Declaring Computer Code Uncopyrightable with a Creative Fair Use Analysis

Adam Mossoff[*]

The past 50 years have been a period of massive innovation in computer technology—the personal computer revolution, the internet, and the mobile revolution. This half-century has been bookended by Stephen Breyer's writings on copyright protection for computer programs. In 1970, then-Professor Breyer at Harvard Law School first expressed his now well-known intellectual property skepticism in *The Uneasy Case for Copyright*.[1] He argued, among other things, that "[c]omputer programs should not receive copyright protection at the present time."[2] Five decades later, now-Justice Breyer wrote the majority opinion in *Google v. Oracle*,[3] holding that Google is not liable for copyright infringement for its unauthorized copying of approximately 11,500 lines of the "declaring code" in Oracle's Java computer program.

Google was a blockbuster copyright case. It was a legal dispute between two titans in the tech industry arising from Google's unauthorized copying of a computer program that has been integral to the interconnected digital world we all live in today—Java. The lawsuit filed by Oracle against Google took a decade to work its way through the courts with multiple trials and appeals. Oracle claimed billions in damages.

[*] Professor of Law, Antonin Scalia Law School at George Mason University. Thank you to Devlin Hartline for his insights and his comments on a draft of this article. For research assistance, thank you to Kevin Beck, Matthew Dollett-Hemphill, and Juliet Lomeo.

[1] See Stephen Breyer, The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs, 84 Harv. L. Rev. 281 (1970).

[2] *Id.* at 351.

[3] 141 S. Ct. 1183 (2021).

Legally, it was just as significant. The Supreme Court promised to decide for the first time the scope of protection for computer programs under the Copyright Act.[4] It was also the first case in over two decades in which the Court addressed a fair use defense, and this was the first time the Court considered how fair use applied to the copying of a computer program. To add icing to the cake, the Court separately added a third issue concerning the standard of review for the fair use doctrine. Fifty-nine amicus briefs were filed with the Court after certiorari was granted, in addition to the filing by the solicitor general. Everyone seemed to have something to say about this case.[5]

Surprisingly, Google won in a 6-2 decision with the majority opinion written by Justice Breyer and a dissent by Justice Clarence Thomas (joined by Justice Samuel Alito). The oral argument appeared to go badly for Google, so many observers thought Google was going to lose. But the majority opinion was surprising beyond just the outcome itself. Justice Breyer's majority opinion is unusual in both the form and substance of copyright law.

As a matter of fair use doctrine, the Court held for the first time that a commercial firm that copied a copyrighted work without authorization for a commercial purpose, namely to create a competing product in the marketplace, was immune from liability. The Court appeared to punt on the question of copyrightability, assuming for the purposes of the opinion that the computer program at issue was copyrightable. But that punt will prove to be a false hope for owners of computer programs like Java. Justice Breyer's fair use analysis breaks new ground from prior Court decisions in both form and substance, providing a very expansive fair use defense for anyone who engages in

[4] The Supreme Court almost ruled on this issue in 1995 in *Lotus Dev. Corp. v. Borland Int'l,* but the Court split 4-4 due to the happenstance of a snowstorm that prevented Justice John Paul Stevens from participating in oral argument. 516 U.S. 233 (1996). Similar tribulations struck again in *Google*. The COVID-19 pandemic delayed oral argument, originally scheduled for late March 2020, until October 2020. Shortly before oral argument, Justice Ruth Bader Ginsburg passed away. Justice Ginsburg was known for her copyright jurisprudence, and she was often on opposite sides of cases from that of Justice Breyer. Thus, only eight justices (again) heard and decided *Google*.

[5] This included me, as I joined one of several amicus briefs filed by academics in support of Oracle.

the unauthorized copying of a computer software program.[6] The end result is that, despite the Court's denial, *Google* was a decision about the copyrightability of the computer code at issue in the case—the "declaring code" in the Java computer program.[7]

Of course, this article cannot possibly address all the legal and policy issues for copyright law in Justice Breyer's pathbreaking opinion. It would be surprising to see any single law journal article do this, even one of the monstrously long law journal articles that certainly will be published on *Google* in the years to come. Thus, I will focus here on the surprising and novel elements in the opinion in this academic version of a "hot take" to *Google*. First, I will briefly detail the background to the case—the nature of the technology and the licensing-negotiation breakdown between Google and Oracle. Second, I will discuss the copyrightability issue, which is necessary to properly frame the fair use analysis that follows. Third and finally, I will focus in depth on the Court's fair use analysis, as it drives the legal result in the case and opens up whole new vistas of doctrine and policy in copyright law. Along the way, Justice Thomas will make appearances in the discussion for various insights from his dissent. I hope the reader will enjoy the ride, as it has lots of twists and turns.

How *Google v. Oracle* Came to Be

First, a description of the background to the case, detailing the technology and the interactions between Google and Oracle, is necessary. Of course, this history is necessarily abridged, so professional programmers and legal experts will likely be unhappy with the details omitted in this survey. For those unhappy with this section, please consider it version 1.0. Later, longer, and more bloated versions will certainly patch the holes and debug the glitches.

Marc Andreessen, the programmer of Mosaic (the first web browser) and now venture capitalist, put it perfectly: software has

[6] See Google, 141 S. Ct. at 1213–14 (Thomas, J., dissenting) (observing that "the majority's application of fair use is far from ordinary").

[7] The third issue on the standard of review ended up being the proverbial dog that did not bark: commentators thought the Court would reverse and remand on the grounds that the Federal Circuit applied the wrong standard of review, but the Court decided this issue without having to reverse.

eaten the world.[8] It's omnipresent in our lives today. It goes far beyond computers, tablets, and smartphones. It runs planes, trains, and automobiles. It runs elevators, thermostats, and coffee machines. There are more transistors at work today than there are leaves on all the trees on the planet—approximately *15 quintillion* transistors as of 2018—and all of them are running computer programs.[9]

Everyone knows the computer programs we use daily (such as Outlook, Chrome, Word, or iTunes), but innumerable computer programs are running under the hood of our devices beyond the computer programs we directly use—which is why tech geeks call us "end-users." The programs that are buried deep in our operating systems and apps are the means by which all these programs interface with each other so that our computers, smartphones, and other devices work. For instance, your email client interfaces with your operating system, which interfaces with a company server, which interfaces with the structural code of the internet, which interfaces with the server receiving the email, and so on, and so on.

