

**Brian Clark** 💡 for Microsoft Azure

Posted on 30 мар. 2019 г. • Updated on 5 апр. 2019 г.



82



27

# Building a Visual Studio Code Extension

#vscode #javascript #typescript #extension

## Visual Studio Code Extension Development

Я не люблю длинных вступлений в сообщениях, потому что вы знаете, что привело вас сюда, и вам не нужны подробности. Давайте сразу перейдем к делу...

### Предположения

- Вы знаете о [Visual Studio Code \(VS Code\)](#)
- Вы знакомы с [расширениями в VS Code](#) и использовали некоторые из них, но не создавали ни одного раньше
- Вы хотя бы немного знакомы с [TypeScript](#) / JavaScript и Node.js

Если вы на самом деле не знаете ни одного из них, это все равно нормально, поскольку мы не уделяем слишком много внимания коду, но если вы чего-то не понимаете, просто оставьте комментарий ниже, и я свяжусь с вами, как только смогу.

- У вас установлено следующее:
  - [Node.js](#)
  - [npm](#) (или [yarn](#))
  - [Yeoman](#)  
`npm i -g yo`
  - [VS Code Yeoman Generator](#)  
`npm i -g generator-code`

### Какие расширения я могу создать?

Вы можете создавать все, что душе угодно. Однако вы можете создавать несколько основных "типов" расширений, которые дают вам лучшее

представление о том, с чего начать. Давайте разберем их по тому, что вы, возможно, захотите сделать.

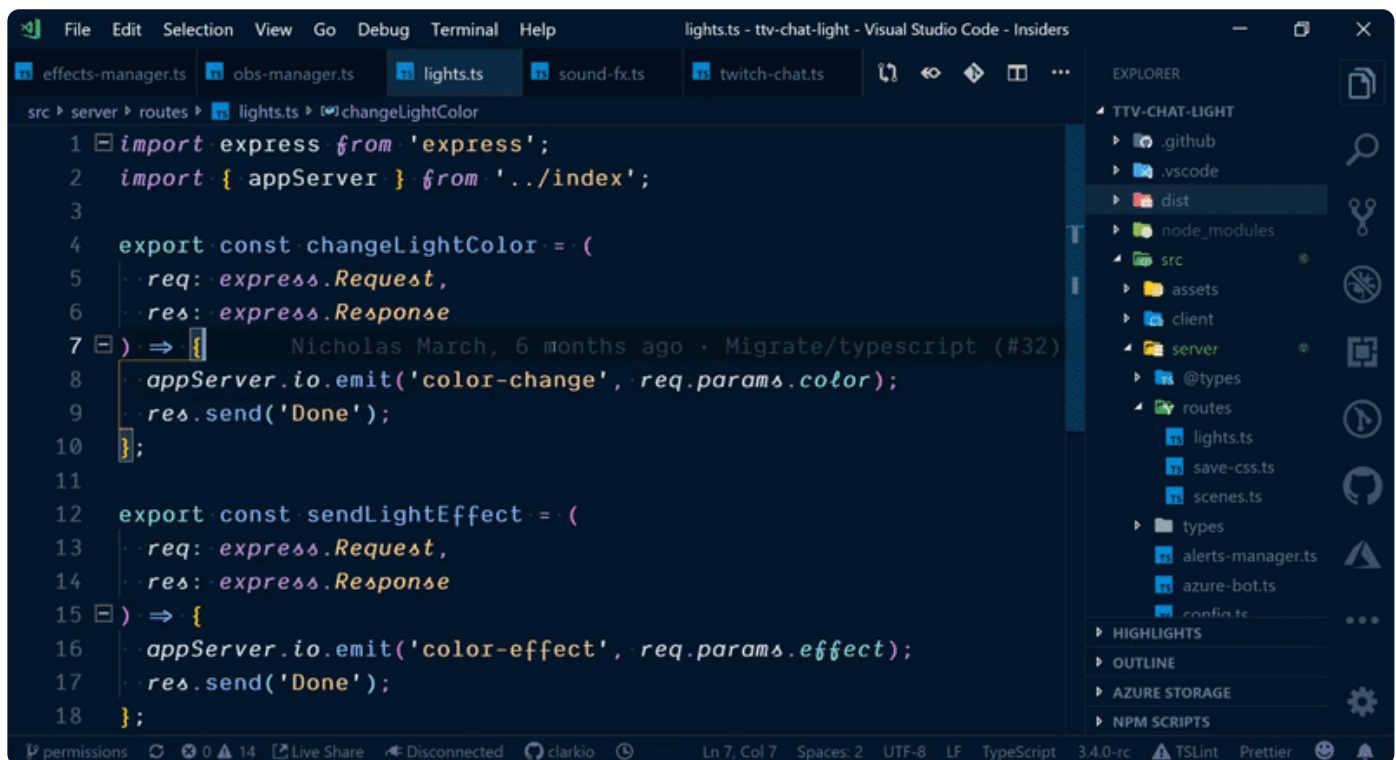
## 1. Настройте внешний вид VS Code

Что это значит? Цветовые темы и темы значков. Цветовая тема на самом деле является отличным началом для вашего первого расширения. Давайте сначала немного разберем их для лучшего понимания.

- Цветная тема

Этот тип расширений настраивает визуальный вид VS Code. Вы можете изменять цвета ооочень многих различных аспектов в пользовательском интерфейсе (UI). это может быть довольно сложно. Использование существующей темы, которая вам нравится, но которую вы хотите доработать, - отличный способ начать с создания собственной темы, которую вы действительно будете использовать. Это даст вам множество примеров изменений цвета, которые вы можете настроить и быстро увидеть результат. Однако вскоре мы увидим, насколько легко начать с нуля.

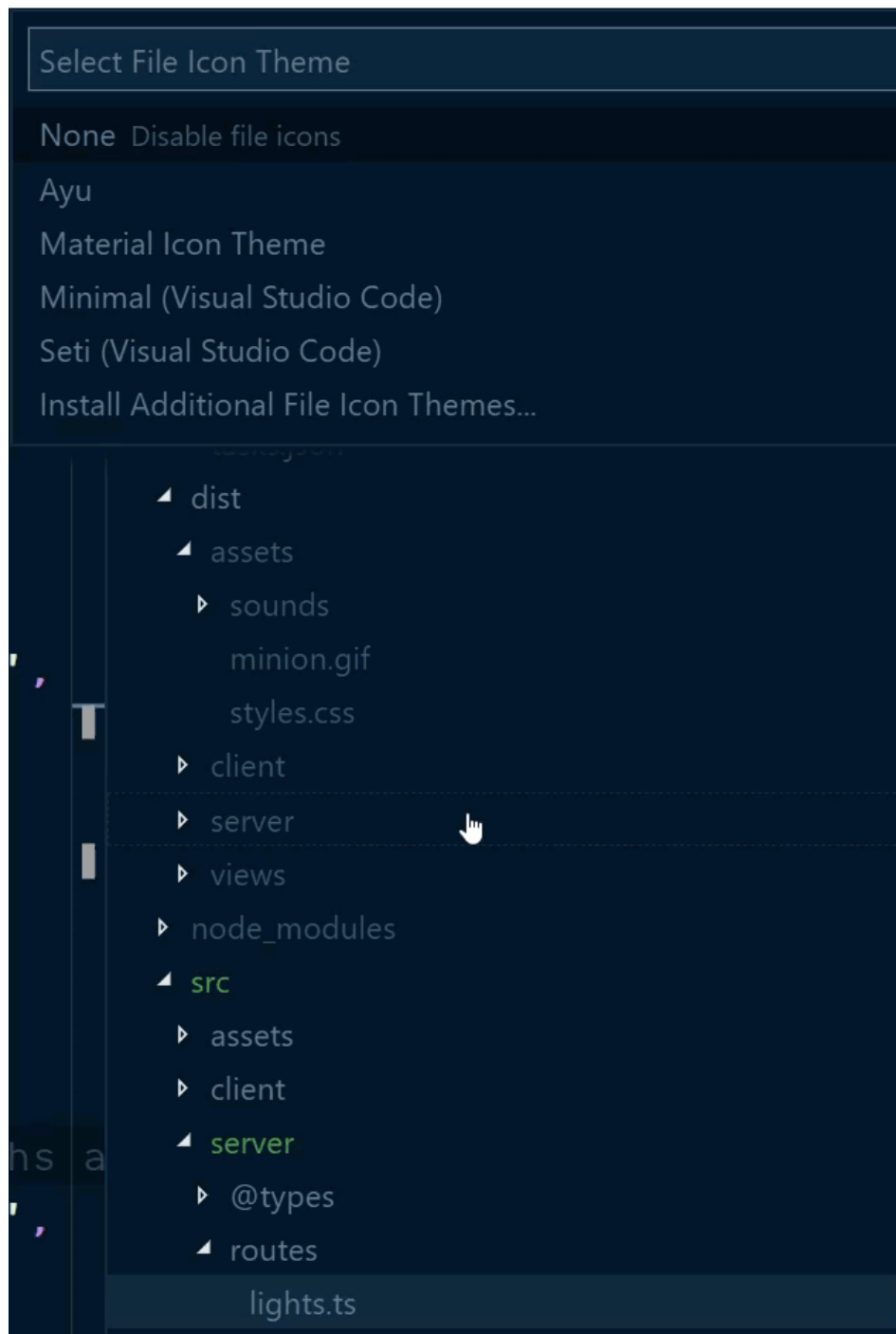
Моя любимая тема на момент написания этого поста - [Night Owl](#) [Сары Драснер](#)



- Тема значка

Этот тип расширений настраивает графику, связанную с типами файлов / папок. Отличным примером этого является [тема значков материалов](#) (которую

я использую).



GIF

## 2. Расширяйте существующую функциональность

Что это значит? Это означает, что в VS Code есть какая-то существующая возможность / фича, но вы хотели бы, чтобы в ней было что-то большее. Простым примером этого является создание новых раскладки клавиш, которые облегчают работу тем, кто привык к определенным сочетаниям клавиш, найденным в другом редакторе. Подобный пример представляет собой расширение, которое содержит фрагменты кода, чтобы создать общий код, который используется в Node.js приложения.

