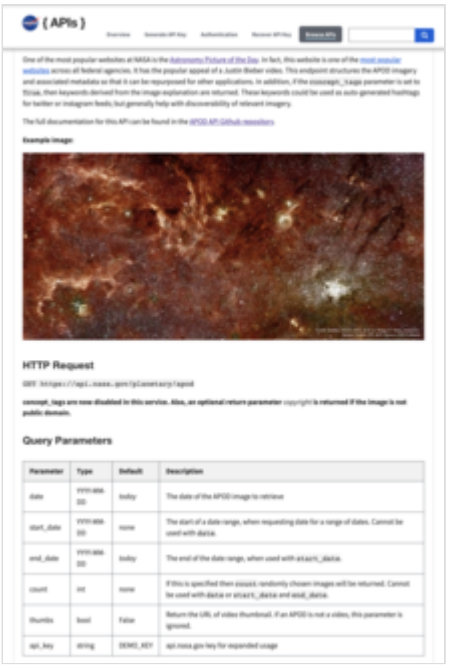
API

An **application programming interface** (**API**) is a way for two or more computer programs or components to communicate with each other. It is a type of software interface, offering a service to other pieces of software.[1] A document or standard that describes how to build or use such a connection or interface is called an *API specification*. A computer system that meets this standard is said to *implement* or *expose* an API. The term API may refer either to the specification or to the implementation. Whereas a system's user interface dictates how its end-users interact with the system in question, its API dictates how to write code that takes advantage of that system's capabilities.



Screenshot of web API documentation written by NASA demonstrating the use of APOD

In contrast to a user interface, which connects a computer to a person, an application programming interface connects computers or pieces of software to each other. It is not intended to be used directly by a person (the end user) other than a computer programmer who is incorporating it into the software. An API is often made up of different parts which act as tools or services that are available to the programmer. A program or a programmer that uses one of these parts is said to *call* that portion of the API. The calls that make up the API are also known as subroutines, methods, requests, or endpoints. An API specification *defines* these calls, meaning that it explains how to use or implement them.

One purpose of APIs is to hide the internal details of how a system works, exposing only those parts that a programmer will find useful, and keeping them consistent even if the internal details change later. An API may be custom-built for a particular pair of systems, or it may be a shared standard allowing interoperability among many systems.

There are APIs for programming languages, software libraries, computer operating systems, and computer hardware. APIs originated in the 1940s, though the term did not emerge until the 1960s and 1970s. Contemporary usage of the term API often refers to web APIs,[2] which allow communication between computers that are joined by the internet. Recent developments in APIs have led to the rise in popularity of microservices, which are loosely coupled services accessed through public APIs.[3]

APIs should be versioned. There are two common versioning strategies:[4]

- Additive change strategy: new features are added without modifying existing ones. Any update must be backward compatible. This strategy is suitable for small projects with low rate of change.

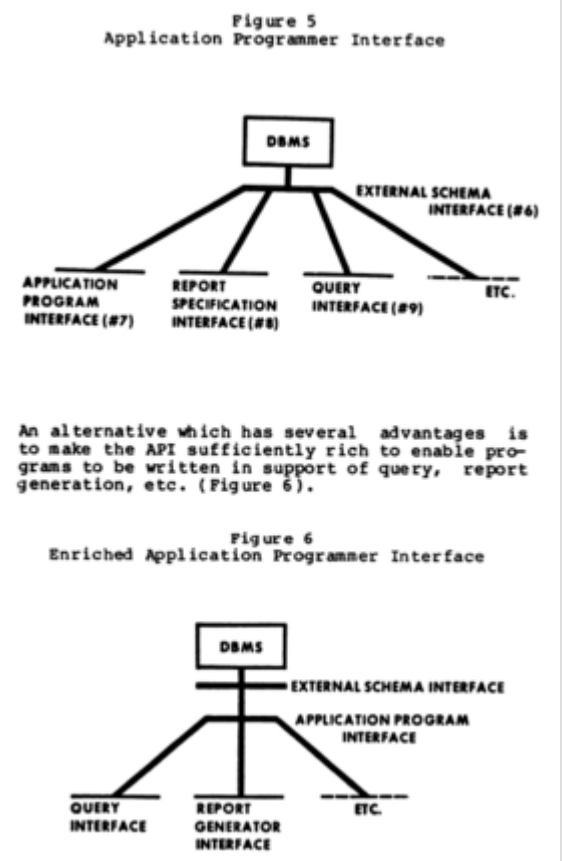
- Explicit version strategy: this strategy allows making any changes including breaking changes. This strategy is suitable for complex applications and complex changes.

Purpose

In building applications, an API simplifies programming by abstracting the underlying implementation and only exposing objects or actions the developer needs. While a graphical interface for an email client might provide a user with a button that performs all the steps for fetching and highlighting new emails, an API for file input/output might give the developer a function that copies a file from one location to another without requiring that the developer understand the file system operations occurring behind the scenes.[5]

History of the term

The term *API* initially described an interface only for end-user-facing programs, known as application programs. This origin is still reflected in the name "application programming interface." Today, the term is broader, including also utility software and even hardware interfaces.[7]



A diagram from 1978 proposing the expansion of the idea of the API to become a general programming interface, beyond application programs alone[6]

1940s and 50s

The idea of the API is much older than the term itself. British computer scientists Maurice Wilkes and David Wheeler worked on a modular software library in the 1940s for EDSAC, an early computer. The subroutines in this library were stored on punched paper tape organized in a filing cabinet. This cabinet also contained what Wilkes and Wheeler called a "library catalog" of notes about each subroutine and how to incorporate it into a program. Today, such a catalog would be called an API (or an API specification or API documentation) because it instructs a programmer on how to use (or "call") each subroutine that the programmer needs.[7]

Wilkes and Wheeler's 1951 book *The Preparation of Programs for an Electronic Digital Computer* contains the first published API specification. Joshua Bloch considers that Wilkes and Wheeler "latently invented" the API because it is more of a concept that is discovered than invented.[7]

