication. Following from the previous section, by using Entrez Util's API one can easily search and retrieve abstracts in JSON format from PubMed, for example to keep up-to-date with recent articles containing specific keywords or authors as exemplified in Figure 2B. JavaScript can further ask PubMed for the full abstract, which could then be analyzed through one of the text-mining libraries discussed in the previous section. As seen above too, one can obtain the full article (if open-access) in XML format through PubMed's OAI-PMH Service.

Essentially any API that provides information in JSON, XML, text formats, or even databases, can be called from JavaScript. Examples of APIs to scientific servers of use in biomolecular sciences are RCSB PDB's RESTful Web Service interface, PDBe's REST API, PUG REST for PubChem, the FlyBase API for *Drosophila* genetics, the Biomolecular Interaction Network Database API, openFDA to query data from the Federal Drug Administration agency of the USA [61], the BLAST API for achieving sequence alignments, the Human Genome Variation Archive's RESTful API to access genomic variability in humans [62], and ChemCalc [20] for multiple mass spectrometry-related calculations.

Other remarkable APIs of potential use in science are those from major search servers (which give access to search functions programmatically), APIs for geographical information like Google Maps', APIs from newspapers and news agencies notably the New York Times' API (which gives access to all articles published since 1851), and the DBpedia APIs (to get structured content from Wikipedia).

7. State-of-the-Art JavaScript Web Apps to Inspire Advanced Applications in Molecular Biosciences

The JavaScript universe offers an astonishing range of web apps that proof its intrinsic power and the power it harnesses from interaction with HTML objects, WebGL, access to hardware inputs and to online resources, and from how easily different JavaScript libraries can be integrated within the same webpage. A list of some interesting examples that illustrate the power of integrability is given in Table 3, most not specifically originally devised for scientific applications but certainly applicable to science.

A first example of state-of-the-art JavaScript web apps are emulators for many hardware devices, including emulators of personal computers loaded with standard operating systems. These operating systems can, in turn, run code interpreters and compilers, such that one could practically run any piece of code inside for example Linux, inside the web browser. One remarkable example is the jor1k OpenRISC 1000 emulator by Sebastian Macke (http://jor1k.com) which includes network and graphics support, easy file transfer to local drives, and a functional Linux distribution. Inside this Linux environment, one can edit and run programs written in bash, awk, C, Lua, Python, and Perl languages (and other languages if their interpreters or compilers are installed). Naturally, having the interpreter run inside an emulated environment which is itself running inside the web browser makes it slower than running the interpreter directly on a Linux terminal. Yet, for small programs that are seldom used, for example to address a specific problem in the lab or for teaching, the potential and simplicity offered by jor1k is remarkable.

Another series of state-of-the-art JavaScript web apps and libraries are those for handling images, videos, and audio, either generated by the browser itself or from files or even obtained live from hardware feed. Libraries such as tracking.js, js-aruco, AR.js, awe.js, and argon.js can perform computer vision in the browser, i.e., detection and tracking of the position and orientation of objects, colors, and markers. One step further from marker detection, WebGL JavaScript libraries like Three.js, GLGE or SceneJS can be used to project 3D objects on the camera feed, allowing for virtual and augmented reality applications in the browser. Applications of augmented reality are slowly being explored in science [63–68]; however, most available tools run offline and use multiple pipelined programs specific for each task (marker tracking, camera feed, video generation) thus being complex to set up. Web-based technologies have therefore special potential in simplifying both development and usage. One particular library, A-Frame, provides a framework that facilitates even further the integration of marker detection by AR.js and 3D graphics superposition on a webcam feed with Three.js. Using A-Frame, an example provided here (Figure 3E and link therein) allows the user to drive two virtual molecules with a marker around the screen, reporting the distance and electrostatic potential between their charged atoms with just a few lines of HTML and JavaScript code.