Вот несколько простых примеров:

- Расширение раскладки [Sublime Text](#)
- Расширение фрагмента кода [Vue.js Фрагменты](#)

Более сложные примеры:

- Улучшение управления версиями с помощью Git: [GitLens](#)
  - Улучшение читаемости кода: [раскрашиватель пар скобок](#)
- 

### 3. Добавьте новую функциональность

Что это значит? Это означает, что VS Code не предоставляет какой-либо функциональности, которую вы хотите использовать. В этом случае мы создаем расширение, которое добавляет эту функциональность.

Примеры:

- **Проблема:** отсутствует встроенная поддержка подключения к отладчику Chrome для интерфейсной веб-разработки.  
**Решение:** [отладчик для расширения Chrome](#)
  - **Проблема:** отсутствует встроенная поддержка интеллектуальных завершений (Intellisense) при написании кода на Python  
**Решение:** [расширение Python](#)
  - **Проблема:** отсутствует встроенная поддержка интеграции сообщений Twitch chat для управления выделением строк кода  
**Решение:** [выделение линий для расширения Twitch](#)
- 

### 4. Объедините другие расширения в одном месте

Что это значит? Вы объединяете множество расширений в одном пакете, которые помогают выполнять определенный фокус / категорию действий. По сути, существует множество расширений, которые сами по себе отлично подходят для того типа разработки, которым вы занимаетесь, но вы можете не знать о них всех. Кто-нибудь может собрать их вместе, чтобы упростить их поиск и установку сразу.

Примером этого является пакет расширений [Angular Essentials](#), который объединяет множество связанных с Angular расширений в одном месте. Ознакомьтесь с исходным кодом на [GitHub](#), чтобы узнать, что требуется для его создания.

Эти типы расширений определяются их `package.json` файлом для определения других расширений, которые будут включены в пакет. Смотрите пример из [package.json от Angular Essentials](#)

## Как мне создать расширение?

Одно из самых простых расширений, с которого вы можете начать, - это расширение цветовой темы. Давайте перейдем к созданию одного из них прямо сейчас.

### Создание расширения цветовой темы

Давайте создадим расширение цветовой темы с нуля, просто чтобы получить представление о создании нашего первого расширения. Мы собираемся использовать генератор VS Code Yeoman, поэтому убедитесь, что вы выполнили следующую команду в вашем терминале / командной строке:

```
npm i -g generator-code
```

В нем в качестве опции устанавливается инструмент командной строки Yeoman и генератор шаблонов VS Code / scaffolding generator

После его установки перейдите в свою основную папку / каталог, который вы хотели бы использовать для разработки (пример: `cd /my/dev/folder`) и запустите следующую команду, чтобы начать:

```
yo code
```

Вам будет предложено выбрать тип расширения, которое вы хотите создать. Используйте клавиши со стрелками на клавиатуре, чтобы перейти к опции "Новая цветовая тема" и нажать Enter клавишу.

```

yo
clarkio /mnt/c/Users/bc/dev/temp yo code

Welcome to the Visual
Studio Code Extension
generator!

? What type of extension do you want to create?
  New Extension (TypeScript)
  New Extension (JavaScript)
> New Color Theme
  New Language Support
  New Code Snippets
  New Keymap
  New Extension Pack
(Move up and down to reveal more choices)

```

Далее вам будет предложено импортировать или преобразовать существующую цветовую тему. Выберите опцию "Нет, начать заново".

```

yo

? What type of extension do you want to create? New Color Theme
? Do you want to import or convert an existing TextMate color theme? (Use arrow keys)
> No, start fresh
  Yes, import an existing theme but keep it as tmTheme file.
  Yes, import an existing theme and inline it in the Visual Studio Code color theme file.

```

Отсюда в приглашении будет предложено предоставить подробную информацию о расширении.

- `name`: это определяет папку, которую оно создаст в вашем текущем каталоге.
- `identifier`: это то, что будет использоваться на рынке расширений, чтобы другие могли найти его, поэтому убедитесь, что оно уникально, если вы планируете опубликовать его позже (обычно я использую дескриптор моего имени пользователя, за которым следует название расширения / темы).
- `description`: более длинный текст в форме описания вашего расширения

- название вашей темы: это текст, который пользователи будут видеть в качестве опции при переключении тем в VS Code (после его установки).
- `base theme`: это даст вам отправную точку для создания вашей темы вместо того, чтобы пытаться выяснить, как создать ее полностью с нуля. Выберите тот вариант, который лучше всего соответствует типу цветовой темы, которую вы хотите создать: темный, светлый, высококонтрастный

```

.s/bc/dev/temp
? What type of extension do you want to create? New Color Theme
? Do you want to import or convert an existing TextMate color theme? No, start fresh
? What's the name of your extension? clarkio-blue
? What's the identifier of your extension? clarkio-blue
? What's the description of your extension? A blue theme made by Clarkio
? What's the name of your theme shown to the user? Clarkio Blue
? Select a base theme: Dark

```

Когда вы закончите вводить все параметры для своей цветовой темы, нажмите Enter клавишу для последнего приглашения, чтобы начать генерацию всего. Вы увидите некоторый прогресс в терминале / командной строке, в которой отображаются файлы и папки, которые он создает для вас.

```

create clarkio-blue/themes/Clarkio Blue-color-theme.json
create clarkio-blue/.vscode/launch.json
create clarkio-blue/package.json
create clarkio-blue/vsc-extension-quickstart.md
create clarkio-blue/README.md
create clarkio-blue/CHANGELOG.md
create clarkio-blue/.vscodeignore

Your extension clarkio-blue has been created!

To start editing with Visual Studio Code, use the following commands:

cd clarkio-blue
code .

Open vsc-extension-quickstart.md inside the new extension for further instructions
on how to modify, test and publish your extension.

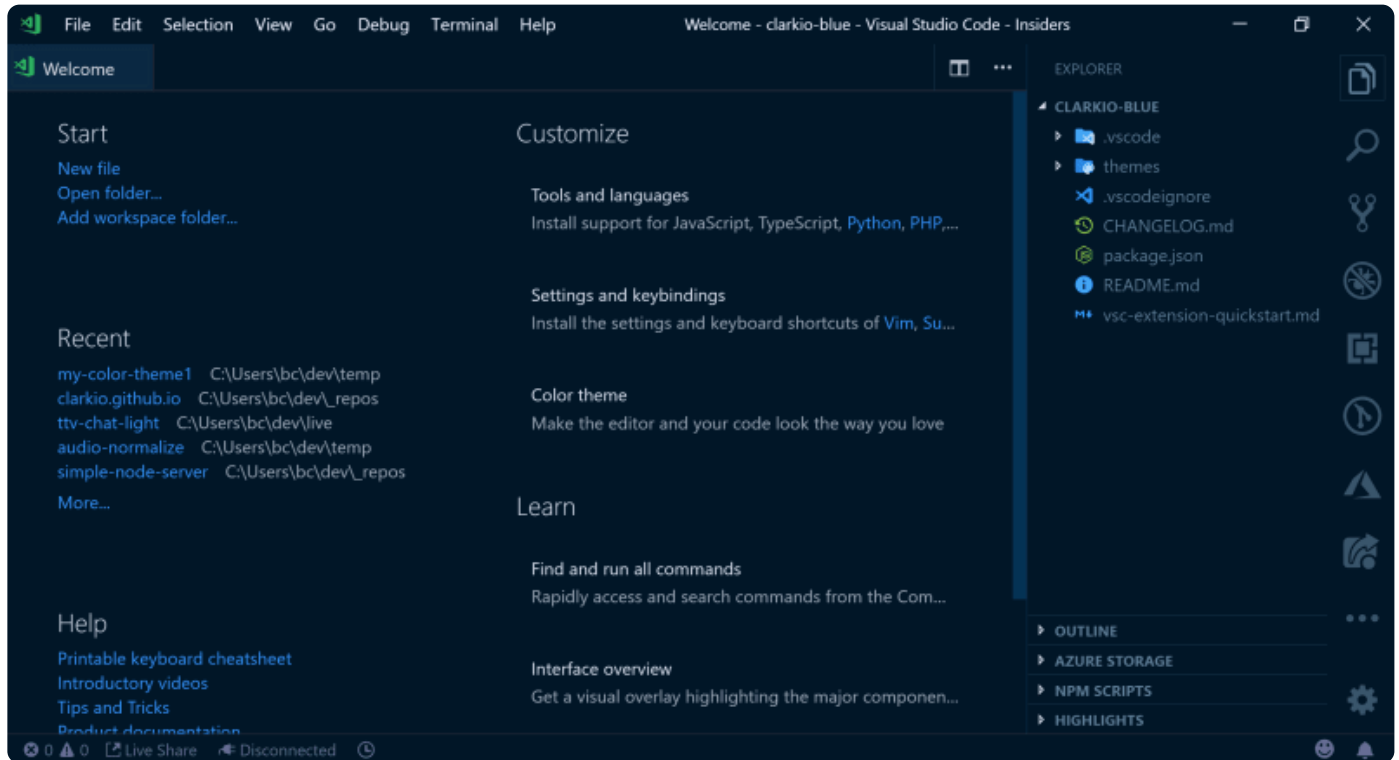
For more information, also visit http://code.visualstudio.com and follow us @
code.

clarkio /mnt/c/Users/bc/dev/temp