1960s and 70s

The term "application program interface" (without an *-ing* suffix) is first recorded in a paper called *Data structures and techniques for remote computer graphics* presented at an AFIPS conference in 1968.[9][7] The authors of this paper use the term to describe the interaction of an application—a graphics program in this case—with the rest of the computer system. A consistent application interface (consisting of Fortran subroutine calls) was intended to free the programmer from dealing with idiosyncrasies of the graphics display device, and to provide hardware independence if the computer or the display were replaced.[8]

Although the people who coined the term API were implementing software on a Univac 1108, the goal of their API was to make hardware independent programs possible.[8]

The term was introduced to the field of databases by C. J. Date[10] in a 1974 paper called *The Relational and Network Approaches: Comparison of the Application Programming Interface*.[11] An API became a part of the ANSI/SPARC framework for database management systems. This framework treated the application programming interface separately from other interfaces, such as the query interface. Database professionals in the 1970s observed these different interfaces could be combined; a sufficiently rich application interface could support the other interfaces as well.[6]

This observation led to APIs that supported all types of programming, not just application programming.

1990s

By 1990, the API was defined simply as "a set of services available to a programmer for performing certain tasks" by technologist Carl Malamud.[12]

The idea of the API was expanded again with the dawn of remote procedure calls and web APIs. As computer networks became common in the 1970s and 1980s, programmers wanted to call libraries located not only on their local computers but on computers located elsewhere. These remote procedure calls were well supported by the Java language in particular. In the 1990s, with the spread of the internet, standards like CORBA, COM, and DCOM competed to become the most common way to expose API services.[13]

2000s

Roy Fielding's dissertation *Architectural Styles and the Design of Network-based Software Architectures* at UC Irvine in 2000 outlined Representational state transfer (REST) and described the idea of a "network-based Application Programming Interface" that Fielding contrasted with traditional "library-based" APIs.[14] XML and JSON web APIs saw widespread commercial adoption beginning in 2000 and continuing as of 2022. The web API is now the most common meaning of the term API.[2]

The Semantic Web proposed by Tim Berners-Lee in 2001 included "semantic APIs" that recasts the API as an open, distributed data interface rather than a software behavior interface.[15] Proprietary interfaces and agents became more widespread than open ones, but the idea of the API as a data interface took hold.

Because web APIs are widely used to exchange data of all kinds online, API has become a broad term describing much of the communication on the internet.[13] When used in this way, the term API has overlap in meaning with the term communication protocol.

Usage

Libraries and frameworks

The interface to a software library is one type of API. The API describes and prescribes the "expected behavior" (a specification) while the library is an "actual implementation" of this set of rules.

A single API can have multiple implementations (or none, being abstract) in the form of different libraries that share the same programming interface.

The separation of the API from its implementation can allow programs written in one language to use a library written in another. For example, because Scala and Java compile to compatible bytecode, Scala developers can take advantage of any Java API.[16]

API use can vary depending on the type of programming language involved. An API for a procedural language such as Lua could consist primarily of basic routines to execute code, manipulate data or handle errors while an API for an object-oriented language, such as Java, would provide a specification of classes and its class methods.[17][18] Hyrum's law states that "With a sufficient number of users of an API, it does not matter what you promise in the contract: all observable behaviors of your system will be depended on by somebody."[19] Meanwhile, several studies show that most applications that use an API tend to use a small part of the API.[20]

Language bindings are also APIs. By mapping the features and capabilities of one language to an interface implemented in another language, a language binding allows a library or service written in one language to be used when developing in another language.

Tools such as SWIG and F2PY, a Fortran-to-Python interface generator, facilitate the creation of such interfaces.[21]

An API can also be related to a software framework: a framework can be based on several libraries implementing several APIs, but unlike the normal use of an API, the access to the behavior built into the framework is mediated by extending its content with new classes plugged into the framework itself.

Moreover, the overall program flow of control can be out of the control of the caller and in the framework's hands by inversion of control or a similar mechanism.[22][23]

Operating systems

An API can specify the interface between an application and the operating system.[24] POSIX, for example, provides a set of common API specifications that aim to enable an application written for a POSIX conformant operating system to be compiled for another POSIX conformant operating system.

Linux and Berkeley Software Distribution are examples of operating systems that implement the POSIX APIs.[25]

Microsoft has shown a strong commitment to a backward-compatible API, particularly within its Windows API (Win32) library, so older applications may run on newer versions of Windows using an executable-specific setting called "Compatibility Mode".[26]

An API differs from an application binary interface (ABI) in that an API is source code based while an ABI is binary based. For instance, POSIX provides APIs while the Linux Standard Base provides an ABI.[27][28]

Remote APIs

Remote APIs allow developers to manipulate remote resources through protocols, specific standards for communication that allow different technologies to work together, regardless of language or platform. For example, the Java Database Connectivity API allows developers to query many different types of databases with the same set of functions, while the Java remote method invocation API uses the Java Remote Method Protocol to allow invocation of functions that operate remotely but appear local to the developer.[29][30]

Therefore, remote APIs are useful in maintaining the object abstraction in object-oriented programming; a method call, executed locally on a proxy object, invokes the corresponding method on the remote object, using the remoting protocol, and acquires the result to be used locally as a return value.

A modification of the proxy object will also result in a corresponding modification of the remote object.[31]

Web APIs

Web APIs are a service accessed from client devices (mobile phones, laptops, etc.) to a web server using the Hypertext Transfer Protocol (HTTP). Client devices send a request in the form of an HTTP request, and are met with a response message usually in JavaScript Object Notation (JSON) or Extensible Markup Language (XML) format. Developers typically use Web APIs to query a server for a specific set of data from that server.

