No. 14-410

In The
Supreme Court of the United States

GOOGLE INC.,

Petitioner,

v.

ORACLE AMERICA, INC.,

Respondent.

On Petition For Writ Of Certiorari To The United States Court Of Appeals For The Federal Circuit

---

BRIEF OF *AMICI CURIAE* THE OPEN SOURCE INITIATIVE, MOZILLA CORP., AND ENGINE IN SUPPORT OF PETITIONER

JASON M. SCHULTZ
(*Counsel of Record*)
NYU TECHNOLOGY LAW AND POLICY CLINIC
NYU SCHOOL OF LAW
245 Sullivan Street, 609
New York, NY 10012
Telephone: (212) 992-7365
Jason.schultz@exchange.law.nyu.edu

Counsel for Amici Curiae

COCKLE LEGAL BRIEFS (800) 225-6964
WWW.COCKLELEGALBRIEFS.COM

TABLE OF CONTENTS

TABLE OF CONTENTS – Continued

TABLE OF AUTHORITIES

TABLE OF AUTHORITIES – Continued

TABLE OF AUTHORITIES – Continued

INTERESTS OF *AMICI CURIAE*[1]

The Open Source Initiative (OSI) is a nonprofit organization founded in 1998 in order to promote the benefits of open source software through both education and advocacy. OSI also acts as a standards body by maintaining the Open Source Definition, an industry standard that encourages trust among developers, users, corporations, and governments, and that facilitates open source cooperation. The maintenance of this standard allows for the flourishing of alternatives to proprietary software that expand choice in the marketplace, spurring competition and promoting progress of computer arts and sciences.

OSI approves and denies open source licensing applications from companies and organizations including Apple, Computer Associates, the European Union, IBM, Lucent, Microsoft, Mozilla, NASA, Nokia, Oracle, Ricoh, and SUN Microsystems.

Members of OSI's board are leaders in technology, software, and open source communities. Such members include the Debian Project, which produces one of the most popular Linux distributions, and the Wikimedia Foundation, which hosts Wikipedia.

[1] Parties' counsel were given ten days notice of *amici's* intent to file this brief pursuant to the requirements of Rule 37.2(a). Copies of the letters indicating consent have been filed with the Clerk of this Court. No counsel for either party has had any role in authoring this brief, and no persons other than *amici* and their counsel have made any monetary contribution to the preparation or submission of this brief. *See* Rule 37.6.

OSI's interest in this litigation is noncommercial, and it maintains a stalwart public-interest-oriented mission and vision involving open source software. OSI's expertise in open source and computer technology fields is reflected in its co-authorship of recent relevant briefs *amicus curiae* including *Alice Corp. v. CLS Bank International*, 134 S. Ct. 2347 (2014), *Jacobsen v. Katzer*, 609 F. Supp. 2d 925 (N.D. Cal. 2009), and *SCO Group, Inc. v. International Business Machines Corp.*, No. 2:03 CV 294 (DAK), 2005 WL 318784 (D. Utah 2005).

Mozilla Corporation has been a pioneer and advocate for the Web for more than a decade. Mozilla creates and promotes open standards that enable innovation and advance the Web as a platform for all. Today, hundreds of millions of people worldwide use Mozilla Firefox to discover, experience, and connect to the Web on computers, tablets, and mobile phones.

Engine is a technology policy, research, and advocacy organization that bridges the gap between policymakers and startups, working with government and a community of more than 500 high-technology, growth-oriented startups across the nation to support the development of technology entrepreneurship. Engine creates an environment where technological innovation and entrepreneurship thrive by providing knowledge about the start-up economy and constructing smarter public policy. To that end, Engine conducts research, organizes events, and spearheads campaigns to educate elected officials, the entrepreneur community, and the general public on issues vital to fostering technological innovation. Engine has worked with the White House, Congress, federal agencies, and state and local governments to discuss policy issues, write legislation, and introduce the tech community to Washington insiders.

INTRODUCTION AND SUMMARY OF ARGUMENT

Every year, hundreds of thousands of open source programmers create millions of lines of software code. Because the code is open, it is available for the world to see, learn from, and use under extremely liberal conditions. Open source software has resulted in huge contributions – both to commercial and non-commercial firms and communities – providing upwards of $50 billion in annual revenue. There are over 4.9 billion lines of open source code currently in existence, the estimated value of which exceeds $387 billion.