Further applications of object detection and tracking are those related to optical character recognition, mentioned in the section about JavaScript libraries for strings, and those for facial gesture recognition. Among tools for gesture recognition, WebGazer [24] identifies different parts of the user's face (eyes, pupils, eyebrows, etc.) from the webcam feed, and computes what he or she is looking at in the screen. WebGazer could be used for hands-free browsing of complex data on webpages. As a proof of concept (that would benefit from improvements on the resolution of WebGazer-mapped coordinates, currently at ~100 pixels in the author's tests) the example at https://lucianoabriata.altervista.org/jsinscience/jsmolwebgazer/jsmolwebgazer.html shows how WebGazer output can be used to drive JSmol rotations such that whatever region of the molecule the user is looking at always comes to the front and its atoms get labeled.

8. Modularity, Integrability, and Open Nature of JavaScript Libraries

Many examples from this article show that JavaScript libraries behave much like modules which can be put together to build complex applications. Many of the examples revisited throughout this review evidence how easy it is to integrate different JavaScript libraries to create powerful web content: JSmol with WebGazer for eye tracking-based control described earlier herein, or with JSME for interactive editing of structures as in Hack-a-mol (Figure 3D and link therein), or with JSpecView to relate molecular structure to spectral features [29], or even multiple synchronized JSmol instances to facilitate structure comparison (Figure 3B and link therein); WebGL, WebSockets, and Web Workers to achieve concurrent molecular visualizations over remote locations as achieved by Mwalongo et al. [42,43]; AR.js with Three.js through A-Frame to achieve augmented reality extremely easily (Figure 3E and link therein); and plotting and mathematics libraries to compute and display data as in the online PCA tool (Figure 2A and link therein). Modularity and integrability are powerful themselves and facilitated by the open source nature of most libraries, and by the existence of established standards and of large communities devoted to development and support.

Last but not least, a highly positive aspect of JavaScript web apps is their reach. Being essentially text files delivered together with HTML, and taking into account that they run in freely available web browsers, they are easily accessible to anyone intending to learn or to improve or adapt libraries and pieces of code to specific problems—especially highly motivated students with limited access to other didactical resources. These and other advantages, as well as the corresponding disadvantages, are discussed from a practical point of view in Table 4.

Table 4. Advantages and disadvantages of JavaScript for use in scientific research, from a practical viewpoint.

Hardware Capabilities	Programming for Non-Experts	Accessibility
• High speeds when optimized, but still behind the fastest compiled languages. Although speed is highly discussed and varies among interpreters and even according to coding practices, JavaScript is certainly not as fast as optimized C/C++ or fortran for numerical calculations. • Speed is anyway a concern being addressed by major efforts (see asm.js, WebAssembly, and emscripten projects). Also background calculations and parallelization through web workers are under development [69].	• Easy to implement rich graphical user interfaces (through HTML). • Simplified access to peripheral hardware such as built-in accelerometers and gyroscopes. • Simple connection to databases and servers providing data and text, through asynchronous requests. • Data does not leave the user's computer, thus inherently protecting privacy (except for server requests).	• Fully cross-platform, can run in any device with a web browser. • Free, and only a web browser is needed to execute it. (Applications on servers available through Node.js as well). • If not obfuscated, code is human-readable directly from HTML or JS file. • Currently one of the most widely used languages according to several unofficial surveys, with a growing community of developers that contribute packages, libraries, and knowledge.

9. Conclusions

Beyond the increasing use of JavaScript libraries to provide interactive data visualization and small client-side calculations, full JavaScript-based web apps are now showing up, and as intended to forecast and stimulate throughout this review, will become increasingly more common. Specific to biomolecular sciences, examples of full web apps include the very complete molecular visualization libraries and web apps discussed above, the very successful Sequence Manipulation Suite for analyzing and editing nucleic acid and protein sequences [17] and an upcoming PDB manipulation suite to analyze and edit PDB files (available from the author's website), the PsychoProt server for analysis of amino acid variability and tolerance to substitutions in proteins [16,44], the Wikipedia Chemical Structure Explorer [19], and PDB-Explorer [18], among others.