An example might be a shipping company API that can be added to an eCommerce-focused website to facilitate ordering shipping services and automatically include current shipping rates, without the site developer having to enter the shipper's rate table into a web database. While "web API" historically has been virtually synonymous with web service, the recent trend (so-called Web 2.0) has been moving away from Simple Object Access Protocol (SOAP) based web services and service-oriented architecture (SOA) towards more direct representational state transfer (REST) style web resources and resource-oriented architecture (ROA).[32] Part of this trend is related to the Semantic Web movement toward Resource Description Framework (RDF), a concept to promote web-based ontology engineering technologies. Web APIs allow the combination of multiple APIs into new applications known as mashups.[33]

In the social media space, web APIs have allowed web communities to facilitate sharing content and data between communities and applications. In this way, content that is created in one place dynamically can be posted and updated to multiple locations on the web.[34] For example, Twitter's REST API allows developers to access core Twitter data and the Search API provides methods for developers to interact with Twitter Search and trends data.[35]

Design

The design of an API has a significant impact on its usage.[5] First of all, the design of programming interfaces represents an important part of software architecture, the organization of a complex piece of software.[36] The principle of information hiding describes the role of programming interfaces as enabling modular programming by hiding the implementation details of the modules so that users of modules need not understand the complexities inside the modules.[37] Aside from the previous underlying principle, other metrics for measuring the usability of an API may include properties such as functional efficiency, overall correctness, and learnability for novices.[38] One straightforward and commonly adopted way of designing APIs is to follow Nielsen's heuristic evaluation guidelines. The Factory method pattern is also typical in designing APIs due to their reusable nature.[39] Thus, the design of an API attempts to provide only the tools a user would expect.[5]

Synchronous versus asynchronous

An application programming interface can be synchronous or asynchronous. A synchronous API call is a design pattern where the call site is blocked while waiting for the called code to finish.[40] With an asynchronous API call, however, the call site is not blocked while waiting for the called code to finish, and instead the calling thread is notified when the reply arrives.

Security

API security is very critical when developing a public facing API. Common threats include SQL injection, Denial-of-service attack (DoS), broken authentication, and exposing sensitive data.[41] Without ensuring proper security practices, bad actors can get access to information they should not have or even gain privileges to make changes to your server. Some common security practices include proper connection security using HTTPS, content security to mitigate data injection attacks, and requiring an API key to use your service.[42] Many public facing API services require you to use an assigned API key, and will refuse to serve data without sending the key with your request.[43]

Release policies

APIs are one of the more common ways technology companies integrate. Those that provide and use APIs are considered as being members of a business ecosystem.[44]

The main policies for releasing an API are:[45]

- Private: The API is for internal company use only.

- Partner: Only specific business partners can use the API. For example, vehicle for hire companies such as Uber and Lyft allow approved third-party developers to directly order rides from within their apps. This allows the companies to exercise quality control by curating which apps have access to the API and provides them with an additional revenue stream.[46]
- Public: The API is available for use by the public. For example, Microsoft makes the Windows API public, and Apple releases its API Cocoa so that software can be written for their platforms. Not all public APIs are generally accessible by everybody. For example, Internet service providers like Cloudflare or Voxility, use RESTful APIs to allow customers and resellers access to their infrastructure information, DDoS stats, network performance, or dashboard controls.[47] Access to such APIs is granted either by "API tokens", or customer status validations.[48]

Public API implications

An important factor when an API becomes public is its "interface stability". Changes to the API—for example adding new parameters to a function call—could break compatibility with the clients that depend on that API.[49]

When parts of a publicly presented API are subject to change and thus not stable, such parts of a particular API should be documented explicitly as "unstable". For example, in the Google Guava library, the parts that are considered unstable, and that might change soon, are marked with the Java annotation `@Beta`.[50]

A public API can sometimes declare parts of itself as *deprecated* or rescinded. This usually means that part of the API should be considered a candidate for being removed, or modified in a backward incompatible way. Therefore, these changes allow developers to transition away from parts of the API that will be removed or not supported in the future.[51]

On February 19, 2020, Akamai published their annual "State of the Internet" report, showcasing the growing trend of cybercriminals targeting public API platforms at financial services worldwide. From December 2017 through November 2019, Akamai witnessed 85.42 billion credential violation attacks. About 20%, or 16.55 billion, were against hostnames defined as API endpoints. Of these, 473.5 million have targeted financial services sector organizations.[52]

Documentation

API documentation describes the services an API offers and how to use those services, aiming to cover everything a client would need to know for practical purposes.

Documentation is crucial for the development and maintenance of applications using the API.[53] API documentation is traditionally found in documentation files but can also be found in social media such as blogs, forums, and Q&A websites.[54]

Traditional documentation files are often presented via a documentation system, such as Javadoc or Pydoc, that has a consistent appearance and structure. However, the types of content included in the documentation differ from API to API.[55]

In the interest of clarity, API documentation may include a description of classes and methods in the API as well as "typical usage scenarios, code snippets, design rationales, performance discussions, and contracts", but implementation details of the API services themselves are usually omitted.

Reference documentation for a REST API can be generated automatically from an OpenAPI document, which is a machine-readable text file that uses a prescribed format and syntax defined in the OpenAPI Specification. The OpenAPI document defines basic information such as the API's name and description, as well as describing operations the API provides access to.[56]

API documentation can be enriched with metadata information like Java annotations. This metadata can be used by the compiler, tools, and by the *run-time* environment to implement custom behaviors or custom handling.[57]

Dispute over copyright protection for APIs

In 2010, Oracle Corporation sued Google for having distributed a new implementation of Java embedded in the Android operating system.[58] Google had not acquired any permission to reproduce the Java API, although permission had been given to the similar OpenJDK project. Google had approached Oracle to negotiate a license for their API, but were turned down due to trust issues. Despite the disagreement, Google chose to use Oracle's code anyway. Judge William Alsup ruled in the *Oracle v. Google* case that APIs cannot be copyrighted in the U.S and that a victory for Oracle would have widely expanded copyright protection to a "functional set of symbols" and allowed the copyrighting of simple software commands:

> To accept Oracle's claim would be to allow anyone to copyright one version of code to carry out a system of commands and thereby bar all others from writing its different versions to carry out all or part of the same commands.[59][60]

Alsup's ruling was overturned in 2014 on appeal to the Court of Appeals for the Federal Circuit, though the question of whether such use of APIs constitutes fair use was left unresolved.[61][62]

In 2016, following a two-week trial, a jury determined that Google's reimplementation of the Java API constituted fair use, but Oracle vowed to appeal the decision.[63] Oracle won on its appeal, with the Court of Appeals for the Federal Circuit ruling that Google's use of the APIs did not qualify for fair use.[64] In 2019, Google appealed to the Supreme Court of the United States over both the copyrightability and fair use rulings, and the Supreme Court granted review.[65] Due to the COVID-19 pandemic, the oral hearings in the case were delayed until October 2020.[66]

