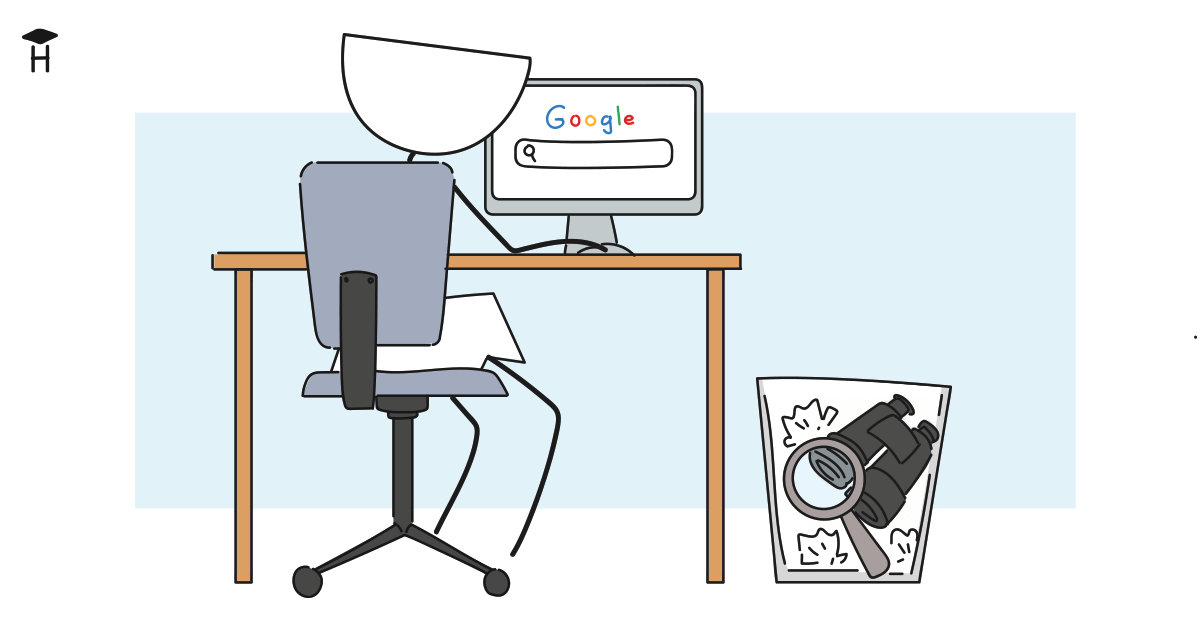
# **Как искать техническую информацию**



**Одним из важнейших навыков для программиста является умение находить ответы и решать проблемы с помощью Google. Многие проблемы, с которыми сталкиваются новички, уже были решены и описаны ранее. Все, что вам нужно, это научиться находить эти решения и ответы.**

* [Веб-сайты](https://guides.hexlet.io/how-to-search/#websites)
* [Язык](https://guides.hexlet.io/how-to-search/#language)
* [Поисковая система](https://guides.hexlet.io/how-to-search/#search-engine)
* [Язык запросов](https://guides.hexlet.io/how-to-search/#query-language)
* [Поиск в библиотеке](https://guides.hexlet.io/how-to-search/#library-search)
* [Поиск по сообщению об ошибке](https://guides.hexlet.io/how-to-search/#search-by-error-message)
* [Поиск по поведению](https://guides.hexlet.io/how-to-search/#behavior-search)
* [Официальная документация](https://guides.hexlet.io/how-to-search/#official-documentation)

## Веб-сайты

Основные сайты, на которых можно получить ответы на свои вопросы:

* [GitHub](https://github.com/)
* [Stackoverflow](https://stackoverflow.com/)

## Язык

Как показывает практика, вначале большинство программистов пытаются искать информацию на своем родном языке. Иногда таким образом можно найти ответ, но чаще — нет. **Универсальным языком для программистов является английский** – на нем говорит весь мир. Объем информации в англоязычном сегменте заметно выше, чем в любом другом. Кроме того, это намного свежее. Научитесь более точно формулировать свои мысли, а во время поиска - составлять набор слов на английском. Так вы быстрее освоите терминологию.

[Code Basics: бесплатные курсы программирования](https://ru.code-basics.com/?utm_source=hexlet-guides&utm_medium=banner&utm_campaign=site-code-basics)

## Поисковая система

В продолжение предыдущего раздела важно отметить, что лучший способ поиска информации - это использование Google. Хотя локальные поисковые системы хороши для определенных задач, англоязычный сегмент не является их основным рынком, и они намного уступают Google. Например, вы заметите, что через некоторое время Google адаптируется к вашим запросам и начинает показывать более релевантные ссылки. Он может понять, какой язык программирования вы предпочитаете, и выдаст соответствующие ответы.