```

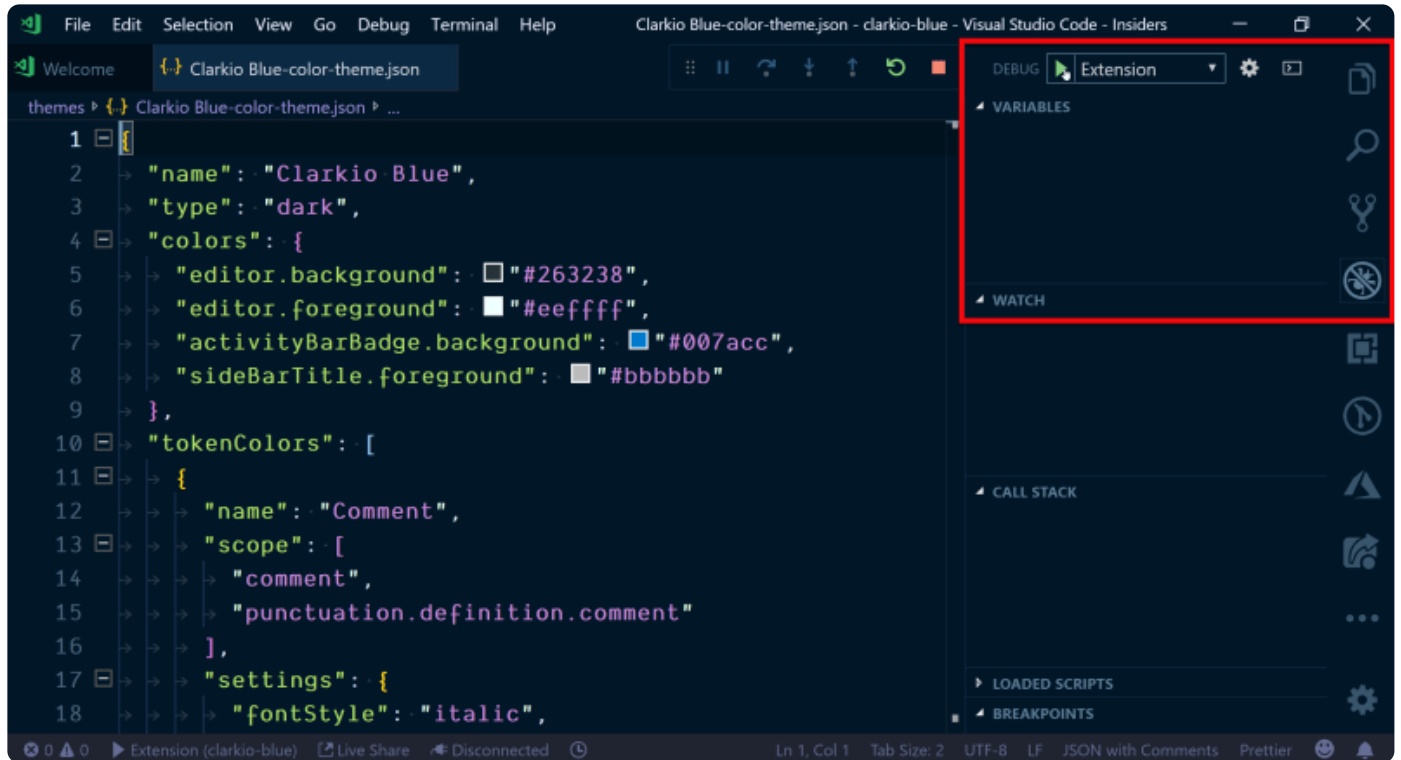


Когда это будет сделано, вы должны увидеть сообщение следующего содержания: "Чтобы начать редактирование с помощью Visual Studio Code, используйте следующие команды": Выполните команды, показанные под ним, и вы увидите, как VS Code открывает папку для вашего расширения цветовой темы.

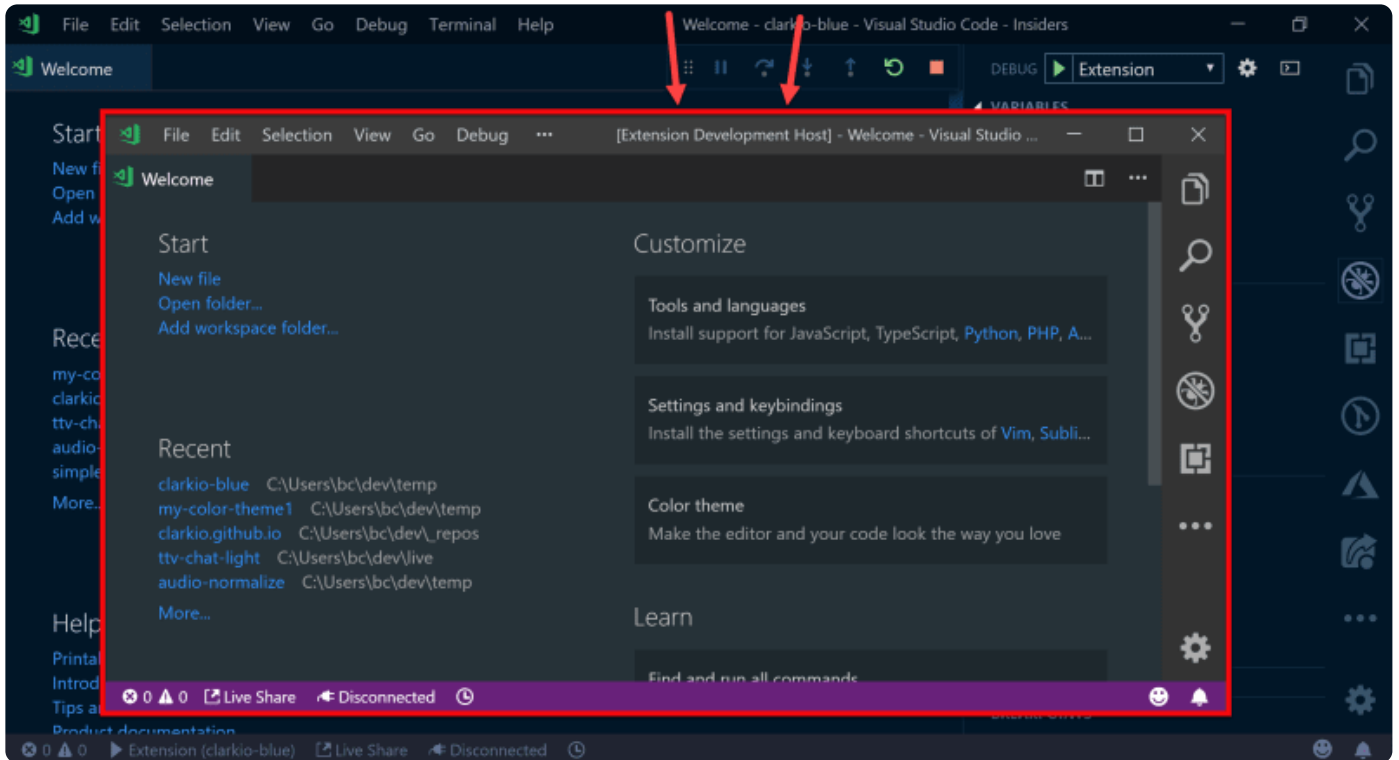


Откройте представление debugger в VS Code с помощью сочетания клавиш CTRL/CMD + SHIFT + D (или щелкните значок с ошибкой в нем). Нажмите на зеленую кнопку "воспроизвести" (как видно на скриншоте ниже), чтобы начать отладку / запуск расширения цветовой темы.