The case was decided by the Supreme Court in Google's favor with a ruling of 6–2. Justice Stephen Breyer delivered the opinion of the court and at one point mentioned that "The declaring code is, if copyrightable at all, further than are most computer programs from the core of copyright." This means the code used in APIs are more similar to dictionaries than novels in terms of copyright protection.[67]

Examples

- ASPI for SCSI device interfacing

- Cocoa and Carbon for the Macintosh
- DirectX for Microsoft Windows
- Ark Engine for HarmonyOS
- EHLLAPI
- Java APIs
- ODBC for Microsoft Windows
- OpenAL cross-platform sound API
- OpenCL cross-platform API for general-purpose computing for CPUs & GPUs
- OpenGL cross-platform graphics API
- OpenMP API that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran on many architectures, including Unix and Microsoft Windows platforms.
- Server application programming interface (SAPI)
- Simple DirectMedia Layer (SDL)

See also

- API testing
- API writer
- Augmented web
- Calling convention
- Common Object Request Broker Architecture (CORBA)
- Comparison of application virtual machines
- Document Object Model (DOM)
- Double-chance function
- Foreign function interface
- Front and back ends
- Interface (computing)
- Interface control document
- List of 3D graphics APIs
- Microservices
- Name mangling
- Open API
- Open Service Interface Definitions
- Parsing
- Plugin
- RAML (software)
- Software development kit (SDK)
- Structured Financial Messaging System
- Web API
- Web content vendor
- XPCOM

References

1. Reddy, Martin (2011). *API Design for C++* (https://books.google.com/books?id=IY29LylT85wC). Elsevier Science. p. 1. ISBN 9780123850041. Archived (https://web.archive.org/web/20230415001843/https://books.google.com/books?id=IY29LylT85wC) from the original on 2023-04-15. Retrieved 2023-03-21.
2. Lane, Kin (October 10, 2019). "Intro to APIs: History of APIs" (https://blog.postman.com/intro-to-apis-history-of-apis/). *Postman*. Archived (https://web.archive.org/web/20200911053834/https://blog.postman.com/intro-to-apis-history-of-apis) from the original on September 11, 2020. Retrieved September 18, 2020. "When you hear the acronym "API" or its expanded version "Application Programming Interface", it is almost always in reference to our modern approach, in that we use HTTP to provide access to machine readable data in a JSON or XML format, often simply referred to as "web APIs." APIs have been around almost as long as computing, but modern web APIs began taking shape in the early 2000s."

3. Wood, Laura (2021-08-25). "Global Cloud Microservices Market (2021 to 2026)" (https://www.businesswire.com/news/home/20210825005630/en/Global-Cloud-Microservices-Market-2021-to-2026---Growth-Trends-COVID-19-Impact-and-Forecasts---ResearchAndMarkets.com). *businesswire.com*. Archived (https://web.archive.org/web/20220408091236/https://www.businesswire.com/news/home/20210825005630/en/Global-Cloud-Microservices-Market-2021-to-2026---Growth-Trends-COVID-19-Impact-and-Forecasts---ResearchAndMarkets.com) from the original on 2022-04-08. Retrieved 2022-03-29.
4. *Designing Web APIs Building APIs That Developers Love*. O'Reilly Media. 2018. ISBN 9781492026877.
5. Clarke, Steven (2004). "Measuring API Usability" (http://www.drdobbs.com/windows/measuring-api-usability/184405654). *Dr. Dobb's*. Archived (https://web.archive.org/web/20220303235859/http://www.drdobbs.com/windows/measuring-api-usability/184405654) from the original on 3 March 2022. Retrieved 29 July 2016.
6. Database architectures – a feasibility workshop (https://hdl.handle.net/2027/mdp.39015077587742?urlappend=%3Bseq=53) (Report). Washington, DC: U.S. Department of Commerce, National Bureau of Standards. April 1981. pp. 45–47. hdl:2027/mdp.39015077587742 (https://hdl.handle.net/2027%2Fmdp.39015077587742?urlappend=%3Bseq=53). LCCN 81600004 (https://lccn.loc.gov/81600004). NBS special publication 500-76. Retrieved September 18, 2020.
7. Bloch, Joshua (August 8, 2018). *A Brief, Opinionated History of the API* (https://www.infoq.com/presentations/history-api/) (Speech). QCon. San Francisco: InfoQ. Archived (https://web.archive.org/web/20200922200610/https://www.infoq.com/presentations/history-api/) from the original on September 22, 2020. Retrieved September 18, 2020.
8. Cotton, Ira W.; Greatorex, Frank S. (December 1968). "Data structures and techniques for remote computer graphics" (https://www.computer.org/csdl/pds/api/csdl/proceedings/download-article/12OmNyRPgFZ/pdf). *AFIPS '68: Proceedings of the December 9–11, 1968, Fall Joint Computer Conference*. AFIPS 1968 Fall Joint Computer Conference. Vol. I. San Francisco, California: Association for Computing Machinery. pp. 533–544. doi:10.1145/1476589.1476661 (https://doi.org/10.1145%2F1476589.1476661). ISBN 978-1450378994. OCLC 1175621908 (https://www.worldcat.org/oclc/1175621908). Archived (https://web.archive.org/web/20201020123943/https://www.computer.org/csdl/pds/api/csdl/proceedings/download-article/12OmNyRPgFZ/pdf) from the original on 2020-10-20. Retrieved 2020-09-19.
9. "application program interface" (https://www.oed.com/search/dictionary/?q=application+program+interface). *Oxford English Dictionary* (Online ed.). Oxford University Press. (Subscription or participating institution membership (https://www.oed.com/public/login/loggingin#withyourlibrary) required.)
10. Date, C. J. (2019). *E. F. Codd and Relational Theory: A Detailed Review and Analysis of Codd's Major Database Writings* (https://books.google.com/books?id=2Sy4DwAAQBAJ&pg=PA135). Lulu.com. p. 135. ISBN 978-1684705276.
11. Date, C. J.; Codd, E. F. (January 1975). "The relational and network approaches: Comparison of the application programming interfaces". In Randall Rustin (ed.). *Proceedings of 1974 ACM-SIGMOD Workshop on Data Description, Access and Control*. SIGMOD Workshop 1974. Vol. 2. Ann Arbor, Michigan: Association for Computing Machinery. pp. 83–113. doi:10.1145/800297.811532 (https://doi.org/10.1145%2F800297.811532). ISBN 978-1450374187. OCLC 1175623233 (https://www.worldcat.org/oclc/1175623233).
12. Carl, Malamud (1990). *Analyzing Novell Networks* (https://babel.hathitrust.org/cgi/pt?id=mdp.39015018454903&seq=314). Van Nostrand Reinhold. p. 294. ISBN 978-0442003647. Archived (https://web.archive.org/web/20210126190257/https://babel.hathitrust.org/cgi/pt?id=mdp.39015018454903&seq=314) from the original on 2021-01-26. Retrieved 2020-09-19.