Yet in order for open source to thrive, programmers need to reuse and reimplement existing Application Programming Interfaces (APIs) – the technical standards according to which computer programs interoperate with each other. Without such interoperability, open source programs lose much of their value as creative and competitive alternatives to proprietary – or "closed source" – programs. By making the information needed for interoperability available to all and thereby making the choice to switch software both cost-effective and frictionless, APIs ensure that consumers have open source alternatives to proprietary software. In misinterpreting 17 U.S.C. § 102(b) and holding APIs copyrightable, the Federal Circuit struck a severe blow to the future of open source software, limiting its ability to compete within software markets dominated by single proprietary players. *Amici* ask this Court to grant certiorari and correct this mistake that undermines the constitutional purpose of copyright, the status quo in copyright law, and a significant sector of our high tech economy.

Imagine a single company owning the right to control the way appliances plug into electrical outlets. If a company wanted to build a new toaster, no matter how innovative, affordable, or otherwise competitive, it would still need to ask permission from the outlet owner to make the toaster compatible with the wall's sockets. And if the outlet company already made toasters? Old, clunky, and poorly designed, they would still dominate the market if they were the only appliance able to plug into the wall.

This is the power of interoperability – and this is how APIs work. APIs are the outlet sockets of the software world, and those who control them often dictate how software works in particular areas of technology. While some control over APIs is possible under patent law, Congress and this Court have long recognized that the law cannot and should not cede control of such technological "rules of the road" to copyright owners. Both § 102(b)'s exclusion of any "method of operation" from copyright's purview and this Court's seminal decision in *Baker v. Selden*, 101 U.S. 99 (1879), preclude this outcome.

The Federal Circuit's decision below upends the long-standing status quo that programmers have relied on for decades. It directly contradicts *Baker* by improperly holding that APIs are not uncopyrightable methods of operation like the accounting tables in that case. APIs are not computer programs, but the Federal Circuit treats them as such. Under the Federal Circuit's rule, the simple action of saving a word document would no longer be as straightforward as clicking "File" and selecting "Save" because that menu hierarchy would be owned and copyrightable. Every program, open source or closed, would have to create some new way of helping a user save a document instead of adopting the tried and true standards that users already understand. And the inefficient and wasteful consequences of the Federal Circuit's rule would not be limited to software user interfaces – many of the open source alternatives in cloud computing, mobile mapping, and online encyclopedias could come under legal threat if they tried to create interoperability with the API methods of their competitors.

For these reasons, and for the reasons that follow, this Court should grant certiorari.

ARGUMENT

I. The Federal Circuit's incorrect reading of § 102(b) threatens to undermine the open source software industry, an important driver of creativity and innovation.

Application Programming Interfaces (APIs) are a critical component of software development. This is especially true for open source software development, a segment of the industry that has become a key driver of creativity and innovation. Open source software is "software that can be freely used, changed, and shared . . . by anyone." Open Source Initiative, *The Open Source Initiative*, http://opensource.org (last visited Nov. 3, 2014), http://opensource.org. The Federal Circuit's ruling that APIs are copyrightable would damage open source software's viability and conflict with the express constitutional purpose of copyright.

The Federal Circuit, by reading § 102(b)'s prohibition of copyright protection for methods of operation out of the Copyright Act, severely limited the ability of open source programmers to build cost-efficient programs that compete with proprietary ones. Open source programming relies on the reimplementation of APIs to build new interoperable software. By allowing copyright protection of APIs, the Federal Circuit's ruling makes reimplementation exceedingly risky, and under certain circumstances, unlawful.

