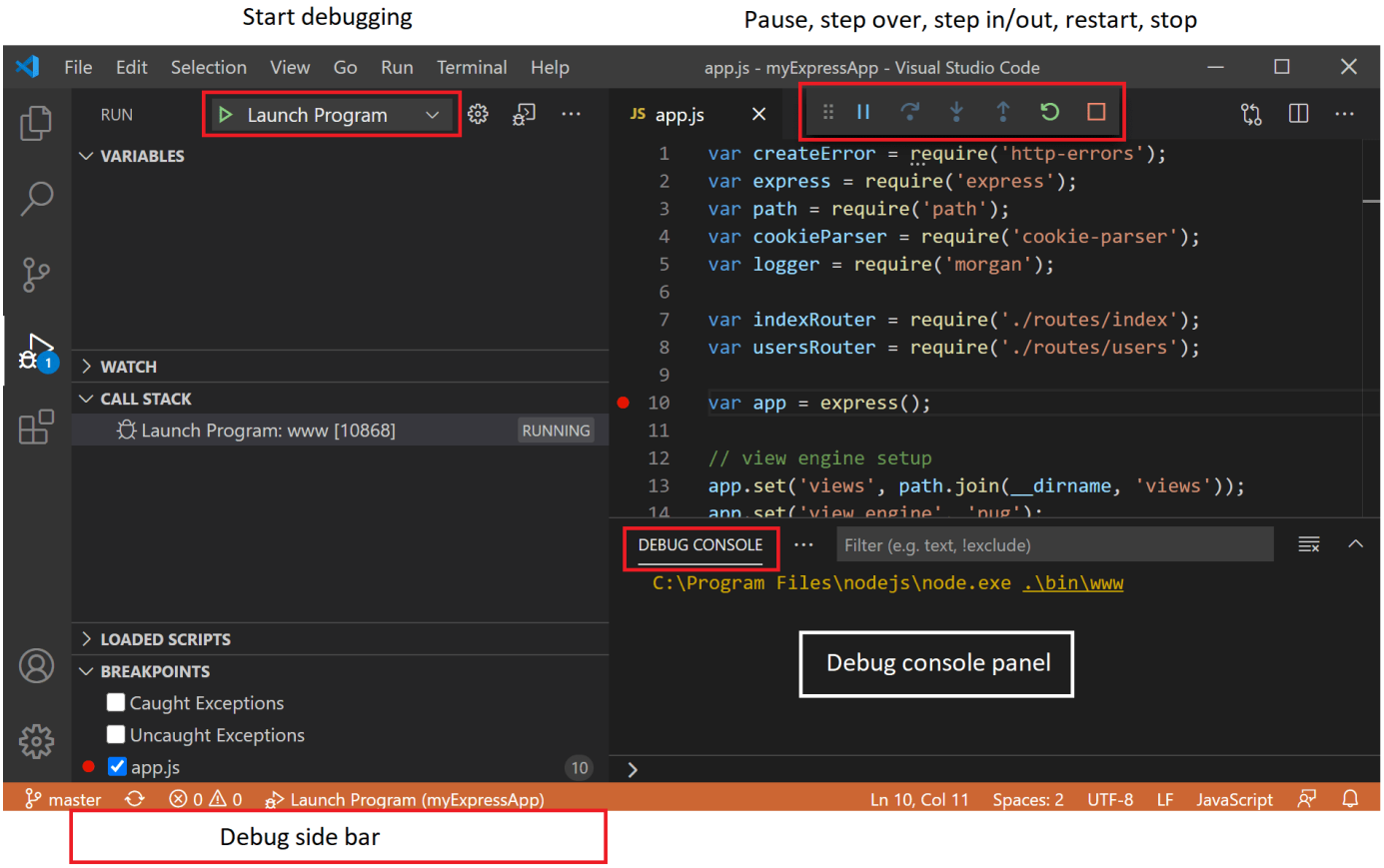
Topics Debugging

In this article Debugger extensions

(https://vscode.dev/github/microsoft/vscode-docs/blob/main/docs/editor/debugging.md)

Отладка

Одной из ключевых особенностей Visual Studio Code является его отличная поддержка отладки. Встроенный отладчик VS Code помогает ускорить цикл редактирования, компиляции и отладки.



Расширения отладчика

VS Code имеет встроенную поддержку отладки для Node.js (https://nodejs.org/) среды выполнения и может отлаживать JavaScript, TypeScript или любой другой язык, который переносится на JavaScript.

Для отладки других языков и сред выполнения (включая PHP (https://marketplace.visualstudio.com/items?itemName=xdebug.php-debug), Ruby (https://marketplace.visualstudio.com/items?itemName=rebornix.Ruby), Go

(https://marketplace.visualstudio.com/items?itemName=golang.go), C # (https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.csharp), Python (https://marketplace.visualstudio.com/items?itemName=ms-python.python), C ++ (https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools), PowerShell (https://marketplace.visualstudio.com/items?itemName=ms-vscode.PowerShell) и многих других (https://marketplace.visualstudio.com/search?term=debug&target=VSCode&category=Debuggers&sortBy=Relevance)) поищите Debuggers расширения (/docs/editor/extension-marketplace) в Marketplace (https://marketplace.visualstudio.com/vscode/Debuggers) VS Code или выберите "Установить дополнительные отладчики" в меню "Выполнить" верхнего уровня.

Ниже приведены несколько популярных расширений, которые включают поддержку отладки:

Python
ms-python 113.6M

(https://marketplace.visualstudio.com/items?itemName=ms-python.python)

C / C ++
ms-vscode 60.5M

(https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools)

Инструменты CMake
ms-vscode 29.5M

(https://marketplace.visualstudio.com/items?itemName=ms-vscode.cmake-tools)

Отладчик для Java
vscjava 29.3M

(https://marketplace.visualstudio.com/items?itemName=vscjava.vscode-java-debug)

> Совет: Расширения, показанные выше, запрашиваются динамически. Выберите плитку расширений выше, чтобы прочитать описание и обзоры и решить, какое расширение лучше для вас.

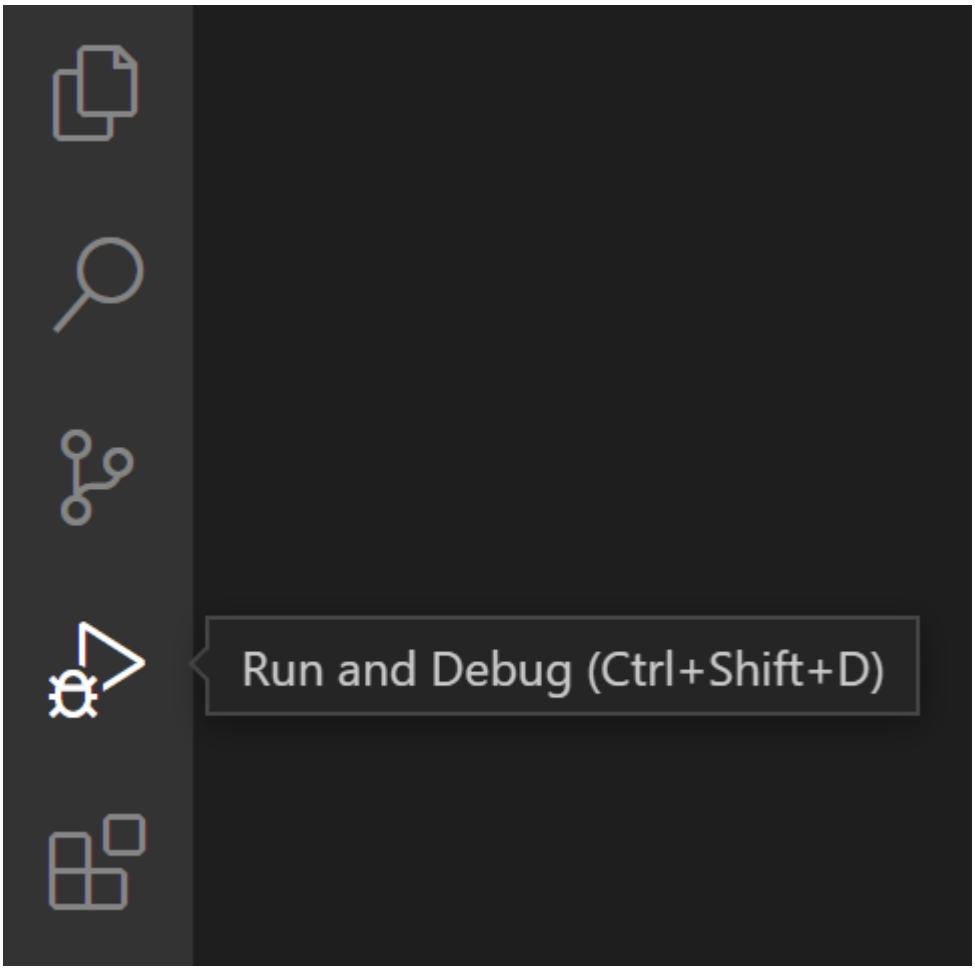
Начать отладку

Следующая документация основана на встроенном Node.js (https://nodejs.org/) отладчике, но большинство концепций и функций применимы и к другим отладчикам.

Полезно сначала создать образец Node.js приложения, прежде чем читать об отладке. Вы можете следовать Node.js пошаговому руководству (/docs/nodejs/nodejs-tutorial) по установке Node.js и созданию простого JavaScript-приложения "Hello World" ("Привет, мир") (`app.js`). После того, как вы настроите простое приложение, на этой странице вы познакомитесь с функциями отладки VS Code.

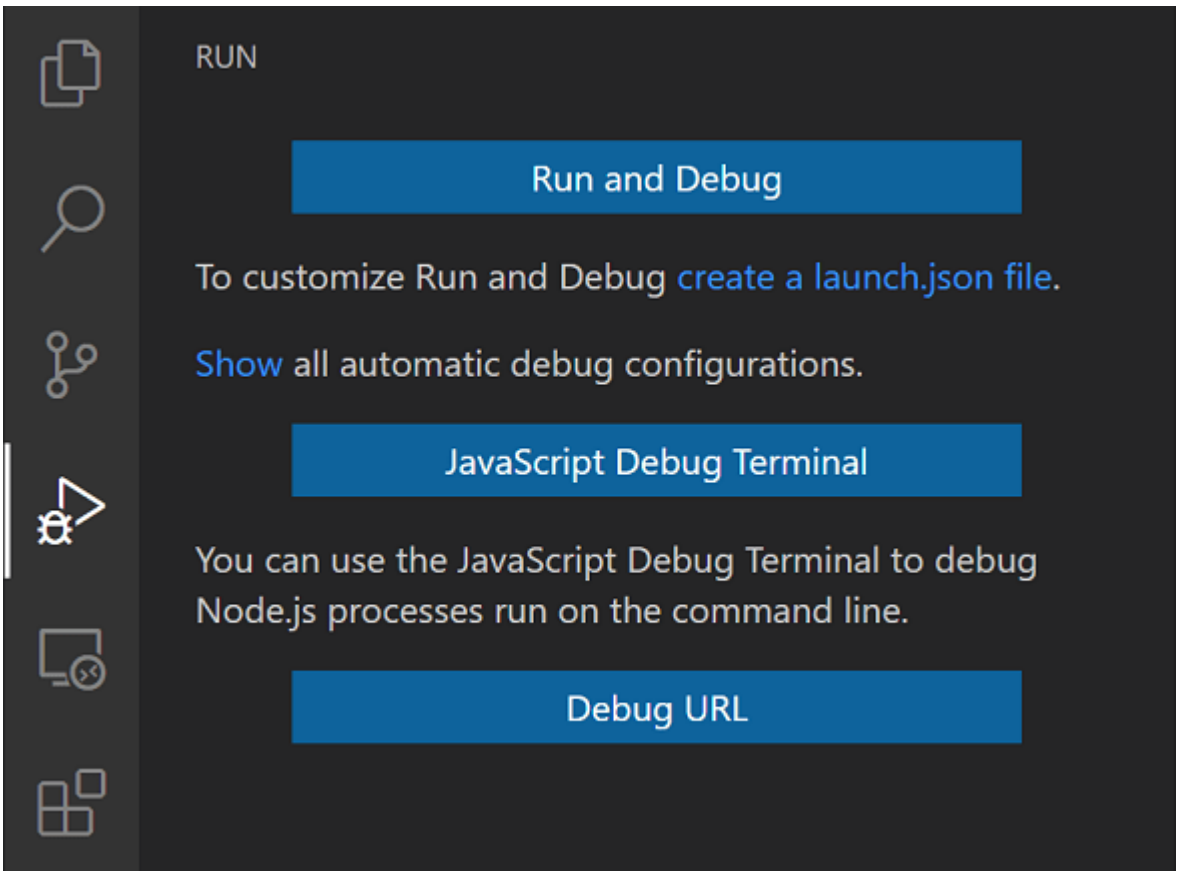
Запуск и просмотр отладки

Чтобы вызвать представление "Запуск и отладка", выберите значок "Запуск и отладка" в панели действий сбоку от VS Code. Вы также можете использовать сочетание клавиш `Ctrl + Shift + D` .