One such under-the-hood program is Java, which contains numerous software interfaces known as Application Programming Interfaces (APIs). APIs allow different computer programs to communicate with each other. Java, created by Sun Microsystems in 1995, has been integral to the interconnected world of the internet and mobile revolution of the past several decades. Programmers could use Java to write a program to run on any electronic device that had the Java Virtual Machine, another computer program, installed on it without having to write separate programs to run on each type of device; thus, Sun's famous slogan, "write once, run anywhere."

Java has been massively successful. Over the decades, Sun Microsystems and Oracle, which purchased Sun Microsystems in 2009, have earned billions through licensing of Java to companies creating computer programs to run on a myriad of digital devices and

[8] See Marc Andreessen, "Why Software Is Eating the World," Wall St. J., Aug. 20, 2011, https://bit.ly/2VgD7hW.

[9] See Simon Winchester, The Perfectionists: How Precision Engineers Created the Modern World 280 (2018).

products. (Since Oracle now owns Java and Oracle was the plaintiff that sued Google in 2010, I'll refer to Oracle as the owner of Java, even if a date occurs before 2009.)

Enter Google and its Android smartphone platform. In competition with Apple's iPhone, Google designed the Android smartphone to be open source, but not on the same basis as the well-known General Public License (GPL) that governs most open-source programs. Google gives away Android for free and permits customization of this smartphone operating system, but it does so with another open-source license (the Apache License). Google's business model isn't licensing; rather, it earns tens of billions annually by collecting and monetizing end-user data via targeted advertising and other uses. So, Google wanted Android to be free and available as open source for modification in different smartphone devices—thus the differences between a Samsung Galaxy and an old Motorola Droid—but it didn't want the universal interoperability required by Oracle for devices using Java.

Still, Google wanted programmers familiar with the widely known and used Java API to easily start making lots of new apps for Android. This would immediately create positive network effects, resulting in tons of sales of Android smartphones. More end users creating data with Android and its accompanying apps would create even more revenue for Google through its ad-based business model. Google thus copied the computer code in Java used by programmers in writing APIs, known as the "declaring code." This fact is undisputed, and Justice Breyer acknowledged it in his opinion: "Because Google wanted millions of programmers, familiar with Java, to be able easily to work with its new Android platform, it also copied roughly 11,500 lines of code from the Java SE program."[10]

Google didn't have to copy the code from Java to make an out-of-the-gate-successful device. Apple and Microsoft developed their own declaring code in their own APIs. Justice Thomas made this point in his dissenting opinion, since it is unacknowledged in Justice Breyer's majority opinion.[11] No one would think Apple

[10] Google, 141 S. Ct. at 1191.

[11] See *id.* at 1212, 1214 (Thomas, J., dissenting).

or Microsoft is hurting for innovation or commercial success. But Google didn't have to go the Apple or Microsoft route, either, and write its own API program. Google could have licensed Java. Oracle offered three separate licensing options for Java, two of which were paid-for licenses for proprietary versions of Java and a third which was a free, open-source GPL.[12] In fact, Google at first engaged in extensive licensing negotiations with Oracle to obtain permission to use Java in its new Android smartphone platform.

These negotiations were unsuccessful. Google embraced an open-source, proprietary model for Android, and thus none of Oracle's licensing options worked for Google. The open-source GPL did not work because Google wanted programs designed for Android to be proprietary to Android and not interoperable on any other device—Android is free and open source for only Android. Oracle's second licensing option permitted Google to have a proprietary API, as it could license the valuable declaring code and develop its own proprietary API program, but this license still required interoperability with any device or machine with a Java Virtual Machine. Again, Google did not want this result; Google did not want Android apps to be interoperable beyond Android. Google did not want to pay for a license for a proprietary version of the Java API that required interoperability, and the free, open-source GPL for Java was incompatible with Google's open-source and proprietary licensing model for Android. Given its plans for Android, Google was unwilling to take any of the three license options from Oracle.

Acknowledging the value in the Java declaring code known to programmers worldwide, Google copied the 11,500 lines of declaring code. It released Android in 2008, first available on an HTC device but really taking off commercially with the Motorola Droid released in 2009 (my own first smartphone). The rest is history. Android is the top-selling smartphone platform in terms of numbers of devices sold worldwide. More people worldwide have smartphones now than have access to potable water, and most of these smartphones are Android devices.

[12] Oracle v. Google, 750 F.3d 1339, 1350 (Fed. Cir. 2014) (describing the three licensing programs for Java).

But before Google made smartphone history, Oracle sued Google for patent and copyright infringement in 2010. The patent claims were later dropped, but this is how the case ended up before the United States Court of Appeals for the Federal Circuit, which is the appellate court that hears all patent appeals. The jury found for Oracle that Google infringed its copyright, but the district court ruled as a matter of law that the Java declaring code was uncopyrightable. Oracle appealed, and the Federal Circuit reversed and remanded for a trial on Google's fair use defense. Google then filed for certiorari with the Supreme Court on the copyrightability issue, but the Court denied the petition. After the second trial, the jury found that Google was immune from liability under the fair use doctrine. Oracle appealed again and won a second reversal by the Federal Circuit. Embracing the adage, "if at first you don't succeed, try, try again," Google filed for certiorari again. This time, the Court granted Google's petition on all the issues raised in the case—the copyrightability of the Java API and Google's fair use defense—as well as a third issue of the standard of review for an appeal from a district court's fair use decision.

The Copyrightability of Computer Software Programs

The first issue in *Google* was whether a software program like an API is copyrightable. This has been a longstanding dispute in copyright law and policy reaching back to when the digital revolution took off like a rocket in the 1960s and 1970s. (The Apollo program would not have been possible without computers, as the newly invented integrated circuits got us to the moon.)