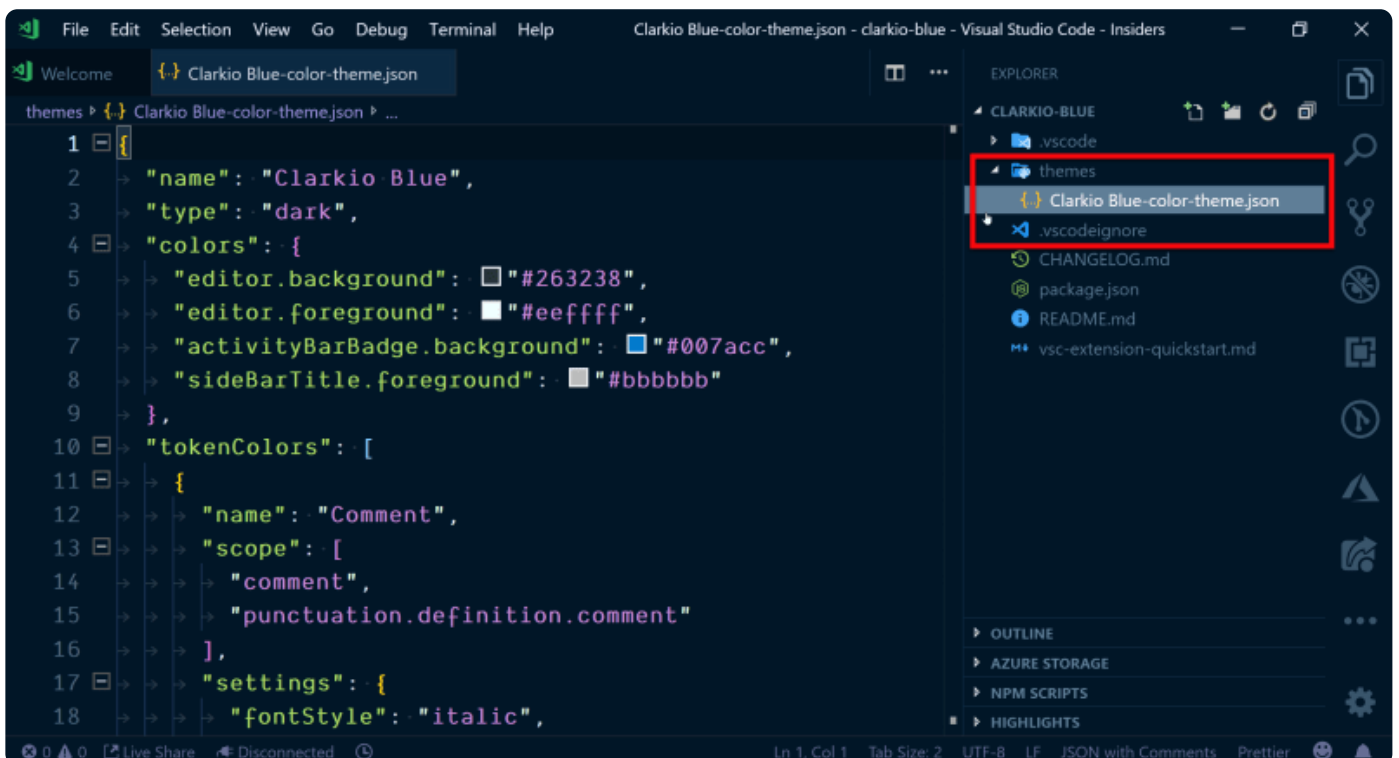




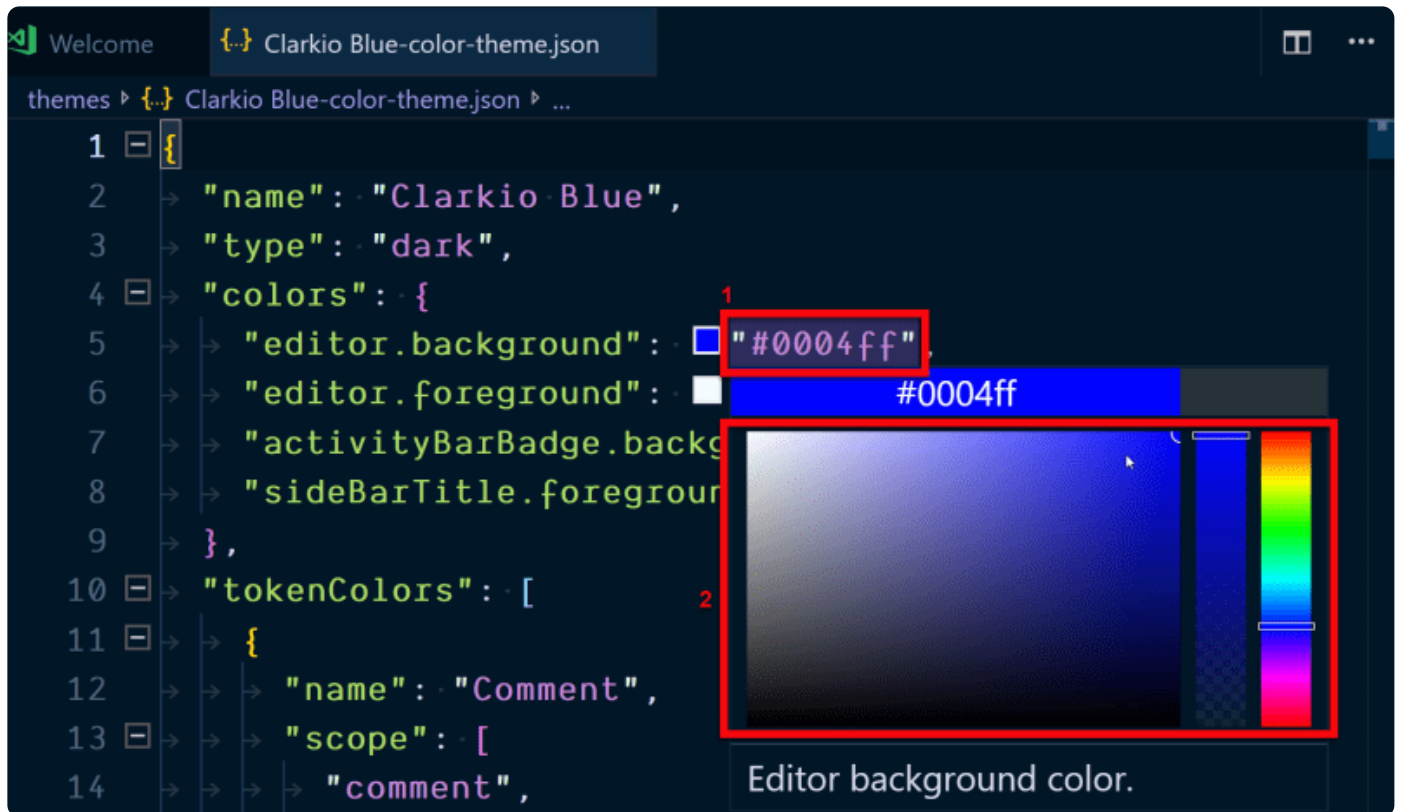
После запуска вы увидите, что откроется новый экземпляр VS Code с названием "Узел разработки расширения". Это отдельный экземпляр VS Code, запущенный с загруженным в него вашим расширением. На предыдущем шаге при создании расширений этой цветовой темы в терминале / командной строке я выбрал "Темный" вариант для своей базовой темы, поэтому я вижу этот цвет как цвет по умолчанию в экземпляре узла разработки расширений VS Code. Вы можете увидеть разные цвета в зависимости от базовой темы, выбранной вами на этом шаге. Давайте протестируем изменение цвета в соответствии с темой.



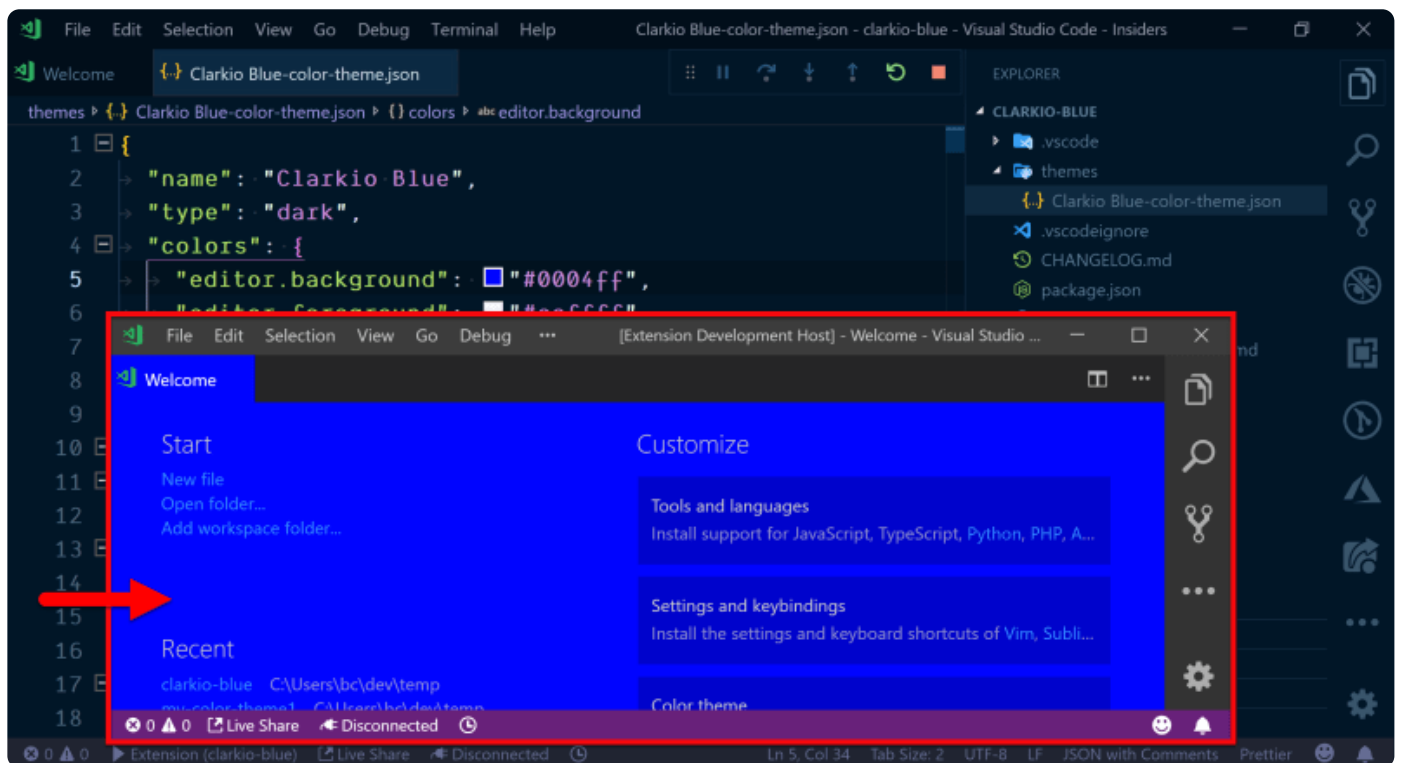
Вернитесь к другому экземпляру VS Code, у которого открыта папка расширения. Разверните папку "темы" и дважды щелкните по .json файлу, чтобы открыть / просмотреть содержимое.



Теперь найдите `editor.background` ключ в файле JSON под `colors` ключом. Наведите указатель мыши на `string` значение, чтобы открыть окно выбора цвета в VS Code. Осторожно наведите курсор мыши на всплывающее окно выбора цвета и измените цвет на что-нибудь другое. Вы увидите, что `string` значение изменится.



Как только вы сохраните внесенные изменения, вы увидите, что "Узел разработки расширения" VS Code автоматически обновится и отобразит изменение цвета.



Вы сделали это! Поздравляем с созданием вашего первого расширения для Visual Studio Code! Не стесняйтесь продолжать настраивать параметры цвета в `.json` файле, чтобы дополнительно настроить расширение цветовой темы.

