Skip to main content	
[GitHub Docs](/en)	
Version: Free, Pro, & Team	
Search GitHub DocsSearch	
Select language: current language is English	
Open Search BarClose Search Bar	
Open Menu	
Open Sidebar	
* [Search on GitHub](/en/search-github "Search on GitHub")/	
* [GitHub Code Search](/en/search-github/github-code-search "GitHub Code Search")/	
* [Code searc	h
syntax](/en/search-github/github-code-search/understanding-github-code-search-syntax "Cod	е
search syntax")	
[Home](/en)	
[Search on GitHub](/en/search-github)	
* Start with search on GitHub	

*	[About	searching	on
GitHub](/en/search-github/getting-started-with-searc	hing-on-github/a	bout-searching-on-git	hub)
	*	[Understand	search
syntax](/en/search-github/getting-started-with-search	ning-on-github/u	nderstanding-the-sear	rch-syntax)
	* [7	Froubleshoot	search
queries](/en/search-github/getting-started-with-searc	:hing-on-github/t	roubleshooting-search	n-queries)
	*	[Sorting	search
results](/en/search-github/getting-started-with-search	ning-on-github/s	orting-search-results)	
* Searching on GitHub			
* [Finding files on GitHub](/en/search-github/search	ching-on-github/f	inding-files-on-github))
* [Search for repositories](/en/search-github/search	:hing-on-github/s	searching-for-reposito	ries)
* [Searching topics](/en/search-github/searching-c	on-github/search	ing-topics)	
* [Searching code (legacy)](/en/search-github/sea	rching-on-github	o/searching-code)	
* [Searching commits](/en/search-github/searching	g-on-github/sear	rching-commits)	
*	[Search	n issues	&
PRs](/en/search-github/searching-on-github/searchir	ng-issues-and-p	ull-requests)	
* [Searching discussions](/en/search-github/searc	hing-on-github/s	earching-discussions)
	*	[Search	GitHub
Marketplace](/en/search-github/searching-on-github/	searching-githul	b-marketplace)	
* [Searching users](/en/search-github/searching-o	n-github/searchi	ing-users)	
* [Searching for packages](/en/search-github/sear	ching-on-github	/searching-for-packag	es)
* [Searching wikis](/en/search-github/searching-or	n-github/searchi	ng-wikis)	

* [Searching in forks](/en/search-github/searching-on-github/searching-in-forks)

* [Searching gists](/en/search-github/searching-on-github/searching-gists)

* GitHub Code Search * [About GitHub Code Search](/en/search-github/github-code-search/about-github-code-search) * [Using GitHub Code Search](/en/search-github/github-code-search/using-github-code-search) [Code search syntax](/en/search-github/github-code-search/understanding-github-code-search-syntax) * [Search on GitHub](/en/search-github "Search on GitHub")/ * [GitHub Code Search](/en/search-github/github-code-search "GitHub Code Search")/ [Code search syntax](/en/search-github/github-code-search/understanding-github-code-search-syntax "Code search syntax") # Understanding GitHub Code Search syntax You can build search queries for the results you want with specialized code qualifiers, regular expressions, and boolean operations. ## In this article * About code search query structure * Query for an exact match * Searching for quotes and backslashes * Using boolean operations

* Using qualifiers

* Using regular expressions

* Separating search terms

* Case sensitivity

About code search query structure

The search syntax in this article only applies to searching code with GitHub code search. Note that the syntax and qualifiers for searching for non-code content, such as issues, users, and discussions, is not the same as the syntax for code search. For more information on non-code search, see [About searching on GitHub](/en/search-github/getting-started-with-searching-on-github/about-searching-on-github) and [Searching on GitHub](/en/search-github/searching-on-github).

Search queries consist of search terms, comprising text you want to search for, and qualifiers, which narrow down the search.

A bare term with no qualifiers will match either the content of a file or the file's path.

For example, the following query:

http-push

The above query will match the file `docs/http-push.txt`, even if it doesn't

contain the term `http-push`. It will also match a file called `example.txt` if it contains the term `http-push`.

You can enter multiple terms separated by whitespace to search for documents that satisfy both terms.