Initially, courts and commentators were strongly divided on whether computer code was copyrightable. Congress resolved this debate when it enacted the Computer Software Copyright Act of 1980.[13] Coincidentally, that same year, the Supreme Court ruled in *Diamond v. Chakrabarty* that biotech innovations were patentable inventions.[14] Nineteen eighty was thus an important year in intellectual property law. That year Congress and the Court established a

[13] Pub. L. No. 96-517, § 117, 94 Stat. 3015, 3028 (1980).

[14] Diamond v. Chakrabarty, 447 U.S. 303 (1980).

stable legal foundation for the biotech and personal computer revolutions that developed in the ensuing years.[15]

The Computer Software Copyright Act is clear and straightforward: computer programs are copyrightable. The legislation very broadly defines a copyrightable computer program as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[16] The broad statutory definition in the Computer Software Copyright Act makes sense. The purpose of this legislation was to resolve definitively the split in the courts in favor of the copyrightability of computer software programs.[17]

Since Java is a computer program that represents "a set of . . . instructions to be used directly or indirectly in a computer in order to bring about a certain result," it seems to easily fall within the scope of the text of the Computer Software Copyright Act. Ergo, it is copyrightable. The statutory text does not distinguish between different types of computer programs, such as operating systems, applications, or the many programs that work below the surface that end users never directly experience. This is the meaning of "indirectly" as an adjective in the statutory definition of a computer program "used . . . to bring about a certain result." Nor does the statutory language distinguish between types of code, such as source code or object code.[18] This is the meaning of the "set of statements or instructions" in the statutory definition, the subject of the proposition that encompasses all computer programs and code.

Not so fast, argued Google. In defending its unauthorized copying of the 11,500 lines of code in the Java program, Google first argued

[15] See Adam Mossoff, A Brief History of Software Patents (and Why They're Valid), 56 Ariz. L. Rev. Syllabus 62, 74 (2014) ("It is significant that the Computer Software Copyright Act was enacted in the early 1980s because it was during this time—the late 1970s and early 1980s—that the personal computer ('PC') Revolution began."); Adam Mossoff & Kevin Madigan, Turning Gold to Lead: How Patent Eligibility Doctrine Is Undermining U.S. Leadership in Innovation, 24 Geo. Mason L. Rev. 939, 943–44 (2017) (discussing how *Chakrabarty* launched the biotech revolution).

[16] 17 U.S.C. § 101.

[17] Mossoff, A Brief History of Software Patents, *supra* note 15, at 73–74.

[18] See Comput. Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 702 (2d Cir. 1992).

that the code that it copied was not copyrightable. Google copied the declaring code, which is the code representing commands entered by the programmer to make the implementing code in Java function as an API—it is the implementing code that interfaces between apps and other programs. Thus, Google argued that the declaring code is inherently or entirely functional because it represents a "method of operation" for programmers in writing a Java API, and a "method of operation" is excluded from copyright protection under § 102(b) in the Copyright Act.[19] By the time of oral argument, Google's copyrightability argument had morphed from one of statutory exclusion into a broader merger doctrine argument that expression is uncopyrightable if it is inextricably intertwined (merged) with an idea and functionality.[20]

Google likely shifted in its argument because, as Justice Thomas pointed out in his dissent, the statutory text in the Computer Software Copyright Act is clear: "Congress rejected any categorical distinction" between types of computer code when it amended the Copyright Act to protect code that functioned "directly or indirectly" in a computer software program.[21]

Much time at oral argument was spent on the copyrightability issue, and, as a result, many commentators on both sides concluded that Google was likely going to lose.[22] During oral argument, Justice Elena Kagan said that she was "surprised or confused" by Google's

[19] 17 U.S.C. § 102(b) ("In no case does copyright protection for an original work of authorship extend to any . . . method of operation.").

[20] The merger doctrine is a longstanding doctrine in copyright law that creates an exception for the copyrightability of a written work. When an idea can only be expressed in one or a few ways in writing, the expression becomes inextricably bound up with the idea being expressed and thus cannot be protected. Protecting the expression would mean protecting the idea, which is impermissible in copyright law. See Baker v. Selden, 101 U.S. 99 (1879). In this case, the Federal Circuit held that there were numerous options available to both Google and Oracle in expressing their ideas and thus the merger doctrine did not apply. See Oracle 750 F.3d at 1358–62 (Fed. Cir. 2014).

[21] Google, 141 S. Ct. at 1213 (Thomas, J., dissenting).

[22] See, e.g., Timothy B. Lee, "Google's Supreme Court Faceoff with Oracle Was a Disaster for Google," Ars Technica, Oct. 8, 2020, https://bit.ly/3ibRHiU; Kevin Madigan, "Media Coverage of Google v. Oracle Oral Arguments Recounts Tough Day in Court for Google," Copyright Alliance, Oct. 20, 2020, https://bit.ly/3hGtvWA.

copyrightability argument.[23] Justice Neil Gorsuch agreed with her, confessing that he was "stuck in a similar place as Justice Kagan."[24] Expressions of surprise or confusion are not what a lawyer wants to hear during oral argument before the Court, and not just by one justice, but by two justices from across the jurisprudential spectrum. The general mood after oral argument was that Google was going to lose, and not just on the issue of copyrightability, but the entire case.[25]

Surprise and confusion continued to be the refrain when the Court issued its decision on April 5, 2021. Despite the concerns and skepticism expressed at oral argument, the Court soundly ruled in favor of Google and punted on the copyrightability issue, at least nominally. The *Google* Court addressed the copyrightability issue in a single sentence: "We shall assume, but purely for argument's sake, that the entire Sun Java API falls within the definition of that which can be copyrighted."[26]

This was surprising for several reasons. First, Justice Breyer's opinion for the Court was clear that its affirmation of the copyrightability of the API computer program was "purely for argument's sake." If there was any doubt about this, Justice Breyer reiterated this caveat—"if copyrightable at all"—later in the opinion, in the conclusion of his fair use analysis, holding that declaring code was far "from the core of copyright" and thus received minimal protection.[27] Yet, the Court could easily have held that the declaring code copied by Google is copyrightable, but that, after a balancing of the fair use factors, Google was still safe from liability. It is a head scratcher, to say the least, why the Court believed that it could conclude only for the sake of argument and in a single sentence an issue that consumed

[23] See Transcript of Oral Argument at 24–25, Google v. Oracle, 141 S. Ct. 1183 (2021) (No. 18-956).