Notably, also classical stand-alone programs and scripts can profit from the power of JavaScript to become available online. This can happen either directly by running them online inside JavaScript-based Linux emulators like jor1k mentioned above, or by converting code into JavaScript with tools like Emscripten, or simply by rewriting from scratch (possibly profiting from existing libraries for efficiency and speed) as in an online web app for analysis and simulation of protein circular dichroism spectra (link in Table 2) rewritten from a macro-containing spreadsheet [70] to HTML5/JavaScript.

Computers and the internet are established, essential instruments at the core of most disciplines, from economics to marketing to social studies to engineering and scientific research. The global trend is that online content and calculations tend to become increasingly more important than offline work, and science is no exception. Whereas, on one side, the cyberinfrastructures for data storage and heavy computations are essential [71], web apps are also essential to display this content in informative, complete, versatile, and interactive ways, and to allow more specific calculations to be carried out by the end user. JavaScript and web components have matured enough for this, and it is now the time when the most exciting developments are starting and will continue to fructify.

Acknowledgments: I thank Fabien Lauer (University of Lorraine, France) for help optimizing the SVD calculations for the online PCA tool, Jerome Etienne and Pyry Matikainen for help implementing the augmented reality example, Ángel Herráez (University of Alcalá, Spain), Robert Hanson (St. Olaf College, USA), and Alexandra Papoutsaki (Brown University, USA) for help integrating WebGazer with JSmol. This work was not funded.

Author Contributions: L.A.A. designed and executed literature research for this review, wrote code for the examples (with help as detailed under acknowledgements), and wrote and revised the manuscript.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Stein, L. Web applets: Java, JavaScript and ActiveX. *Trends Genet. TIG* **1996**, *12*, 484–485. [CrossRef]
2. Sedova, M.; Jaroszewski, L.; Godzik, A. Protael: Protein data visualization library for the web. *Bioinform. Oxf. Engl.* **2016**, *32*, 602–604. [CrossRef] [PubMed]
3. Martin, A.C.R. Viewing multiple sequence alignments with the JavaScript Sequence Alignment Viewer (JSAV). *F1000Research* **2014**, *3*, 249. [CrossRef] [PubMed]
4. Yachdav, G.; Wilzbach, S.; Rauscher, B.; Sheridan, R.; Sillitoe, I.; Procter, J.; Lewis, S.E.; Rost, B.; Goldberg, T. MSAViewer: Interactive JavaScript visualization of multiple sequence alignments. *Bioinform. Oxf. Engl.* **2016**, *32*, 3501–3503. [CrossRef] [PubMed]
5. Deu-Pons, J.; Schroeder, M.P.; Lopez-Bigas, N. jHeatmap: An interactive heatmap viewer for the web. *Bioinform. Oxf. Engl.* **2014**, *30*, 1757–1758. [CrossRef] [PubMed]
6. Gómez, J.; García, L.J.; Salazar, G.A.; Villaveces, J.; Gore, S.; García, A.; Martín, M.J.; Launay, G.; Alcántara, R.; Del-Toro, N.; et al. BioJS: An open source JavaScript framework for biological data visualization. *Bioinform. Oxf. Engl.* **2013**, *29*, 1103–1104. [CrossRef] [PubMed]
7. Corpas, M. The BioJS article collection of open source components for biological data visualisation. *F1000Research* **2014**, *3*, 56. [CrossRef] [PubMed]
8. Kreft, L.; Botzki, A.; Coppens, F.; Vandepoele, K.; Van Bel, M. PhyD3: A phylogenetic tree viewer with extended phyloXML support for functional genomics data visualization. *Bioinform. Oxf. Engl.* **2017**. [CrossRef] [PubMed]
9. Hanson, R.M.; Prilusky, J.; Renjian, Z.; Nakane, T.; Sussman, J.L. JSmol and the Next-Generation Web-Based Representation of 3D Molecular Structure as Applied to Proteopedia. *Isr. J. Chem.* **2013**, *53*, 207–216. [CrossRef]
10. Rose, A.S.; Hildebrand, P.W. NGL Viewer: A web application for molecular visualization. *Nucleic Acids Res.* **2015**, *43*, W576–W579. [CrossRef] [PubMed]
11. Rego, N.; Koes, D. 3Dmol.js: Molecular visualization with WebGL. *Bioinform. Oxf. Engl.* **2015**, *31*, 1322–1324. [CrossRef] [PubMed]
12. Bekker, G.-J.; Nakamura, H.; Kinjo, A.R. Molmil: A molecular viewer for the PDB and beyond. *J. Cheminform.* **2016**, *8*, 42. [CrossRef] [PubMed]
13. Bienfait, B.; Ertl, P. JSME: A free molecule editor in JavaScript. *J. Cheminform.* **2013**, *5*, 24. [CrossRef] [PubMed]
14. Earley, C.W. CH5M3D: An HTML5 program for creating 3D molecular structures. *J. Cheminform.* **2013**, *5*, 46. [CrossRef] [PubMed]
15. Mohebifar, M.; Sajadi, F. Chemozart: A web-based 3D molecular structure editor and visualizer platform. *J. Cheminform.* **2015**, *7*, 56. [CrossRef] [PubMed]
16. Abriata, L.A.; Bovigny, C.; Dal Peraro, M. Detection and sequence/structure mapping of biophysical constraints to protein variation in saturated mutational libraries and protein sequence alignments with a dedicated server. *BMC Bioinform.* **2016**, *17*, 242. [CrossRef] [PubMed]
17. Stothard, P. The sequence manipulation suite: JavaScript programs for analyzing and formatting protein and DNA sequences. *BioTechniques* **2000**, *28*, 1102–1104. [PubMed]
18. Jin, X.; Awale, M.; Zasso, M.; Kostro, D.; Patiny, L.; Reymond, J.-L. PDB-Explorer: A web-based interactive map of the protein data bank in shape space. *BMC Bioinform.* **2015**, *16*, 339. [CrossRef] [PubMed]

