No. 18-956

IN THE
Supreme Court of the United States

GOOGLE LLC,
Petitioner,

v.

ORACLE AMERICA, INC.,
Respondent.

On Petition for a Writ of Certiorari to the United States Court of Appeals for the Federal Circuit

AMICUS CURIAE BRIEF OF DEVELOPERS ALLIANCE IN SUPPORT OF PETITIONER

BRUCE GUSTAFSON
DEVELOPERS ALLIANCE
1015 7th Street, NW
2nd Floor
Washington, DC 20001
(202) 735-7333
bruce@developersalliance.org

JAMES H. HULME
Counsel of Record
NADIA A. PATEL
ARENT FOX LLP
1717 K Street, NW
Washington, DC 20006
(202) 857-6000
james.hulme@arentfox.com

TABLE OF CONTENTS

Page

TABLE OF AUTHORITIES

Page(s)

Cases

Statutes

Other Authorities

INTEREST OF *AMICUS CURIAE*

The Developers Alliance is a non-profit corporation that advocates for software developers.[1] Our corporate mission is to "[a]dvocate on behalf of developers and the companies that depend on them, support the industry's continued growth, and promote innovation."[2]

Alliance members include industry leaders in consumer, enterprise, industrial, and emerging software, and a global network of more than 75,000 developers.[3]

Amici have no direct financial interest in the outcome of this case, but have a strong interest in seeing that the law continues to support innovation in the software industry. Due to the importance of the issues presented to the developer community, the Developers Alliance has been following this litigation closely. The Developers Alliance has previously joined

---

[1] No counsel for any party authored this brief in whole or part, and no person other than *amicus curiae* or its counsel made a monetary contribution to the preparation or submission of this brief. All parties received timely notice of the Developers Alliance's intent to file and consented to the filing of this brief.

[2] https://www.developersalliance.org/about/about-the-alliance/.

[3] A list of Developers Alliance members is available at https://www.developersalliance.org/member-directory/. Google is a Developers Alliance member but took no part in the preparation of this brief.

two *amicus* briefs in this matter before the Federal Circuit.[4]

SUMMARY OF THE ARGUMENT

The current case has implications that go far beyond the two litigants involved. In 2017 there were an estimated three million software developers in the United States, and their collective work added an estimated $565 billion to the country's gross domestic product.[5,6] As a result of the current litigation, developers are now confused about whether and

---

[4] *See* Brief of *Amici Curiae* Rackspace US, Inc., Application Developers Alliance, TMSoft, LLC, and Stack Exchange Inc., *Oracle America, Inc. v. Google Inc.*, Nos. 13-1021, 13-1022 (Fed. Cir. May 30, 2013); *see also* Brief of *Amici Curiae* Engine Advocacy, The App Developers Alliance, and Github Inc., *Oracle America, Inc. v. Google Inc.*, Nos. 17-1118, 17-1202 (Fed. Cir. June 1, 2017).

[5] There are nearly three million professionals that are involved in software development and programming as part of their jobs. Over half of those are strictly software developers while the rest have occupations that require programming as a secondary component of their work, such as computer scientists, data analysts, and database administrators. Developers Alliance & NDP Analytics, *Quantifying Risks to Interoperability in the Software Industry* (2017), https://www.developersalliance.org/interoperability-report-december-2017.

[6] In 2017, Software.org, the BSA Foundation, commissioned The Economist Intelligence Unit (EIU) to assess the economic impact of the software industry. The EIU collected and analyzed the most recent data available from several recognized and reputable sources. *The Growing $1 Trillion Economic Impact of Software* (2017), https://software.org/reports/2017-us-software-impact/.

where established practices constitute copyright infringement. Specifically, developers now question their ability to freely create interoperable software across projects and platforms, as has been common practice. The inevitable result of this uncertainty will be reduced innovation, higher industry costs, and increased litigation.

The record in this case provides ample background for the Court to address and clarify the copyright issues arising from software interfaces and their fair use. The courts of appeals have taken divergent approaches to the application of copyright law to computer software, and the Court should provide much-needed certainty for the software developer community. For these reasons we ask that the petition for a writ of certiorari be granted.

ARGUMENT

Technology has progressed since this Court decided *Baker* v. *Selden*, 101 U.S. 99 (1880). While the nineteenth century saw great strides in the development of programmable machines, the "software" and "hardware" of the age were predominantly paper tapes, punch cards and electromechanical systems. Since then, the ability to repurpose complex equipment without completely reinventing them has become a fundamental driver of innovation.