[24] *Id*. at 29–30.

[25] See, e.g., Connor Hansen & Stefan Szpajda, "Google v. Oracle: What We Learned from Oral Argument," JDSupra.com, Oct. 22, 2020, https://bit.ly/2Ujjhm9 (noting that "the Justices . . . on balance[] signaled skepticism of Google's positions on each of the issues before the Court"); Eileen McDermott, "Justices Look for Reassurance That the Sky Won't Fall When They Rule in Google v. Oracle," IPWatchdog.com, Oct. 7, 2020, https://bit.ly/3hHXm0R (quoting Gene Quinn that "it seems to me that unless the Supreme Court fundamentally changes the law, Google will lose").

[26] Google, 141 S. Ct. at 1197.

[27] *Id*. at 1202.

hundreds of pages of legal briefs and tens of pages of court opinions over a decade, as well as a substantial portion of oral argument before the Court.

Second, Justice Breyer conceded the issue of the copyrightability of the declaring code for argument's sake about halfway through his opinion for the Court, which left scant time for him to address the other two issues. Yet, after detailing the nature of APIs as computer software programs, reviewing the facts, summarizing copyright policy and doctrine, and restating the two issues posed by Google in its cert petition, one thing is clear: all of the Court's legal work would be in a fair use analysis, constituting about half of the Court's entire opinion. Justice Breyer spent a surprisingly small portion of his opinion on the substantive legal analysis that produced the heart and core of the Court's decision, in much the same way that Google, in Justice Breyer's view, had copied only a relatively small portion of the Java program when viewed in the context of Java's millions of lines of code.

Finally, this conditional assumption on the copyrightability of the declaring code was surprising because, as Justice Thomas stated in his dissent, the majority "opinion . . . makes it difficult to imagine any circumstance in which declaring code will remain protected by copyright."[28] It may seem easy to dismiss Justice Thomas's statement as the standard hyperbole of dissenting opinions, in which justices are wont to identify a parade of horribles that will follow from majority opinions. But in *Google*, this statement was not hyperbole: an API program is not protectible by copyright after this decision. In the prophetic words used by Justice Antonin Scalia in one of his dissents, "this wolf comes as a wolf."[29] The fair use defense enunciated by Justice Breyer is so expansive, and the protections offered by copyright for an API program are conversely so minimal, that it is virtually impossible to think of a scenario in which an unauthorized use of an API program would not be deemed justifiable as fair use.

Indeed, Justice Thomas was not alone in reaching this conclusion when the decision came down.[30] A month later, an expert

[28] *Id.* at 1214 (Thomas, J., dissenting).

[29] Morrison v. Olson, 487 U.S. 654, 699 (1988) (Scalia, J., dissenting).

[30] See IP Watchdog, "Stakeholders Speak Out on Google v. Oracle," IPWatchdog.com, Apr. 7, 2021, https://bit.ly/3epUcwV (quoting Bob Zeidman that the "decision effectively made software uncopyrightable").

testified in the trial for Epic's antitrust lawsuit against Apple; when pressed under cross examination on the copyrightability of Apple's APIs used in its App Store, he acknowledged that "based on [*Google v. Oracle*], not all software code is protected under the IP laws."[31] How and why an API program is no longer effectively protectible under copyright is the subject of the next section.

Transforming Fair Use into a Copyrightability Doctrine[32]

Fair use doctrine did all of the heavy lifting for the Court to reach its decision in *Google*, and Justice Breyer's opinion for the Court was novel and groundbreaking in copyright law. For the first time, the Court held that a company was immune from liability when it deliberately copied without authorization a copyrighted work for a commercial purpose, not because it was engaged in parody or commentary, but because it made and sold a competing product in the marketplace. That's definitely a "whoa!" moment in copyright law.

Although constituting about half of the entire opinion, Justice Breyer packed a lot into this fair use analysis. As with my review of the background of the case and the API program at issue, I can't possibly explore here all of the nooks and crannies of this portion of the opinion. Thus, this section details some key portions of the novel structure and substantive analysis that has led to the widely recognized result that API programs are effectively no longer protectible under copyright law.

What Is Fair Use?

Fair use doctrine is a multifactor legal doctrine that commentators and scholars have long complained is rife with indeterminacy and unpredictability. Fair use doctrine is often identified as "equitable,"

[31] Dorothy Atkins, "Google v. Oracle Hangs Over Apple's IP Defense in Epic Trial," Law360, May 20, 2021, https://bit.ly/3hG7xTz (reporting on testimony by John Malackowski, chief executive and cofounder of Ocean Tomo).

[32] With apologies to Laura G. Lape, Transforming Fair Use: The Productive Use Factor in Fair Use Doctrine, 58 Albany L. Rev. 677 (1994–1995).

which Justice Breyer detailed a bit in *Google*,[33] given that its roots are found in court cases, not in legislation. The foundational case for fair use doctrine was the 1841 decision in *Folsom v. Marsh* by Justice Joseph Story, riding circuit.[34] The doctrine remained a judicial gloss on the copyright statutes until Congress codified fair use in § 107 of the 1976 Copyright Act.[35] Even so, it continues to be largely defined in substance by judicial decisions.

Section 107 lists four factors for courts to assess in determining when unauthorized copying or use is immunized from liability:

(1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;
(2) the nature of the copyrighted work;
(3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and
(4) the effect of the use upon the potential market for or value of the copyrighted work.