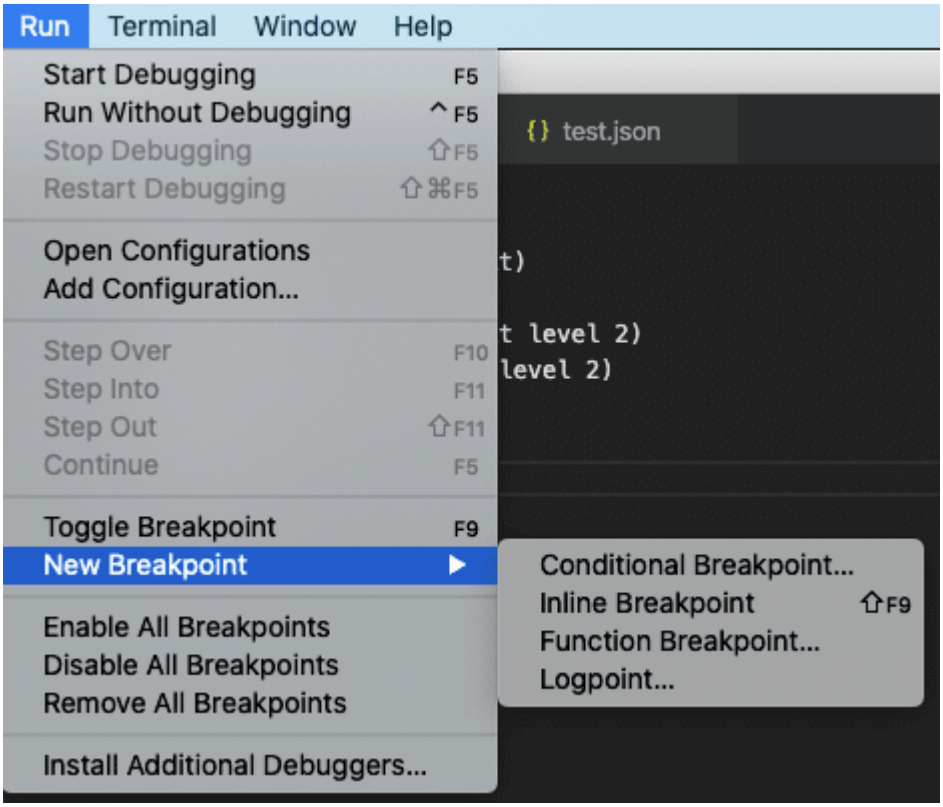
В представлении "Запуск и отладка" отображается вся информация, относящаяся к запуску и отладке, и есть верхняя панель с командами отладки и параметрами конфигурации.

Если запуск и отладка еще не настроены (не `launch.json` созданы), VS Code отображает представление запуска Run.



Меню "Выполнить"

Меню Выполнить верхнего уровня содержит наиболее распространенные команды запуска и отладки:

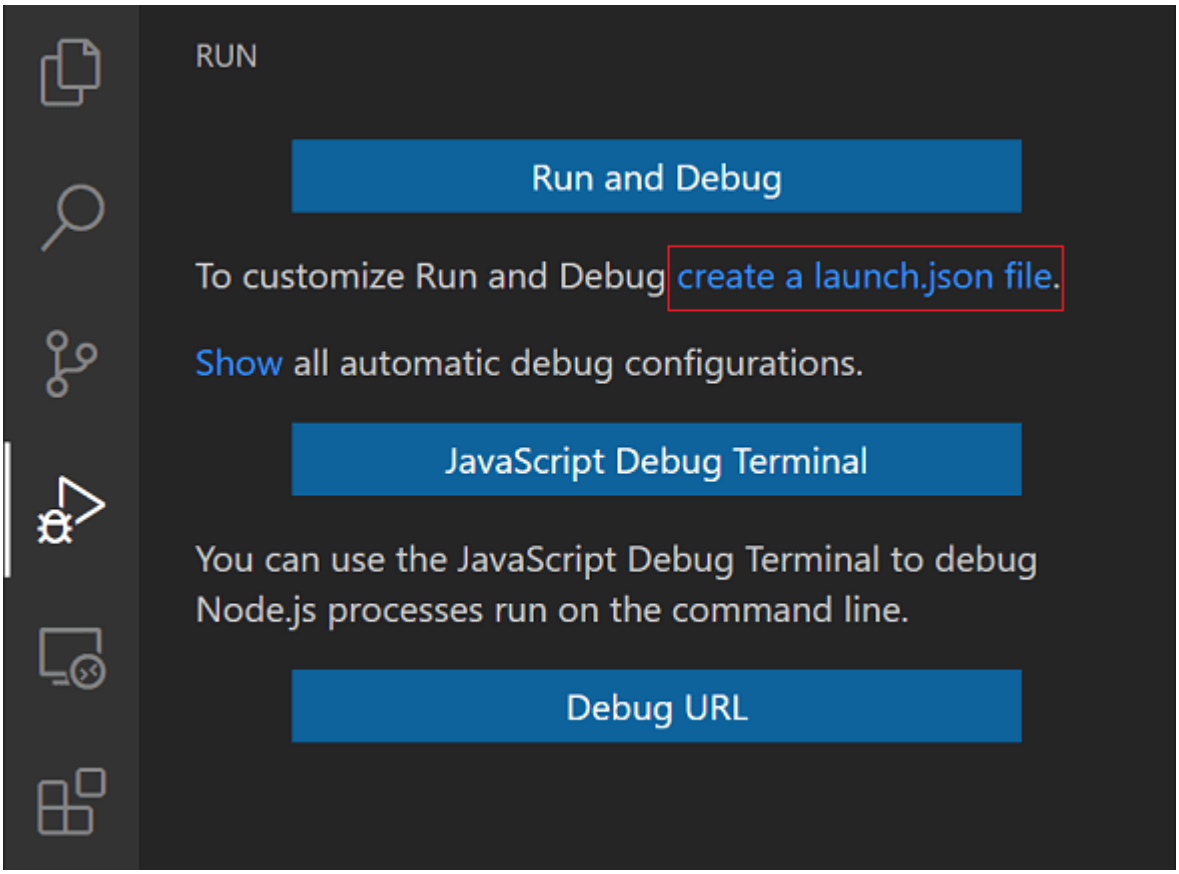


Конфигурации запуска

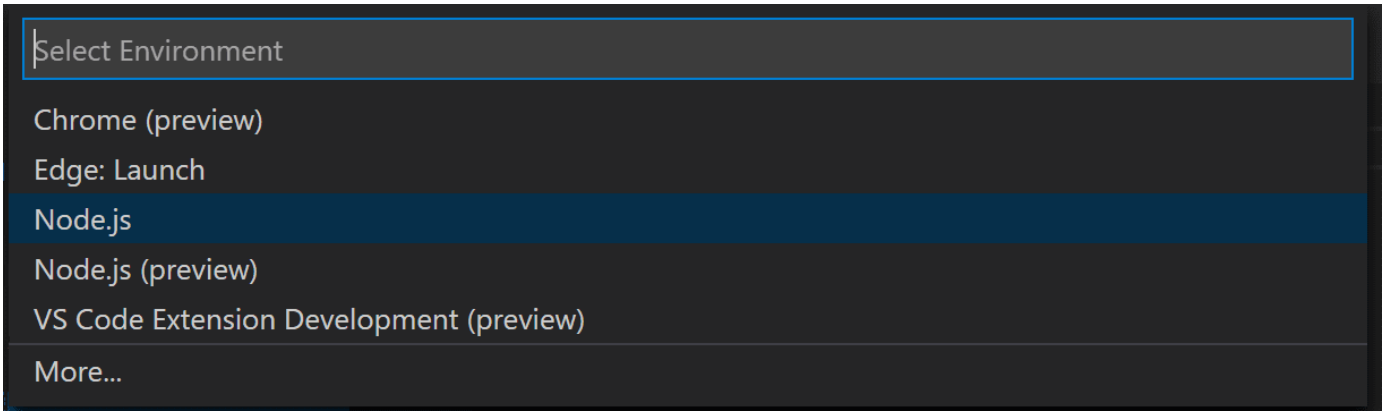
Чтобы запустить или отладить простое приложение в VS Code, выберите Выполнить и отладить в окне Запуска отладки или нажмите F5 , и VS Code попытается запустить ваш текущий активный файл.

Однако для большинства сценариев отладки полезно создать файл конфигурации запуска, поскольку он позволяет настроить и сохранить сведения о настройке отладки. VS Code хранит информацию о конфигурации отладки в launch.json файле, расположенном в .vscode папке в вашей рабочей области (корневая папка проекта) или в ваших пользовательских настройках (/docs/editor/debugging#_global-launch-configuration) или в настройках рабочей области (/docs/editor/multi-root-workspaces#_workspace-launch-configurations).

To create a launch.json file, click the create a launch.json file link in the Run start view.