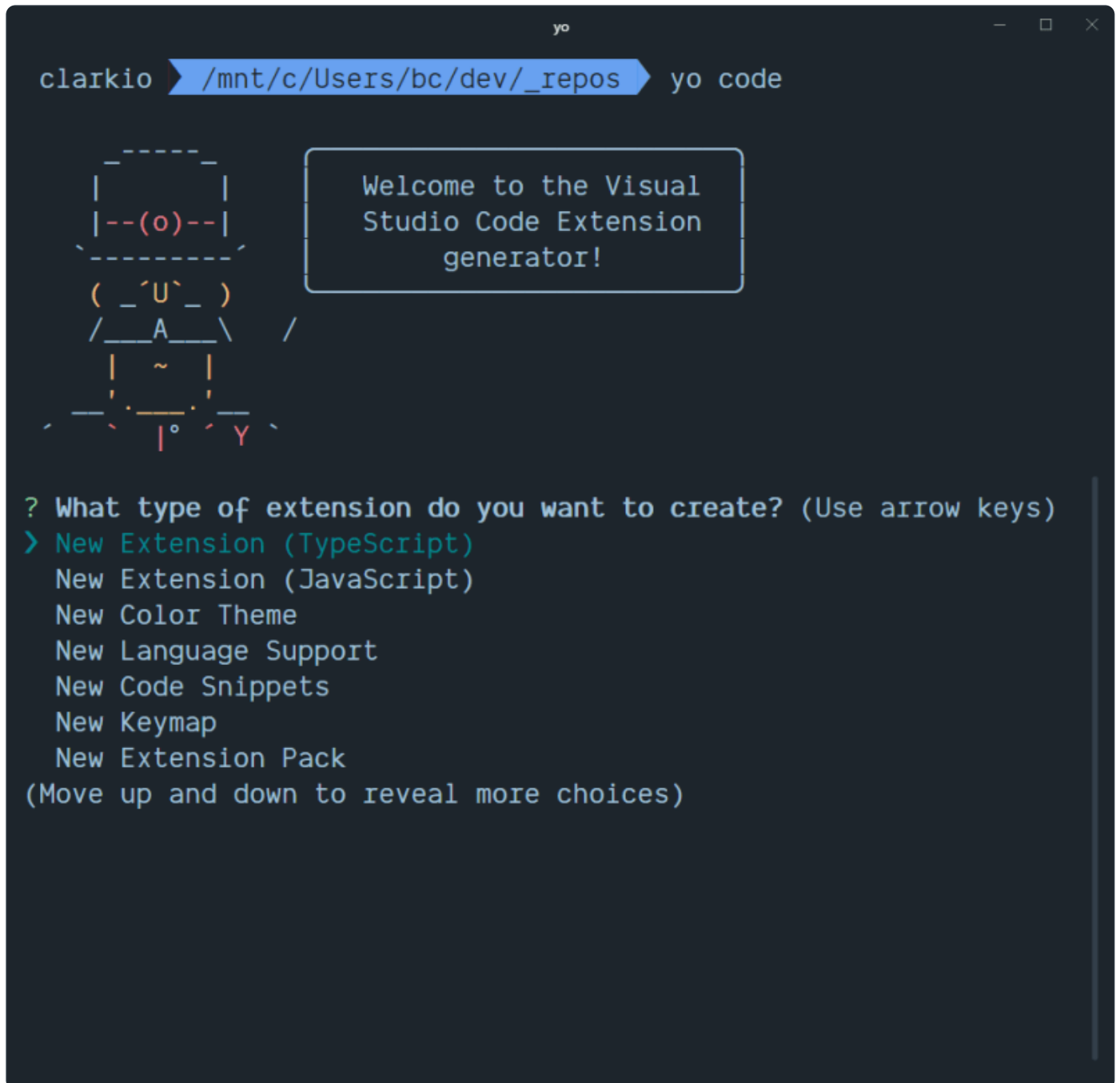
## Создание чего-то большего

Создание расширения цветовой темы - это здорово и все такое, но оно не погружает глубоко в улучшение или создание функций в VS Code. Давайте рассмотрим процесс создания расширения, результатом которого станет создание файла со списком всех расширений, которые мы используем в VS Code.

Перейдите в свою основную папку, которую вы хотели бы использовать для разработки (пример: `cd /my/dev/folder`) и запустите следующую команду, чтобы начать:

```
yo code
```

Вам будет предложено выбрать тип расширения, которое вы хотите создать. С помощью клавиш со стрелками на клавиатуре перейдите к опции "Новое расширение (TypeScript)" и нажмите `Enter` клавишу.



```
clarkio /mnt/c/Users/bc/dev/_repos yo code

Welcome to the Visual
Studio Code Extension
generator!

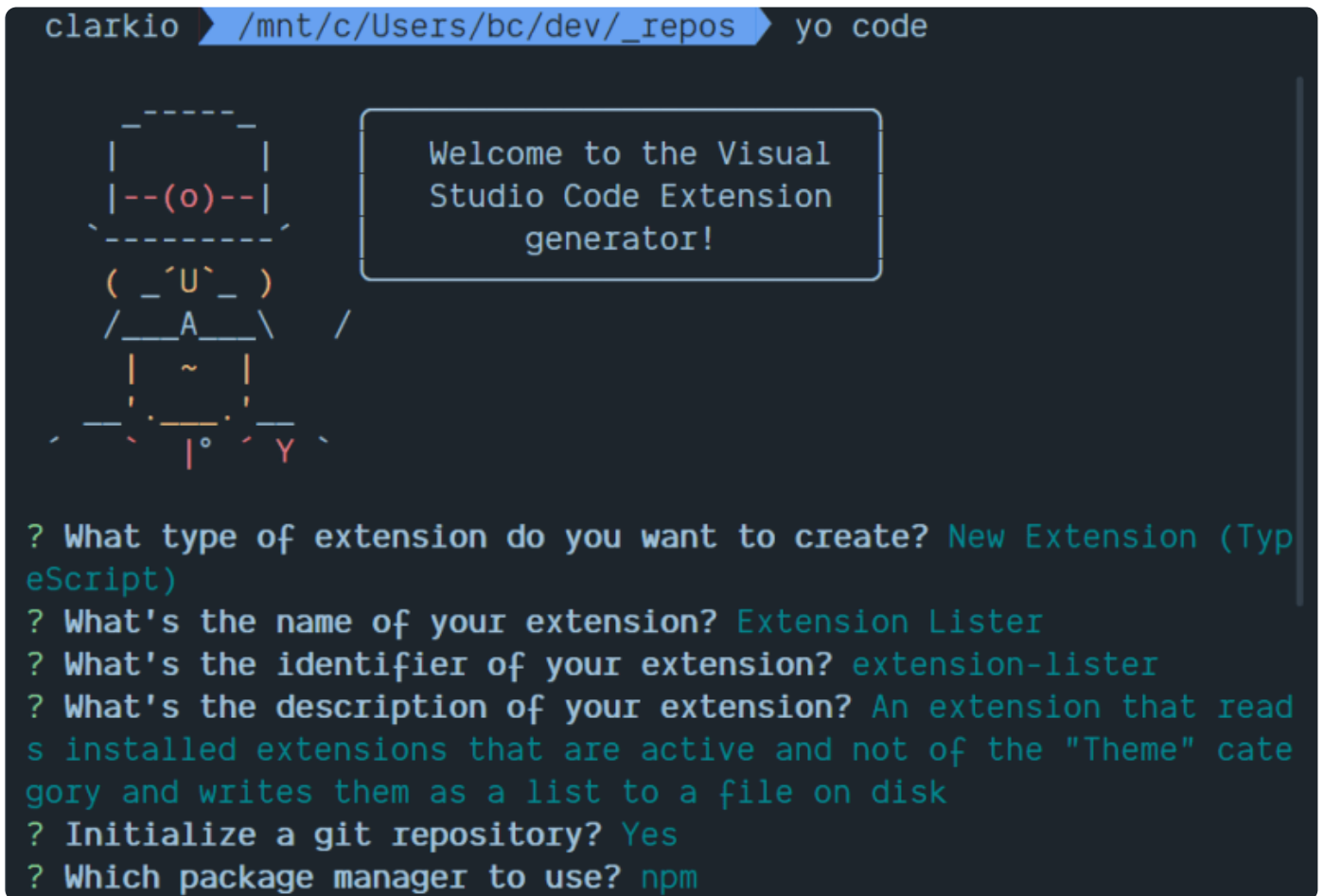
? What type of extension do you want to create? (Use arrow keys)
> New Extension (TypeScript)
  New Extension (JavaScript)
  New Color Theme
  New Language Support
  New Code Snippets
  New Keymap
  New Extension Pack
(Move up and down to reveal more choices)
```

Отсюда в приглашении будет предложено предоставить подробную информацию о расширении.

- `name`: это определяет папку, которую оно создаст в вашем текущем каталоге.
- `identifier`: это то, что будет использоваться на рынке расширений, чтобы другие могли найти его, поэтому убедитесь, что оно уникально, если вы планируете опубликовать его позже (обычно я использую дескриптор моего имени пользователя, за которым следует название расширения / темы).
- `description`: более длинный текст в форме описания вашего расширения
- название вашей темы: это текст, который пользователи будут видеть в качестве опции при переключении тем в VS Code (после его установки).

- `git repository`: это дает вам возможность инициализировать проект как новый репозиторий `git` или нет
- `package manager`: выберите то, что вы предпочитаете между `npm` и `yarn`

```
clarkio /mnt/c/Users/bc/dev/_repos yo code
```



```
? What type of extension do you want to create? New Extension (TypeScript)
? What's the name of your extension? Extension Lister
? What's the identifier of your extension? extension-lister
? What's the description of your extension? An extension that reads installed extensions that are active and not of the "Theme" category and writes them as a list to a file on disk
? Initialize a git repository? Yes
? Which package manager to use? npm
```

Перейдите в каталог для вновь созданного расширения и откройте его в VS Code

```
clarkio /mnt/c/Users/bc/dev/_repos cd extension-lister
clarkio /mnt/c/Users/bc/dev/_repos/extension-lister master code .
```

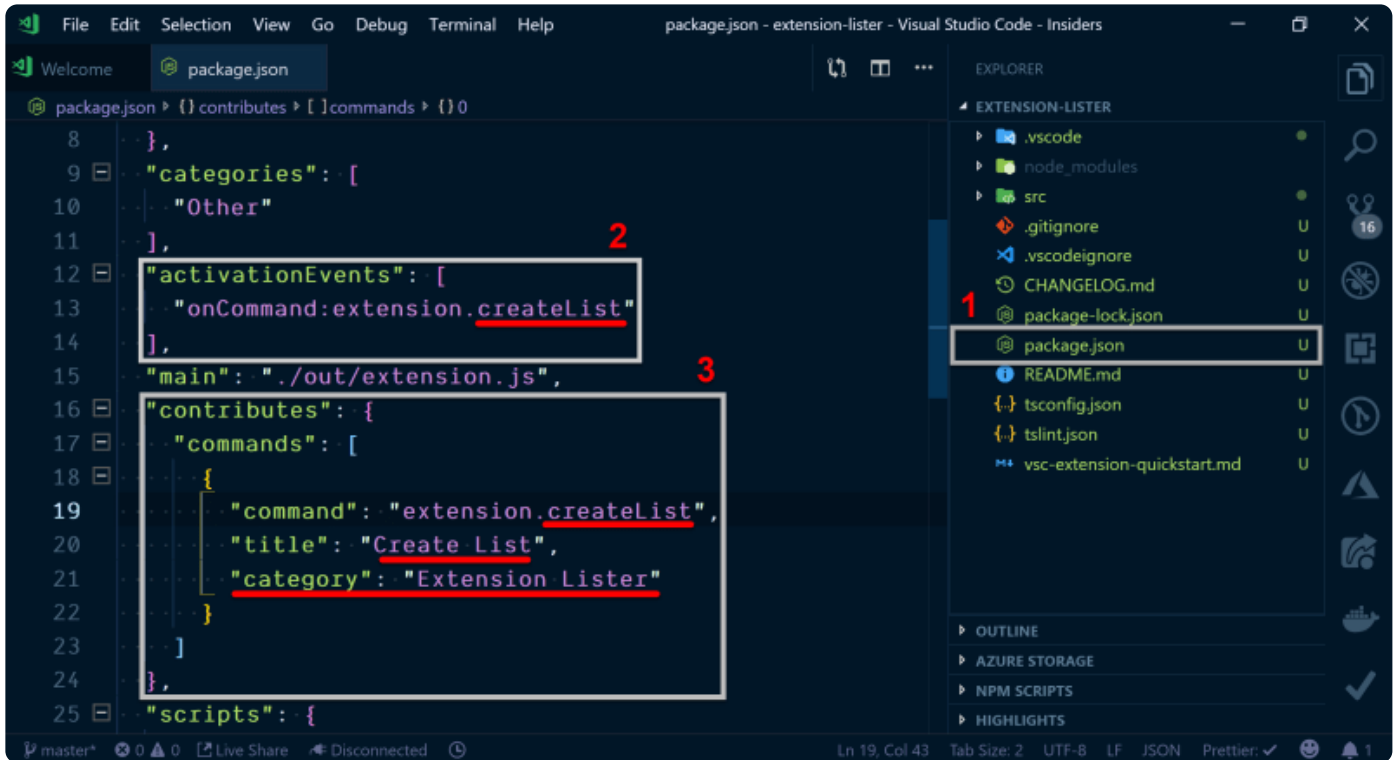
Расширения VS Code используют `package.json` в качестве своего манифеста и расширяют его некоторыми дополнительными опциями, выходящими за рамки обычных опций. Мы не будем вдаваться в подробности по всем из них, но нам нужно будет изменить несколько для целей этого расширения.

