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

I don't like long introductions in posts because you know what brought you here and don't need the fluff. Let's get right into it...

Assumptions

- You know about Visual Studio Code (VS Code)
- You're familiar with extensions in VS Code and have used some but haven't built one before
- You know at least a little of TypeScript/JavaScript and Node.js

> If you don't really know one or any of these that's still ok as we don't focus too much on the code but if there is something you don't understand just leave a comment below and I'll get back to you as soon as I can.

- You have the following installed:
  - Node.js
  - npm (or yarn)
  - Yeoman
    ```
    npm i -g yo
    ```
  - VS Code Yeoman Generator
    ```
    npm i -g generator-code
    ```

What kind of extensions can I build?

You can build whatever your heart desires to build. There are a few main "types" of extensions you can build though which give you a better idea of where to start. Let's break them down by what you might want to do.

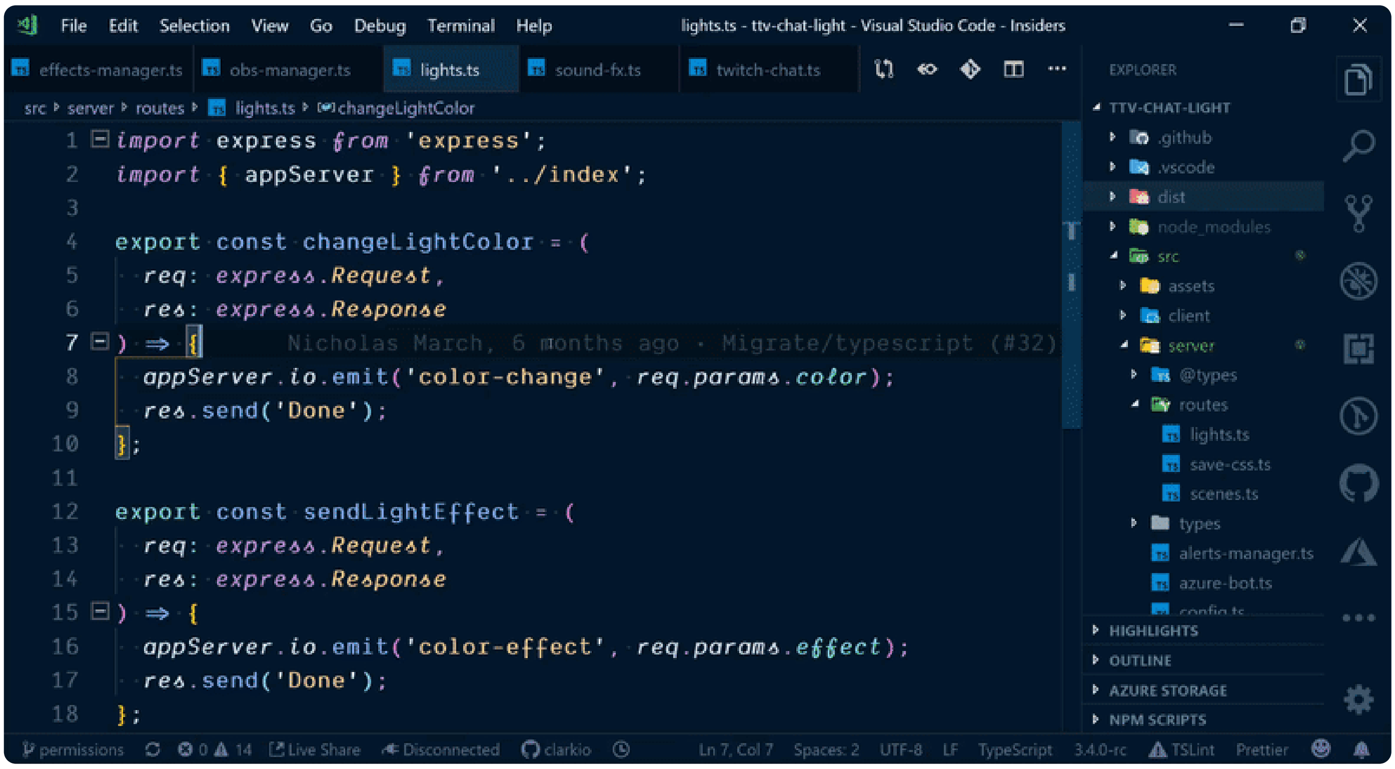
1. Customize VS Code's Look and Feel

What does this mean? Color themes and Icon Themes. A Color theme is actually a great place to start for your first extension. Let's first break them down a bit more for better understanding.