Although § 107 states that these factors are nonexclusive, courts have proven that codification often means ossification. Judges have generally applied only these four factors and usually in lockstep fashion in resolving defendants' claims that they are engaging in fair use.[36]

The overarching theme of the four factors, which are taken primarily from Justice Story's analysis in *Folsom*, is that the unauthorized copying and use of the work should not interfere with the commercial exploitation of the work by the copyright owner. Conversely, users of copyrighted works should be free to engage in activities or speech that are distinct from the primary markets in which

[33] Google, 141 S. Ct. at 1196 (quoting Stewart v. Abend, 495 U.S. 207, 236 (1990) that the fair use doctrine arose as an "equitable rule of reason"). See also Weissmann v. Freeman, 868 F.2d 1313, 1323 (2d Cir. 1989) ("Analysis begins not by elevating the statutory guides into inflexible rules, but with a review of the underlying equities.").

[34] See Folsom v. Marsh, 9 F. Cas. 342 (C.C.D. Mass. 1841) (No. 4,901) (Story, Circuit Justice).

[35] See Pub. L. No. 94-553, 90 Stat. 2541, 2546 (1976).

[36] Barton Beebe, An Empirical Study of U.S. Copyright Fair Use Opinions, 1978–2005, 156 U. Pa. L. Rev. 549, 561–62 (2008).

a copyrighted work is sold. Thus, for instance, a news article may report on a famous book, such as one of the Harry Potter novels, or a scholar may quote the book in an academic article or discuss its characters in teaching a class. These are all express fair uses of a copyrighted work.[37] Again, these are activities that do not produce market substitutes for the copyrighted work in the relevant market in which the work is sold and used.

Unsurprisingly, factor one and factor four have been mostly front and center in judicial analyses of fair use claims with heavy emphases on the nature of the allegedly infringing use and the impact on the current or potential market for the copyrighted work. This is also reflected in the now-dominant judicial standard for assessing factor one, which asks whether the use by the defendant is *transformative*.[38] Again, this standard makes sense from a commercialization perspective. A transformative use of a work neither serves as a market substitute nor forecloses potential commercial uses within a relevant market.

The Nature of Computer Code as an Expressive Work

As noted, courts typically begin their fair use analysis with factor one—the purpose and character of the use and whether it is commercial or not—and this is where courts address the question of whether the unauthorized use is transformative or not. But Justice Breyer began his fair use analysis with factor two—the nature of the work—and not with factor one. Although skipping over factor one is not completely unheard of in copyright law, it is very uncommon among lower courts, and the Supreme Court had not done so before in any of its prior fair use cases under § 107.

Justice Breyer invoked the "equitable," context-specific nature of fair use doctrine to justify starting with the second factor (nature of the work), and this is significant for two reasons. First, Justice Breyer argued that computer code is different from literary works like Harry Potter because computer programs "always serve

[37] In addition to the four factors, § 107 expressly states that "criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research, is not an infringement of copyright." 17 U.S.C. § 107.

[38] See Pierre N. Leval, Toward a Fair Use Standard, 103 Harv. L. Rev. 1105 (1990).

functional purposes."[39] He identified the Java API as a "user interface" that is ultimately "inextricably bound together with a general system, the division of computer tasks, that no one claims is a proper subject of copyright."[40] If you are feeling a bit of déjà vu at this moment, you are right. This was Google's method of operation/merger doctrine argument for why the declaring code was not copyrightable, which had surprised and confused both Justice Kagan and Justice Gorsuch during oral argument.[41]

For lawyers and commentators knowledgeable about these legal and technical issues, Justice Breyer confirmed this sense of déjà vu with a classic poker "tell": he cited in his factor two analysis the 1995 decision in *Lotus Development Corp. v. Borland International* by the U.S. Court of Appeals for the First Circuit.[42] In fact, he quoted from or cited *Lotus* four times in *Google*, and three of these four citations were to the concurring opinion by Judge Michael Boudin. This is odd, if only because *Lotus* was not a fair use decision.

The *Lotus* court famously held that Lotus could not copyright the pull-down (or drop-down) menu with a list of commands to select in its then-famous Lotus 1-2-3 spreadsheet program, as this was a graphical user interface that was a purely functional "method of operation" under § 102(b). In *Lotus*, Borland had replicated the pull-down menu *without* copying the computer code; it used entirely different computer code. The *Lotus* court ruled that Lotus could not extend its copyright protection over its computer code to the functional pull-down menu as a graphical interface for end users. In other words, *Lotus* is a *copyrightability decision*. Of course, this was the first issue in *Google* that the Court skipped for the sake of argument—whether Java (and APIs more generally) was uncopyrightable because the declaring code inherently merged with its programming function.[43]

[39] Google, 141 S. Ct. at 1198.

[40] *Id.* at 1201. See also *id.* at 1202 (stating that with declaring code, "unlike many other programs, its use is inherently bound together with uncopyrightable ideas (general task division and organization) and new creative expression (Android's implementing code)").

[41] See *supra* notes 17–24 and accompanying text.

[42] 49 F.3d 807 (1st Cir. 1995), aff'd by an equally divided Court, 516 U.S. 233 (1996).

[43] See *supra* notes 17–24 and accompanying text.

In referencing *Lotus*, Justice Breyer's statement that an API is a "user interface" is both telling and also a bit odd. The declaring code copied by Google, which was specifically at issue in this case, is not a user interface in the normal sense of this term. An API is not something an end user interfaces with in the way an end user interfaces with Outlook to write and send an email; rather, an API is an under-the-hood program in the guts of an operating system or app. In *Lotus* the company tried to copyright the "method of operation" in a user interface—the pull-down menu in a spreadsheet program—and it seems there was an important equivocation occurring in *Google* under factor two of the fair use doctrine between written code and user interface. This equivocation motivated the conclusion under factor two that an API is inherently bound up with its functionality such that the declaring code should receive very little copyright protection, or effectively none.