1. Откройте `package.json` файл и найдите `activationEvents` ключ. Это определяет, какое действие активирует ваше расширение. На данный момент определено активировать ваше расширение при запуске команды "HelloWorld". Вы увидите это как строковое значение `"onCommand:extension.helloWorld"`. Чтобы убедиться, что мы сохраняем

соответствие намерениям этого нового расширения, замените его на `helloWorld` на `createList`, поскольку это будет название команды, которую мы создадим.

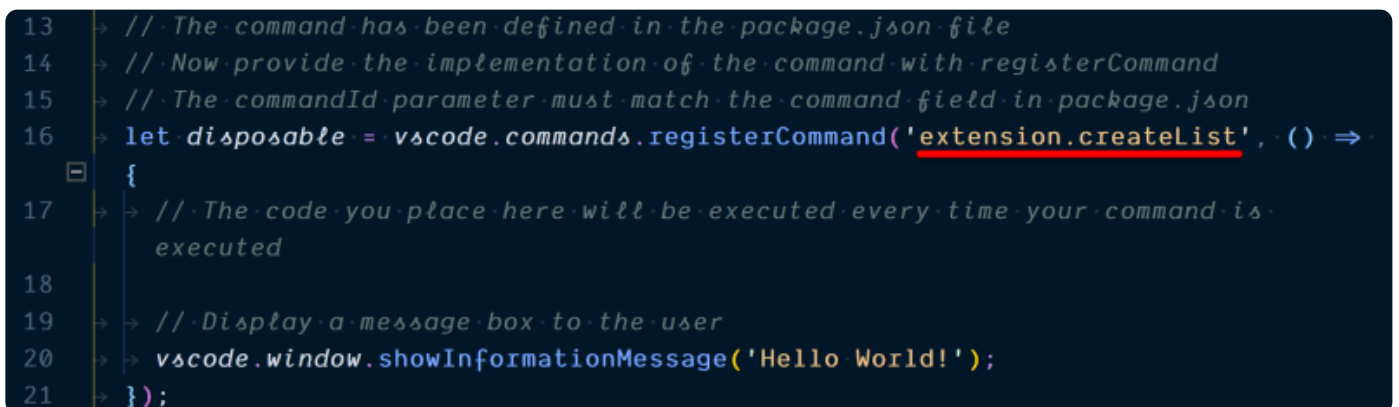
2. Вскоре под `activationEvents` ключом вы найдете другой ключ `contributes`. Это определяет множество способов, которыми ваше расширение может "вносить вклад" в VS Code. На данный момент у него определена одна опция, и это с помощью `commands` опции. Это массив, определяющий различные команды, которые расширение сделает доступными для VS Code через палитру команд. Вы должны увидеть одну команду, определенную как `extension.helloWorld`. Часть "расширение" этой команды действует как пространство имен при подключении команды в VS Code, а часть "HelloWorld" создает уникальное имя для команды в вашем расширении. Нам нужно убедиться, что это будет изменено в соответствии с тем, что мы обновили в `activationEvents` поэтому замените `helloWorld` здесь на `createList`.
3. `title` Ключ в определении команды предоставляет текст, который будет показан пользователям в палитре команд. Измените значение этого ключа на `Create List`. Последнее, что мы изменим в этом файле, - это добавим еще один ключ ниже, `title` поэтому поставьте запятую в конце `title` значения и нажмите клавишу "Enter" для новой строки. Добавьте ключ `"category"` (здесь тоже должен появиться `intellisense`, который поможет) и присвойте ему значение `"Extension Lister"`. Это `category` помогает визуальнo сгруппировать команды таким образом, чтобы они отображались в палитре команд следующим образом `" "` (Пример: "Прослушиватель расширений: Создать список")





Давайте приступим к созданию функциональности для этого расширения. В src папке вы увидите файл `extension.ts`. Это основной файл, который используется для загрузки вашего расширения и соответствующей функциональности. Найдите минутку, чтобы прочитать комментарии в коде, которые помогают описать некоторые части кода, сгенерированные для вас... Seriously, прочтите это и вернитесь.

Теперь, когда вы немного лучше знакомы с кодом, давайте настроим нашу первую команду. Сначала нам нужно обновить наш командный код, чтобы он соответствовал изменениям, которые мы внесли в `package.json`. Найдите код `registerCommand('extension.helloWorld')` и замените `helloWorld` на `createList`.



Хорошо, команда подключена правильно, и мы готовы приступить к сбору списка расширений. VS Code API предоставляет пространство имен, `extensions` которое содержит список всех расширений, доступных в экземпляре в `all`

массиве. Мы получаем доступ к API через `import * as vscode from 'vscode';` инструкцию. Поскольку это уже было сделано в `registerCommand` (в строке 17), давайте возьмем массив и присвоим ему константу. Замените строку кода `vscode.window.showInformationMessage('Hello World!');` (и комментарий над ней) следующим кодом:

```
const activeExtensions = vscode.extensions.all;
```

Если вы запустите и отладите расширение с точкой останова в этой строке, вы заметите, что массив содержит буквально все установленные и доступные расширения для экземпляра VS Code, в котором выполняется расширение. Не стесняйтесь попробовать это самостоятельно прямо сейчас, но это не обязательно. Это хорошее начало, но если вы проверите коллекцию расширений, вы заметите, что она включает расширения, встроенные в VS Code, те, которые в данный момент не активны, и расширения, являющиеся темами.

Поскольку мы получаем расширения, выходящие за рамки того, что мы хотим включить в список (активные, не связанные с темой типы, расширения), нам сначала нужно отфильтровать некоторые из этих результатов. Добавьте следующий метод `array`, `filter`, чтобы охватить расширения в пределах области видимости.

```
const activeExtensions = vscode.extensions.all.filter(  
  (extension: vscode.Extension<any>) => {  
    return (  
      extension.isActive && // make sure it is active  
      !extension.packageJSON.isBuiltin && // don't include built in  
      !extension.packageJSON.categories.some(  
        // don't include themes  
        (category: string) => category.toLocaleLowerCase() === 'themes'  
      )  
    );  
  });
```

Далее мы хотим отформатировать содержимое того, что будет записано в файл, следующим образом: `<extension name>: <extension marketplace URL>`. Это упростит общий доступ и позволит другим пользователям быстро устанавливать расширения. Для этого давайте переберем массив расширений, чтобы создать

строку содержимого для записи в файл. Для этого используйте следующий код и добавьте его после определения и назначения `activeExtensions` массива:

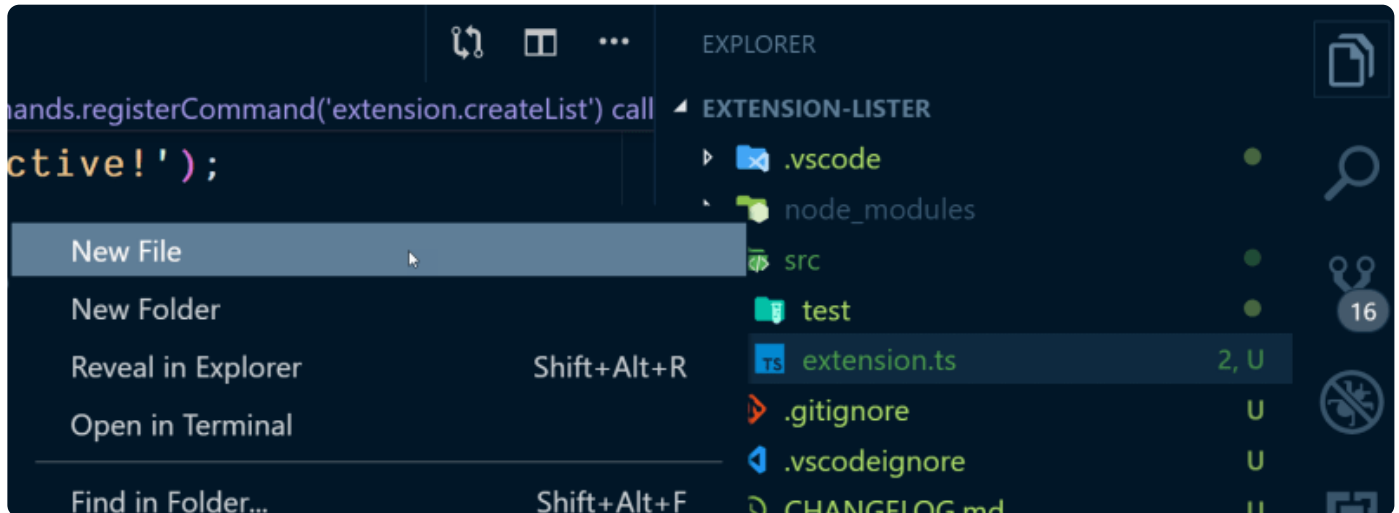
```
let extensionListData = ''; // variable to hold the file contents as a string
activeExtensions.forEach((extension: vscode.Extension<any>) => {
  // thanks to TypeScript and the exposed Extension type from the VS Code API we get
  // In particular we want to read the display name property found in the `packageJSON`
  extensionListData += `${
    extension.packageJSON.displayName
  }: https://marketplace.visualstudio.com/items?itemName=${extension.id}\n`;
});
```

На данный момент у нас есть необходимые данные в удобочитаемом формате, поэтому мы готовы спросить пользователя, где он хочет сохранить этот файл. Для визуальных подсказок и очередей VS Code предоставляет опции через свое `window` пространство имен. В частности, нам нужна `showSaveDialog` функция. Мы можем предоставить ему несколько опций / аргументов, которые помогут улучшить пользовательский интерфейс в этом диалоговом окне. Мы знаем, что это будет просто текстовый файл, поэтому мы предоставим опцию фильтра, которая позволяет ограничить тип файла `*.txt`.