A. APIs have the functional purpose of creating interoperability and compatibility between computer programs.

In order to contextualize the scope and impact of the Federal Circuit's ruling, it is important to understand certain technical aspects of APIs. APIs are exact specifications prescribed by a program or operating system that enable communication between various aspects of a computer program or between computer programs. Efthimios Parasidis, *A Sum Greater Than Its Parts?: Copyright Protection For Application Program Interfaces*, 14 Tex. Intell. Prop. L.J. 59, 63-64 (2005). However, APIs are not – as the Federal Circuit treated them – computer programs. Many of the most popular services of today, such as Google Maps or Twitter, make use of APIs to increase access to their services by third-party software programs and websites. For example, Google Maps offers an API that enables websites like Yelp.com to display the location of a popular restaurant on a Google Map instead of Yelp needing to write its own map program from scratch. *See* Brian Proffitt, *What APIs Are and Why They're Important*, Readwrite (Sept. 19, 2013), http://www.readwrite.com/2013/09/19/api-defined. The Yelp website simply communicates with Google Maps and asks for map data on the specific location of the restaurant. *Id.* Google Maps then sends that data to the Yelp site to be displayed. The method of communication between the two sites is defined by the Google Maps API. *Id.* Similarly, the "copy and paste" function of most word processors and web browsers is an API – it enables those programs to interact and move a block of text between applications, or between a website and an application. *See Clipboard*, Windows Dev Center – Desktop http://msdn.microsoft.com/en-us/library/windows/desktop/ms648709(v=vs.85).aspx (last visited Oct. 25, 2014) (showing that the link "clip-board reference" contains the API reference). As these examples show, APIs are the fundamental building blocks that enable communication between two computer programs or applications.

At the most basic level, an API is much like a standard electrical outlet: it allows two programs to plug into each other. The standard outlet serves the same purpose. It allows the circuitry in an electric device to interoperate with the circuitry in a house or office. When Cuisinart designs a new toaster oven, its electrical engineers do not need to know how a house is wired as long as they configure the plug for their toaster oven according to the standard outlet specifications. Likewise, electricians who wire or rewire homes do not have to anticipate which toaster oven the homeowner will purchase so long as they set up standard outlets. When Cuisinart builds a new toaster oven, it can save costs by using the existing plug. It also knows that it can market its product to potential customers who want to switch – all a switching customer has to do is unplug his existing oven from the socket and plug in the new Cuisinart toaster. Cuisinart could theoretically design a new plug that works with a home's wiring, but consumers are used to seeing the two- and three-pronged plugs, and they would have to figure out how to implement the new plug, possibly requiring a converter. Requiring each appliance company to create its own plug is an inefficient anti-consumer, anti-competitive system that would confuse users and render older toaster ovens incompatible because homeowners would constantly be changing the sockets in their house.

APIs are much like this standard outlet. So long as programmers, analogous here to electricians and electrical engineers, know the standard terminology and structure of an API, they can write their code to call on those standard terms, lowering transaction costs. And just as an engineer relies on the standard electrical socket to design a toaster that will plug into any home's wall, a computer programmer relies on APIs to design a computer program that will work with other programs.

Similarly, interoperability dictates the standardization of light bulb bases. Because light bulb bases are standardized, a purchaser has access to a multitude of cheap options when selecting a bulb. No matter the bulb, the purchaser can be confident that it will work with any given base to transfer electricity from the socket in the lamp to the light bulb. A homeowner could have a black-light bulb, a fluorescent bulb, or an energy-efficient bulb, yet because the socket is standardized, none of those outputs matter to the electrician who wires a lamp. In the same way, an API allows two programs to interact with each other through a standardized language.

Yet another way to understand APIs is to consider the order of the gears on a car's gearshift – Park, Reverse, Neutral, Drive, Low (PRNDL). The driver uses the gearshift lever to move the transmission from one state to another. How would the driver know how to operate a car if each manufacturer were required to organize the gears differently? Each of these familiar and logically named settings sends information to the vehicle, which then acts according to the driver's commands. In this way, the gearshift lever acts as an interface between the driver and the automobile – and because it is standardized, any driver can enter any automatic automobile in the United States and be confronted with the same PRNDL sequence. While the gears in PRNDL could be arranged in multiple different ways, or even renamed entirely, changing this interface would result in potentially dangerous (and undoubtedly inefficient) consequences for the driver. Further, the use of a uniform interface insulates the user from underlying changes or complexity in the actual gear mechanisms. The automobile manufacturer or mechanic is free to make modifications to the engine in order to make it more powerful or efficient, but the user experience with the gearshift remains the same.