Second, Justice Breyer downplayed the creativity exercised by the programmers in selecting the declaring code when designing the Java API. He emphasized instead the inherently functional or mechanical nature of the declaring code in computer operations as an "interface." Here, Justice Breyer referenced his earlier, vivid description of an API program at the start of his opinion. Breyer had used numerous metaphors to characterize what an API program is and how it functions as a digital mechanism, allowing a person using an app on a digital device to work with that device and other devices to achieve the goals of the end-user app. He compared an API to an automobile gas pedal, the QWERTY keyboard, the Dewey Decimal system for categorizing books in a library, office file cabinets, and a programmable cooking robot.[44]

The combination of these two arguments—that an API is inherently functional and that, as a user interface, its functionality is necessarily intertwined with its unprotectible features—naturally led Justice Breyer to the conclusion that "the declaring code [in the Java API] is, if copyrightable at all, further than are most computer programs (such as the implementing code) from the core of copyright."[45] By reanimating the method of operation/merger doctrine argument

[44] Google, 141 S. Ct. at 1192–93.

[45] *Id.* at 1202.

from the copyrightability issue nominally skipped over by the Court, Justice Breyer was able to achieve under the second factor of the fair use doctrine what the Computer Software Copyright Act expressly disallowed: narrowing the scope of copyright protection for declaring code by distinguishing between types of computer programs and giving declaring code a more "thin" protection[46] than other types of computer code or expressive works deemed to be "closer to the core of copyright."[47]

This reading is a disservice to programmers and the companies that employ them to produce the innovative products and services that have driven the high-tech industry and the U.S. innovation economy for the past several decades. In each context of creative expression, whether a novel, an engraving, a map, or a computer program, the protectible creative elements are always intertwined with unprotectible facts, ideas, or functions. As Justice Thomas pointed out, even books are "inherently bound with uncopyrightable ideas—the use of chapters, having a plot, or including dialogue or footnotes. This does not place books far 'from the core of copyright.'"[48]

Programmers engage in creative labors in producing code that is often characterized as "elegant" or even "beautiful." Bad code is "cludgy." Beyond its technical sense, "cludgy" clearly conveys an artistic connotation. All engineers always aspire to find the "elegant solution" to a problem.

In denying this creativity, Justice Breyer was reaffirming a position he held 50 years ago—coding is not really a creative endeavor. In *The Uneasy Case for Copyright*, he stated that computer programs were not even as creative as the architectural drawings, photographs, or code words that were eventually brought within the protections of the copyright laws. Computer programs, he argued, "are neither of literary or artistic intent nor are they intended to convey information to another person."[49]

[46] *Id.* at 1197–98.

[47] *Id.* at 1202.

[48] *Id.* at 1215 (Thomas, J., dissenting).

[49] Breyer, *supra* note 1, at 340 n.233.

In fact, one of the reasons for the success of Java was precisely because of the selection of the declaring code by its programmers. That's what made it appealing for programmers, just as the selection of names of characters and even the descriptions of objects in a book can make it more appealing for readers—think Harry Potter. This creativity in selecting the declaring code used by programmers who would use it to create an API contributed in part to Java becoming massively popular as the internet itself began to grow at an even more explosive rate.

Java was so valuable that Google did not want to launch its Android smartphone platform without it, just like Harry Potter became so popular that its readers wanted more—movies, games, costumes, and other derivative works. This value is what copyright promotes and secures for its owners. It ensures they may profit from their productive labors by exclusively controlling the use of the work and derivative uses as well. Contrary to Justice Breyer's claim that the declaring code was far from the "core of copyright," the declaring code in Java seems exactly the type of valuable expressive work that is—and ought to be—protected by copyright. This is confirmed by the Software Copyright Protection Act, and by Google's desire to use the valuable declaring code for its own products.

The Purpose and Character of Google's Use of the Declaring Code

Google's unauthorized use of the declaring code leads to the first fair use factor—the purpose and character of the use. Given the equitable nature of fair use doctrine, Justice Breyer had the discretion to address the first factor second. Indeed, the equitable nature of fair use doctrine proved to be a lynchpin in *Google*. The equitable nature of the doctrine is what Justice Breyer invoked to give him the discretion to ultimately conclude under the second factor (analyzed first) that there was very *narrow* copyright protection for the declaring code, as opposed to other copyrighted works. Now, under the first factor, this same discretion worked in the other direction: Justice Breyer *expansively* construed the "transformative" nature of Google's copying of the Java declaring code for use in the Android smartphone platform.

Justice Breyer noted that Google's copying of the Java declaring code for use in its Android smartphone platform was clearly transformative. Notably, Justice Breyer continued to refer to the

Java declaring code as an "interface."[50] He explained, in the high-tech industry, programmers often "reimplement" a computer program interface by adopting it for "a distinct and different computing environment."[51] This is certainly correct if one is speaking about interfaces, such as the pull-down menus now ubiquitous in end-user apps using graphical user interfaces such as word processors, spreadsheets, and email clients. Justice Breyer extended this reimplementation theory to the copying of the declaring code in the Java API. He argued that, in copying the declaring code for the new smartphone platform in Android, Google was simply engaging in its own reimplementation of a computer interface, transforming it for a new computing environment (smartphones) in an innovative and creative way.

Apparently, Justice Breyer wanted to have his equity cake and eat it too. First, he invoked the equitable nature of fair use doctrine to give him the discretion to apply fair use in novel and unusual ways—both formally and substantively. Then he shunted to the side the important inquiries courts typically make when a defendant invokes an equitable doctrine to shield itself from legal liability. One such inquiry is whether the defendant acted with bad faith, or, in the case of copyright infringement, engaged in explicit piracy in deliberately copying a work for commercial gain. Here, Justice Breyer addressed these issues in just two paragraphs and deemed them to be either "not dispositive" or "not determinative" in this case.[52]

Google did not dispute that it deliberately copied 11,500 lines of Oracle's Java declaring code after a breakdown in lengthy negotiations for a license. It couldn't deny this inescapable fact. As one revealing internal email to Andy Rubin, head of the Android team at Google, stated: "What we've actually been asked to do (by Larry [Page] and Sergey [Brin]) is to investigate what technical alternatives exist to Java for Android and Chrome. We've been over a bunch of these, and think they all suck. We conclude that we need

[50] Google, 141 S. Ct. at 1203–04. Again, this is, at best, loose and confusing terminology, or, at worst, simply incorrect, as code is not the same thing as the appearance or function of a computer program.

[51] *Id.* at 1203.