19. Ertl, P.; Patiny, L.; Sander, T.; Rufener, C.; Zasso, M. Wikipedia Chemical Structure Explorer: Substructure and similarity searching of molecules from Wikipedia. *J. Cheminform.* **2015**, *7*, 10. [CrossRef] [PubMed]
20. Patiny, L.; Borel, A. ChemCalc: A building block for tomorrow's chemical infrastructure. *J. Chem. Inf. Model.* **2013**, *53*, 1223–1228. [CrossRef] [PubMed]
21. Ovchinnikov, S.; Park, H.; Varghese, N.; Huang, P.-S.; Pavlopoulos, G.A.; Kim, D.E.; Kamisetty, H.; Kyrpides, N.C.; Baker, D. Protein structure determination using metagenome sequence data. *Science* **2017**, *355*, 294–298. [CrossRef] [PubMed]
22. Marks, D.S.; Hopf, T.A.; Sander, C. Protein structure prediction from sequence variation. *Nat. Biotechnol.* **2012**, *30*, 1072–1080. [CrossRef] [PubMed]
23. Wang, S.; Sun, S.; Li, Z.; Zhang, R.; Xu, J. Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model. *PLoS Comput. Biol.* **2017**, *13*, e1005324. [CrossRef] [PubMed]
24. Papoutsaki, A.; Sangkloy, P.; Laskey, J.; Daskalova, N.; Huang, J.; Hays, J. WebGazer: Scalable Webcam Eye Tracking Using User Interactions. In Proceedings of the 25th International Joint Conference on Artificial Intelligence, New York, NY, USA, 9–15 July 2016; pp. 3839–3845.
25. Ince, D.C.; Hatton, L.; Graham-Cumming, J. The case for open computer programs. *Nature* **2012**, *482*, 485–488. [CrossRef] [PubMed]
26. Evans, J.A.; Reimer, J. Open access and global participation in science. *Science* **2009**, *323*, 1025. [CrossRef] [PubMed]
27. Hanson, B.; Sugden, A.; Alberts, B. Making data maximally available. *Science* **2011**, *331*, 649. [CrossRef] [PubMed]
28. Beisken, S.; Conesa, P.; Haug, K.; Salek, R.M.; Steinbeck, C. SpeckTackle: JavaScript charts for spectroscopy. *J. Cheminform.* **2015**, *7*, 17. [CrossRef] [PubMed]
29. Lancashire, R.J. The JSpecView Project: An Open Source Java viewer and converter for JCAMP-DX, and XML spectral data files. *Chem. Cent. J.* **2007**, *1*, 31. [CrossRef] [PubMed]
30. Vosegaard, T. jsNMR: An embedded platform-independent NMR spectrum viewer. *Magn. Reson. Chem. MRC* **2015**, *53*, 285–290. [CrossRef] [PubMed]
31. Mukhyala, K.; Masselot, A. Visualization of protein sequence features using JavaScript and SVG with pViz.js. *Bioinform. Oxf. Engl.* **2014**, *30*, 3408–3409. [CrossRef] [PubMed]
32. Jaschob, D.; Davis, T.N.; Riffle, M. SnipViz: A compact and lightweight web site widget for display and dissemination of multiple versions of gene and protein sequences. *BMC Res. Notes* **2014**, *7*, 468. [CrossRef] [PubMed]
33. Cui, Y.; Chen, X.; Luo, H.; Fan, Z.; Luo, J.; He, S.; Yue, H.; Zhang, P.; Chen, R. BioCircos.js: An interactive Circos JavaScript library for biological data visualization on web applications. *Bioinform. Oxf. Engl.* **2016**, *32*, 1740–1742. [CrossRef] [PubMed]
34. Laird, M.R.; Langille, M.G.I.; Brinkman, F.S.L. GenomeD3Plot: A library for rich, interactive visualizations of genomic data in web applications. *Bioinform. Oxf. Engl.* **2015**, *31*, 3348–3349. [CrossRef] [PubMed]
35. Vanderkam, D.; Aksoy, B.A.; Hodes, I.; Perrone, J.; Hammerbacher, J. pileup.js: A JavaScript library for interactive and in-browser visualization of genomic data. *Bioinform. Oxf. Engl.* **2016**, *32*, 2378–2379. [CrossRef] [PubMed]
36. Bernal-Rusiel, J.L.; Rannou, N.; Gollub, R.L.; Pieper, S.; Murphy, S.; Robertson, R.; Grant, P.E.; Pienaar, R. Reusable Client-Side JavaScript Modules for Immersive Web-Based Real-Time Collaborative Neuroimage Visualization. *Front. Neuroinform.* **2017**, *11*, 32. [CrossRef] [PubMed]
37. Virag, I.; Stoicu-Tivadar, L.; Crişan-Vida, M. Gesture Interaction Browser-Based 3D Molecular Viewer. *Stud. Health Technol. Inform.* **2016**, *226*, 17–20. [PubMed]
38. Rose, A.S.; Bradley, A.R.; Valasatava, Y.; Duarte, J.M.; Prlić, A.; Rose, P.W. Web-based Molecular Graphics for Large Complexes. In Proceedings of the 21st International Conference on Web3D Technology, Anaheim, CA, USA, 22–24 July 2016; pp. 185–186.
39. Yuan, S.; Chan, H.C.S.; Hu, Z. Implementing WebGL and HTML5 in Macromolecular Visualization and Modern Computer-Aided Drug Design. *Trends Biotechnol.* **2017**. [CrossRef] [PubMed]
40. Saita, E.; Abriata, L.A.; Tsai, Y.T.; Trajtenberg, F.; Lemmin, T.; Buschiazzo, A.; Dal Peraro, M.; de Mendoza, D.; Albanesi, D. A coiled coil switch mediates cold sensing by the thermosensory protein DesK. *Mol. Microbiol.* **2015**, *98*, 258–271. [CrossRef] [PubMed]