When a developer writes original software (or "code" in vernacular), there is a universal understanding that they hold protected rights in their

work.[7] To enable collaborative development and interoperability, however, developers must be free to connect their own code to code that other developers have written. This is commonly done through software interfaces called Application Programming Interfaces (APIs). Shared APIs are the established industry mechanism that promotes innovation while protecting the creative works of individual developers. Without shared APIs, every device and program is an island, and modern software development simply cannot happen.

Interoperability through software interfaces increases innovation by allowing independent developers to build on the work of others. Interoperability allows for independent innovation in logically separate sections of a complex computer program by defining how to pass information from one program section to another. For instance, by using software interfaces between the firmware of a mobile device, its operating system, and the application software developers have written, consumers can freely add, delete and update apps without purchasing a new phone.[8] In fact, it is now easy for users to port their entire application library from one

---

[7] Developers often refer to writing computer software as "writing code," analogous to "writing prose" or "writing poetry" for other authors. "Writing code" and the verb "coding" are equivalent terms.

[8] Traditionally, "hardware" refers to the physical aspects of a device, while "software" is the broad term for programs that run on hardware. "Firmware" is software that is semi-permanently placed in hardware, often forming part of the interface between the two.

device to another. Interoperability also allows developers to specialize and thus creates efficiencies in the use of scarce programming skills. Finally, interoperability helps drive innovation by balancing the market power of the various participants in a complex software ecosystem.

It is critical for the developer community to understand the intellectual property rights of the various software ecosystem participants. The role of both patent law and copyright law is to promote innovation.[9] Without clear and consistent rulings from U.S. courts on the application of these rules to modern software, developers will be reluctant to collaborate or to create interoperable systems.

I. Software interfaces are the universally accepted mechanism allowing developers to write software that is interoperable and independent of the underlying hardware.

Just as specialization of labor revolutionized the manufacturing sector, the ability to separate hardware and software into interoperable parts has revolutionized the technology industry. The key that has unlocked software innovation is the ability for software and hardware from many independent

---

[9] *See* U.S. Const. art 1, § 8, cl. 8 ("Congress shall have Power . . . To promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their Writings and Discoveries . . . .").

creators to interoperate with software and hardware from many other independent creators.

The size and complexity of software projects is growing steadily. A multi-player online game today can contain five million lines of code, while a luxury car might contain 100 million lines.[10] Software development no longer occurs only inside corporations, but is now distributed both geographically and across time zones, with many independent developers contributing their effort and knowledge to a single project. The distribution of development effort is now so broad that any particular developer can expect to work on many software projects during their careers, often concurrently.

To support the tremendous demand for new software, the developer community has adopted a number of universal practices. First, developers rely on libraries of common software functions written by others, rather than recode these functions themselves for every project. This improves developer efficiency. It is also standard practice for developers working on large projects to coordinate the efforts of several individuals and create interoperable code modules that can be connected and reconnected to achieve the larger programming goal. This allows developers to specialize and to tackle larger tasks as part of a community. Thirdly, industry has focused on isolating the underlying device hardware from the software

---

[10] An informative chart with comparable code sizes for various technologies is available at https://www.visualcapitalist.com/millions-lines-of-code/.

above it by adopting a number of more standardized platforms that bridge the gap between generic software and proprietary hardware. This creates opportunities to develop software programs that run on many different devices without having to rewrite new code for each device. In all cases, the key enablers are software interfaces that connect code blocks and manage the controlled transfer of formatted information between layers and blocks of software.

Because the use of software interfaces drives such universal benefit, it is generally accepted in the developer community that these structures should be widely available and easy to implement. The result has been the emergence of the open-source software community to share code and publish APIs, and the rise of software platforms like Java and Android to enable greater interoperability across a wide range of devices. It has also led to the emergence of reusable software tools tailored to the most popular software languages. A developer who can master these tools gains efficiency, and can then apply these gains to a wide range of projects. This in turn has led to competition amongst the software tool builders to capture a broad community of developers to increase the value of their programming platform—a virtuous cycle. Simply put, interoperability is a deeply embedded principle in modern software development today.

The next phase of technology evolution is already upon us. Fully interoperable devices are being linked together to form the internet of things (IoT). Household appliances, mobile phones, computers, wristwatches, doorbells and a vast array of sensors

and devices are already sharing information through a dynamic and evolving network of interfaces. Even more than in today's software industry, interoperability is a fundamental enabler of IoT. It is estimated that the global economic productivity resulting from IoT would drop by $77 billion over the next eight years if current interoperability practices were restricted.[11]

II. Developers rely on intellectual property law to both protect their independent work and promote their ability to work collectively.