[52] *Id.* at 1204.

to negotiate a license for Java under the terms we need."[53] Another email to Rubin stated: "With talks with Sun broken off where does that leave us regarding Java class libraries? Ours are half-ass at best. We need another half of an ass."[54] Desiring a better half of an ass, Google deliberately made the business decision to pirate the 11,500 lines of valuable declaring code. This was the easiest solution, given that Google could not square the circle of making Android immediately appealing to programmers while rejecting the interoperability requirement in Oracle's three license options, including even in its free, open-source GPL.

Contrary to Justice Breyer's claim that "it would have been difficult, perhaps prohibitively so, to attract programmers to build its Android smartphone system without" the Java declaring code, Google could have written its own API program without copying the Java declaring code. As noted earlier, Apple did exactly this. So did Microsoft. Alternatively, Google could have simply rethought its business model for Android as an open source, proprietary smartphone platform, as opposed to the open source, interoperable version that Oracle implemented through Java. Oracle had licensed the successful Java program through multiple commercial options for many years, contributing to the explosive growth of the interoperable internet and mobile devices. Instead, Google copied the declaring code because it was simply a cheaper, easier, and quicker route to its commercial goal of producing a successful smartphone platform—it engaged in what is now called predatory infringement (and what policy wonks call "efficient infringement").[55]

Justice Breyer neither acknowledged this piracy nor even disputed the evidence of the piracy in his equitable analysis of Google's fair use claim. This point bears highlighting: Google committed piracy. It deliberately copied the declaring code for its own commercial

[53] Exhibit C to Oracle's Opposition to Motion to Preclude Submission of Willfulness to Jury, Oracle America, Inc. v. Google Inc., Docket No. 3:10-cv-03561 (N.D. Cal., Aug. 12, 2010) (Doc. 1299-4).

[54] Exhibit L to Oracle's Opposition to Motion to Preclude Submission of Willfulness to Jury, Oracle America, Inc. v. Google Inc., Docket No. 3:10-cv-03561 (N.D. Cal., Aug. 12, 2010) (Doc. 1299-13).

[55] See Adam Mossoff & Bhamati Viswanathan, Explaining Efficient Infringement, Ctr. for Intellectual Prop. & Innovation Pol'y, May 11, 2017, https://bit.ly/3wKbhrJ.

benefit to make a commercial substitute for the original copyrighted Java computer program in its extremely successful smartphone platform. In his earlier review of the facts and procedural history of the case, Justice Breyer stated some of the facts of Google's piracy, but did not identify it as piracy. He had to acknowledge these facts, if only because they were undeniable and undisputed. "Because Google wanted millions of programmers, familiar with Java, to be able to easily work with its new Android program, it also copied roughly 11,500 lines of code from the Java SE program."[56] This decision to copy the valuable declaring code, contributing to Android's success, was immensely profitable for Google. As of 2015, Google had earned more than $42 billion from the Android smartphone platform; it has earned tens of billions more since that time.[57]

Justice Breyer addressed only obliquely and in a few sentences the longstanding recognition in fair use doctrine that the "good faith" of the defendant is a legitimate concern. In highly generalized language (declining to engage with the specific facts of the piracy, in contrast to Justice Thomas in his dissent), Justice Breyer noted that good faith was not necessarily dispositive in a fair use decision. This is certainly true as a matter of general legal doctrine; it is rare for courts to apply per se rules within an equitable doctrine.[58] Presumptions tend to be more common, and here the Court did adopt in its 1984 decision in *Sony v. Universal Studios* a presumption against fair use if the copying was done for a commercial purpose,[59] but it abandoned this presumption in its 1994 decision in *Campbell v. Acuff-Rose Music*.[60]

[56] Google, 141 S. Ct. at 1191.

[57] *Id*. at 1194.

[58] See Sony Corp. of Am. v. Universal City Studios, Inc., 464 U.S. 417, 448 n.31 (1984) (quoting from the House Report for § 107 that fair use is an "equitable rule of reason" and noting both the House and Senate followed the courts' jurisprudence in adopting § 107 and thus "eschewed a rigid, bright line approach to fair use").

[59] See *id*. at 451 (stating that "every commercial use of copyrighted material is presumptively an unfair exploitation of the monopoly privilege that belongs to the owner of the copyright").

[60] See Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 584 (1994) ("The language of [§ 107] makes clear that the commercial or nonprofit educational purpose of a work is only one element of the first factor enquiry into its purpose and character.").

Justice Breyer was correct that a commercial purpose pursued with bad faith does not automatically doom a defendant's fair use defense, but it has proven to be highly dispositive in practice. This result is unsurprising for an "equitable" doctrine that naturally involves inquiries into a litigant's state of mind or willingness to engage in strategic behavior. The question reaches back to the foundation of fair use doctrine in the 1841 *Folsom v. Marsh* decision, which was cited in *Google* (as it has been in almost all major fair use decisions).[61] In *Folsom*, Circuit Justice Story stated simply as legal truth that a copyright is "private property" and that the law secures an owner, including a commercial intermediary like a publisher, against "piracy."[62] An oft-cited, rigorous empirical study of modern fair use cases found that findings of bad faith are not common, but "in the few cases where courts explicitly found that the defendant's conduct was undertaken in bad faith, courts almost invariably found no fair use."[63] Thus, despite no per se rule or presumption against fair use for commercial use undertaken with bad faith, courts routinely find that piracy is ultimately a disqualifying factor for claiming as a matter of right that one should be immune from liability for copyright infringement under an equitable doctrine.

In his dissent, Justice Thomas pointed out (under factor four) that Google's piracy of the Java declaring code destroyed Oracle's licensing business model, which was distinct from Google's ad-based business model.[64] Accordingly, Google sought to maximize as quickly as possible market adoption of the Android smartphone platform, which ultimately brought the company billions in ad-based revenue. Google was not worried about the negative commercial implications of its copying of the declaring code for Oracle, the company that actually owned this declaring code and relied on licensing as its business model to profit from its property.