41. Abriata, L.A.; Albanesi, D.; Dal Peraro, M.; de Mendoza, D. Signal Sensing and Transduction by Histidine Kinases as Unveiled through Studies on a Temperature Sensor. *Acc. Chem. Res.* **2017**. [CrossRef] [PubMed]
42. Mwalongo, F.; Krone, M.; Becher, M.; Reina, G.; Ertl, T. GPU-based remote visualization of dynamic molecular data on the web. *Graph. Model.* **2016**, *88*, 57–65. [CrossRef]
43. Mwalongo, F.; Krone, M.; Becher, M.; Reina, G.; Ertl, T. Remote Visualization of Dynamic Molecular Data Using WebGL. In Proceedings of the 20th International Conference on 3D Web Technology, Heraklion, Greece, 18–21 June 2015; pp. 115–122.
44. Abriata, L.A.; Palzkill, T.; Dal Peraro, M. How structural and physicochemical determinants shape sequence constraints in a functional enzyme. *PLoS ONE* **2015**, *10*, e0118684. [CrossRef] [PubMed]
45. Hanson, R.M.; Lu, X.-J. DSSR-enhanced visualization of nucleic acid structures in Jmol. *Nucleic Acids Res.* **2017**. [CrossRef] [PubMed]
46. DeLano, W.L. *The PyMOL Molecular Graphics System*; DeLano Scientific: San Carlos, CA, USA, 2002.
47. Rothenbreger, O.; Newton, T.; Hanson, R.; Sitzmann, M. The Jmol Virtual Molecular Model Kit: A Resource for Teaching and Learning Chemistry. In *CHED Committee on Computers in Chemical Education*; American Chemical Society: Washington, DC, USA, 2011.
48. Abriata, L.A. Structural database resources for biological macromolecules. *Brief. Bioinform.* **2016**. [CrossRef] [PubMed]
49. Jensen, J.H.; Kromann, J.C. The Molecule Calculator: A Web Application for Fast Quantum Mechanics-Based Estimation of Molecular Properties. *J. Chem. Educ.* **2013**, *90*, 1093–1095. [CrossRef]
50. Schmidt, M.W.; Baldridge, K.K.; Boatz, J.A.; Elbert, S.T.; Gordon, M.S.; Jensen, J.H.; Koseki, S.; Matsunaga, N.; Nguyen, K.A.; Su, S.; et al. General Atomic and Molecular Electronic Structure System. *J. Comput. Chem.* **1993**, *14*, 1347–1363. [CrossRef]
51. Scalfani, V.F.; Williams, A.J.; Tkachenko, V.; Karapetyan, K.; Pshenichnov, A.; Hanson, R.M.; Liddie, J.M.; Bara, J.E. Programmatic conversion of crystal structures into 3D printable files using Jmol. *J. Cheminform.* **2016**, *8*, 66. [CrossRef] [PubMed]
52. Jiang, C.; Jin, X.; Dong, Y.; Chen, M. Kekule.js: An Open Source JavaScript Chemoinformatics Toolkit. *J. Chem. Inf. Model.* **2016**, *56*, 1132–1138. [CrossRef] [PubMed]