13. Jin, Brenda; Sahni, Saurabh; Shevat, Amir (2018). *Designing Web APIs* (https://books.google.com/books?id=Dg1rDwAAQBAJ). O'Reilly Media. ISBN 9781492026877. Archived (https://web.archive.org/web/20230410040816/https://books.google.com/books?id=Dg1rDwAAQBAJ) from the original on 2023-04-10. Retrieved 2023-03-21.
14. Fielding, Roy (2000). *Architectural Styles and the Design of Network-based Software Architectures* (https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm) (PhD). University of California, Irvine. Archived (https://web.archive.org/web/20200122200214/https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm) from the original on January 22, 2020. Retrieved September 18, 2020.
15. Dotsika, Fefie (August 2010). "Semantic APIs: Scaling up towards the Semantic Web". *International Journal of Information Management*. **30** (4): 335–342. doi:10.1016/j.ijinfomgt.2009.12.003 (https://doi.org/10.1016%2Fj.ijinfomgt.2009.12.003).
16. Odersky, Martin; Spoon, Lex; Venners, Bill (10 December 2008). "Combining Scala and Java" (http://www.artima.com/pins1ed/combining-scala-and-java.html). *artima.com*. Archived (https://web.archive.org/web/20160808030830/http://www.artima.com/pins1ed/combining-scala-and-java.html) from the original on 8 August 2016. Retrieved 29 July 2016.
17. de Figueiredo, Luiz Henrique; Ierusalimschy, Roberto; Filho, Waldemar Celes (1994). "The design and implementation of a language for extending applications" (https://www.researchgate.net/publication/2778436). *TeCGraf Grupo de Tecnologia Em Computacao Grafica*: 273–284. CiteSeerX 10.1.1.47.5194 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.5194). S2CID 59833827 (https://api.semanticscholar.org/CorpusID:59833827). Retrieved 29 July 2016.
18. Sintes, Tony (13 July 2001). "Just what is the Java API anyway?" (https://www.infoworld.com/article/2077392/just-what-is-the-java-api-anyway.html). *JavaWorld*. Archived (https://web.archive.org/web/20201019213926/https://www.infoworld.com/article/2077392/just-what-is-the-java-api-anyway.html) from the original on 2020-10-19. Retrieved 2020-07-18.
19. Winters, Titus; Tom Manshreck; Hyrum Wright, eds. (2020). *Software engineering at Google: lessons learned from programming over time*. Sebastopol, CA: O'Reilly Media. ISBN 9781492082798. OCLC 1144086840 (https://www.worldcat.org/oclc/1144086840).
20. Mastrangelo, Luis; Ponzanelli, Luca; Mocci, Andrea; Lanza, Michele; Hauswirth, Matthias; Nystrom, Nathaniel (2015-10-23). "Use at your own risk: the Java unsafe API in the wild". *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*. OOPSLA 2015. New York, NY, US: Association for Computing Machinery. pp. 695–710. doi:10.1145/2814270.2814313 (https://doi.org/10.1145%2F2814270.2814313). ISBN 978-1-4503-3689-5.
21. "F2PY.org" (http://www.f2py.org/). F2PY.org. Archived (https://web.archive.org/web/20110704234152/http://www.f2py.org/) from the original on 2011-07-04. Retrieved 2011-12-18.
22. Fowler, Martin. "Inversion Of Control" (http://martinfowler.com/bliki/InversionOfControl.html). Archived (https://web.archive.org/web/20110123051630/http://martinfowler.com/bliki/InversionOfControl.html) from the original on 2011-01-23. Retrieved 2011-08-25.
23. Fayad, Mohamed. "Object-Oriented Application Frameworks" (http://www.dre.vanderbilt.edu/~schmidt/CACM-frameworks.html). Archived (https://web.archive.org/web/20131105141605/http://www.dre.vanderbilt.edu/~schmidt/CACM-frameworks.html) from the original on 2013-11-05. Retrieved 2013-11-05.
24. Lewine, Donald A. (1991). *POSIX Programmer's Guide* (http://shop.oreilly.com/product/9780937175736.do). O'Reilly & Associates, Inc. p. 1. ISBN 9780937175736. Archived (https://web.archive.org/web/20160822175942/http://shop.oreilly.com/product/9780937175736.do) from the original on 22 August 2016. Retrieved 2 August 2016.