Когда пользователь завершит свои действия в диалоговом окне сохранения, результирующий URI (универсальный индикатор ресурсов) для файла будет возвращен нам в функции `promise .then() resolve`. Сначала нам нужно убедиться, что URI действительно был предоставлен. Если это не так, мы можем показать диалоговое окно с ошибкой с помощью `vscode.window.showErrorMessage()`. Как только мы убедимся, что у нас по крайней мере есть значение, мы сможем записать его в файл, но давайте ненадолго задержимся на этом. Смотрите следующий код, который выполняет то, что мы обсуждали в этом разделе:

```
vscode.window.showSaveDialog({ filters: { '*': ['txt'] } }).then(uri => {
  if (!uri) {
    // This pops up an error notification dialog
    vscode.window.showErrorMessage(
      'You must select a file location to create the extension list'
    );
    return; // Don't proceed if we don't have a file URI to write to
  }
  // We'll add the code to write to a file here next...
});
```

Мы хотим реализовать запись в файл в файловой системе операционной системы, однако мы не должны загрязнять наш основной код расширения этим кодом. Давайте создадим отдельный файл для обработки записи файла в файловую систему. Откройте проводник VS Code, щелкнув по значку или используя сочетание клавиш `CTRL/CMD + SHIFT + E`. Щелкните правой кнопкой мыши на `src` папке и выберите "Новый файл".



Присвойте ему имя `file-manager` и нажмите `enter`, чтобы завершить создание файла. VS Code должен автоматически открыть этот файл, но если нет, дважды щелкните по нему. Мы не будем рассматривать следующий код, поскольку он на самом деле не имеет отношения к созданию расширений VS Code, но знаем, что он обрабатывает запись содержимого в файл. Не стесняйтесь просмотреть код, если хотите. Добавьте это в `file-manager.ts` файл и сохраните:

```
import { promises } from 'fs'; // leverage the Node.js file system module's functions

export function writeExtensionListFile(uri: string, data: any) {
  return promises.writeFile(uri, data);
}
```

Теперь мы можем импортировать `writeExtensionListFile()` функцию из `file-manager.ts` файла, который мы создали для выполнения последнего шага. Снова откройте `extension.ts` файл и добавьте инструкцию `import` в начало его после `vscode` импорта (находится в строке 3):

```
import { writeExtensionListFile } from './file-manager';
```

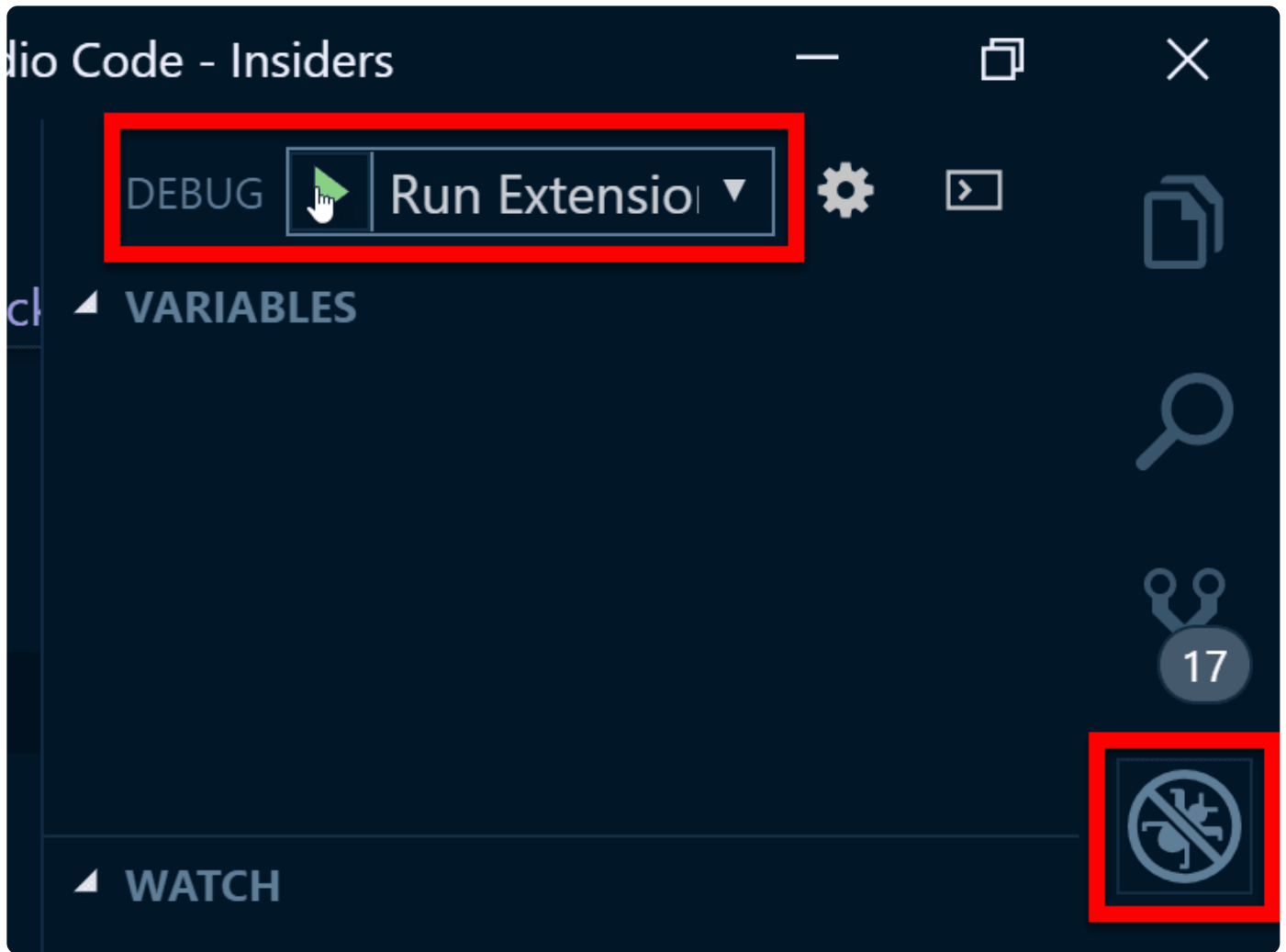
Отлично, давайте воспользуемся им. Перейдите туда, где у нас есть `showSaveDialog`, и после части, которая проверяет `uri` значение, добавьте следующий код:

```
// Provide the full path on the file system for the file to write to and the contents
writeExtensionListFile(uri.fsPath, extensionListData)
  .then(() => {
    // if the file was created successfully show an alert notification
    vscode.window.showInformationMessage(
      'Extension list was successfully created'
    );
  })
  .catch((error: any) => {
    // if the file failed to be created show an error notification
    vscode.window.showErrorMessage(
      'There was an issue creating the extension list'
    );
    console.error(error);
  });
```

Что делает этот код? Он вызывает функцию `writeExtensionListFile` и передает полное значение пути к файловой системе (свойство `uri` объекта) и содержимое, которое мы хотим записать в него. Если это проходит успешно, мы сообщаем пользователю об этом с помощью `vscode.window.showInformationMessage()` функции, в противном случае мы показываем сообщение об ошибке с помощью `vscode.window.showErrorMessage()` функции.

В случае, если у вас что-то работает неправильно, используйте следующий репозиторий GitHub в качестве ссылки на готовый код: [Список расширений VS Code](#)

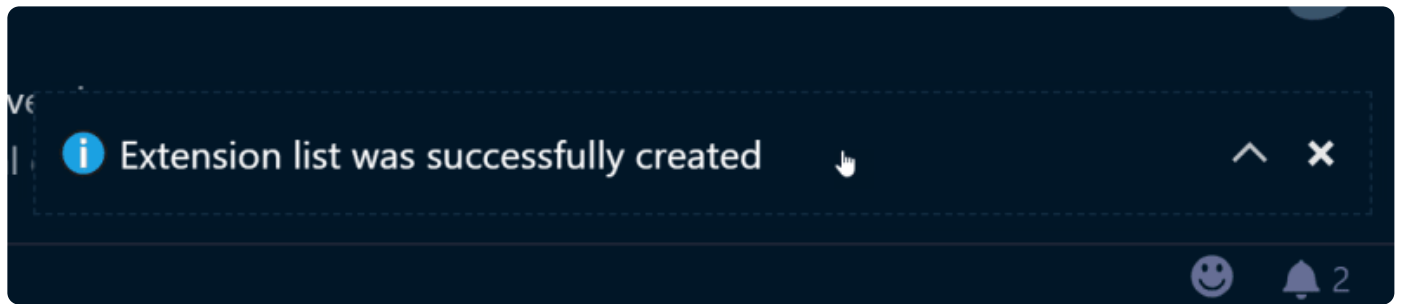
Как насчет того, чтобы запустить его и протестировать? Нажмите `F5`, чтобы начать отладку, или откройте представление отладки в VS Code и нажмите зеленую кнопку "Воспроизвести".



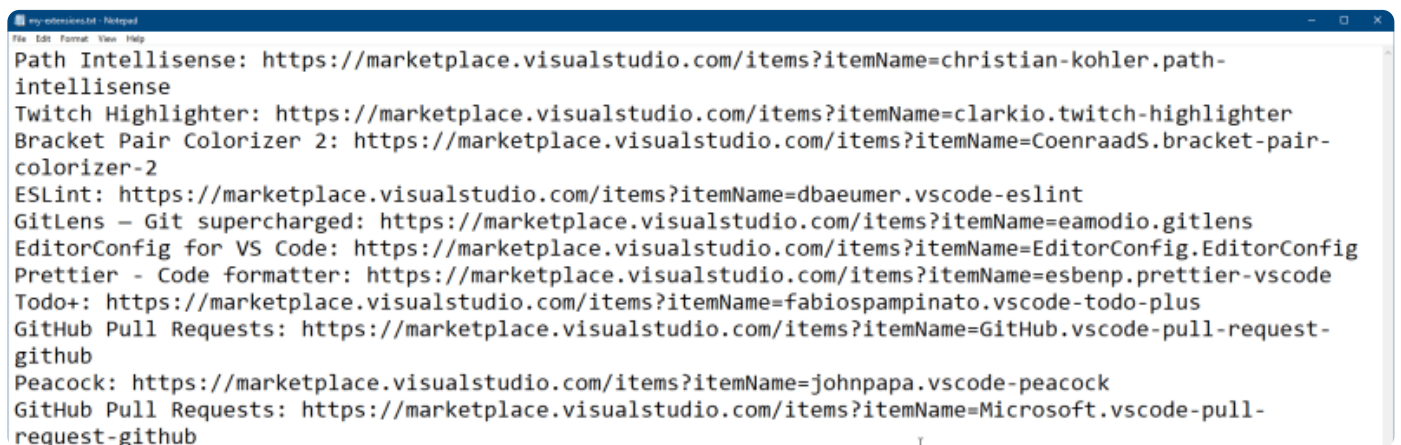
You'll then see a new VS Code instance load up with the title "Extension Development Host" that has your extension running within it. Open the command palette using the keyboard shortcut `CTRL/CMD + SHIFT + P` and type `lister` to see the available command for our extension.