For example, the following query:

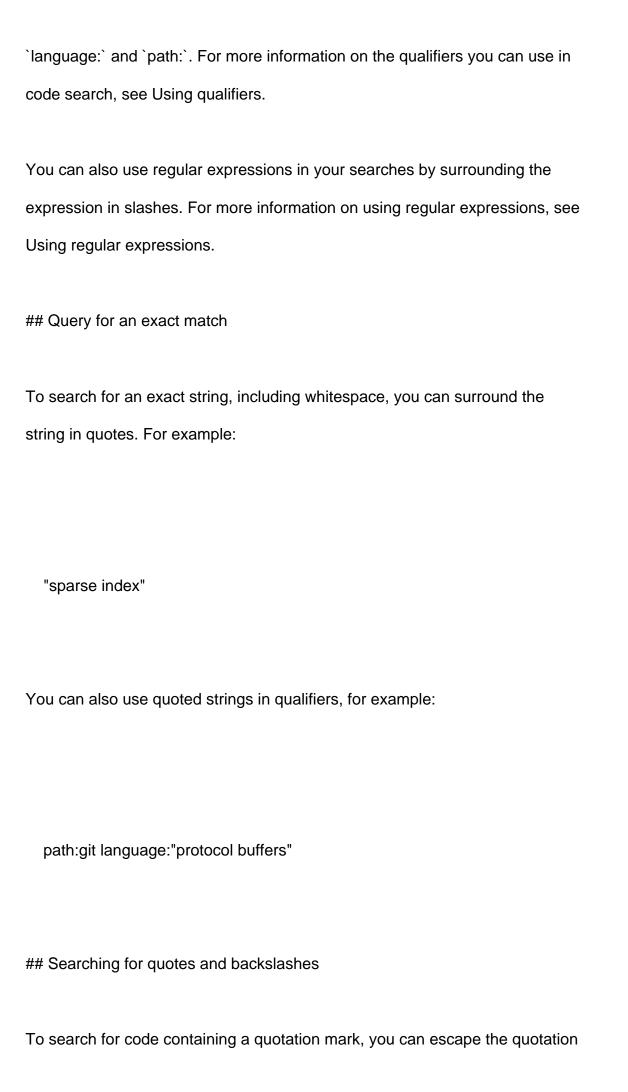
sparse index

The search results would include all documents containing both the terms 'sparse' and 'index', in any order. As examples, it would match a file containing 'SparseIndexVector', a file with the phrase 'index for sparse trees', and even a file named 'index.txt' that contains the term 'sparse'.

Searching for multiple terms separated by whitespace is the equivalent to the search `hello AND world`. Other boolean operations, such as `hello OR world`, are also supported. For more information about boolean operations, see Using boolean operations.

Code search also supports searching for an exact string, including whitespace. For more information, see Query for an exact match.

You can narrow your code search with specialized qualifiers, such as `repo:`,



mark using a backslash. For example, to find the exact string `name = "tensorflow"`, you can search:

"name = \"tensorflow\""

To search for code containing a backslash, '\', use a double backslash, '\\'.

The two escape sequences `\\` and `\"` can be used outside of quotes as well.

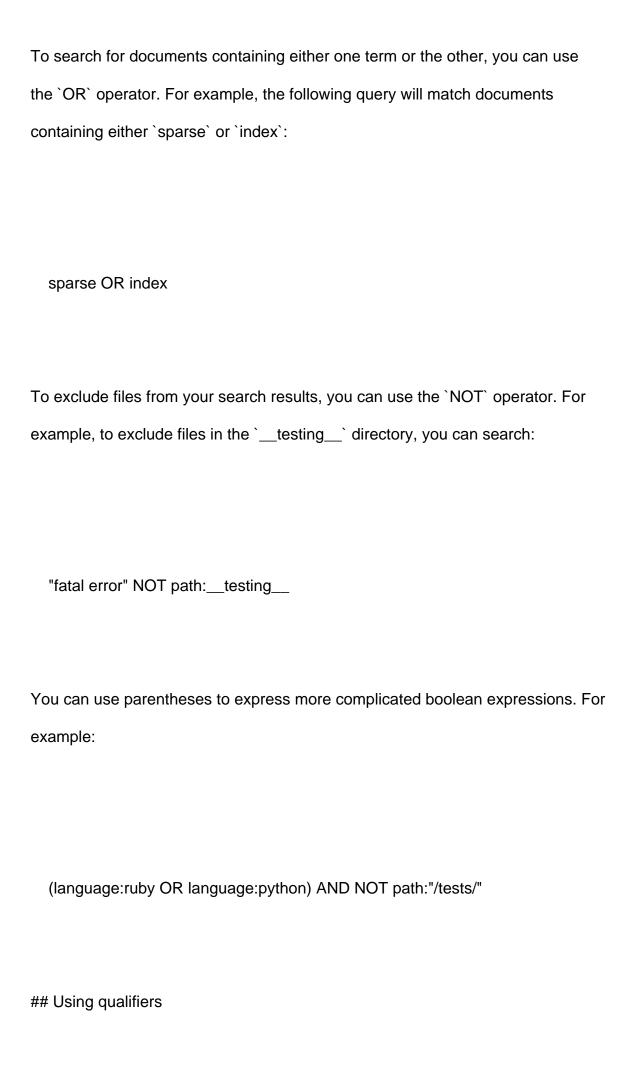
No other escape sequences are recognized, though. A backslash that isn't followed by either `"` or `\` is included in the search, unchanged.