25. West, Joel; Dedrick, Jason (2001). "Open source standardization: the rise of Linux in the network era" (http://www.joelwest.org/Papers/WestDedrick2001b.pdf) (PDF). *Knowledge, Technology & Policy*. **14** (2): 88–112. doi:10.1007/PL00022278 (https://doi.org/10.1007%2FPL00022278). S2CID 46082812 (https://api.semanticscholar.org/CorpusID:46082812). Archived (https://web.archive.org/web/20160827180926/http://www.joelwest.org/Papers/WestDedrick2001b.pdf) (PDF) from the original on 27 August 2016. Retrieved 2 August 2016.
26. Microsoft (October 2001). "Support for Windows XP" (https://web.archive.org/web/20090926235439/http://www.microsoft.com/windowsxp/using/helpandsupport/learnmore/appcompat.mspx). Microsoft. p. 4. Archived from the original (http://www.microsoft.com/windowsxp/using/helpandsupport/learnmore/appcompat.mspx) on 2009-09-26.
27. "LSB Introduction" (https://web.archive.org/web/20150402094250/http://www.linuxfoundation.org/collaborate/workgroups/lsb/lsb-introduction). Linux Foundation. 21 June 2012. Archived from the original (http://www.linuxfoundation.org/collaborate/workgroups/lsb/lsb-introduction) on 2015-04-02. Retrieved 2015-03-27.
28. Stoughton, Nick (April 2005). "Update on Standards" (https://db.usenix.org/publications/login/2005-04/openpdfs/standards2004.pdf) (PDF). USENIX. Archived (https://web.archive.org/web/20090327123725/https://db.usenix.org/publications/login/2005-04/openpdfs/standards2004.pdf) (PDF) from the original on 2009-03-27. Retrieved 2009-06-04.
29. Bierhoff, Kevin (23 April 2009). *API Protocol Compliance in Object-Oriented Software* (https://www.cs.cmu.edu/~kbierhof/thesis/bierhoff-thesis.pdf) (PDF) (PhD). Carnegie Mellon University. ISBN 978-1-109-31660-5. ProQuest 304864018 (https://search.proquest.com/docview/304864018). Archived (https://web.archive.org/web/20161011151916/https://www.cs.cmu.edu/~kbierhof/thesis/bierhoff-thesis.pdf) (PDF) from the original on 11 October 2016. Retrieved 29 July 2016.
30. Wilson, M. Jeff (10 November 2000). "Get smart with proxies and RMI" (https://www.infoworld.com/article/2076234/get-smart-with-proxies-and-rmi.html). *JavaWorld*. Archived (https://web.archive.org/web/20200720092311/https://www.infoworld.com/article/2076234/get-smart-with-proxies-and-rmi.html) from the original on 2020-07-20. Retrieved 2020-07-18.
31. Henning, Michi; Vinoski, Steve (1999). *Advanced CORBA Programming with C++* (https://archive.org/details/advancedcorbapro00henn). Addison-Wesley. ISBN 978-0201379273. Retrieved 16 June 2015.
32. Benslimane, Djamal; Schahram Dustdar; Amit Sheth (2008). "Services Mashups: The New Generation of Web Applications" (https://corescholar.libraries.wright.edu/cgi/viewcontent.cgi?article=2126&context=knoesis). *IEEE Internet Computing, vol. 12, no. 5*. Institute of Electrical and Electronics Engineers. pp. 13–15. Archived (https://web.archive.org/web/20231007175339/https://corescholar.libraries.wright.edu/cgi/viewcontent.cgi?article=2126&context=knoesis) from the original on 2023-10-07. Retrieved 2019-10-01.
33. Niccolai, James (2008-04-23), "So What Is an Enterprise Mashup, Anyway?" (https://web.archive.org/web/20171010045104/https://www.pcworld.com/article/145039/article.html), *PC World*, archived from the original (https://www.pcworld.com/article/145039/so_what_is_an_enterprise_mashup_anyway.html) on Oct 10, 2017
34. Parr, Ben (21 May 2009). "The Evolution of the Social Media API" (https://web.archive.org/web/20160811142540/http://mashable.com/2009/05/21/social-media-api/#9Yqe0USxKGqn). *Mashable*. Archived from the original (http://mashable.com/2009/05/21/social-media-api/) on Aug 11, 2016. Retrieved 26 July 2016.
35. "GET trends/place" (https://developer.twitter.com/en/docs/trends/trends-for-location/api-reference/get-trends-place). *Twitter Developer Platform*. Archived (https://web.archive.org/web/20200617023554/https://developer.twitter.com/en/docs/trends/trends-for-location/api-reference/get-trends-place) from the original on 2020-06-17. Retrieved 2020-04-30.

36. Garlan, David; Shaw, Mary (January 1994). "An Introduction to Software Architecture" (https://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf) (PDF). *Advances in Software Engineering and Knowledge Engineering*. **1**. Archived (https://web.archive.org/web/20210506203152/http://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf) (PDF) from the original on 6 May 2021. Retrieved 8 August 2016 – via CMU School of Computer Science.
37. Parnas, D.L. (1972). "On the Criteria To Be Used in Decomposing Systems into Modules" (https://doi.org/10.1145%2F361598.361623). *Communications of the ACM*. **15** (12): 1053–1058. doi:10.1145/361598.361623 (https://doi.org/10.1145%2F361598.361623). S2CID 53856438 (https://api.semanticscholar.org/CorpusID:53856438).
38. Myers, Brad A.; Stylos, Jeffrey (2016). "Improving API usability" (https://doi.org/10.1145%2F2896587). *Communications of the ACM*. **59** (6): 62–69. doi:10.1145/2896587 (https://doi.org/10.1145%2F2896587). S2CID 543853 (https://api.semanticscholar.org/CorpusID:543853).
39. Brian Ellis, Jeffrey Stylos, and Brad Myers. 2007. "The Factory Pattern in API Design: A Usability Evaluation (http://www.cs.cmu.edu/~NatProg/papers/Ellis2007FactoryUsability.pdf) Archived (https://web.archive.org/web/20220321043301/http://www.cs.cmu.edu/~NatProg/papers/Ellis2007FactoryUsability.pdf) 2022-03-21 at the Wayback Machine". In *Proceedings of the 29th international conference on Software Engineering* (*ICSE '07*). IEEE Computer Society, USA, 302–312. doi:10.1109/ICSE.2007.85 (https://doi.org/10.1109%2FICSE.2007.85).
40. "Synchronous vs. Asynchronous Writes - Packaged Contact Center Enterprise" - Cisco DevNet (https://developer.cisco.com/docs/packaged-contact-center/#!synchronous-vs-asynchronous-writes) Archived (https://web.archive.org/web/20220803065429/https://developer.cisco.com/docs/packaged-contact-center/#!synchronous-vs-asynchronous-writes) 2022-08-03 at the Wayback Machine.
41. Silva, Paulo (2019). "Global Cloud Microservices Market (2021 to 2026)" (https://owasp.org/www-project-api-security/). Archived (https://web.archive.org/web/20200218212644/https://owasp.org/www-project-api-security/) from the original on 2020-02-18. Retrieved 2022-03-29.
42. "Web Security" (https://developer.mozilla.org/en-US/docs/Web/Security). 2022-02-18. Archived (https://web.archive.org/web/20220402211512/https://developer.mozilla.org/en-US/docs/Web/Security) from the original on 2022-04-02. Retrieved 2022-03-29.
43. "API Keys – What Is an API Key? | APILayer Blog" (https://blog.apilayer.com/api-keys-what-is-an-api-key/). 2022-03-01. Archived (https://web.archive.org/web/20220516143559/https://blog.apilayer.com/api-keys-what-is-an-api-key/) from the original on 2022-05-16. Retrieved 2022-07-15.
44. de Ternay, Guerric (Oct 10, 2015). "Business Ecosystem: Creating an Economic Moat" (https://web.archive.org/web/20160917121456/https://boostcompanies.com/business-ecosystem/). *BoostCompanies*. Archived from the original (http://boostcompanies.com/business-ecosystem) on 2016-09-17. Retrieved 2016-02-01.
45. Boyd, Mark (2014-02-21). "Private, Partner or Public: Which API Strategy Is Best for Business?" (http://www.programmableweb.com/news/private-partner-or-public-which-api-strategy-best-business/2014/02/21). *ProgrammableWeb*. Archived (https://web.archive.org/web/20160718172056/http://www.programmableweb.com/news/private-partner-or-public-which-api-strategy-best-business/2014/02/21) from the original on 2016-07-18. Retrieved 2 August 2016.
46. Weissbrot, Alison (7 July 2016). "Car Service APIs Are Everywhere, But What's In It For Partner Apps?" (https://www.adexchanger.com/mobile/car-service-apis-everywhere-whats-partner-apps/). *AdExchanger*. Archived (https://web.archive.org/web/20200728095644/https://www.adexchanger.com/mobile/car-service-apis-everywhere-whats-partner-apps/) from the original on 28 July 2020. Retrieved 14 August 2020.