Еще один важный момент: поиск в Google работает даже лучше, чем поиск по сайту. Если вам что-то нужно, например, на Github, лучше сделать правильный запрос в Google, чтобы получить лучший и быстрый результат. В приведенном ниже разделе "Язык запросов" приведены дополнительные сведения.

## Язык запросов

Каждая поисковая система имеет свой язык запросов. Она включает специальные операторы поиска, позволяющие более точно указать, что вам нужно. Вот несколько распространенных методов поиска:

* site:stackoverflow.com how to test react code — этот поиск будет выполняться через [Stackoverflow](https://stackoverflow.com/).
* add class to element -jquery — дефис обозначает стоп-слова, поэтому будут найдены все варианты, исключающие эти слова.
* "immutable js" — двойные кавычки, указывающие на поиск точного соответствия.

С полным списком можно ознакомиться [на сайте поддержки Google](https://support.google.com/websearch/answer/2466433?visit_id=1-636424030566191968-2246914586&p=adv_operators&hl=en&rd=1).

## Поиск в библиотеке

Подавляющее большинство библиотек расположено (они говорят, "размещено") на [GitHub](https://github.com/). Допустим, вам нужно найти библиотеку для выполнения HTTP-запросов на JavaScript. Для этого вы можете сформировать следующий запрос: github js http client. Google покажет вам дюжину ссылок на различные репозитории. Конечно, вы также можете использовать язык запросов: site:github.com js http client, но, как правило, достаточно просто указать github. Та же стратегия поиска может быть использована для известных библиотек: github express.

## Поиск по сообщению об ошибке

Прежде чем выполнять поиск по сообщению об ошибке, необходимо понять, где именно здесь находится *сообщение об ошибке*. Хотя выходные данные содержат много соответствующей информации, это не описание этой ошибки. Например:

There was 1 failure:  
1) App\SolutionTest::testResult with data set *#2 (0, 2, 2, 1, 2)*  
Failed asserting that '1' matches expected 0.  
/usr/src/app/tests/App/Tests/SolutionTest.php:15  
FAILURES**!**  
Tests: 3, Assertions: 3, Failures: 1.  
Makefile:2: recipe for target 'test' failed  
make: Leaving directory '/usr/src/app'  
make: \*\*\* [test] Error 1

В выходных данных много текста, но есть только одно реальное сообщение об ошибке: Failed asserting that '1' matches expected 0.. В данном конкретном случае примерно ясно, что не так и где искать (файл и строка указаны в трассировке стека ниже). Но так происходит не всегда. Если вы правильно определили сообщение об ошибке, возможно, вам также захочется сделать одну вещь. Часто такие сообщения являются персонализированными: они принимают конкретные значения параметров, которые относятся к вашей среде. Например, пути к файлам. Следовательно, если вы будете искать по всему сообщению об ошибке, то, скорее всего, Google ничего не найдет. Например, в сообщении выше такими параметрами являются '1' и 0. Если вы очистите фразу, появится Failed asserting that matches expected. Это то, что вам нужно искать. Вы также можете добавить название языка: php Failed asserting that matches expected.

## Поиск по поведению

Иногда сообщение об ошибке либо отсутствует, либо не может привести к правильному ответу (поскольку это следствие, а не причина). На этом этапе вам нужно проявить творческий подход и составить предложение на английском языке. Также подойдет набор ключевых слов. Если поиск не был успешным, попробуйте добавитьsite:stackoverflow.com. Stackoverflow - уникальное место, где есть ответы практически на все подобные вопросы. Если вы знаете, к какой библиотеке или программе точно относится ошибка, было бы полезно найти ее репозиторий на Github и изучить раздел "Проблемы". Если ошибка определенно есть, и она актуальна, то, скорее всего, кто-то уже упоминал о ней.

## Официальная документация

Поиск - это хорошо, но не забудьте заглянуть в официальную документацию по инструментам (включая языки программирования). Официальная (и не только) документация, как правило, делится на следующие типы:

1. [Начало работы](https://guides.rubyonrails.org/getting_started.html) - это небольшое (не всегда) пошаговое руководство, помогающее создать минимальную рабочую версию и быстро начать работу с инструментом, чтобы увидеть его в действии. Это первый документ, который вам нужно найти.
2. [Руководства](https://laravel.com/docs/5.5/routing) представляют собой описания компонентов инструмента. Они написаны в повествовательной форме, поэтому прочитать их от начала до конца не составит труда. Они облегчают изучение больших блоков информации.
3. [API](https://bit.ly/2uq98XM) - это краткая документация по всем возможным функциям приложения. Она больше предназначена для поиска ответов на конкретные вопросы.
4. [Учебные пособия](https://blog.codeship.com/an-introduction-to-apis-with-phoenix/), в отличие от quides, нацелены на различные варианты использования инструмента.