53. Burger, M.C. ChemDoodle Web Components: HTML5 toolkit for chemical graphics, interfaces, and informatics. *J. Cheminform.* **2015**, *7*, 35. [CrossRef] [PubMed]
54. Sousa da Silva, A.W.; Vranken, W.F. ACPYPE—AnteChamber PYthon Parser interfacE. *BMC Res. Notes* **2012**, *5*, 367. [CrossRef] [PubMed]
55. Zoete, V.; Cuendet, M.A.; Grosdidier, A.; Michielin, O. SwissParam: A fast force field generation tool for small organic molecules. *J. Comput. Chem.* **2011**, *32*, 2359–2368. [CrossRef] [PubMed]
56. Przybyła, P.; Shardlow, M.; Aubin, S.; Bossy, R.; Eckart de Castilho, R.; Piperidis, S.; McNaught, J.; Ananiadou, S. Text mining resources for the life sciences. *Database* **2016**, *2016*. [CrossRef]
57. Chen, H.; Sharp, B.M. Content-rich biological network constructed by mining PubMed abstracts. *BMC Bioinform.* **2004**, *5*, 147. [CrossRef] [PubMed]
58. Krallinger, M.; Rabal, O.; Leitner, F.; Vazquez, M.; Salgado, D.; Lu, Z.; Leaman, R.; Lu, Y.; Ji, D.; Lowe, D.M.; et al. The CHEMDNER corpus of chemicals and drugs and its annotation principles. *J. Cheminform.* **2015**, *7*, S2. [CrossRef] [PubMed]
59. Choi, W.; Kim, B.; Cho, H.; Lee, D.; Lee, H. A corpus for plant-chemical relationships in the biomedical domain. *BMC Bioinform.* **2016**, *17*, 386. [CrossRef] [PubMed]
60. Kim, J.; Ohta, T.; Tateisi, Y.; Tsujii, J. GENIA corpus—Semantically annotated corpus for bio-textmining. *Bioinformatics* **2003**, *19*, i180–i182. [CrossRef] [PubMed]
61. Kass-Hout, T.A.; Xu, Z.; Mohebbi, M.; Nelsen, H.; Baker, A.; Levine, J.; Johanson, E.; Bright, R.A. OpenFDA: An innovative platform providing access to a wealth of FDA's publicly available data. *J. Am. Med. Inform. Assoc.* **2016**, *23*, 596–600. [CrossRef] [PubMed]
62. Lopez, J.; Coll, J.; Haimel, M.; Kandasamy, S.; Tarraga, J.; Furio-Tari, P.; Bari, W.; Bleda, M.; Rueda, A.; Gräf, S.; et al. HGVA: The Human Genome Variation Archive. *Nucleic Acids Res.* **2017**. [CrossRef] [PubMed]
63. Chastine, J.W.; Brooks, J.C.; Zhu, Y.; Owen, G.S.; Harrison, R.W.; Weber, I.T. AMMP-Vis: A Collaborative Virtual Environment for Molecular Modeling. In Proceedings of the ACM Symposium on Virtual Reality Software and Technology, Monterey, CA, USA, 7–9 November 2005; pp. 8–15.