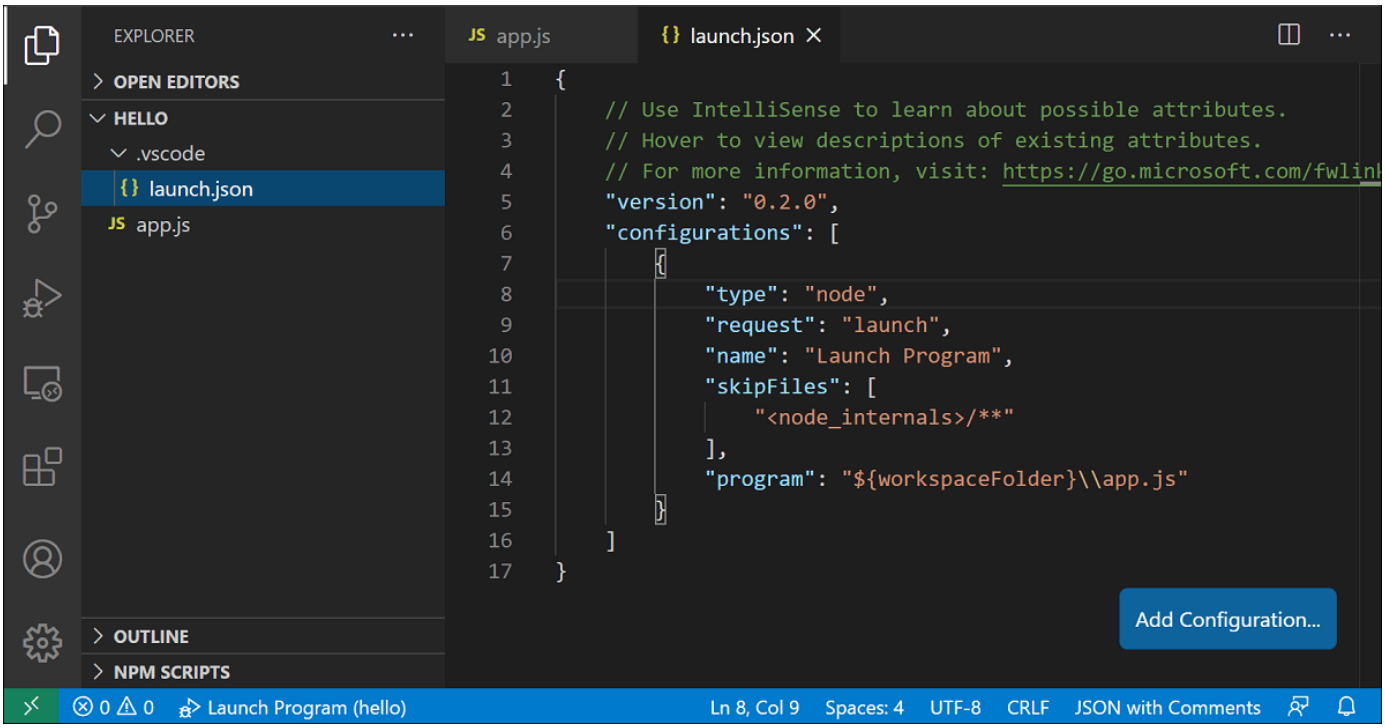
VS Code will try to automatically detect your debug environment, but if this fails, you will have to choose it manually:



Here is the launch configuration generated for Node.js debugging:

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "node",
      "request": "launch",
      "name": "Launch Program",
      "skipFiles": ["<node_internals>/**"],
      "program": "${workspaceFolder}\\app.js"
    }
  ]
}
```

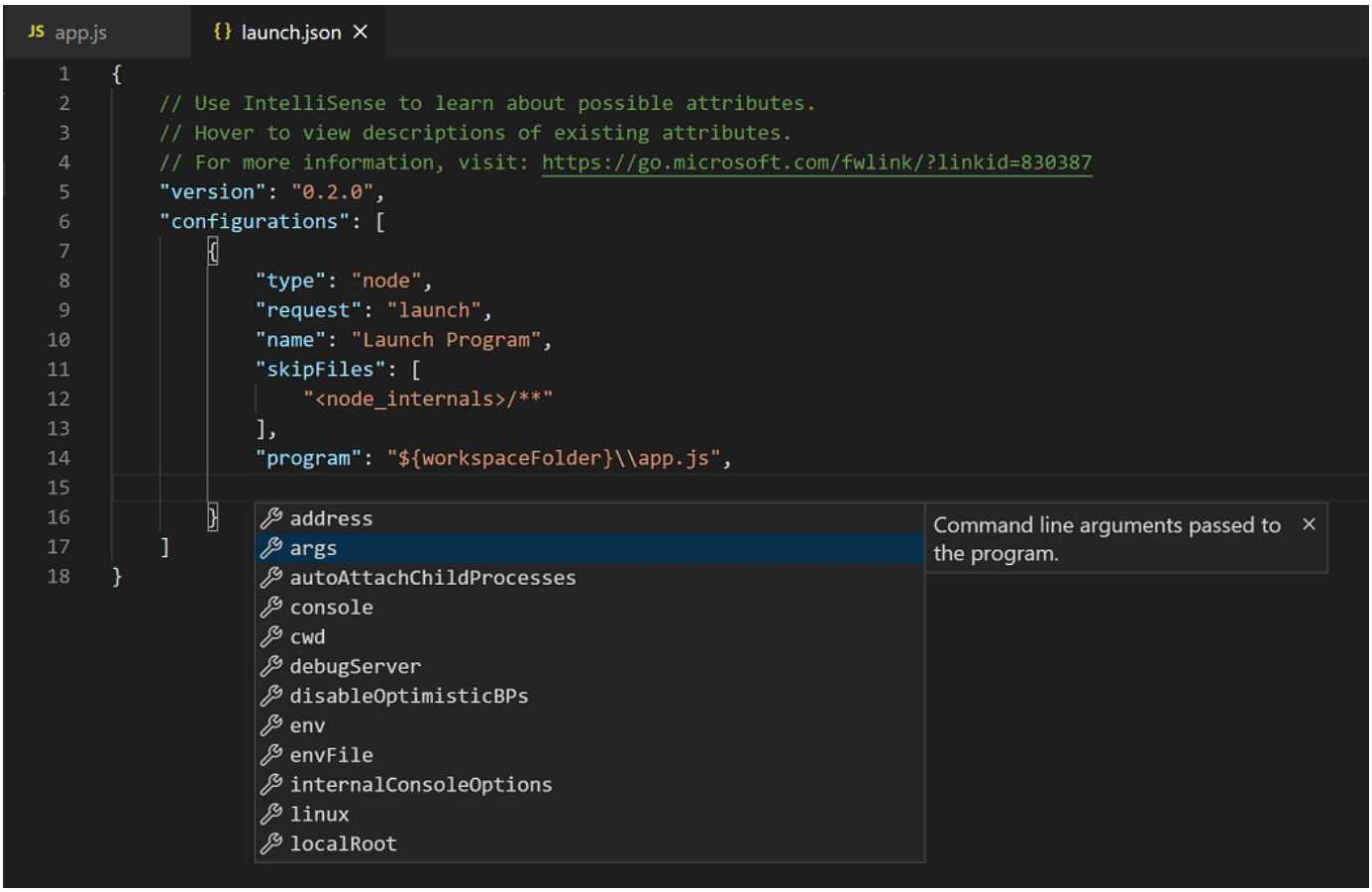
Если вы вернетесь в проводник (Ctrl + Shift + E), вы увидите, что VS Code создал .vscode папку и добавил launch.json файл в вашу рабочую область.



> Примечание: Вы можете отлаживать простое приложение, даже если у вас нет открытой папки в VS Code, но управлять конфигурациями запуска и настраивать расширенную отладку невозможно. Строка состояния VS Code становится фиолетовой, если у вас не открыта папка.

Обратите внимание, что атрибуты, доступные в конфигурациях запуска, различаются от отладчика к отладчику. Вы можете использовать подсказки IntelliSense (`Ctrl + Пробел`), чтобы узнать, какие атрибуты существуют для конкретного отладчика. Справка по наведению курсора мыши также доступна для всех атрибутов.

Не предполагайте, что атрибут, доступный для одного отладчика, автоматически работает и для других отладчиков. Если вы видите красные закорючки в конфигурации запуска, наведите на них курсор, чтобы узнать, в чем проблема, и попробуйте исправить их перед запуском сеанса отладки.



Просмотрите все автоматически сгенерированные значения и убедитесь, что они имеют смысл для вашего проекта и среды отладки.

Конфигурации запуска и присоединения

В VS Code есть два основных режима отладки: Launch и Attach, которые управляют двумя разными рабочими процессами и сегментами разработчиков. В зависимости от вашего рабочего процесса может возникнуть путаница при определении того, какой тип конфигурации подходит для вашего проекта.

Если вы работаете с инструментами разработчика браузера, возможно, вы не привыкли "запускать из своего инструмента", поскольку экземпляр вашего браузера уже открыт. Когда вы открываете DevTools, вы просто прикрепляете DevTools к открытой вкладке вашего браузера. С другой стороны, если вы работаете с сервера или с рабочего стола, вполне нормально, что ваш редактор запускает ваш процесс за вас, и ваш редактор автоматически подключает свой отладчик к только что запущенному процессу.

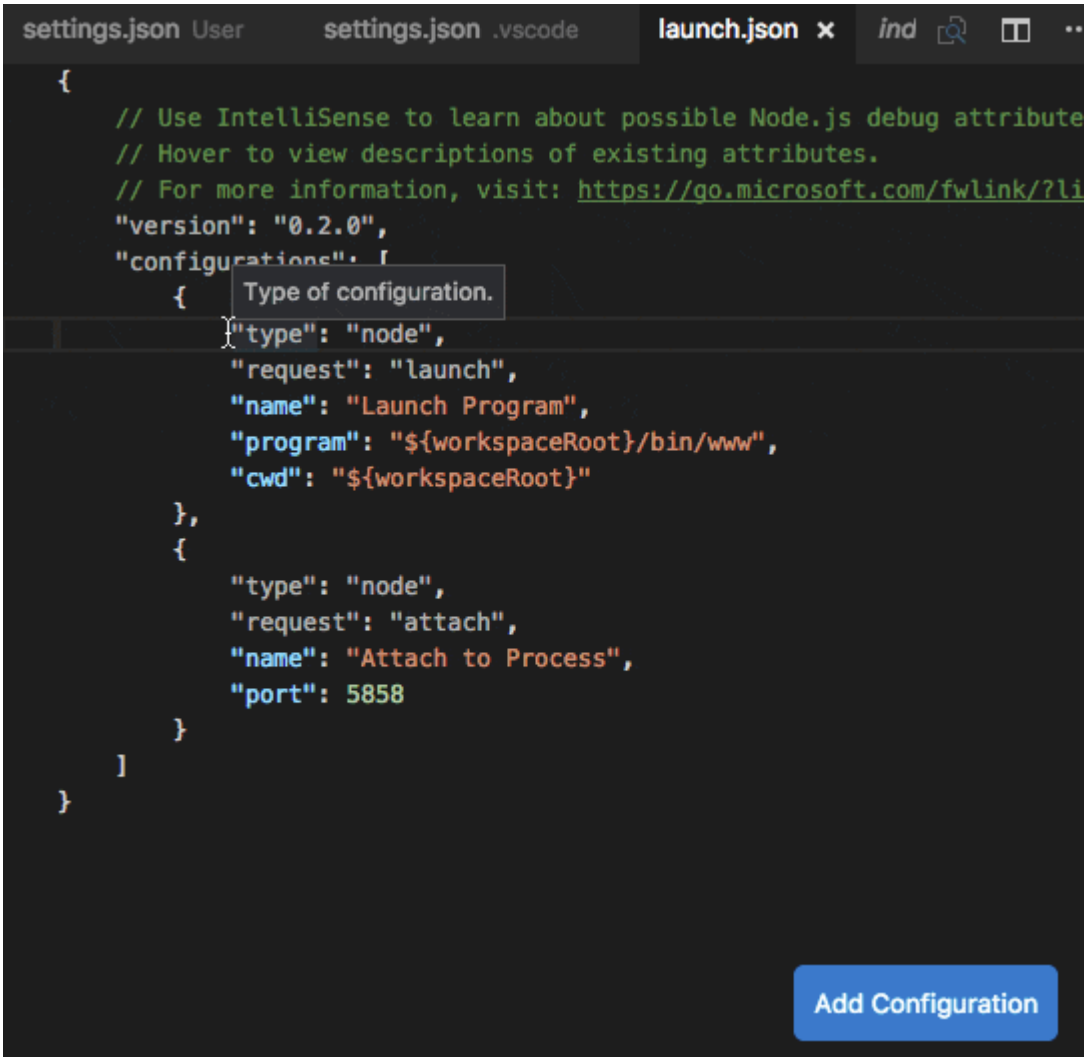
Лучший способ объяснить разницу между launch и attach - рассматривать конфигурацию launch как рецепт запуска вашего приложения в режиме отладки, прежде чем к нему подключится VS Code, в то время как конфигурация attach - это рецепт подключения отладчика VS Code к приложению или процессу, которые уже запущены.

Отладчики VS Code обычно поддерживают запуск программы в режиме отладки или подключение к уже запущенной программе в режиме отладки. В зависимости от запроса (`attach` или `launch`) требуются разные атрибуты, и `launch.json` проверка и предложения VS Code должны помочь в этом.

Добавление новой конфигурации

Чтобы добавить новую конфигурацию к существующей `launch.json` , используйте один из следующих методов:

- Используйте IntelliSense, если ваш курсор находится внутри массива configuration .
- Нажмите кнопку Добавить конфигурацию, чтобы вызвать фрагмент IntelliSense в начале массива.
- Выберите опцию Добавить конфигурацию в меню "Выполнить".

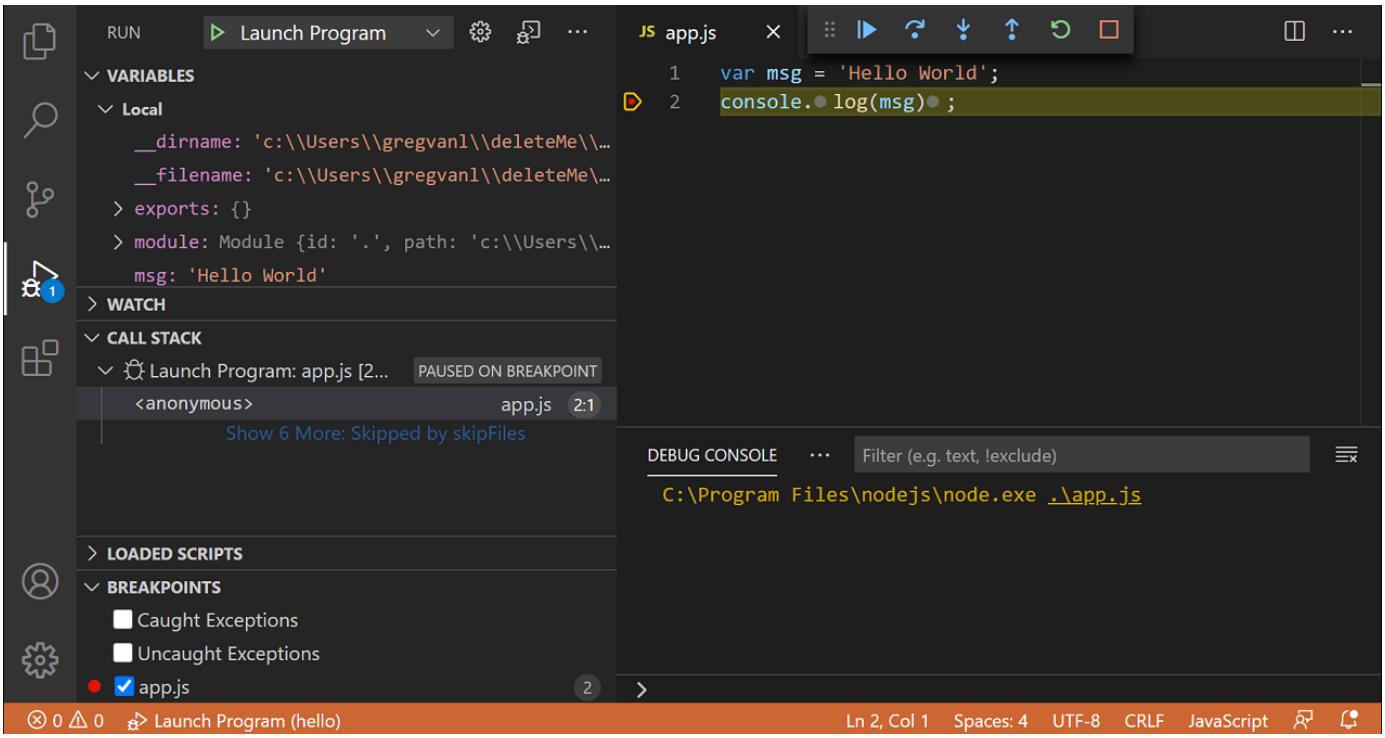


VS Code также поддерживает составные конфигурации запуска для одновременного запуска нескольких конфигураций; для получения более подробной информации, пожалуйста, прочтите этот раздел.

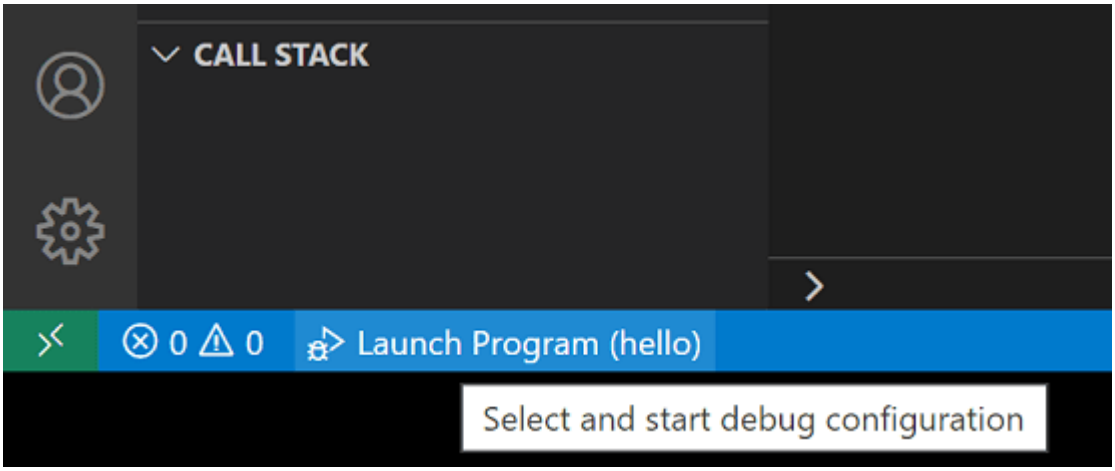
Чтобы запустить сеанс отладки, сначала выберите конфигурацию с именем Launch Program, используя раскрывающийся список "Конфигурация" в представлении "Запуск и отладка". После того, как вы настроили конфигурацию запуска, начните сеанс отладки с помощью F5 .

Alternatively, you can run your configuration through the Command Palette (Ctrl+Shift+P) by filtering on Debug: Select and Start Debugging or typing 'debug ' and selecting the configuration you want to debug.

As soon as a debugging session starts, the DEBUG CONSOLE panel is displayed and shows debugging output, and the Status Bar changes color (orange for default color themes):



Кроме того, в строке состояния отображается состояние отладки, показывающее активную конфигурацию отладки. Выбрав состояние отладки, пользователь может изменить активную конфигурацию запуска и начать отладку без необходимости открывать представление "Запуск и отладка".



Действия по отладке

После запуска сеанса отладки в верхней части редактора появится панель инструментов отладки.

Экшен	Объяснение
Продолжить / Приостановить F5	Продолжить: Возобновить обычное выполнение программы / скрипта (до следующей точки останова). Приостановка: проверьте код, выполняющийся в текущей строке, и отладьте построчно.

Экшен	Объяснение
Пошаговое руководство F10	Выполните следующий метод как отдельную команду, не проверяя и не следуя шагам его компонента.
Переход к F11	Введите следующий метод, чтобы построчно проследить за его выполнением.
Выходим Shift + F11	Находясь внутри метода или подпрограммы, вернитесь к предыдущему контексту выполнения, заполнив оставшиеся строки текущего метода, как если бы это была одна команда.
Перезапуск Ctrl + Shift + F5	Завершите текущее выполнение программы и снова запустите отладку, используя текущую конфигурацию запуска.
Остановка Shift + F5	Завершите текущее выполнение программы.

> Совет: Используйте параметр debug.toolBarLocation для управления расположением панели инструментов отладки. Это может быть значение по умолчанию floating , docked для представления Run and Debug или hidden . floating Панель инструментов отладки можно перетаскивать по горизонтали, а также вниз, в область редактора.

Режим запуска

В дополнение к отладке программы, VS Code поддерживает запуск программы. Действие Debug: Run (Запуск без отладки) запускается с помощью Ctrl + F5 и использует выбранную в данный момент конфигурацию запуска. Многие атрибуты конфигурации запуска поддерживаются в режиме "Выполнить". VS Code поддерживает сеанс отладки во время работы программы, и нажатие кнопки Stop завершает работу программы.

> Совет: Действие Run доступно всегда, но не все расширения отладчика поддерживают "Run". В этом случае 'Run' будет таким же, как 'Debug'.

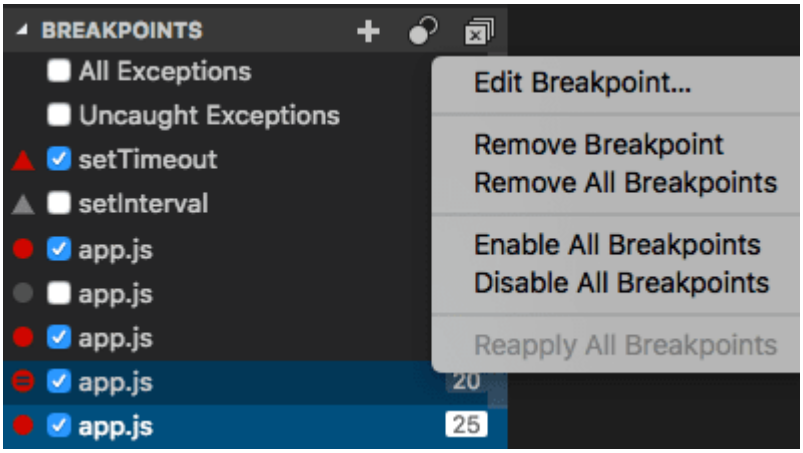
Точки останова

Точки останова можно переключать, нажимая на поле редактора или используя клавишу F9 в текущей строке. Более точное управление точками останова (включение / выключение / повторное применение) можно выполнить в разделе ТОЧЕК ОСТАНОВА представления Run and Debug.

- Точки останова на полях редактора обычно отображаются в виде красных закрашенных кружков.
- Отключенные точки останова отмечены закрашенным серым кружком.