There is no doubt that a developer can hold a copyright in the software they have written.[12] Because modern software development is such a collaborative enterprise, the current level of innovation rests on the developer community's shared understanding of the "rules of the road." In order for collaborative development to occur, developers acknowledge that others must be free to connect their own code to code that is already written. This is done through interfaces that rely on a shared knowledge of the structure in which information must be passed. Developers have long operated on the understanding that these software interfaces could be shared while

---

[11] Developers Alliance & NDP Analytics, *Quantifying Risks to Interoperability in the Software Industry* (2017), https://www.developersalliance.org/interoperability-report-december-2017.

[12] The legislative history of 17 U.S.C. § 102(b) makes this clear. *See* H.R. Rep. No. 94-1476, at 56–57 (1976); S. Rep. No. 94-473, at 54 (1975).

the enabling code behind them could remain proprietary.

The advantage of this arrangement is that it places no penalty on developers for sharing, and it encourages market competition by creating a mechanism for developers to easily call on the comparable APIs of many competing implementing code authors. More effective implementing code gets re-used more often by the community, and thus the community as a whole produces better software. The reputation and prospects of the best authors rise accordingly.

The Java programming language owes its success to developer demands for an efficient way to create interoperable software. It promised developers a "write once, run anywhere" programming language where their investment in learning and mastering the programming tool would translate into better and faster code production across many projects and in many environments. In turn, Java's creators were able to market a platform to millions of hardware designers that would bring a large and established developer workforce to work on their projects. Java is now one of the world's most popular programming languages and platforms.[13] The key to this remarkable ecosystem was the publication of free and open APIs which enabled interoperability. Without open APIs, it is unlikely that Java would have grown

---

[13] Brief of Petitioner at 216a.

to be as popular as it is or that developers would continue to support it.[14]

III. Control of APIs and software interfaces will have a fundamental influence on the future of software innovation.

The interoperability of software systems hinges on the interfaces between the many component parts. Who has rights to these interfaces, and how those rights are allocated, is critical to the future of software development. This cannot be overstated.

If access to APIs remains separate and independent of rights to the implementing code in the background, then the ability to collaboratively create interoperable software and hardware will proceed at its current frenetic pace and the goals of intellectual property law will be met. If, on the other hand, both the use and implementation of APIs are subject to arbitrary control, then these interfaces become a choke point reducing innovation.

APIs exist at the interface between two software environments. In most cases, the code on one side is pre-written and waiting for appropriate parameters to be passed so that it can execute and perform its prescribed function. On the other side, the assumption is that new code is regularly being

---

[14] A recent study of developers' views on APIs supports this statement. *Developers Voice Concerns over Court Ruling Upending Use of APIs* (2019), https://www.developersalliance.org/spring-2019-api-survey.

written that seeks to pass the appropriate parameters and off-load the effort of independently writing the established code being called.

If APIs can be free and open on one side, but owned and licensed on the other, then the value in the platform implementing code can shift from the cost to re-implement, to the cost to replace a developer community invested in the related tools (once the community is established). If, by extension, it becomes common for all languages and platforms to manufacture this value shift once their developer communities mature, then developers will limit their investment in any particular system, knowing its popularity is finite, and efficiency and innovation will suffer.

Developers also invest heavily in mastering a particular programming language. Each programming language is simply a syntax and vocabulary for writing software; the underlying logical structures are universal. But the more places where a specific language can be used, and the more portable the resulting code is, the more valuable a language is to learn and to master. If someone builds an interpreter and a library of APIs that allows this common language to be used efficiently and in many foreign contexts, then the language gains in popularity and its attractiveness increases. The interpreter itself has value to third parties as a shortcut to a bottom-up reimplementation of the enabling platform code. If the interpreter and APIs are later restricted, however, developer investment in the language is lost and innovation is reduced.

IV. Current case law has left it unclear how and when a developer's work is protected intellectual property.

Both patent law and copyright exist to promote creativity. The law does this by carefully balancing the rights of creators in their works against the rights of others to build upon existing art. In the case of patents, strict examination and detailed boundaries help ensure there is space for future innovation. Copyright, however, is more fluid, and creators have had to rely more heavily on exceptions, precedent and analogy to define individual rights against future creators.