Just like a driver uses the PRNDL interface to operate a car, a computer program uses an API's list of known commands in order to interoperate with another program. In both cases, consistency in the names and order of the commands are critical to enable any programmer to operate another program.

Changing the names of the components or the structure, sequence, and organization of an API would make it unrecognizable to another program so that the programs could no longer interoperate. Standard interfaces in these contexts allow both the mechanic to make modifications to the engine of a car and the software programmer to make modifications to her code. In both cases they are free to do so, because they know the driver or third-party programmer will still be confronted by the same standard interface they have always used to operate the underlying machinery or code. Outlets, light bulbs, and gearshifts are not supposed to have complicated interfaces; their interaction is about simplicity and necessity. The whole idea behind an API is that the most efficient means of production for Cuisinart is to reuse the existing method by which the plug interoperates with the wall outlet.

Much like in electrical engineering and engine design, in software development, once an API is established, innovation occurs through implementations that rely on its standard form. A programmer examines her implementing code and removes a redundancy or inefficient sequence – but the API remains unchanged. When her program receives instructions from another program using the precise language of the shared API, her program is able to produce the same requested result, but with greater speed and reliability due to revisions made to the implementing code. There are only a few ways in which APIs can be developed, and common practice dictates that developers reference and reuse existing APIs that achieve the same function. This practice improves speed and efficiency for developers and results in more competition and consumer choice. Just as the standardized outlet leads to more, better, less expensive toaster ovens on the market, programmers' ability to reimplement APIs leads to more, better, and less expensive alternatives to proprietary programs.

B. Open source use and reimplementation of APIs lowers the cost of software development and enables more diverse participation, greater competition, and expanded consumer choice in the software industry.

Ultimately, the ability to rely on existing APIs results in double savings for the open source programmer: she does not need to rewrite the subroutines for the basic tasks the API is designed to handle, and she does not need to duplicate the functions provided by compatible programs. These savings are passed on to other developers who duplicate and make use of that open source code, as well as to consumers. And these benefits are not limited to open source developers. Incumbent developers receive network-effect benefits of more compatible programs and therefore more users. The low cost of software development fostered by freely usable APIs has also greatly benefitted startups. Using preexisting APIs allows entrepreneurs to cheaply build software that can interoperate with larger systems, opening the software market to a broader pool of developers and leading to the creation of innovative new products. As the number of programs compatible with proprietary software and its APIs increases, the likelihood that future programs will use the same APIs in order to remain compatible likewise increases. Further, because APIs make it less necessary to attempt to access or decompile source code in order to ensure compatibility, originators of popular software such as Microsoft can provide APIs for its Windows software to encourage compatible products that expand the possibilities of existing programs.

The threat of liability resulting from the Federal Circuit's ruling would have a chilling effect on open source software innovation because it discourages individual developers from tapping into existing frameworks. Open source software encourages collaborative development of widely dispersed individuals to produce software in small and frequent updates. Pekka Abrahamsson et al., *Agile Software Development Methods: Review and Analysis* 74 (2002). If there is uncertainty about what developers can and cannot reuse, those developers are unlikely to work on a project at all. The consequences of that kind of chill are felt by many. Open source programmers often address deficiencies in a program's code or function, and a hobbyist or individual who cares passionately about a software project might want to make a beneficial change to its source code. If part of this change involves an API, under the Federal Circuit's rule, the specter of litigation will discourage the hobbyist from making a positive contribution to the software.

Proprietary software owners could rely on the Federal Circuit's rule to threaten open source software developers who are trying to compete, killing off all kinds of potential new innovation in the process. *Cf.* Branford L. Smith & Susan O. Mann, *Innovation and Intellectual Property Protection in the Software Industry: An Emerging Role for Patents*, 71 U. Chi. L. Rev. 241, 242 (2004).

The Federal Circuit's rule also threatens to allow "vendor lock-in" and reduce competition by mandating higher switching costs between vendors. If Microsoft were able to copyright the API that allows a user to open a Microsoft Word document, it would be exceedingly difficult for a user to switch to a different program because his or her files would be trapped in Microsoft's proprietary format. Open source software reduces the risk of vendor lock-in. Cheap and efficient use of existing APIs is critical to getting these open source alternatives to the market, and thus, increasing software competition.