- При запуске сеанса отладки точки останова, которые не могут быть зарегистрированы в отладчике, изменяются на серый пустой круг. То же самое может произойти, если исходный код редактируется во время сеанса отладки без поддержки оперативного редактирования.

Если отладчик поддерживает прерывание при различных типах ошибок или исключений, они также будут доступны в представлении ТОЧЕК ОСТАНОВА.

Команда Повторно применить все контрольные точки снова устанавливает все контрольные точки в их исходное местоположение. Это полезно, если ваша среда отладки "ленива" и "неправильно расставляет" точки останова в исходном коде, который еще не был выполнен.

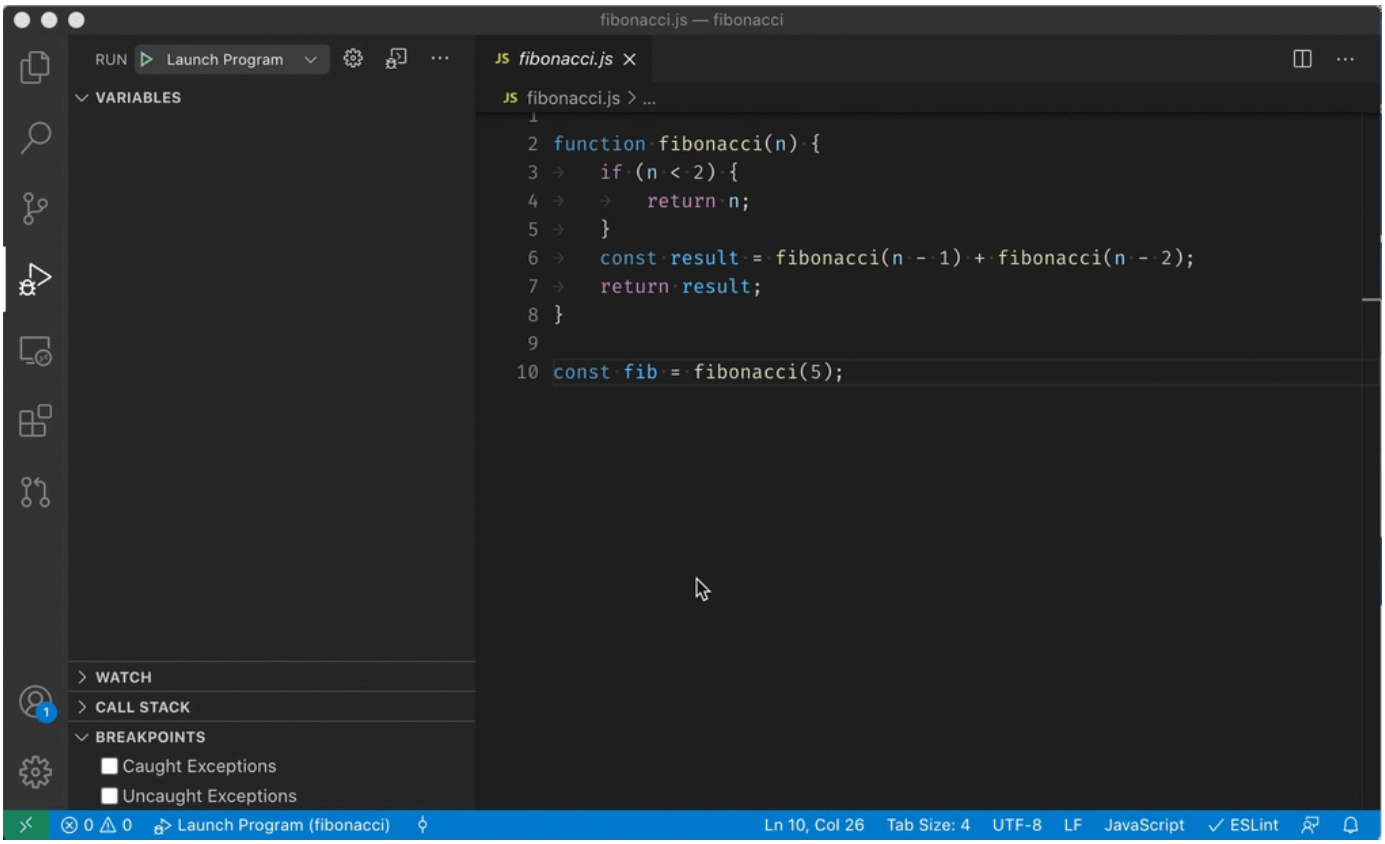


Optionally, breakpoints can be shown in the editor's overview ruler by enabling the setting `debug.showBreakpointsInOverviewRuler`:

Точки входа в систему

Точка входа - это вариант точки останова, которая не "проникает" в отладчик, а вместо этого регистрирует сообщение на консоли. Точки входа особенно полезны для внедрения ведения журнала при отладке производственных серверов, которые нельзя приостановить или остановить.

Точка входа представлена значком в форме ромба. Сообщения журнала представляют собой обычный текст, но могут включать выражения, которые должны оцениваться в фигурных скобках ('{}').



Как и обычные точки останова, точки входа могут быть включены или отключены, а также могут управляться условием и / или количеством совпадений.

Примечание: Точки входа поддерживаются встроенным в VS Code отладчиком Node.js, но могут быть реализованы другими расширениями отладки. Расширения Python (/docs/python/python-tutorial) и Java (/docs/java/java-tutorial), например, поддерживают точки входа.

Срабатывающие точки останова

Триггерная точка останова - это точка останова, которая автоматически включается при достижении другой точки останова. Они могут быть очень полезны при диагностике случаев сбоев в коде, которые происходят только после выполнения определенного предварительного условия.

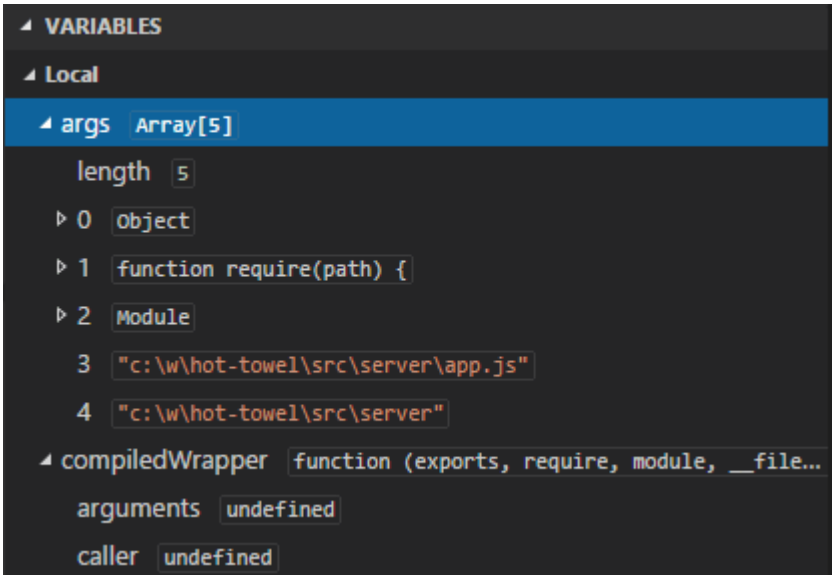
Срабатывающие точки останова можно установить, щелкнув правой кнопкой мыши на поле глифа, выбрав Добавить срабатывающую точку останова, а затем выбрав, какая другая точка останова включает эту точку останова.

0:00 / 0:13

Срабатывающие точки останова работают для всех языков, и условные точки останова также могут использоваться в качестве триггера.

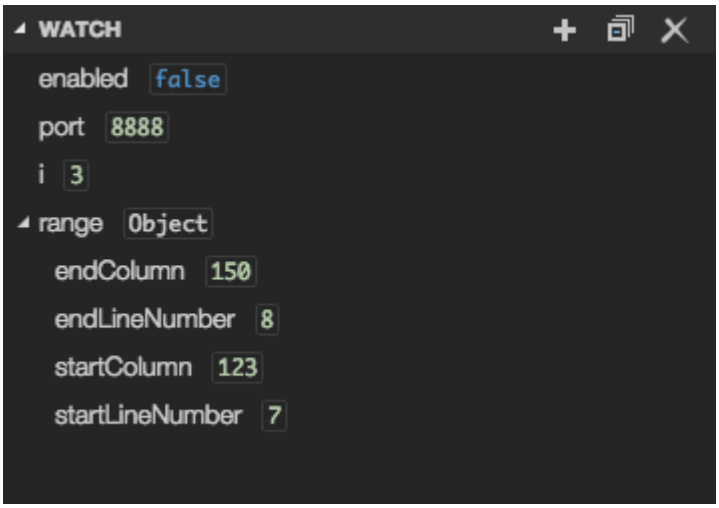
Проверка данных

Переменные можно просмотреть в разделе ПЕРЕМЕННЫЕ представления Запуск и отладка или наведя указатель мыши на их исходный код в редакторе. Значения переменных и вычисление выражений относятся к выбранному фрейму стека в разделе СТЕК ВЫЗОВОВ.

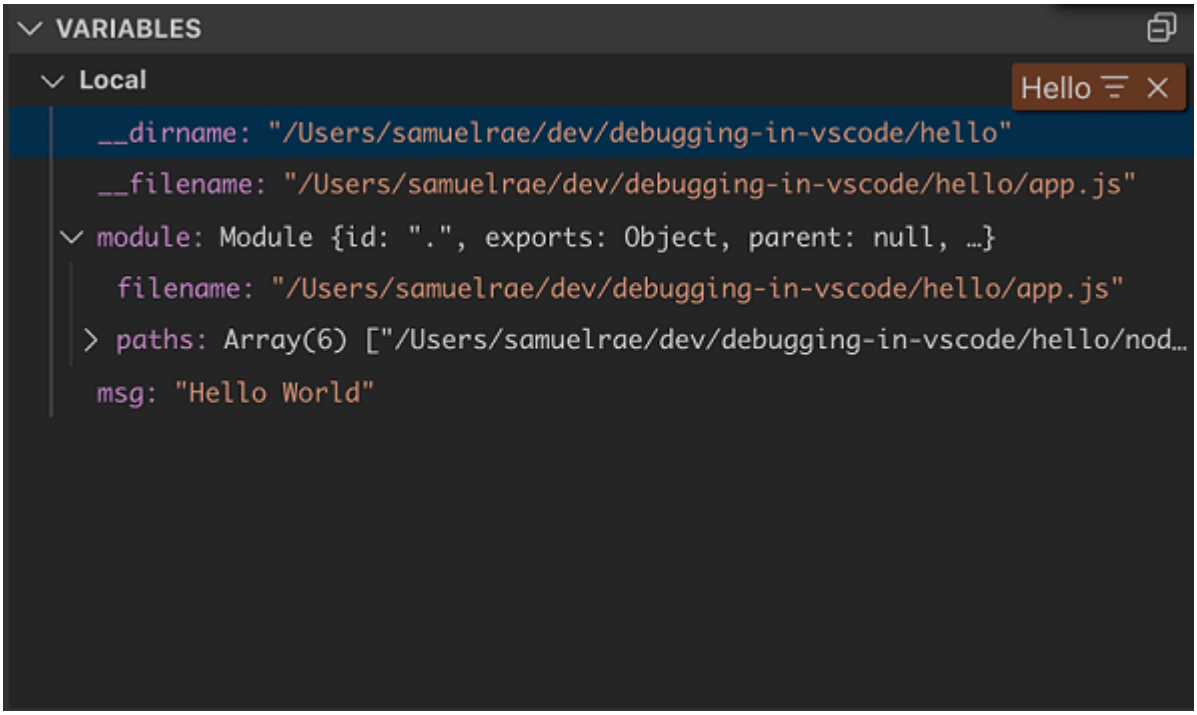


Значения переменных можно изменять с помощью действия Set Value из контекстного меню переменной. Кроме того, вы можете использовать действие Копировать значение, чтобы скопировать значение переменной, или действие Копировать как выражение, чтобы скопировать выражение для доступа к переменной.

Переменные и выражения также можно вычислять и просматривать в разделе WATCH представления Run and Debug.

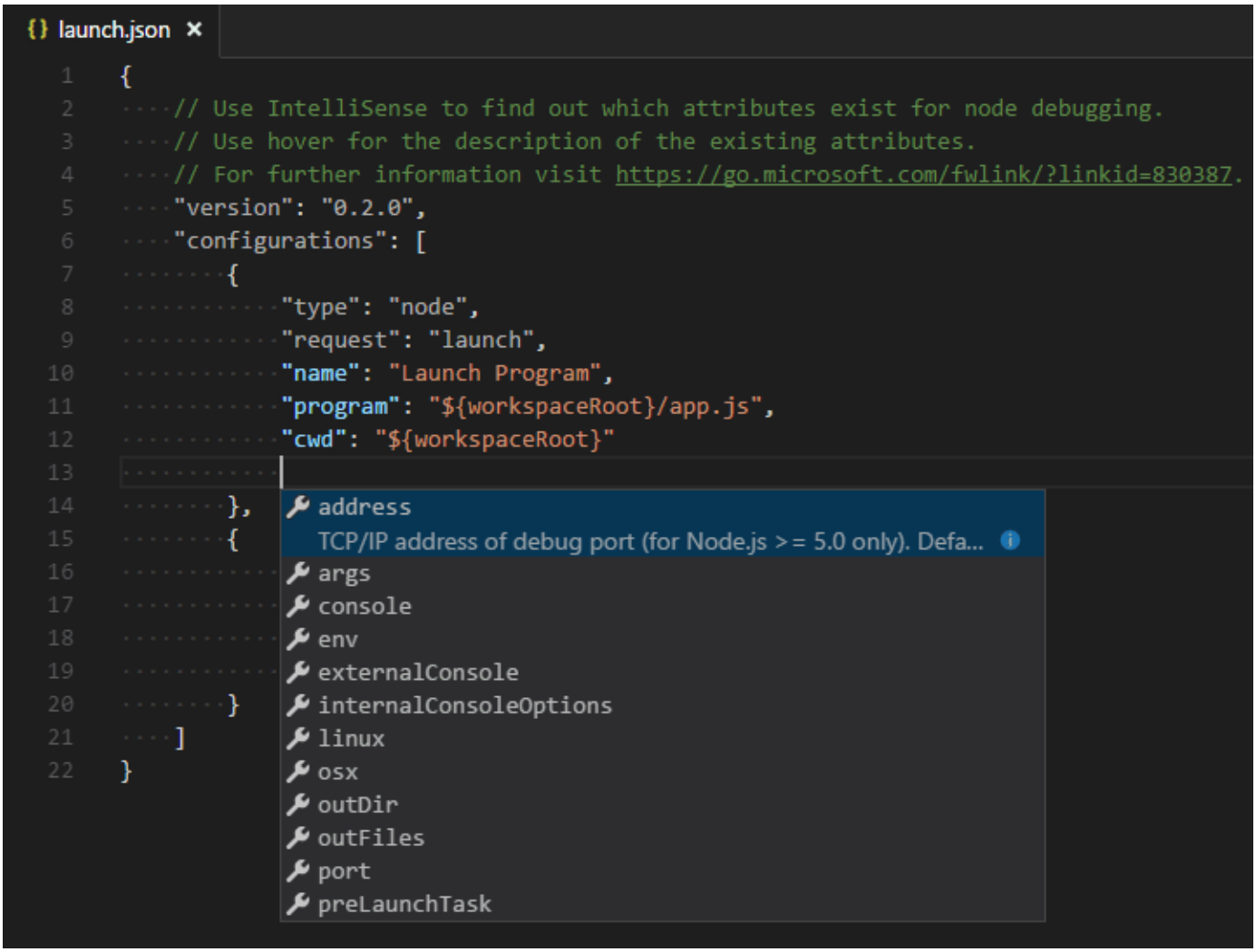


Имена и значения переменных можно отфильтровать, введя их, пока основное внимание уделяется разделу ПЕРЕМЕННЫЕ.



Атрибуты Launch.json

Существует множество `launch.json` атрибутов, помогающих поддерживать различные отладчики и сценарии отладки. Как упоминалось выше, вы можете использовать IntelliSense (`Ctrl + Пробел`), чтобы просмотреть список доступных атрибутов после того, как вы указали значение для `type` атрибута.



Следующие атрибуты обязательны для каждой конфигурации запуска:

- `type` - тип отладчика, который будет использоваться для данной конфигурации запуска. Каждое установленное расширение debug вводит тип: `node` например, для встроенного отладчика узлов или `php` и `go` для расширений PHP и Go.
- `request` - тип запроса для этой конфигурации запуска. В настоящее время поддерживаются `launch` и `attach`.
- `name` - удобное для чтения имя, которое отображается в раскрывающемся списке конфигурации запуска отладки.

Вот некоторые необязательные атрибуты, доступные для всех конфигураций запуска:

- `presentation` - используя `order`, `group` и `hidden` атрибуты в `presentation` объекте, вы можете сортировать, группировать и скрывать конфигурации и соединения в раскрывающемся списке Debug configuration и в окне быстрого выбора Debug.
- `preLaunchTask` - чтобы запустить задачу перед началом сеанса отладки, установите этот атрибут на метку задачи, указанную в tasks.json (/docs/editor/tasks) (в `.vscode` папке workspace). Или это значение можно установить в `${defaultBuildTask}`, чтобы использовать вашу задачу сборки по умолчанию.
- `postDebugTask` - чтобы запустить задачу в самом конце сеанса отладки, присвоите этому атрибуту имя задачи, указанное в задачи.json (/docs/editor/tasks) (в `.vscode` папке рабочей области).
- `internalConsoleOptions` - этот атрибут управляет видимостью панели консоли отладки во время сеанса отладки.

- `debugServer` - только для авторов расширений debug: этот атрибут позволяет подключаться к указанному порту вместо запуска адаптера отладки.
- `serverReadyAction` - если вы хотите открывать URL-адрес в веб-браузере всякий раз, когда программа в режиме отладки выводит определенное сообщение на консоль отладки или встроенный терминал. Подробнее смотрите раздел Автоматическое открытие URI при отладке серверной программы ниже.

Многие отладчики поддерживают некоторые из следующих атрибутов:

- `program` - исполняемый файл для запуска при запуске отладчика
- `args` - аргументы, передаваемые программе для отладки
- `env` - переменные среды (значение `null` может использоваться для "отмены определения" переменной)
- `envFile` - путь к файлу dotenv с переменными окружения
- `cwd` текущий рабочий каталог для поиска зависимостей и других файлов
- `port` - перенос при подключении к запущенному процессу
- `stopOnEntry` - прерывается немедленно при запуске программы
- `console` - какую консоль использовать, например, `internalConsole`, `integratedTerminal` или `externalTerminal`

Подстановка переменных

VS Code предоставляет часто используемые пути и другие значения в качестве переменных и поддерживает замену переменных внутри строк в `launch.json`. Это означает, что вам не нужно использовать абсолютные пути в конфигурациях отладки. Например, `${workspaceFolder}` указывает корневой путь к папке рабочей области, `${file}` файл, открытый в активном редакторе, и `${env:Name}` переменную среды 'Name'. Вы можете просмотреть полный список предопределенных переменных в ссылке на переменные (/docs/editor/variables-reference) или вызвав IntelliSense внутри `launch.json` строковых атрибутов.

