Reflection (بازتاب)

In [computer science](https://en.wikipedia.org/wiki/Computer_science), **reflection** is the ability of a [process](https://en.wikipedia.org/wiki/Process_(computing)) to examine, [introspect](https://en.wikipedia.org/wiki/Introspection_(computer_science)), and modify its own structure and behavior.[[1]](https://en.wikipedia.org/wiki/Reflection_(computer_programming)#cite_note-1)

در علم رایانه ، بازتاب توانایی یک فرایند برای بررسی ، درونگرا و اصلاح ساختار و رفتار خود است

Historical background[پس زمینه تاریخی]

The earliest computers were programmed in their native [assembly language](https://en.wikipedia.org/wiki/Assembly_language), which were inherently reflective, as these original architectures could be programmed by defining instructions as data and using [self-modifying code](https://en.wikipedia.org/wiki/Self-modifying_code). As programming moved to compiled higher-level languages such as [Algol](https://en.wikipedia.org/wiki/ALGOL), [Cobol](https://en.wikipedia.org/wiki/Cobol), and [Fortran](https://en.wikipedia.org/wiki/Fortran) (but also [Pascal](https://en.wikipedia.org/wiki/Pascal_(programming_language)) and [C](https://en.wikipedia.org/wiki/C_(programming_language)) and many other languages), this reflective ability largely disappeared until programming languages with reflection built into their type systems appeared

[Brian Cantwell Smith](https://en.wikipedia.org/wiki/Brian_Cantwell_Smith)'s 1982 doctoral dissertation[[2]](https://en.wikipedia.org/wiki/Reflection_(computer_programming)#cite_note-2)[[3]](https://en.wikipedia.org/wiki/Reflection_(computer_programming)#cite_note-3) introduced the notion of computational reflection in procedural [programming languages](https://en.wikipedia.org/wiki/Programming_languages) and the notion of the [meta-circular interpreter](https://en.wikipedia.org/wiki/Meta-circular_interpreter) as a component of [3-Lisp](https://en.wikipedia.org/w/index.php?title=3-Lisp&action=edit&redlink=1).

اولین کامپیوترها به زبان مونتاژ مادری خود برنامه ریزی شده بودند که ذاتاً بازتابی بودند ، زیرا این معماری های اصلی می توانند با تعیین دستورالعمل ها به عنوان داده و با استفاده از کد اصلاح شده خود برنامه ریزی شوند. از آنجا که برنامه نویسی به سمت زبان های سطح بالاتر مانند Algol ، Cobol و Fortran ( بلکه Pascal و C و بسیاری از زبان های دیگر) منتقل شده است ، این توانایی بازتاب تا حد زیادی ناپدید شد تا اینکه زبان های برنامه نویسی با تأمل در سیستم های نوع خود ظاهر شدند.

رساله دکترا در سال 1982 برایان کانتول اسمیت مفهوم تأمل محاسباتی را در زبانهای برنامه نویسی رویه ای و مفهوم مفسر مدور دایره ای به عنوان مؤلفه ای از 3-Lisp معرفی کرد

Uses[[استفاده](https://en.wikipedia.org/w/index.php?title=Reflection_(computer_programming)&action=edit&section=2) ها]

Reflection helps programmers make generic software libraries to display data, process different formats of data, perform [serialization](https://en.wikipedia.org/wiki/Serialization) or deserialization of data for communication, or do bundling and unbundling of data for containers or bursts of communication. Effective use of reflection almost always requires a plan: A design framework, encoding description, object library, a map of a database or entity relations.

بازتاب به برنامه نویسان کمک می کند تا کتابخانه های نرم افزاری عمومی را برای نمایش داده ها ، پردازش قالب های مختلف داده ها ، انجام سریال سازی یا سوزاندن داده ها برای برقراری ارتباط ، یا انجام بسته بندی و جدا کردن داده ها برای ظروف یا انفجارهای ارتباطی کمک کنند. استفاده مؤثر از بازتاب تقریباً همیشه به یک برنامه نیاز دارد: یک چارچوب طراحی ، توصیف کدگذاری ، کتابخانه اشیاء ، نقشه یک بانک اطلاعاتی یا روابط موجود.