64. Pence, H.E.; Williams, A.J.; Belford, R.E. New Tools and Challenges for Chemical Education: Mobile Learning, Augmented Reality, and Distributed Cognition in the Dawn of the Social and Semantic Web. In *Chemistry Education*; Wiley-VCH Verlag GmbH & Co. KGaA: Weinheim, Germany, 2015; pp. 693–734. ISBN 978-3-527-67930-0.
65. Gillet, A.; Sanner, M.; Stoffler, D.; Goodsell, D.; Olson, A. Augmented reality with tangible auto-fabricated models for molecular biology applications. *IEEE Vis.* **2004**. [CrossRef]
66. Gillet, A.; Sanner, M.; Stoffler, D.; Olson, A. Tangible interfaces for structural molecular biology. *Structure* **2005**, *13*, 483–491. [CrossRef] [PubMed]
67. Vega Garzón, J.C.; Magrini, M.L.; Galembeck, E. Using augmented reality to teach and learn biochemistry. *Biochem. Mol. Biol. Educ.* **2017**. [CrossRef] [PubMed]
68. Berry, C.; Board, J. A Protein in the palm of your hand through augmented reality. *Biochem. Mol. Biol. Educ.* **2014**, *42*, 446–449. [CrossRef] [PubMed]
69. Wilkinson, S.R.; Almeida, J.S. QMachine: Commodity supercomputing in web browsers. *BMC Bioinform.* **2014**, *15*, 176. [CrossRef] [PubMed]
70. Abriata, L.A. A Simple Spreadsheet Program To Simulate and Analyze the Far-UV Circular Dichroism Spectra of Proteins. *J. Chem. Educ.* **2011**, *88*, 1268–1273. [CrossRef]
71. Stein, L.D. Towards a cyberinfrastructure for the biological sciences: Progress, visions and challenges. *Nat. Rev. Genet.* **2008**, *9*, 678–688. [CrossRef] [PubMed]