- Color Theme

This type of extension customizes the visual look of VS Code. You can change the colors of sooo many different aspects within the user interface (UI). this can be quite overwhelming. Using an existing theme you like but want to tweak is a great way to get started with creating your own theme that you'll actually use. Doing so will give you a lot of example color changes that you can tweak and see the result quickly. However we'll see how easy it is to get started from scratch soon.

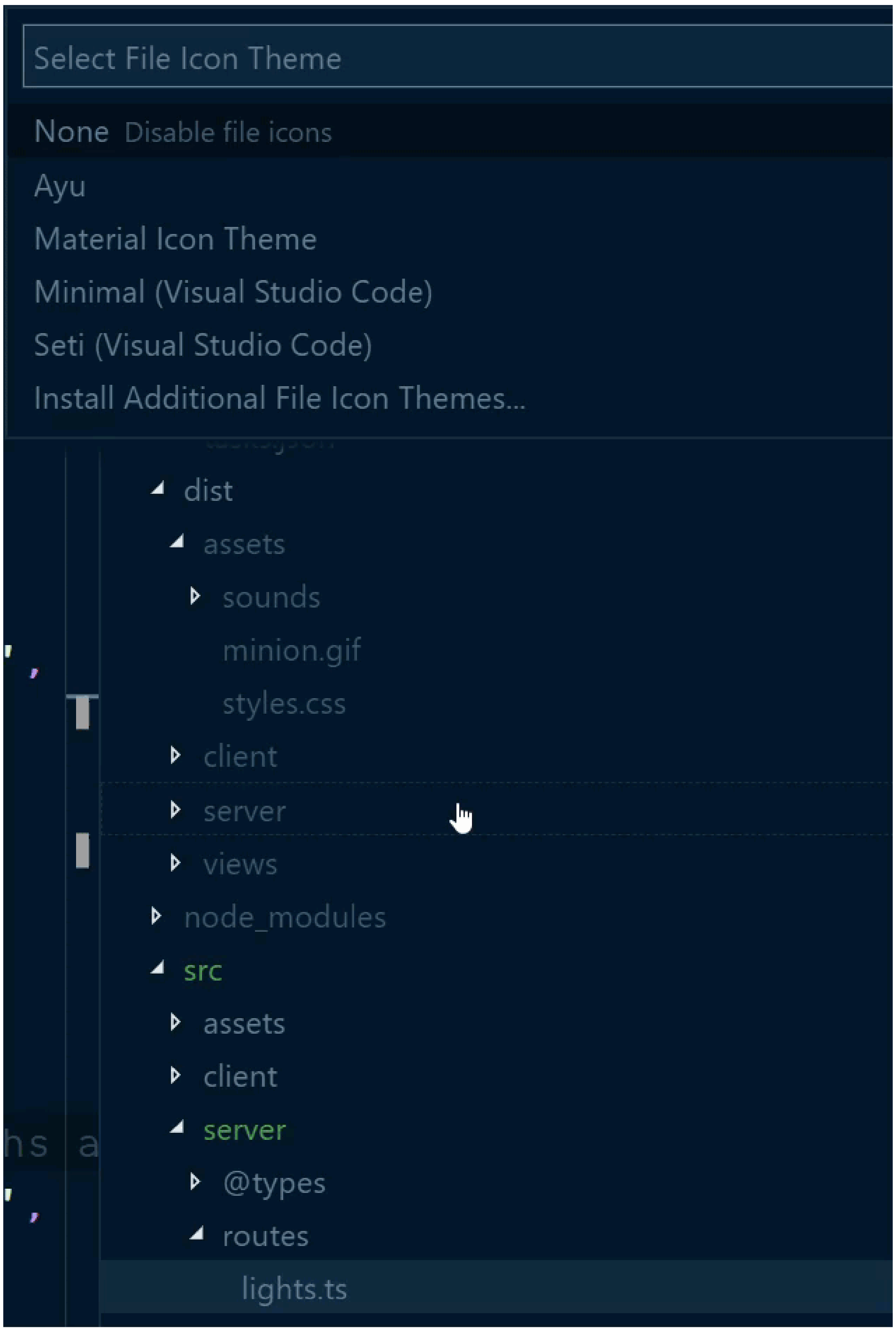
My favorite theme at the time of writing this post is Night Owl by Sarah Drasner



```typescript
import express from 'express';
import { appServer } from '../index';

export const changeLightColor = (
  req: express.Request,
  res: express.Response
) => {        Nicholas March, 6 months ago • Migrate/typescript (#32)
  appServer.io.emit('color-change', req.params.color);
  res.send('Done');
};

export const sendLightEffect = (
  req: express.Request,
  res: express.Response
) => {
  appServer.io.emit('color-effect', req.params.effect);
  res.send('Done');
};
```

- Icon Theme

This type of extension customizes the graphics associated to file/folder types. A great example of this is the Material Icon Theme (which is what I use).