C. Facilitating the existence of successful open source alternatives fulfills copyright's constitutional purpose of promoting creativity and innovation, and the Federal Circuit's ruling threatens a thriving open source industry's ability to realize those objectives.

The constitutional purpose of copyright law is to "promote the Progress of Science and useful Arts." U.S. Const. art. I, § 8, cl. 8. The creative boon represented by open source can be seen in the over 200,000 projects and over 4.9 billion lines of code that comprise the open source software market. *See Black Duck Software Estimates Development Cost of Open Source Software at $387 Billion*, Black Duck Software (Apr. 14, 2009), https://www.blackducksoftware.com/news/releases/2009-04-14. This innovation has contributed immensely to the economy and the public good. *See id.* (estimating the total development cost of open source software at $387 billion); Amanda McPherson et al., *Estimating the Total Development Cost of a Linux Distribution*, The Linux Foundation (Oct. 2008), http://www.linuxfoundation.org/sites/main/files/publications/estimatinglinux.html (approximating the cost of developing a Linux distribution at $10.8 billion); *Investor Relations: Financial Statements*, Red Hat, http://investors.redhat.com/financials-statements.cfm (last visited Oct. 25, 2014) (noting Red Hat's revenue at over $1 billion); *StatCounter Global Stats: Top 5 Desktop, Tablet, & Console Browsers from Sept. 2013 to Sept. 2014*, Statcounter, http://gs.statcounter.com/#browser-ww-monthly-201309-201309-bar (last visited Oct. 25, 2014) (highlighting Mozilla's Firefox browser's market share of 18%).

History shows that promoting interoperability is fundamental to the promotion of "Science and useful Arts." U.S. Const. art. I, § 8, cl. 8. The software written for the first personal computers (PCs) was only compatible with one operating system and IBM's proprietary firmware. Rival PC manufacturers, based on their understanding of these interfaces, were able to develop their own firmware and build PCs that could run software previously only compatible with IBM machines. What followed was explosive worldwide growth in PC production and ownership, with diverse products, reduced prices, and a vastly increased overall demand for software. And the innovation afforded by open interfaces is not limited to PCs. UNIX was the first modern operating system. Heather J. Meeker, *The Open Source Alternative* 4 (2008). Because the computer industry considered the UNIX API to be uncopyrightable, programmers reimplemented the API to create Linux, an open source operating system. *See id.* at 6. In turn, Linux APIs were later used by computer giants Sun and Oracle to ensure that programs written for Linux were compatible with their own operating systems. Opening Expert Report of Dr. Owen Astrachan at 39, *Oracle Am., Inc. v. Google Inc.*, 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA). Likewise, Apple – now the world's largest computer company – changed its interface for its current operating system, OS X, to make use of and rely upon the UNIX API. Joe Wilcox, *Will OS X's Unix Roots Help Apple Grow?*, CNET (May 21, 2001, 12:05 PM), http://news.cnet.com/Will-OS-Xs-Unix-roots-help-Apple-grow/2100-1040_3-257982.html. These examples, among many others, demonstrate that the free implementation and reuse of APIs has been critical to open source growth and the initial success of the technology industry as a whole.