47. "Cloudflare API v4 Documentation" (https://api.cloudflare.com/). *cloudflare*. 25 February 2020. Archived (https://web.archive.org/web/20200226084431/https://api.cloudflare.com/) from the original on 26 February 2020. Retrieved 27 February 2020.
48. Liew, Zell (17 January 2018). "Car Service APIs Are Everywhere, But What's In It For Partner Apps" (https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/). *Smashing Magazine*. Archived (https://web.archive.org/web/20200221101916/https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/) from the original on 21 February 2020. Retrieved 27 February 2020.
49. Shi, Lin; Zhong, Hao; Xie, Tao; Li, Mingshu (2011). "An Empirical Study on Evolution of API Documentation". *Fundamental Approaches to Software Engineering* (https://www.researchgate.net/publication/225147411). International Conference on Fundamental Approaches to Software Engineering. Lecture Notes in Computer Science. Vol. 6603. pp. 416–431. doi:10.1007/978-3-642-19811-3_29 (https://doi.org/10.1007%2F978-3-642-19811-3_29). ISBN 978-3-642-19810-6. Retrieved 22 July 2016.
50. "guava-libraries – Guava: Google Core Libraries for Java 1.6+" (https://web.archive.org/web/20140326101728/https://code.google.com/p/guava-libraries/). *Google Project Hosting*. 2014-02-04. Archived from the original (https://code.google.com/p/guava-libraries/) on Mar 26, 2014. Retrieved 2014-02-11.
51. Oracle. "How and When to Deprecate APIs" (http://docs.oracle.com/javase/7/docs/technotes/guides/javadoc/deprecation/deprecation.html). *Java SE Documentation*. Archived (https://web.archive.org/web/20160409212914/http://docs.oracle.com/javase/7/docs/technotes/guides/javadoc/deprecation/deprecation.html) from the original on 9 April 2016. Retrieved 2 August 2016.
52. Takanashi, Dean (19 February 2020). "Akamai: Cybercriminals are attacking APIs at financial services firms" (https://venturebeat.com/2020/02/19/akamai-cybercriminals-are-attacking-apis-at-financial-services-firms/). *Venture Beat*. Archived (https://web.archive.org/web/20200227153607/https://venturebeat.com/2020/02/19/akamai-cybercriminals-are-attacking-apis-at-financial-services-firms/) from the original on 27 February 2020. Retrieved 27 February 2020.
53. Dekel, Uri; Herbsleb, James D. (May 2009). "Improving API Documentation Usability with Knowledge Pushing". *Institute for Software Research, School of Computer Science*. CiteSeerX 10.1.1.446.4214 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.446.4214).
54. Parnin, Chris; Treude, Cristoph (May 2011). "Measuring API documentation on the web". *Web2SE '11: Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering*. pp. 25–30. doi:10.1145/1984701.1984706 (https://doi.org/10.1145%2F1984701.1984706). ISBN 9781450305952. S2CID 17751901 (https://api.semanticscholar.org/CorpusID:17751901).
55. Maalej, Waleed; Robillard, Martin P. (April 2012). "Patterns of Knowledge in API Reference Documentation" (https://mobis.informatik.uni-hamburg.de/wp-content/uploads/2013/03/TSE-2012-04-0081.R2_Maalej.pdf) (PDF). *IEEE Transactions on Software Engineering*. Archived (https://web.archive.org/web/20160822193338/https://mobis.informatik.uni-hamburg.de/wp-content/uploads/2013/03/TSE-2012-04-0081.R2_Maalej.pdf) (PDF) from the original on 22 August 2016. Retrieved 22 July 2016.
56. "Structure of an OpenAPI Document" (https://web.archive.org/web/20221106000735/https://oai.github.io/Documentation/specification-structure.html). *OpenAPI Documentation*. Archived from the original (https://oai.github.io/Documentation/specification-structure.html) on 2022-11-06. Retrieved 2022-11-06.