```
{
  "type": "node",
  "request": "launch",
  "name": "Launch Program",
  "program": "${workspaceFolder}/app.js",
  "cwd": "${workspaceFolder}",
  "args": ["${env:USERNAME}"]
}
```

Свойства, зависящие от платформы

`Launch.json` поддерживает определение значений (например, аргументов для передачи программе), которые зависят от операционной системы, в которой запущен отладчик. Для этого поместите литерал для конкретной платформы в `launch.json` файл и укажите соответствующие свойства внутри этого литерала.

Ниже приведен пример, который по-другому передается `"args"` программе в Windows:

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "node",
      "request": "launch",
      "name": "Launch Program",
      "program": "${workspaceFolder}/node_modules/gulp/bin/gulpfile.js",
      "args": ["myFolder/path/app.js"],
      "windows": {
        "args": ["myFolder\\path\\app.js"]
      }
    }
  ]
}
```

Допустимые операционные свойства - `"windows"` для Windows, `"linux"` для Linux и `"osx"` для macOS. Свойства, определенные в области, специфичной для операционной системы, переопределяют свойства, определенные в глобальной области.

Пожалуйста, обратите внимание, что `type` свойство не может быть размещено внутри раздела, зависящего от платформы, потому что `type` косвенно определяет платформу в сценариях удаленной отладки, и это привело бы к циклической зависимости.

В приведенном ниже примере отладка программы всегда останавливается при входе, за исключением macOS:

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "node",
      "request": "launch",
      "name": "Launch Program",
      "program": "${workspaceFolder}/node_modules/gulp/bin/gulpfile.js",
      "stopOnEntry": true,
      "osx": {
        "stopOnEntry": false
      }
    }
  ]
}
```

Глобальная конфигурация запуска

VS Code поддерживает добавление `"launch"` объекта в ваши пользовательские настройки (/docs/getstarted/settings). Затем эта `"launch"` конфигурация будет доступна для всех ваших рабочих областей. Например:

```
"launch": {
    "version": "0.2.0",
    "configurations": [{
        "type": "node",
        "request": "launch",
        "name": "Launch Program",
        "program": "${file}"
    }]
}
```

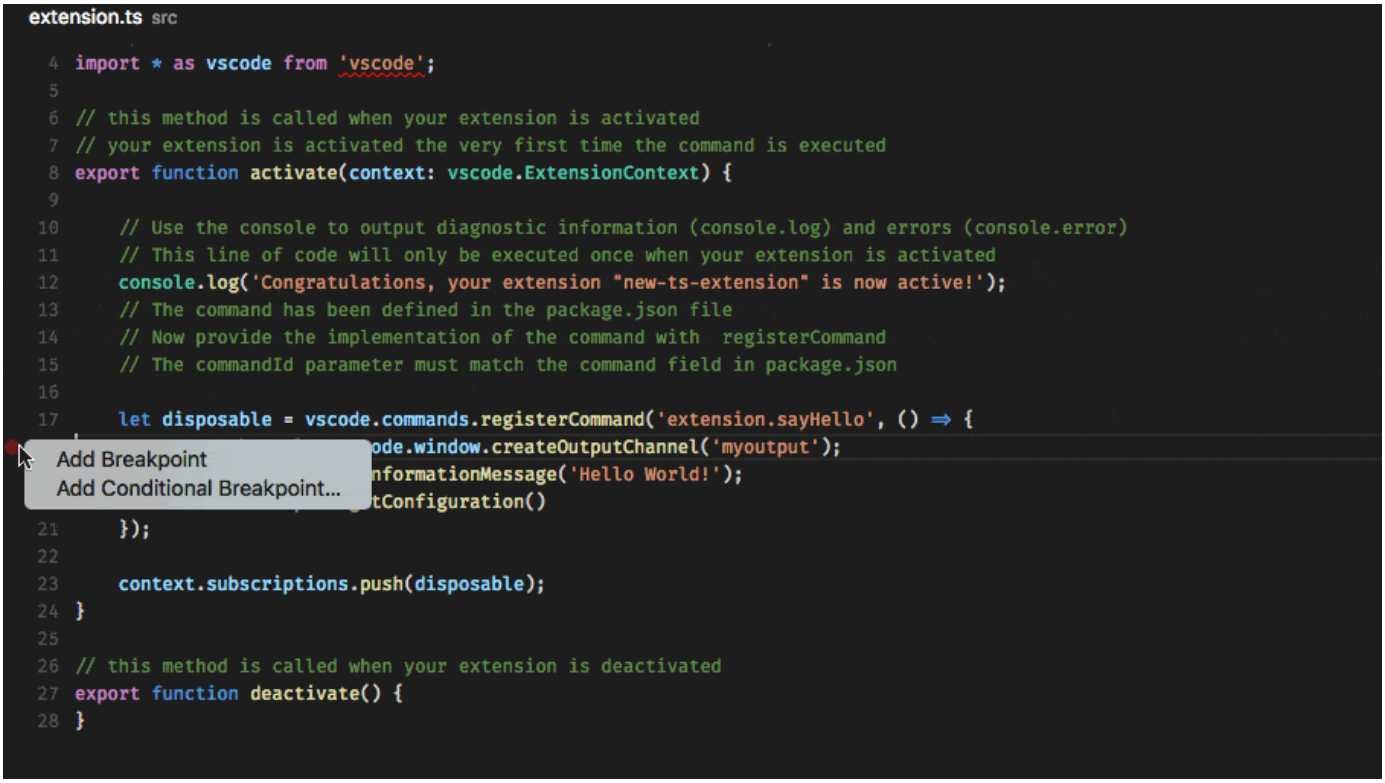
Расширенные разделы о точках останова

Условные точки останова

Мощной функцией отладки VS Code является возможность задавать условия на основе выражений, количества попаданий или комбинации того и другого.

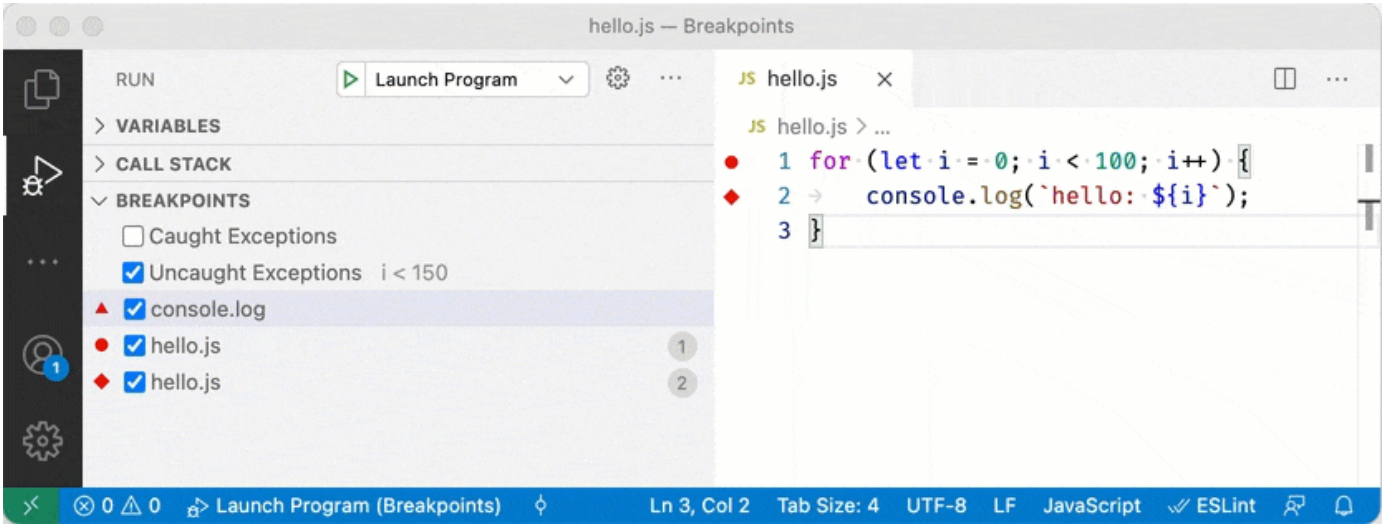
- Условие выражения: точка останова будет достигнута всякий раз, когда выражение принимает значение `true`.
- Количество попаданий: "Количество попаданий" определяет, сколько раз нужно нажать точку останова, прежде чем она "остановит" выполнение. Соблюдается ли "количество попаданий" и точный синтаксис выражения различаются в зависимости от расширений отладчика.

Вы можете добавить условие и / или количество попаданий при создании исходной точки останова (с помощью действия "Добавить условную точку останова") или при изменении существующей точки останова (с помощью действия "Изменить условие"). В обоих случаях открывается встроенное текстовое поле с выпадающим меню, в котором можно вводить выражения:



Поддержка редактирования условий и количества попаданий также поддерживается для точек останова функции и исключения. Вы можете инициировать редактирование условия из контекстного меню или нового встроенного действия Редактировать условие.

Пример редактирования условий в представлении ТОЧЕК ОСТАНОВА:



Если отладчик не поддерживает условные точки останова, действия Добавить условную точку останова и Изменить условие будут отсутствовать.

Встроенные точки останова

Встроенные точки останова будут достигнуты только тогда, когда выполнение достигнет столбца, связанного со встроенной точкой останова. Это особенно полезно при отладке сокращенного кода, который содержит несколько инструкций в одной строке.

Встроенную точку останова можно установить с помощью Shift + F9 или через контекстное меню во время сеанса отладки. Встроенные точки останова отображаются встроенно в редакторе.

Встроенные точки останова также могут иметь условия. Редактирование нескольких точек останова в строке возможно через контекстное меню на левом поле редактора.

Точки останова функций

Вместо размещения точек останова непосредственно в исходном коде отладчик может поддерживать создание точек останова, указывая имя функции. Это полезно в ситуациях, когда исходный код недоступен, но имя функции известно.

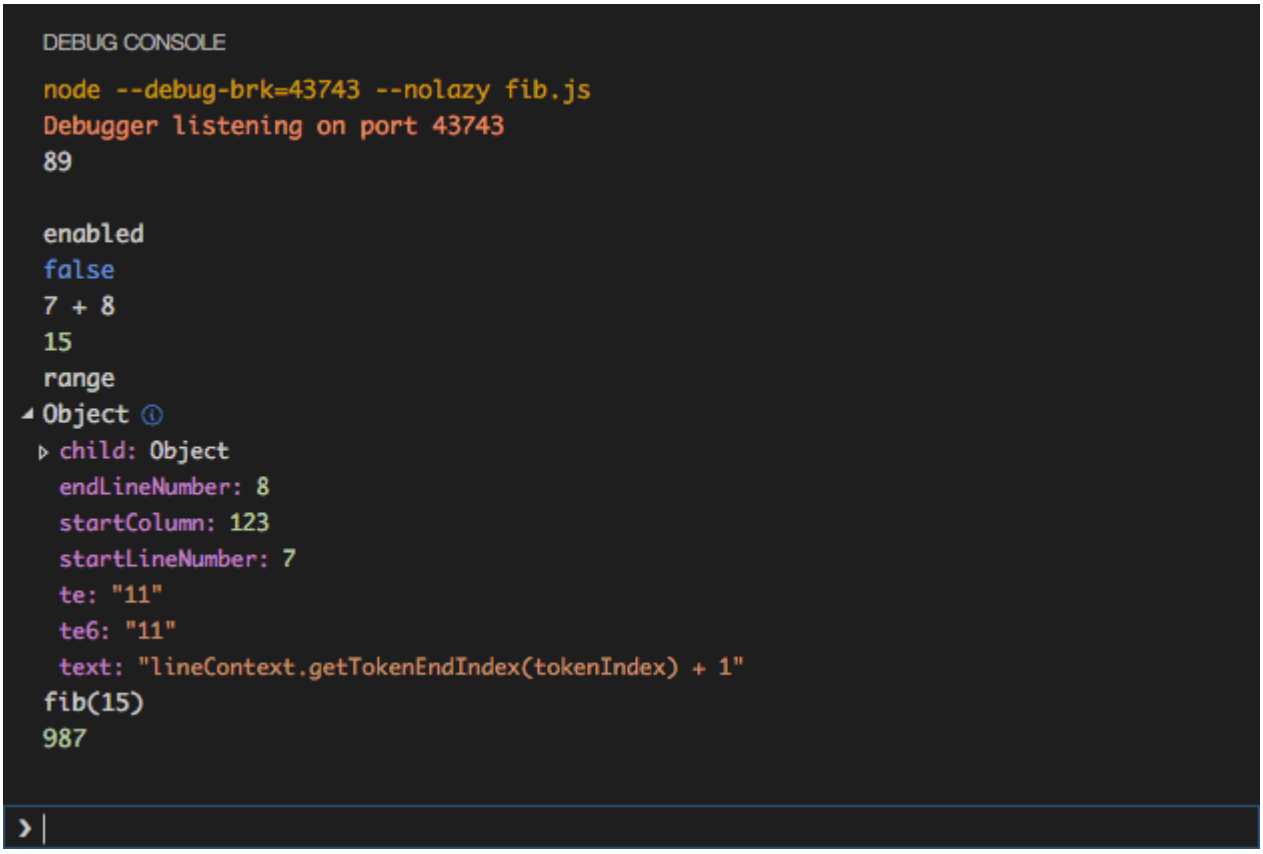
Точка останова функции создается нажатием кнопки + в заголовке раздела ТОЧКИ ОСТАНОВА и вводом имени функции. Точки останова функций показаны красным треугольником в разделе ТОЧКИ ОСТАНОВА.

Точки останова данных

Если отладчик поддерживает точки останова данных, их можно установить из контекстного меню в представлении ПЕРЕМЕННЫЕ. Команды Break on Value Change / Read /Access добавят точку останова данных, которая достигается при изменении значения базовой переменной / считывании / доступе. Точки останова данных отображаются красным шестиугольником в разделе ТОЧКИ ОСТАНОВА.

Консоль отладки REPL

Выражения могут быть вычислены с помощью функции Debug Console REPL (цикл чтения-вычисления-печати (https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop)). Чтобы открыть консоль отладки, используйте действие Консоль отладки в верхней части панели отладки или используйте команду Вид: консоль отладки (Ctrl+Shift+Y). Выражения вычисляются после нажатия Enter , а в консоли отладки REPL отображаются предложения по мере ввода. Если вам нужно ввести несколько строк, используйте Shift + Enter между строками, а затем отправьте все строки на проверку с помощью Enter . Ввод с консоли отладки использует режим активного редактора, что означает, что ввод с консоли отладки поддерживает раскраску синтаксиса, отступы, автоматическое закрытие кавычек и другие языковые функции.



Примечание: Вы должны находиться в запущенном сеансе отладки, чтобы использовать REPL консоли отладки.

Перенаправлять ввод / вывод в / из целевого объекта отладки

Перенаправление ввода / вывода зависит от отладчика / среды выполнения, поэтому в VS Code нет встроенного решения, которое работало бы для всех отладчиков.

Вот два подхода, которые вы, возможно, захотите рассмотреть:

1. Запустите программу для отладки ("debug target") вручную в терминале или командной строке и перенаправьте ввод / вывод по мере необходимости. Убедитесь, что вы передали соответствующие параметры командной строки целевому объекту отладки, чтобы отладчик мог подключиться к нему. Создайте и запустите конфигурацию отладки "attach", которая подключается к целевому объекту отладки.

2. Если используемое вами расширение отладчика может запускать цель отладки во встроенном терминале VS Code (или внешнем терминале), вы можете попробовать передать синтаксис перенаправления оболочки (например, "<" или ">") в качестве аргументов.

Вот пример `launch.json` конфигурации:

```
{
  "name": "launch program that reads a file from stdin",
  "type": "node",
  "request": "launch",
  "program": "program.js",
  "console": "integratedTerminal",
  "args": ["<", "in.txt"]
}
```

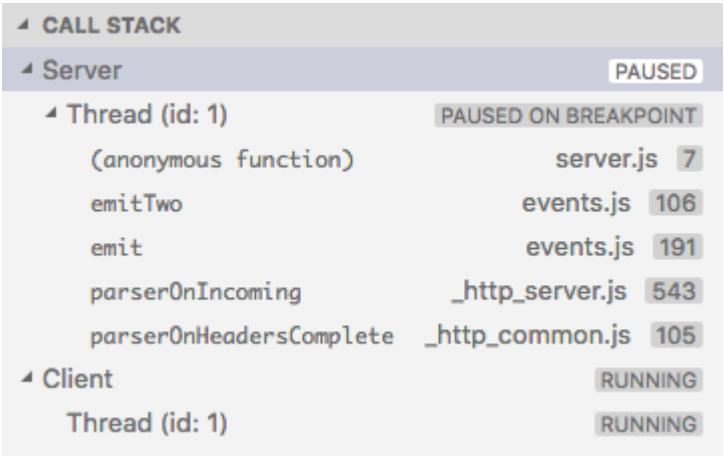
Этот подход требует, чтобы синтаксис "<" передавался через расширение отладчика и в конечном итоге оставался неизмененным во встроенном терминале.

Многоцелевая отладка

Для сложных сценариев, включающих более одного процесса (например, клиента и сервера), VS Code поддерживает многоцелевую отладку.

Использование многоцелевой отладки просто: после запуска первого сеанса отладки вы можете просто запустить другой сеанс. Как только запускается второй сеанс, пользовательский интерфейс VS Code переключается в *многоцелевой режим*:

- Отдельные сеансы теперь отображаются как элементы верхнего уровня в представлении СТЕКА ВЫЗОВОВ.



- Панель инструментов отладки показывает текущий активный сеанс (а все остальные сеансы доступны в раскрывающемся меню).

- Действия по отладке (например, все действия на панели инструментов отладки) выполняются в активном сеансе. Активный сеанс можно изменить либо с помощью выпадающего меню на панели инструментов debug, либо выбрав другой элемент в представлении СТЕКА ВЫЗОВОВ.

Составные конфигурации запуска

Альтернативным способом запуска нескольких сеансов отладки является использование составной конфигурации запуска. В конфигурации составного запуска перечислены имена двух или более конфигураций запуска, которые должны запускаться параллельно. Необязательно `preLaunchTask` может быть указан параметр, который выполняется перед запуском отдельных сеансов отладки. Логический флаг `stopAll` определяет, приведет ли ручное завершение одного сеанса к остановке всех составных сеансов.

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "node",
      "request": "launch",
      "name": "Server",
      "program": "${workspaceFolder}/server.js"
    },
    {
      "type": "node",
      "request": "launch",
      "name": "Client",
      "program": "${workspaceFolder}/client.js"
    }
  ],
  "compounds": [
    {
      "name": "Server/Client",
      "configurations": ["Server", "Client"],
      "preLaunchTask": "${defaultBuildTask}",
      "stopAll": true
    }
  ]
}
```

Составные конфигурации запуска отображаются в раскрывающемся меню "Конфигурация запуска".