The negative impact on Oracle's licensing of Java and its revenue was immediate. After the release of Android with the pirated declaring code, licensees leveraged "the cost-free availability of Android"

[61] Google, 141 S. Ct. at 1197.

[62] Folsom, 9 F. Cas. at 345.

[63] Beebe, *supra* note 36, at 608.

[64] Google, 141 S. Ct. at 1216–17 (Thomas, J., dissenting).

to renegotiate their licenses with Oracle. For example, Amazon reduced its royalty payments to Oracle by 97.5 percent.[65] Similarly, Samsung's license with Oracle "dropped from $40 million to about $1 million."[66] Examples like these provide substantial evidence that Google's copying of the declaring code was not "transformative"—converting the copied work into a new market context with a different function or purpose—but instead was an act of deliberate copyright infringement that struck at the core of a copyright owner's ability to reap the benefits from its property.

Justice Breyer did not address these facts in his opinion; he instead balanced an oblique reference to "good faith" against what he deemed to be the innovative technological and commercial transformation in the copied code—Google "reimplemented" the "interface" of the Java declaring code for a new smartphone device. This was important innovation that a copyright owner, argued Justice Breyer, should not be permitted to prevent. Oracle was willing to license to Google, but Google was not happy with the terms of the licensing options. Oracle, he explained, should not be able to create a "lock limiting the future creativity of new programs," as this "lock would interfere with, not further, copyright's basic creativity objectives."[67] Here, he broadly defined a use of a copied work as transformative if it somehow prevents a generalized concern about stifling of innovation and commercial activities. For support, he cited a few appellate court opinions and some amicus briefs arguing that antitrust policies—concerns about "monopolization" and "anti-competitive power"—are fundamental to a fair use analysis.[68]

Much more can be said about the nature of Google's copying and the unusual nature of Justice Breyer's fair use analysis. For instance, as Justice Thomas pointed out, the expansive definition of transformative use "wrongly conflates transformative use with derivative use." The derivative right—the extension of a copyrighted work into a new commercial context, such as taking a character from a novel and converting it into a character in a movie, video

[65] *Id.* at 1216.

[66] *Id.*

[67] *Id.* at 1208.

[68] *Id.* at 1204, 1208.

game, or even in advertisements for household products—is expressly secured to copyright owners under the Copyright Act.[69] Justice Breyer neither defined nor discussed in the Court's opinion how he would distinguish between innovative transformative uses of explicit copies for a commercial purpose that qualify as a fair use and the exclusive right of a copyright owner to decide how and under what terms a work may be copied and used in new market contexts.

As with the earlier poker "tell" of the repeated citations to *Lotus* in the fair use analysis, another "tell" confirms how unusual and expansive Justice Breyer's fair use analysis is in *Google*. In the conclusion of the Court's opinion, in which he again cited *Lotus*, Justice Breyer stated, "We do not overturn or modify our earlier cases involving fair use—cases, for example, that involve 'knockoff' products, journalistic writings, and parodies."[70] This statement would not be necessary—unless of course the Court's analysis raised the concern by diverging from earlier fair use decisions. The Court's analysis raised many questions: How intertwined does a creative work need to be with unprotectible elements before it receives merely "thin" copyright protection? Does piracy now get a free pass under fair use? Is the derivative right now a similarly "thin" copyright protection in the face of a fair use claim to a transformative use? Justice Breyer's sentence about the limits of the decision—limiting it to the copying and use of the Java API by Google—says far more than just its literal meaning.

The Court's decision may ultimately be limited to the copyright protection for an API computer program. But it is an open question whether lawyers, judges, or academics will delimit their legal and policy analyses of *Google* to merely the facts of the API computer program and Google's copying of the declaring code for its Android smartphone platform. There are early indications of lawyers and judges being as creative with *Google* as *Google* itself was with the fair use doctrine that preexisted it. Given the explicit adoption into fair use doctrine of antitrust policy concerns about promoting

[69] See 17 U.S.C. § 106(2) ("[T]he owner of copyright under this title has the exclusive rights . . . to prepare derivative works based upon the copyrighted work.").

[70] Google, 141 S. Ct. at 1208.

competition and preventing monopolies stifling innovation, *Google* is already making appearances in high-tech antitrust cases, such as Epic's antitrust lawsuit against Apple.[71] In another copyright case involving a fair use defense by the estate of Andy Warhol for the late artist's unauthorized use of an image of the musician Prince in one of his iconic artworks, the Second Circuit ordered the parties to submit briefs on the impact that *Google* had on its earlier decision against the Warhol estate.[72] The Warhol estate predictably argued that *Google* "comprehensively refutes" the earlier decision that the Warhol artwork was not a fair use.[73] The decision in *Google* was surprising, but it is unsurprising to see a decision by the Court—the legal institution responsible for the final say on how legal doctrines should be interpreted and applied by all courts—now being extended beyond the narrow set of facts presented in the case.

Conclusion

Google promised to be a blockbuster case and it did not disappoint—neither with its surprising decision nor with the unusual and novel analysis employed by Justice Breyer to reach this decision. As Justice Thomas pointed out in his dissent, "we have never found fair use for copying that reaches into the tens of billions of dollars and wrecks the copyright holder's market."[74] Given the deliberate, unauthorized copying of Oracle's declaring code by Google, the unusually narrow copyright protection afforded to computer programs under fair use, and the expansive definition of "transformative use" as applied to computer programs, among other issues, I cannot imagine a scenario of unauthorized copying of an API that would not qualify as a fair use. In sum, Justice Breyer's opinion for the Court implemented a creative use of fair use doctrine to reach a result expressly prohibited by the Computer Software Protection Act of 1980: that some computer code is unprotectible under the copyright laws.

[71] See *supra* note 31, and accompanying text.

[72] Bill Donahue, "2nd Cir. Wants to Know If Google Ruling Alters Warhol Case," Law360, Apr. 29, 2021, https://bit.ly/3iODdWq.

[73] *Id.* The court ultimately ruled in August 2021 that nothing changed, despite 60 intellectual property professors filing an amicus brief supporting the Warhol estate.

[74] Google, 141 S. Ct. at 1218 (Thomas, J., dissenting).