57. "Annotations" (https://web.archive.org/web/20110925021948/http://download.oracle.com/javase/1,5.0/docs/guide/language/annotations.html). Sun Microsystems. Archived from the original (http://download.oracle.com/javase/1,5.0/docs/guide/language/annotations.html) on 2011-09-25. Retrieved 2011-09-30..
58. "Oracle and the End of Programming As We Know It" (http://www.drdobbs.com/jvm/232901227). DrDobbs. 2012-05-01. Archived (https://web.archive.org/web/20120509064446/http://www.drdobbs.com/jvm/232901227) from the original on 2012-05-09. Retrieved 2012-05-09.
59. "APIs Can't be Copyrighted Says Judge in Oracle Case" (http://www.tgdaily.com/business-and-law-features/63756-apis-cant-be-copyrighted-says-judge-in-oracle-case). TGDaily. 2012-06-01. Archived (https://web.archive.org/web/20121221090137/http://www.tgdaily.com/business-and-law-features/63756-apis-cant-be-copyrighted-says-judge-in-oracle-case) from the original on 2012-12-21. Retrieved 2012-12-06.
60. "*Oracle America, Inc. vs. Google Inc.*" (https://www.wired.com/wiredenterprise/wp-content/uploads/2012/05/Judge-Alsup-Ruling-on-Copyrightability-of-APIs.pdf) (PDF). Wired. 2012-05-31. Archived (https://web.archive.org/web/20131104001738/http://www.wired.com/wiredenterprise/wp-content/uploads//2012/05/Judge-Alsup-Ruling-on-Copyrightability-of-APIs.pdf) (PDF) from the original on 2013-11-04. Retrieved 2013-09-22.
61. "Oracle Am., Inc. v. Google Inc., No. 13-1021, Fed. Cir. 2014" (https://law.justia.com/cases/federal/appellate-courts/cafc/13-1021/13-1021-2014-05-09.html). Archived (https://web.archive.org/web/20141010070718/http://law.justia.com:80/cases/federal/appellate-courts/cafc/13-1021/13-1021-2014-05-09.html) from the original on 2014-10-10.
62. Rosenblatt, Seth (May 9, 2014). "Court sides with Oracle over Android in Java patent appeal" (https://www.cnet.com/news/court-sides-with-oracle-over-android-in-java-patent-appeal/). *CNET*. Archived (https://web.archive.org/web/20170419063834/https://www.cnet.com/news/court-sides-with-oracle-over-android-in-java-patent-appeal/) from the original on 2017-04-19. Retrieved 2014-05-10.
63. "Google beats Oracle – Android makes "fair use" of Java APIs" (https://arstechnica.com/tech-policy/2016/05/google-wins-trial-against-oracle-as-jury-finds-android-is-fair-use/). *Ars Technica*. 2016-05-26. Archived (https://web.archive.org/web/20170120164551/http://arstechnica.com/tech-policy/2016/05/google-wins-trial-against-oracle-as-jury-finds-android-is-fair-use/) from the original on 2017-01-20. Retrieved 2016-07-28.
64. Decker, Susan (March 27, 2018). "Oracle Wins Revival of Billion-Dollar Case Against Google" (https://www.bloomberg.com/news/articles/2018-03-27/oracle-wins-revival-of-billion-dollar-case-against-google). *Bloomberg Businessweek*. Archived (https://web.archive.org/web/20220109112648/https://www.bloomberg.com/news/articles/2018-03-27/oracle-wins-revival-of-billion-dollar-case-against-google) from the original on January 9, 2022. Retrieved March 27, 2018.
65. Lee, Timothy (January 25, 2019). "Google asks Supreme Court to overrule disastrous ruling on API copyrights" (https://arstechnica.com/tech-policy/2019/01/google-asks-supreme-court-to-overrule-disastrous-ruling-on-api-copyrights/). *Ars Technica*. Archived (https://web.archive.org/web/20190423084450/https://arstechnica.com/tech-policy/2019/01/google-asks-supreme-court-to-overrule-disastrous-ruling-on-api-copyrights/) from the original on April 23, 2019. Retrieved February 8, 2019.
66. vkimber (2020-09-28). "Google LLC v. Oracle America, Inc" (https://www.law.cornell.edu/supct/cert/18-956). *LII / Legal Information Institute*. Archived (https://web.archive.org/web/20210415061113/https://www.law.cornell.edu/supct/cert/18-956) from the original on 2021-04-15. Retrieved 2021-03-06.

67. "Supreme Court of the United States, No. 18–956, GOOGLE LLC, PETITIONER v. ORACLE AMERICA, INC" (https://www.supremecourt.gov/opinions/20pdf/18-956_d18f.pdf) (PDF). April 5, 2021. Archived (https://web.archive.org/web/20210405140150/https://www.supremecourt.gov/opinions/20pdf/18-956_d18f.pdf) (PDF) from the original on April 5, 2021. Retrieved April 25, 2021.

Further reading

- Taina Bucher (16 November 2013). "Objects of Intense Feeling: The Case of the Twitter API" (http://computationalculture.net/article/objects-of-intense-feeling-the-case-of-the-twitter-api). *Computational Culture* (3). ISSN 2047-2390 (https://www.worldcat.org/issn/2047-2390). Argues that "APIs are far from neutral tools" and form a key part of contemporary programming, understood as a fundamental part of culture.
- What is an API? (https://www.supremecourt.gov/opinions/20pdf/18-956_d18f.pdf) – in the U.S. Supreme Court opinion, *Google v. Oracle 2021*, pp. 3–7 – "For each task, there is computer code; API (also known as Application Programming Interface) is the method for calling that 'computer code' (instruction – like a recipe – rather than cooking instruction, this is machine instruction) to be carry out"
- Maury, Innovation and Change (http://ondrejka.net/history/2014/02/28/maury.html) – Cory Ondrejka, February 28, 2014, " ...proposed a public API to let computers talk to each other". (Textise (https://www.textise.net/showText.aspx?strURL=http://ondrejka.net/history/2014/02/28/maury.html) URL)

External links

- Forrester : IT industry : API Case : Google v. Oracle (https://go.forrester.com/what-it-means/ep218-google-oracle-api-case/) – May 20, 2021 – content format: Audio with text – length 26:41
- Geliştiriciler İçin API Listesi (https://api.ahmetcadirci.com) – Feb 23, 2024