Today, the use of shared APIs to drive innovation in computing and software programming is as important as it has ever been. Iain Dey, *Barclays Snubs Tech Titans to Save Billions*, Sunday Times (Jan. 6, 2013), http://www.thesundaytimes.co.uk/sto/business/Finance/article1188867.ece (noting that large companies like Barclays are switching to open source software and saving billions of dollars in the process); Fahmida Y. Rashid, *Open-Source Software Gives Competitive Advantage*, eWeek (Feb. 8, 2011), http://www.eweek.com/c/a/Linux-and-Open-Source/Open-Source-Software-Gives-Competitive-Advantage-Gartner-Survey-729638 (a survey of 547 companies in 11 countries in 2010 found that more than half said they are using open source software). Popular technology companies such as Microsoft, Apple, Twitter and Facebook all rely on open source software, as does the government. *See* Memorandum from David M. Wennergren, Assistant Secretary of Defense, Department of Defense, Chief Information Officer to the Secretaries of the Military Departments (Oct. 16, 2009), *available at* http://dodcio.defense.gov/Portals/0/Documents/OSSFAQ/2009OSS.pdf (encouraging the use of open source software); Cade Metz, *Facebook, Google, and the Rise of Open Source Security Software*, Wired (Oct. 29, 2014), http://www.wired.com/2014/10/facebook-builder-osquery (detailing the efforts of technology companies such as Facebook and Google to release open source security programs); Tim O'Reilly, *Thoughts on the Whitehouse.gov Switch to Drupal*, O'Reilly (Oct. 25, 2009), http://radar.oreilly.com/2009/10/whitehouse-switch-drupal-opensource.html; Dirk A.D. Smith, *Exclusive: Inside the NSA's Private Cloud*, CIO (Sept. 29, 2014, 6:00 AM), http://www.cio.com/article/

2688434/private-cloud/exclusive-inside-the-nsas-private-cloud.html (noting that the secretive NSA is using open source products in its cloud computing system); *Red Hat's Decade of Collaboration with Government and the Open Source Community*, Red Hat (May 11, 2012), http://www.redhat.com/en/about/blog/red-Hats-decade-of-collaboration-with-government-and-the-open-source-community.

Overall, open source software products and services have saved consumers an estimated $60 billion per year and generate an estimated $50.7 billion in annual revenue. Dave Rosenberg, *Study Finds "Free Open Source Software Is Costing Vendors $60 Billion,"* CNET (Apr. 16, 2008, 11:50 AM), http://www.cnet.com/news/study-finds-free-open-source-software-is-costing-vendors-60-billion; *Projected Revenue of Open Source Software from 2008 to 2020*, Statista, http://www.statista.com/statistics/270805/projected-revenue-of-open-source-software-since-2008 (last visited Oct. 25, 2014). These savings are generated as a result of users having the freedom to easily switch from more expensive proprietary systems. Making APIs copyrightable threatens to deny consumers these savings and to deny various organizations a robust software market from which to choose the most suitable product for their needs. Unduly granting APIs copyright protection would transform a market characterized by competition and consumer responsiveness into one controlled by a few dominant players who have incentives to exclude smaller, innovative open source developers. If – as the Federal Circuit held – APIs are copyrightable, the kind of technological innovation and creativity that characterized the birth of personal computing will be severely discouraged, and the reimplementation and reuse of APIs will become much more expensive and restricted.

II. The Federal Circuit's ruling breaks with well-established and well-reasoned precedent that open source programmers have relied upon for decades.

Open source software developers have relied on the decisions of this Court and the First Circuit for decades to guarantee software interface interoperability. *See Baker*, 101 U.S. 99; *Lotus Dev. Corp. v. Borland Int'l, Inc.*, 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996). In *Baker*, Charles Selden developed and published a book containing a system of bookkeeping alongside an essay explaining the system. 101 U.S. at 100. This system streamlined the process of bookkeeping such that what was once accomplished by "double entry" could now be seen on one or two pages due to Selden's creative use of ruled lines and headings. *Id*. This Court held that such a system was not an appropriate subject matter for copyright law. *Id.* at 105. The organized lines and headings were considered to be part of the "method of operation" of the overall bookkeeping system, and thus, were uncopyrightable. *Id.* at 103-04.

Just as Selden's system provided the instructions, rules, and methods for using double entry bookkeeping for accountants and businesses to communicate or "interface," an API provides the methods for computer programs to do the same. Each line of Selden's form can be thought of as a method of operation – and, indeed, the overall organization of the form can be considered an API itself. The layout of the headings at issue in *Baker* is just the structure, sequence, and organization of the bookkeeping system in the same way that an API is part of the structure, sequence, and organization of the computer program. *See* Michael F. Morgan, *The Cathedral and the Bizarre: An Examination of the "Viral" Aspects of the GPL*, 27 J. Marshall J. Computer & Info. L. 349, 422 (2010) (arguing that "[i]t is . . . difficult to envision any strong arguments for the proposition that a[n] . . . API is not a method of operation"). As this Court wrote in *Baker*, "[t]he very object of publishing a book on science or the useful arts is to communicate to the world the useful knowledge which it contains." 101 U.S. at 103. Here, the "very object of publishing" an API is exactly the same. A significant part of the value of the Java programming language is derived from the millions of programmers and users who have put in the time and effort to understand and learn the language. Reading *Baker* as not fundamentally deciding the *un*copyrightability of APIs seriously hinders that objective.