Additional escape sequences, such as '\n' to match a newline character, are supported in regular expressions. See Using regular expressions.

Using boolean operations

Code search supports boolean expressions. You can use the operators `AND`, `OR`, and `NOT` to combine search terms.

By default, adjacent terms separated by whitespace are equivalent to using the `AND` operator. For example, the search query `sparse index` is the same as `sparse AND index`, meaning that the search results will include all documents containing both the terms `sparse` and `index`, in any order.



* Repository qualifier
* Organization and user qualifiers
* Language qualifier
* Path qualifier
* Symbol qualifier
* Content qualifier
* Is qualifier
Repository qualifier
To search within a repository, use the `repo:` qualifier. You must provide the
full repository name, including the owner. For example:
repo:github-linguist/linguist
To search within a set of repositories, you can combine multiple `repo:`
qualifiers with the boolean operator `OR`. For example:
repo:github-linguist/linguist OR repo:tree-sitter/tree-sitter

You can use specialized keywords to qualify your search.

Note
Code search does not currently support regular expressions or partial matching
for repository names, so you will have to type the entire repository name
(including the user prefix) for the `repo:` qualifier to work.
Organization and user qualifiers
To search for files within an organization, use the `org:` qualifier. For
example:
org:github
To search for files within a personal account, use the `user:` qualifier. For
example:
user:octocat
Note

Code search does not currently support regular expressions or partial matching for organization or user names, so you will have to type the entire organization or user name for the qualifier to work.

Language qualifier

To narrow down to a specific languages, use the `language:` qualifier. For example:

language:ruby OR language:cpp OR language:csharp

For a complete list of supported language names, see
[languages.yaml](https://github.com/githublinguist/linguist/blob/main/lib/linguist/languages.yml) in [githublinguist/linguist](https://github.com/github-linguist/linguist). If your
preferred language is not on the list, you can open a pull request to add it.

Path qualifier

To search within file paths, use the `path:` qualifier. This will match files containing the term anywhere in their file path. For example, to find files containing the term `unit_tests` in their path, use:

path:unit_tests

The above query will match both `src/unit_tests/my_test.py` and `src/docs/unit_tests.md` since they both contain `unit_test` somewhere in their path.