Нажмите клавишу "Enter" на клавиатуре, и вам должно быть предложено выбрать местоположение и имя для вашего файла. Выберите местоположение и имя для вашего файла и нажмите кнопку "Сохранить". Вы должны увидеть всплывающее уведомление об успешном завершении в правом нижнем углу VS Code.



Просто чтобы убедиться, перейдите туда, где вы сохранили файл, и откройте его, чтобы просмотреть содержимое. Вы должны увидеть полный список активных расширений, которые вы используете, с их URL-адресами marketplace.



Поздравляем вас с созданием еще одного расширения VS Code, добавляющего новые функциональные возможности!

## Как мне опубликовать расширение?

Вместо того, чтобы снова писать ту же информацию здесь, взгляните на [документацию по расширению публикации](#). Документация, представленная на веб-сайте VS Code, довольно проста и понятна, поэтому ознакомьтесь с ней для получения дополнительной информации.

## Куда мне обратиться, чтобы узнать больше?

Лучшее место, где можно прочитать больше о разработке расширений VS Code, - на официальном сайте [VS Code API Docs](#). В левой части вы найдете оглавление с возможностью расширения, в котором рассматриваются важные темы, связанные с разработкой расширения.

## Заккрытие

Если у вас есть еще вопросы, на которые здесь нет ответов, дайте мне знать в комментариях. Я буду регулярно проверять и следить за ответом или



публикацией, если это более уместно в этом формате.

Спасибо, что прочитали этот пост, и я надеюсь, что вы нашли его полезным.

Подключайтесь к:

- [Twitter](#)
- [Twitch](#)
- [Instagram](#)
- [GitHub](#)
- [YouTube](#)
- [Веб-сайт](#)

## Лучшие комментарии (7) ↕



Джеймс Аллен • 10 мая 19 г.



Привет, Брайан, из-за твоей записи здесь я хочу спросить ... может ли расширение считывать код пользователя и отображать что-то встроенное? Например, если у меня есть комментарий в моем коде, такой как...

```
/* Azure DevOps: Project ABC
 * #255127 - My user story's name here
 */
```

Я бы хотел, чтобы расширение заменило это ссылкой на тикет 255127 моего проекта Azure DevOps ABC.

Я нашел страницу с [ограничениями](#) в Code API docs и предполагаю, что у меня не может быть расширения для этого. Есть ли что-нибудь возможное, о чем вы знаете?



Таннер Б. Хесс Уэббер • 12 сент. 19 г.



В самом худшем случае вы сможете использовать что-то вроде Acorns для генерации AST, а затем пройти по дереву, чтобы найти комментарии, соответствующие этому шаблону, запросить достоверность вашего источника DevOps и добавить текст во вторую строку. Это требует больших затрат, поэтому, надеюсь, вы сможете найти способ получше.

На основе одного экземпляра вы должны быть в состоянии сделать это легко, просто выделив комментарий, запустив команду, проанализировав

комментарий, запросив свой источник достоверности, а затем используя API "editBuilder" для заполнения информации о тикете. Возможно, вам придется повозиться с некоторыми функциями аутентификации, но это не должно быть ужасно.



Брайан Кларк 💡 ⭐ • 12 сент. 19 г.



Привет, Джеймс, я знаю, что это очень затянулось, но я не верю, что эти ограничения помешают вам в том, чего вы пытаетесь достичь. Я не совсем уверен, но я бы изучил возможности языка программирования и, в частности, возможности рефакторинга, чтобы увидеть, сработает ли это для вашего сценария.

ссылка -> [code.visualstudio.com/api/language ...](https://code.visualstudio.com/api/language-extensions)

Надеюсь, это поможет



Таннер Б. Хесс Уэббер • 12 мая 19 г.



Привет, Брайан,

Отличный пост! Мне любопытно, знаете ли вы (или написали) какие-либо хорошие ресурсы для тестирования расширений по мере их создания. До сих пор это было моим самым большим препятствием при разработке расширений

Заранее спасибо за любую помощь!



Брайан Кларк 💡 ⭐ • 12 сент. 19 г.



Привет, Таннер,

Прошу прощения за очень запоздалый ответ. Я пока не углублялся в это, но когда я это сделаю, я буду использовать расширение Джона Папы для VS Code, Peacock, в качестве ссылки: [github.com/johnpapa/vscode-peacock](https://github.com/johnpapa/vscode-peacock)



Таннер Б. Хесс Уэббер • 12 сент. 19 г.



О, это отличный репозиторий для реального изучения того, что вы можете сделать! Сейчас я создал несколько внутренних расширений для своей компании, но еще не добавил к ним тесты (TDD - моя проблема, так что это печально для меня). Время вернуться и добавить тесты! Большое спасибо!

Ашвини Манодж • 26 окт. 19 г. ...

Вау, именно то, что я тоже искал. Я не был уверен, как приступить к тестированию расширения! Спасибо!

[Кодекс поведения](#) • [Сообщить о нарушении](#)Сообщество разработчиков ...

## Тенденции в JavaScript

Сообщество JavaScript в настоящее время обсуждает процесс создания и выпуска побочных проектов, изучает передовые концепции Node.js и экспериментирует с подсветкой синтаксиса только для CSS. Также есть интерес к визуальному пониманию реквизитов React и использованию ExpressoTS для серверных приложений.



### Создание более чем второстепенного побочного проекта: от планирования до выпуска

Лукас Лима ду Насименту • 24 августа  
[#svelte](#) [#javascript](#) [#webdev](#) [#учебник](#)



### React Props: визуальное руководство

Рид Баргер • 18 августа  
[#react](#) [#webdev](#) [#новички](#) [#javascript](#)



### Невозможно!?! Подсветка синтаксиса только CSS 🤯 ...с одним элементом и градиентами 🤯

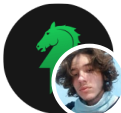
GrahamTheDev • 16 августа  
[#css](#) [#html](#) [#javascript](#) [#webdev](#)



## Примечания: Дополнительно Node.js Концепции Стивена Гридера

Мухаммад Бин Зафар • 19 августа

#javascript #node #webdev #новички



## Первый проект с ExpressoTS

Дэниел Болл для ExpressoTS • 27 августа

#javascript #typescript #expressots #restapi



Microsoft Azure

Invent with purpose

Any language. Any platform.

Try Azure for Free

### More from Microsoft Azure

Добивайтесь большего, используя GitHub Copilot с AI и VS Code

#javascript #githubcopilot #vscode #ai

Давайте #взломаем вместе: Javascript в Azure Keynote

#contosorealestate #javascript #hacktogether, ,взломайте вместе,,

Пошаговое руководство: перенос модели программирования v3 на v4 для функций Azure для Node.Приложение Js

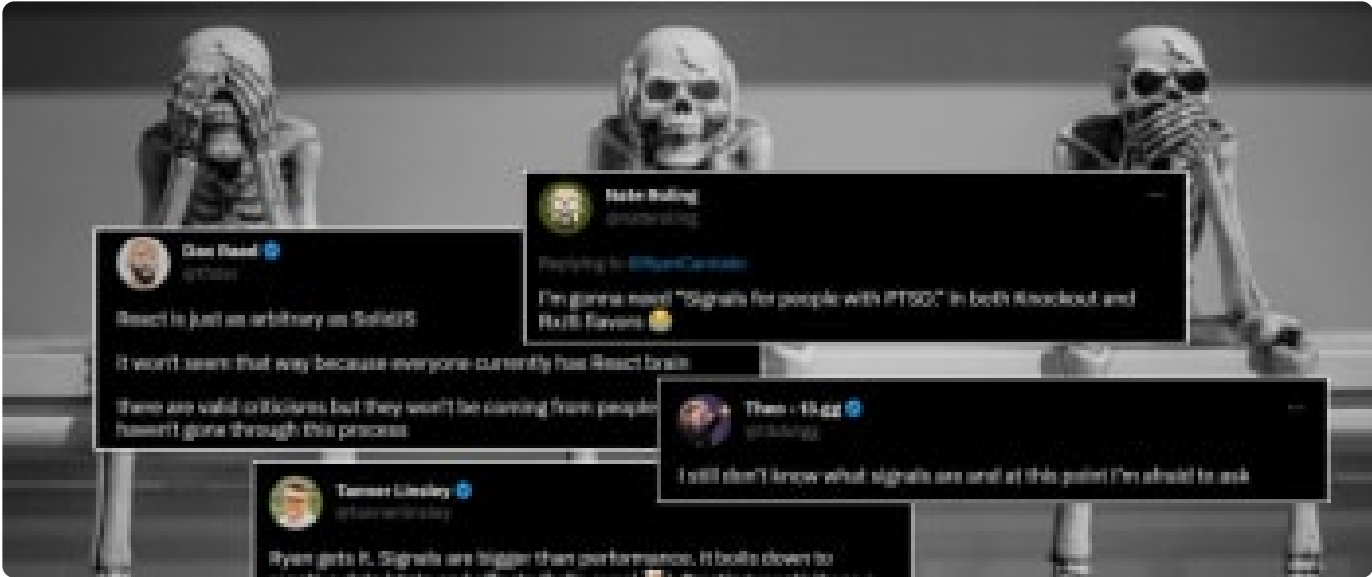
#javascript #hacktogether #contosorealestate #webdev

Сообщество разработчиков

...

Самые важные обсуждения JavaScript происходят в DEV

**"Те, кто не извлекает уроков из истории, обречены повторять ее"**



👉 Райан Карниато и Дэн Абрамов обсуждают основы React!