In *Lotus*, the First Circuit properly interpreted the principle laid out in *Baker* for modern software development, noting that:

> [t]he description of the art in a book, though entitled to the benefit of copyright, lays no foundation for an exclusive claim to the art itself. The object of the one is explanation, the object of the other is use. The former may be secured by copyright. The latter can only be secured, if it can be secured at all, by letters-patent.

49 F.3d at 816-17 (quoting *Baker*, 101 U.S. at 104-05). *Lotus* established that aspects of computer programs considered to be a "method of operation" are not protected by copyright. *Id.* at 815. In particular, the First Circuit held that dropdown file menus, with commands like "Copy," "Print," and "Quit," are simply a set of user-facing instructions or organized commands. *Id.* When the user clicks "File" and then "Open," for example, when opening a new document, he activates the code that contains the instructions that the program uses to communicate with the operating system to retrieve and open that specific document. The organization of those commands, at issue in *Lotus*, is analogous to the organization of the API. While the structure of the file menu today is common among computer users, the structure of Java's API is common to programmers. For purposes of copyright and § 102(b), the structure of an API is thus nearly identical to the structure of the file hierarchy at issue in *Lotus*.

Because the method of operation at issue in *Lotus* – the means of allowing user-written programs to communicate with spreadsheet programs – is effectively an API, open source programmers have reasonably relied on these decisions as precedent that protects the reuse and reimplementation of APIs in computer programs. Open source programmers can attribute part of their success to the use and reimplementation of existing APIs. The fact that these developers have not been sued for this reimplementation demonstrates the reasonableness of their reliance on *Baker* and *Lotus* – and it is that reasonable reliance that the Federal Circuit's holding threatens to unravel. *See* Jonathan Band & Masanobu Katoh, *Interfaces on Trial 2.0*, at 21-22 (2011) (arguing that *Lotus* and its progeny have reduced developer concern over possible liability resulting from the creation of interoperable software); *see also, e.g.*, *Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co.*, 535 U.S. 722, 739 (2002) (prioritizing the concern of disrupting settled business expectations and noting the Court's wariness of the potential consequences). As discussed above, the ultimate effect of the Federal Circuit's rule would be to chill software creation and innovation.

III. The technological world today would look vastly different if the Federal Circuit's ruling had been in effect following *Lotus*, and the consequences would affect many products that millions use daily.

If *Lotus* had been decided according to the Federal Circuit's rule in this case, the world of computing would be vastly different. There would be no basic actions familiar to computer users across platforms, and there would be much less competition and choice among software providers. Indeed, the Internet as the world now knows it would look much different – and the world would be worse off. The now-ubiquitous file drop down menu would be a nonliteral, copyrightable element of a computer program. The ease and familiarity of a simple action like saving a document and then sending that document to the printer would not exist. Use of computers, and the Internet, would be much less universal, and much more difficult.

Instead, these functions and commands that most of us use every day would vary widely across every different program. One would not be able to save a Microsoft Word document by clicking "File" then "Save," because the structure, sequence, and organization of that action would be copyrighted and owned by Lotus. Instead, Microsoft would have had to devise another method by which users could save documents – and that method would *also* be copyrighted. In turn, Adobe – the company that writes the program on which many of us read and edit PDF files – would have to alter its command menu as well. And so on and so forth. The result would cost users countless hours in adjusting to and learning the new menus, and cost developers countless hours in thinking of and creating new drop down menus. The proverbial wheel would need to be reinvented, over and over again.