This case is an ideal one for the Court to bring established software practice into harmony with the copyright framework. Software interfaces are a critical component of a significant and highly innovative industry. The ability for developers, most of whom have no legal background, to proceed with confidence in creating highly collaborative and interoperable software systems relies on a clear articulation of how laws apply to their work.

The current case has created confusion amongst the lower courts and software developers about the extent of copyright protection for certain types of software. It has created further confusion as to when it is appropriate to relax protections to advance innovation. It has also created a mosaic of conflicting opinions amongst the lower courts, leaving developers confused about whether and where established practices constitute copyright

infringement.[15] The mosaic of conflicting opinions is the most detrimental because software development by its nature is widespread and often virtual, as is the market for software.

Java and Android are both textbook examples of interoperable platforms.[16] The Java programming language is one of the most popular programming languages in which to develop interoperable software.[17] Throughout the long history of this litigation, Google, Oracle, and *amici* have addressed the application of copyright to the various component parts of the Java and Android platforms.[18] They have also explored the application of fair use, *scenes a faire*,

---

[15] For example, developers are now questioning whether programming languages themselves are subject to copyright. *See* Charles Duan, *Can Copyright Protect a Language?*, Slate (June 3, 2015), https://slate.com/technology/2015/06/oracle-v-google-klingon-and-copyrighting-language.html.

[16] A recent blog post on Google's developer website cites 16,000 Google Play Certified android devices, and Oracle's website claims that 15 billion devices run Java. Mishaal Rahman, *There are nearly 16,000 Google Play Certified Android Devices*, XDA Developers (Apr. 17, 2018, 2:30 PM), https://www.xda-developers.com/number-of-google-play-certified-android-devices/; Go Java, https://go.java/index.html (last visited Feb. 20, 2019).

[17] A recent developer survey found that almost seventy percent of responding developers use Javascript. *Most used programming languages among developers worldwide, as of early 2018*, Statista, https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/ (last visited Feb. 20, 2019).

[18] *See* Brief of Petitioner at 1 (listing opinions below).

the merger doctrine, and creative versus functional works.[19]

Further, the long history of this case has provided a rich analysis of the specifics of the tools and principles under review, but also of the many ways that existing case law might be applied.[20] Google, Oracle and *amici* have provided numerous analogies from libraries to QWERTY keyboards, and at various stages of litigation the lower courts and *amici* have added to this growing list in an attempt to anchor modern software development in the rich history of creative works and copyright application.[21]

Finally, the long path through the lower courts has brought focus to specific issues they cannot resolve: the software interfaces so critical to developers and their work. These interfaces are the key to interoperability, a characteristic which already defines modern technology development, and which will be of paramount importance as the internet of things grows in the years ahead.

---

[19] Brief of Petitioner at 8–10; *see generally* Petitioner's Appendix.

[20] Brief of Petitioner at 8–10.

[21] The many lower court analogies and holdings are found in Brief of Petitioner in the extensive Appendix. For further examples of *amicus* commentary and review, *see* Jonathan Band, *Interfaces on Trial 3.0: Oracle America v. Google and Beyond*, SSRN (Oct. 19, 2016), https://ssrn.com/abstract=2876853; *see also* Peter S. Menell, *Rise of the API Copyright Dead?: An Updated Epitaph for Copyright Protection of Network and Functional Features of Computer Software*, 31 Harv. J. L. & Tech. 305 (2018).

Developers and the industry they have built require clarity on the boundaries of their rights if they are to continue to innovate. If an interface is the boundary between two co-dependent systems (where each side must mirror the other, like toy building blocks or a plug and socket), this Court is uniquely capable of articulating the rights and obligations for the participants on each side towards the other. Alternatively, if an interface is a separate idea, like a stream of binary digits or a metaphysical sheet of glass which simply separates the shared landscape into two viewpoints, then the Court can define rules specific to interfaces themselves—meta-rules applicable to all participants. What developers require is a uniform understanding because the number of interfaces being implemented across our industry will grow exponentially in the years ahead.

CONCLUSION

To provide the requisite legal clarity and to resolve the conflicting approaches and conclusions of the lower courts, the petition for a writ of certiorari should be granted.

Respectfully submitted,

James H. Hulme
Counsel of Record
Nadia A. Patel
ARENT FOX LLP
1717 K Street, NW
Washington, DC 20006
(202) 857-6000
james.hulme@arentfox.com

Bruce Gustafson
Developers Alliance
1015 7th Street, N.W., 2nd Floor
Washington, DC 20001
bruce@developersalliance.org

Attorneys for Amicus Curiae
Developers Alliance