Reflection makes a language more suited to network-oriented code. For example, it assists languages such as Java to operate well in networks by enabling libraries for serialization, bundling and varying data formats. Languages without reflection (e.g. [C](https://en.wikipedia.org/wiki/C_(programming_language))) have to use auxiliary compilers, e.g. for [Abstract Syntax Notation](https://en.wikipedia.org/wiki/Abstract_Syntax_Notation), to produce code for serialization and bundling.

بازتاب باعث می شود زبانی متناسب با کد شبکه محور باشد. به عنوان مثال ، به زبانهایی مانند جاوا کمک می کند که با فعال کردن کتابخانه ها برای سریال سازی ، بسته بندی و فرمت های مختلف داده ، در شبکه ها به خوبی کار کنند. زبانهای بدون تأمل (به عنوان مثال C) مجبورند از کامپایلرهای کمکی استفاده کنند ، به عنوان مثال برای Abstract Syntax Notation ، برای تولید کد برای سریال سازی و بسته بندی.

Reflection can be used for observing and modifying program execution at runtime. A reflection-oriented program component can monitor the execution of an enclosure of code and can modify itself according to a desired goal related to that enclosure. This is typically accomplished by dynamically assigning program code at runtime.

بازتاب می تواند برای مشاهده و اصلاح اجرای برنامه در زمان اجرا استفاده شود. یک مؤلفه برنامه بازتاب محور می تواند بر اجرای یک محفظه کد نظارت کند و می تواند خود را مطابق هدف مورد نظر مربوط به آن محوطه تغییر دهد. این معمولاً با اختصاص پویا کد برنامه در زمان اجرا انجام می شود

In object-oriented programming languages such as [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), reflection allows *inspection* of classes, interfaces, fields and methods at runtime without knowing the names of the interfaces, fields, methods at compile time. It also allows *instantiation* of new objects and *invocation* of methods.

در زبانهای برنامه نویسی شی گرا مانند جاوا ، بازتاب اجازه می دهد کلاس ها ، رابط ها ، قسمت ها و روش ها را در زمان اجرا بدون دانستن نام واسط ها ، فیلدها ، روشها در زمان کامپایل بازرسی کنید. همچنین امکان فوری اشیاء جدید و فراخوانی روشها را فراهم می آورد

Reflection is often used as part of [software testing](https://en.wikipedia.org/wiki/Software_testing), such as for the runtime creation/instantiation of [mock objects](https://en.wikipedia.org/wiki/Mock_object). Reflection is also a key strategy for [metaprogramming](https://en.wikipedia.org/wiki/Metaprogramming).

بازتاب اغلب به عنوان بخشی از تست نرم افزار ، مانند ایجاد زمان اجرا یا فوری اشیاء mock استفاده می شود. بازتاب همچنین یک استراتژی کلیدی برای برنامه نویسی metaprogramming است

In some object-oriented programming languages, such as [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)) and [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), reflection can be used to bypass [member accessibility](https://en.wikipedia.org/wiki/Member_accessibility) rules. For C#-properties this can be achieved by writing directly onto the (usually invisible) backing field of a non-public property. It is also possible to find non-public methods of classes and types and manually invoke them. This works for project-internal files as well as external libraries (.Net-assemblies and Java-archives).

در برخی از زبانهای برنامه نویسی شی گرا مانند C# و جاوا می توان از انعکاس برای دور زدن قوانین دسترسی اعضای استفاده کرد. برای فیلدهای C# این می تواند با نوشتن مستقیم روی قسمت پشتی (معمولاً نامرئی) یک فیلد غیر عمومی حاصل شود. همچنین می توان روشها و کلاسهای غیر عمومی را پیدا کرد و به صورت دستی از آنها استناد کرد. این کار برای فایل های داخلی پروژه و همچنین کتابخانه های خارجی (بایگانی مجامع خالص و بایگانی جاوا) انجام می شود.

Implementation[[پیاده](https://en.wikipedia.org/w/index.php?title=Reflection_(computer_programming)&action=edit&section=3) سازی]

A language supporting reflection provides a number of features available at runtime that would otherwise be difficult to accomplish in a lower-level language. Some of these features are the abilities to:

* Discover and modify source-code constructions (such as code blocks, [classes](https://en.wikipedia.org/wiki/Class_(computer_science)), methods, protocols, etc.) as [first-class objects](https://en.wikipedia.org/wiki/First-class_object) at runtime.
* Convert a [string](https://en.wikipedia.org/wiki/String_(computer_science)) matching the symbolic name of a class or function into a reference to or invocation of that class or function.
* Evaluate a string as if it were a source-code statement at runtime.
* Create a new interpreter for the language's bytecode to give a new meaning or purpose for a programming construct.