Considering this alternate universe in even more depth shows the drastic impact of the Federal Circuit's decision. For example, something as simple as the act of throwing a file in the "trash" to discard it would be copyrighted and owned. This would mean that every operating system would have to have a different and unique method of discarding files, and that every application would have to be written differently to accommodate each operating system's disparate functions and code. There would be no uniformity of experience for users or developers. Imagine the added unnecessary hours that an Open Office[2] developer would spend writing APIs so that her suite of programs could simply open a file off the hard drive. And the API would have to be different for Apple OS X, for Microsoft Windows, and for Linux.

On an even larger scale, Amazon could "own" cloud computing with a copyright on Amazon Web Services, effectively creating a monopoly. *See* Jillian

[2] Open Office is one of the most popular open source productivity suites. The suite has been downloaded close to 120 million times. *Download Stats*, Apache Open Office, http://www.openoffice.org/stats/downloads.html (last visited Oct. 25, 2014).

D'Onfro, *Here's A Reminder Just How Massive Amazon's Web Services Business Is*, Bus. Insider (June 16, 2014, 9:33 AM), http://www.businessinsider.com/amazon-web-services-market-share-2014-6#ixzz3IA0Bd1fo (describing the dominance of Amazon Web Services in cloud computing). With cloud software becoming a major component of business and personal computing, the implications of an Amazon stranglehold on the market are troubling. Today, Amazon's API plays a critical role in cloud computing, where cloud-based interfaces allow for users to interact with other computers over the Internet. *See* Quentin Hardy, *Active in Cloud, Amazon Reshapes Computing*, N.Y. Times, Aug. 27, 2012, http://www.nytimes.com/2012/08/28/technology/active-in-cloud-amazon-reshapes-computing.html?_r=0. Multiple cloud service providers use a standard API developed by Amazon. *Amazon Web Services (AWS) and Eucalyptus Partner to Bring Additional Compatibility Between AWS and On-Premises IT Environments*, Eucalyptus (Mar. 22, 2012), http://www.eucalyptus.com/news/amazon-web-services-and-eucalyptus-partner. Businesses use this standard API to take advantage of the vast capabilities that cloud computing offers. If the API developed by Amazon were copyrightable, Amazon could deny other cloud services the right to use it and effectively control for itself a sizeable portion of the cloud computing market.

In another example, the Federal Circuit's holding could signal the end of Wikipedia. In a very real sense, Wikipedia is simply a reimplementation of the structure, sequence, and organization of the information contained in encyclopedias like *Britannica* or *Encarta*. APIs, to a large extent, are just strings of information that computers can read, the structure of which is largely dependent on the language being used. *Encyclopedia Britannica* is also just a string of information with a structure, sequence, organization. The facts within the encyclopedia are not copyrightable, but the way those facts are expressed *is*, much like the implementing code. *See Feist Publ'ns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 348 (1991). However, if the headers and categorization of the encyclopedic entries, which constitute the structure, sequence, and organization of the encyclopedia, were copyrightable, Wikipedia would not be able to rely on that structure in organizing one of the world's largest online repositories of factual information. Allowing copyright of the structure, sequence, and organization of a computer program or of an API is akin to granting *Encyclopedia Britannica* a monopoly over the encyclopedia.

These hypothetical results are inimical to the constitutional purpose of copyright protection. U.S. Const. art. I, § 8, cl. 8. Programs would be clunky and inefficient. They would be more expensive. Competition would suffer, leaving proprietary giants controlling the software market, because the cost of development and innovation would be too great for small players or the open source community.

Buying into the idea that a Java API is copyrightable restricts a particular "method of operation," such that 95 years from now, a software company, for the first time, would be able to duplicate and implement a file hierarchy. That means users would not have a seamless, common software experience, even for the most miniscule tasks, like dragging and dropping a file, until the year 2109. If *Lotus* was rightly decided, the Court should grant certiorari since the decision at issue here is clearly at odds with *Lotus*.

CONCLUSION

For the reasons mentioned above, the petition for a writ of certiorari should be granted.

Respectfully submitted,

Jason M. Schultz
(*Counsel of Record*)
NYU Technology Law and
Policy Clinic
NYU School of Law
245 Sullivan Street, 609
New York, NY 10012
Telephone: (212) 992-7365
Jason.schultz@exchange.law.nyu.edu

Counsel for Amici Curiae