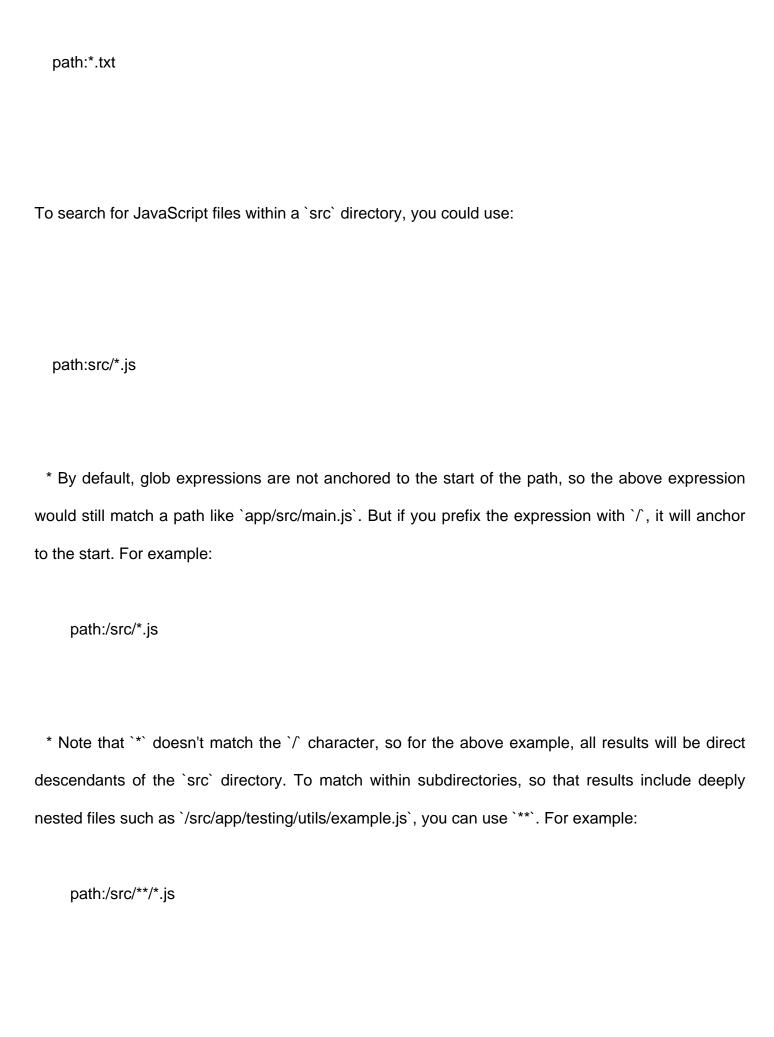
To match only a specific filename (and not part of the path), you could use a regular expression:

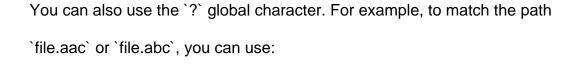
path:/(^|V)README\.md\$/

Note that the `.` in the filename is escaped, since `.` has special meaning for regular expressions. For more information about using regular expressions, see Using regular expressions.

You can also use some limited glob expressions in the `path:` qualifier.

For example, to search for files with the extension `txt`, you can use:





path:*.a?c

To search for a filename which contains a special character like `*` or `?`, just use a quoted string:

path:"file?"

Glob expressions are disabled for quoted strings, so the above query will only match paths containing the literal string `file?`.

Symbol qualifier

You can search for symbol definitions in code, such as function or class definitions, using the `symbol:` qualifier. Symbol search is based on parsing your code using the open source [Tree-sitter](https://github.com/tree-sitter) parser ecosystem, so no extra setup or build tool integration is required.

For example, to search for a symbol called `WithContext`:

language:go symbol:WithContext

In some languages, you can search for symbols using a prefix (e.g. a prefix of their class name). For example, for a method `deleteRows` on a struct `Maint`, you could search `symbol:Maint.deleteRows` if you are using Go, or `symbol:Maint::deleteRows` in Rust.

You can also use regular expressions with the symbol qualifier. For example, the following query would find conversions people have implemented in Rust for the `String` type:

language:rust symbol:/^String::to_.*/

Note that this qualifier only searches for definitions and not references, and not all symbol types or languages are fully supported yet. Symbol extraction is supported for the following languages:

* Bash

- * C
 * C#
 * C++
 * CodeQL
 * Elixir
 - * Go
 - * JSX
 - * Java
 - * JavaScript
 - * Lua
 - * PHP
 - * Protocol Buffers
 - * Python
 - * R
 - * Ruby
 - * Rust
 - * Scala
 - * Starlark
 - * Swift
 - * Typescript

We are working on adding support for more languages. If you would like to help contribute to this effort, you can add support for your language in the open source [Tree-sitter](https://github.com/tree-sitter) parser ecosystem, upon which symbol search is based.

By default, bare terms search both paths and file content. To restrict a search to strictly match the content of a file and not file paths, use the `content:` qualifier. For example:

content:README.md

This query would only match files containing the term `README.md`, rather than matching files named `README.md`.

Is qualifier

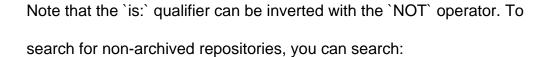
To filter based on repository properties, you can use the `is:` qualifier.

`is:` supports the following values:

- * `archived`: restricts the search to archived repositories.
- * `fork`: restricts the search to forked repositories.
- * `vendored`: restricts the search to content detected as vendored.
- * `generated`: restricts the search to content detected as generated.