یک زبان با پشتیبانی بازتاب تعدادی ویژگی را در زمان اجرا فراهم می کند که در غیر این صورت انجام یک زبان سطح پایین دشوار خواهد بود. برخی از این ویژگی ها توانایی های زیر است:

* ساختارهای منبع منبع (مانند بلوک کد ، کلاس ها ، روش ها ، پروتکل ها و غیره) را به عنوان اشیاء کلاس اول در زمان اجرا کشف و اصلاح کنید.
* یک رشته مطابق با نام نمادین یک کلاس یا عملکرد را به یک مرجع یا فراخوانی آن کلاس یا عملکرد تبدیل کنید.
* یک رشته را طوری ارزیابی کنید که گویی در زمان اجرا یک بیانیه کد منبع است.
* یک مترجم جدید برای کد برنامه زبان ایجاد کنید تا معنای جدیدی را برای یک ساختار برنامه نویسی ارائه دهد

These features can be implemented in different ways. In [MOO](https://en.wikipedia.org/wiki/MOO_(programming_language)), reflection forms a natural part of everyday programming idiom. When verbs (methods) are called, various variables such as *verb* (the name of the verb being called) and *this* (the object on which the verb is called) are populated to give the context of the call. Security is typically managed by accessing the caller stack programmatically: Since *callers*() is a list of the methods by which the current verb was eventually called, performing tests on *callers*()[1] (the command invoked by the original user) allows the verb to protect itself against unauthorised use.

این ویژگی ها می توانند به روش های مختلفی پیاده سازی شوند. در MOO ، بازتاب بخشی طبیعی از اصطلاحات برنامه نویسی روزمره است. وقتی به فعل ها (روش ها) فراخوانی می شود ، متغیرهای مختلفی از جمله فعل (نام فعل نامیده می شود) و این (شیی ای که فعل روی آن خوانده می شود) جمع می شوند تا زمینه تماس را ارائه دهند. امنیت بطور معمول با دسترسی به استک صدا زننده مدیریت می شود: از آنجا که callers() لیستی از روش هایی است که با استفاده از آن فعل فعلی در نهایت فراخوانی می شود ، انجام آزمایشات بر روی callers() (دستور فراخوانی شده توسط کاربر اصلی) به فعل اجازه می دهد تا برای محافظت از خود در برابر استفاده غیرمجاز از خود محافظت کند.

Compiled languages rely on their runtime system to provide information about the source code. A compiled [Objective-C](https://en.wikipedia.org/wiki/Objective-C) executable, for example, records the names of all methods in a block of the executable, providing a table to correspond these with the underlying methods (or selectors for these methods) compiled into the program. In a compiled language that supports runtime creation of functions, such as [Common Lisp](https://en.wikipedia.org/wiki/Common_Lisp), the runtime environment must include a compiler or an interpreter.

زبان های تدوین شده برای تهیه اطلاعات در مورد کد منبع به سیستم زمان اجرا خود تکیه می کنند. به عنوان مثال ، یک کامپایل شده قابل اجرای Objective-C ، اسامی همه روش ها را در یک بخش از اجرایی ثبت می کند ، و جداول تهیه می کند تا این روش ها با روش های زیرساختی (یا انتخاب کنندگان این روش ها) کامپایل شده در برنامه مطابق باشد. در یک زبان کامپایل شده که از ایجاد کارکرد توابع مانند Common Lisp پشتیبانی می کند ، محیط زمان اجرا باید کامپایلر یا مفسر باشد

Reflection can be implemented for languages not having built-in reflection facilities by using a [program transformation](https://en.wikipedia.org/wiki/Program_transformation) system to define automated source-code changes.

بازتاب می تواند برای زبانهایی که از امکانات بازتاب داخلی برخوردار نیستند با استفاده از یک سیستم تبدیل برنامه برای تعریف تغییرات خودکار کد منبع استفاده شود.