Удаленная отладка

VS Code сам по себе не поддерживает удаленную отладку: это функция используемого вами расширения debug, и вам следует обратиться к странице расширения в Marketplace (https://marketplace.visualstudio.com/search?target=VSCode&category=Debuggers&sortBy=Installs) за поддержкой и подробностями.

Однако есть одно исключение: Node.js отладчик, входящий в состав VS Code, поддерживает удаленную отладку. Смотрите раздел Node.js Отладка (/docs/nodejs/nodejs-debugging#_remote-debugging), чтобы узнать, как это настроить.

Автоматическое открытие URI при отладке серверной программы

Разработка веб-программы обычно требует открытия определенного URL-адреса в веб-браузере, чтобы получить доступ к серверному коду в отладчике. В VS Code есть встроенная функция "serverReadyAction" для автоматизации этой задачи.

Вот пример простого Node.js Приложения Express (https://expressjs.com):

```
var express = require('express');
var app = express();

app.get('/', function(req, res) {
  res.send('Hello World!');
});

app.listen(3000, function() {
  console.log('Example app listening on port 3000!');
});
```

Это приложение сначала устанавливает обработчик "Hello World" для URL-адреса "/", а затем начинает прослушивать HTTP-соединения на порту 3000. Порт объявляется в консоли отладки, и обычно разработчик теперь вводит его `http://localhost:3000` в своем приложении браузера.

Функция serverReadyAction позволяет добавить структурированное свойство `serverReadyAction` в любую конфигурацию запуска и выбрать "действие", которое необходимо выполнить:

```
{
  "type": "node",
  "request": "launch",
  "name": "Launch Program",
  "program": "${workspaceFolder}/app.js",

  "serverReadyAction": {
    "pattern": "listening on port ([0-9]+)",
    "uriFormat": "http://localhost:%s",
    "action": "openExternally"
  }
}
```

Здесь свойство `pattern` описывает регулярное выражение для сопоставления выходной строки программы, которая объявляет порт. Шаблон для номера порта заключен в круглые скобки, чтобы он был доступен как группа захвата регулярных выражений. В этом примере мы извлекаем только номер порта, но также возможно извлечь полный URI.

Свойство `uriFormat` описывает, как номер порта преобразуется в URI. Первое значение `%s` заменяется первой группой захвата соответствующего шаблона.

Полученный URI затем открывается вне VS Code ("внешне") с помощью стандартного приложения, настроенного для схемы URI.

Запускаем отладку через Edge или Chrome

В качестве альтернативы, для `action` можно установить значение `debugWithEdge` или `debugWithChrome`. В этом режиме можно добавить `webRoot` свойство, которое передается сеансу отладки Chrome или Edge.

Чтобы немного упростить задачу, большинство свойств являются необязательными, и мы используем следующие резервные значения:

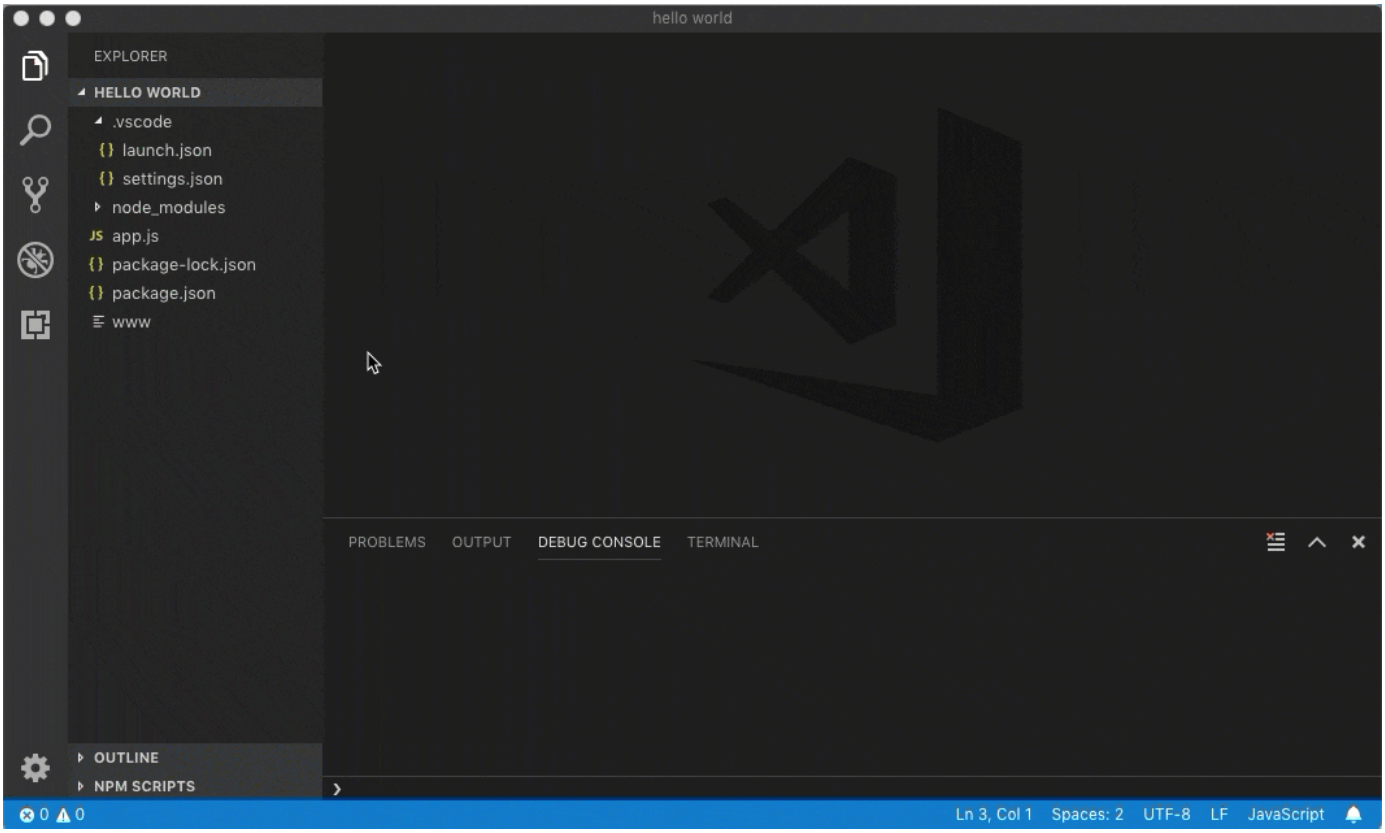
- шаблон: `"listening on.* (https?://\\S+|[0-9]+)"` который соответствует обычно используемым сообщениям "прослушивание на порту 3000" или "Теперь прослушивание: https://localhost:5001".
- uriFormat: `"http://localhost:%s"`
- webRoot: `"${workspaceFolder}"`

Запуск произвольной конфигурации запуска

В некоторых случаях может потребоваться настроить дополнительные параметры для сеанса отладки браузера или использовать полностью другой отладчик. Вы можете сделать это, установив `action` значение `startDebugging` с `name` свойством, заданным для имени конфигурации запуска, которое будет запускаться при совпадении с `pattern`.

Именованная конфигурация запуска должна находиться в том же файле или папке, что и конфигурация с `serverReadyAction`.

Здесь функция serverReadyAction в действии:



Следующие шаги

Чтобы узнать о Node.js против Кодекса поддержку отладки, взгляните на:

- Node.js (/docs/nodejs/nodejs-debugging) - Описывает Node.js отладчик, который включен в VS Code.
- TypeScript (/docs/typescript/typescript-debugging) - Отладчик Node.js также поддерживает отладку на TypeScript.

Чтобы ознакомиться с руководствами по основам Node.js отладки, ознакомьтесь с этими видеороликами:

- Вводное видео - Debugging (/docs/introvideos/debugging) - Демонстрирует основы отладки.
- Начало работы с Node.js отладка (https://www.youtube.com/watch?v=2oFKNL7vYV8) - показывает, как подключить отладчик к запущенному процессу Node.js.

Чтобы узнать о поддержке отладки для других языков программирования с помощью расширений VS Code:

- C ++ (/docs/cpp/cpp-debug)
- Python (/docs/python/debugging)
- Java (/docs/java/java-debugging)

Чтобы узнать о поддержке запуска задач в VS Code, перейдите по ссылке:

- Задачи (/docs/editor/tasks) - Описывает, как запускать задачи с помощью Gulp, Grunt и Jake и как отображать ошибки и предупреждения.

Чтобы написать собственное расширение отладчика, посетите:

- Расширение отладчика (/api/extension-guides/debugger-extension) - Использует макет примера для иллюстрации шагов, необходимых для создания расширения отладки VS Code.

Распространенные вопросы

Какие поддерживаются сценарии отладки?

Отладка приложений на основе Node.js поддерживается в Linux, macOS и Windows "из коробки" с помощью VS Code. Многие другие сценарии поддерживаются расширениями VS Code (https://marketplace.visualstudio.com/vscode/Debuggers?sortBy=Installs), доступными в Marketplace.

Я не вижу никаких конфигураций запуска в раскрывающемся списке "Запуск и отладка". Что не так?

Наиболее распространенная проблема заключается в том, что вы не настроили `launch.json` или в этом файле синтаксическая ошибка. В качестве альтернативы вам может потребоваться открыть папку, поскольку отладка без папок не поддерживает конфигурации запуска.

Была ли эта документация полезной?

Да **Нет**

02/1/2024

Subscribe(/feed.xml) Ask questions(https://stackoverflow.com/questions/tagged/vscode)

Follow @code(https://go.microsoft.com/fwlink/?LinkID=533687)

Request features(https://go.microsoft.com/fwlink/?LinkID=533482)

Report issues(https://www.github.com/Microsoft/vscode/issues)

Watch videos(https://www.youtube.com/channel/UCs5Y5_7XK8HLDX0SLNwkd3w)

Привет из Сиэтла. Следуйте @code (https://go.microsoft.com/fwlink/?LinkID=533687)

Star 156,478