For example:

path:/^MIT.txt\$/	is:archived
-------------------	-------------



log4j NOT is:archived

To exclude forks from your results, you can search:

log4j NOT is:fork

Using regular expressions

Code search supports regular expressions to search for patterns in your code.

You can use regular expressions in bare search terms as well as within many

qualifiers, by surrounding the regex in slashes.

For example, to search for the regular expression `sparse.*index`, you would

use:

/sparse.*index/

Note that you'll have to escape any forward slashes within the regular expression. For example, to search for files within the `App/src` directory, you would use:

/^App\/src\/

Inside a regular expression, '\n' stands for a newline character, '\t' stands for a tab, and '\x{hhhh}' can be used to escape any Unicode character. This means you can use regular expressions to search for exact strings that contain characters that you can't type into the search bar.

Most common regular expressions features work in code search. However, "lookaround" assertions are not supported.

Separating search terms

All parts of a search, such as search terms, exact strings, regular expressions, qualifiers, parentheses, and the boolean keywords `AND`, `OR`, and `NOT`, must be separated from one another with spaces. The one exception

is that items inside parentheses, `(` `)`, don't need to be separated from the parentheses.

If your search contains multiple components that aren't separated by spaces, or other text that does not follow the rules listed above, code search will try to guess what you mean. It often falls back on treating that component of your query as the exact text to search for. For example, the following query:

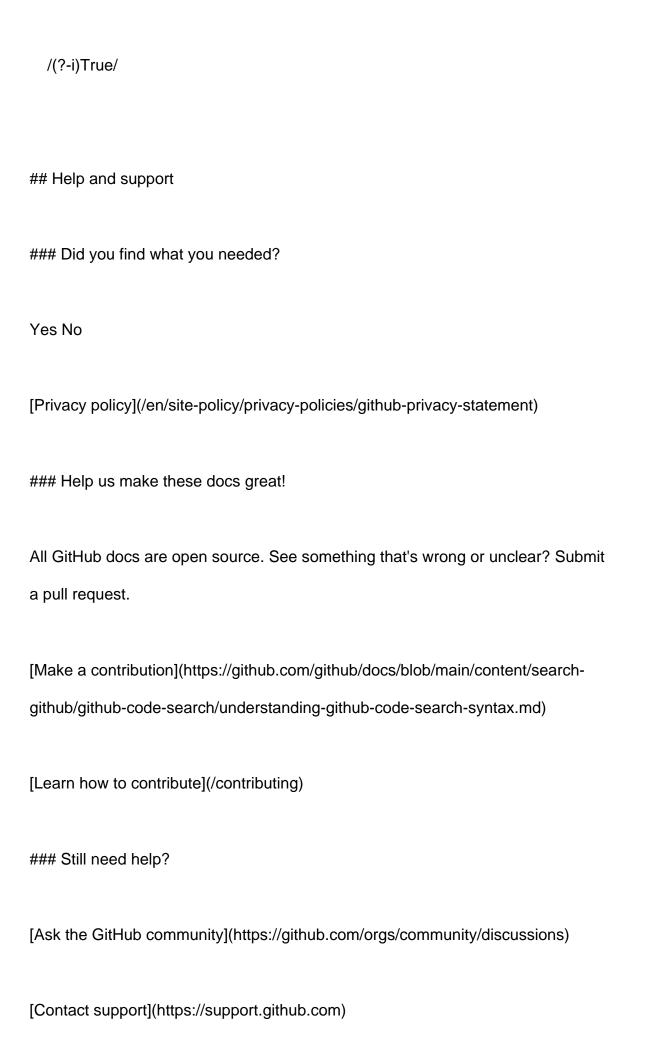
printf("hello world\n");

Code search will give up on interpreting the parentheses and quotes as special characters and will instead search for files containing that exact code.

If code search guesses wrong, you can always get the search you wanted by using quotes and spaces to make the meaning clear.

Case sensitivity

By default, code search is case-insensitive, and results will include both uppercase and lowercase results. You can do case-sensitive searches by using a regular expression with case insensitivity turned off. For example, to search for the string "True", you would use:



Legal

- * © 2025 GitHub, Inc.
- * [Terms](/en/site-policy/github-terms/github-terms-of-service)
- * [Privacy](/en/site-policy/privacy-policies/github-privacy-statement)
- * [Status](https://www.githubstatus.com/)
- * [Pricing](https://github.com/pricing)
- * [Expert services](https://services.github.com)
- * [Blog](https://github.blog)