GIF

2. Extend Existing Functionality

What does this mean? It means there is some existing capability/feature in VS Code but you wish it had something more to it. A simple example of this is creating new keymaps that make it easier for someone who's used to certain keyboard shortcuts found in another editor. A similar example is an extension that provides code snippets to generate common code that is reused in Node.js applications.

Here are some simple examples:

- Keymap Extension Sublime Text
- Code Snippet Extension Vue.js Snippets

More complex examples:

- Enhancing source control with Git: GitLens
- Improving code readability: Bracket Pair Colorizer

---

3. Add New Functionality

What does this mean? It means VS Code doesn't provide some sort of functionality that you're looking to use. In that case we build out an extension that adds that functionality.

Examples:

- **Problem**: No native support for connecting with Chrome's debugger for front end web development.
  Solution: Debugger for Chrome Extension
- **Problem**: No native support for smart completions (Intellisense) when writing Python code
  Solution: Python Extension
- **Problem**: No native support for integrating Twitch chat messages to control highlighting lines of code
  Solution: Line Highlighter for Twitch Extension

---

4. Aggregate Other Extensions into One Place

What does this mean? You bring many extensions together under one pack that help fulfill a specific focus/category of actions. Essentially there are many extensions out there that alone are great for the type of development you may be doing but you may not be aware of them all. Someone can gather them together to make them easier to find and install at once.

An example of this is the Angular Essentials extension pack that packs a bunch of Angular related extensions into one place. Check out the source code on GitHub to see what's involved in creating one.

> These types of extensions are driven by their `package.json` file to define the other extensions to be included in the pack. See the example from Angular Essentials' package.json

How do I build an extension?

One of the simplest extensions you can start with is a Color Theme extension. Let's step through building one of those now.

Building a Color Theme Extension

Let's build a color theme extension from scratch just to get a feel for creating our first extension. We're going to use the VS Code Yeoman Generator so make sure you executed the following command in your terminal/command prompt:

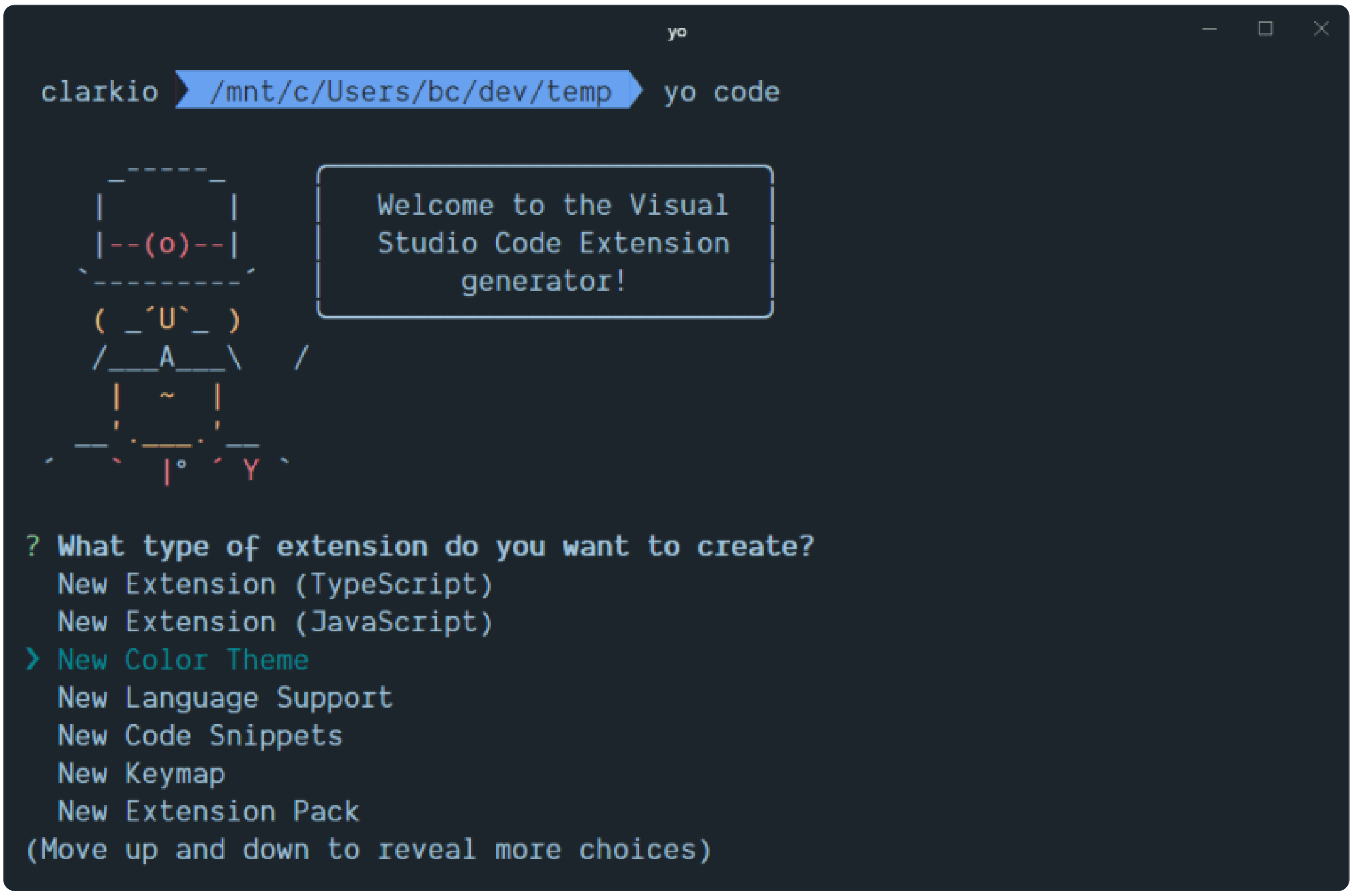
```
npm i -g generator-code
```

> This installs the Yeoman CLI tool and the VS Code templating/scaffolding generator as an option within it

Once you've installed that go to your main folder/directory that you'd like to use for development (example: `cd /my/dev/folder`) and run the following command to get started:

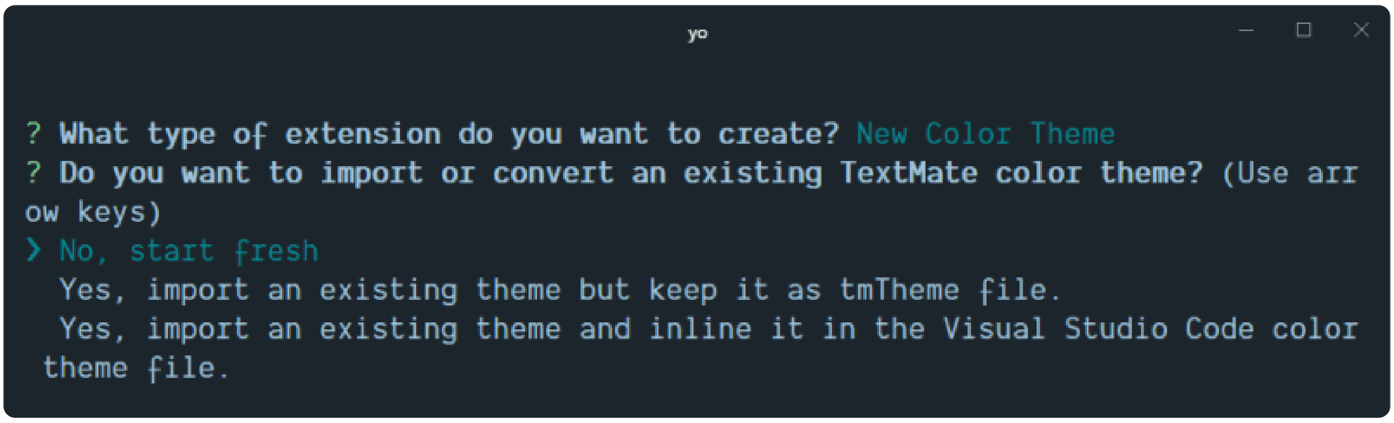
```
yo code
```

You'll be prompted to choose the type of extension you want to create. Use your keyboard arrow keys to move to the "New Color Theme" option and press your `Enter` key.



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

     _-----_     ╭──────────────────────────╮
    |       |    │   Welcome to the Visual  │
    |--(o)--|    │   Studio Code Extension  │
   `---------´   │        generator!        │
    ( _´U`_ )    ╰──────────────────────────╯
    /___A___\   /
     |  ~  |
   __'.___.'__
 ´   `  |° ´ Y `

? 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)
```

Next, you'll be prompted with an option to import or convert an existing color theme. Choose the option "No, start fresh"

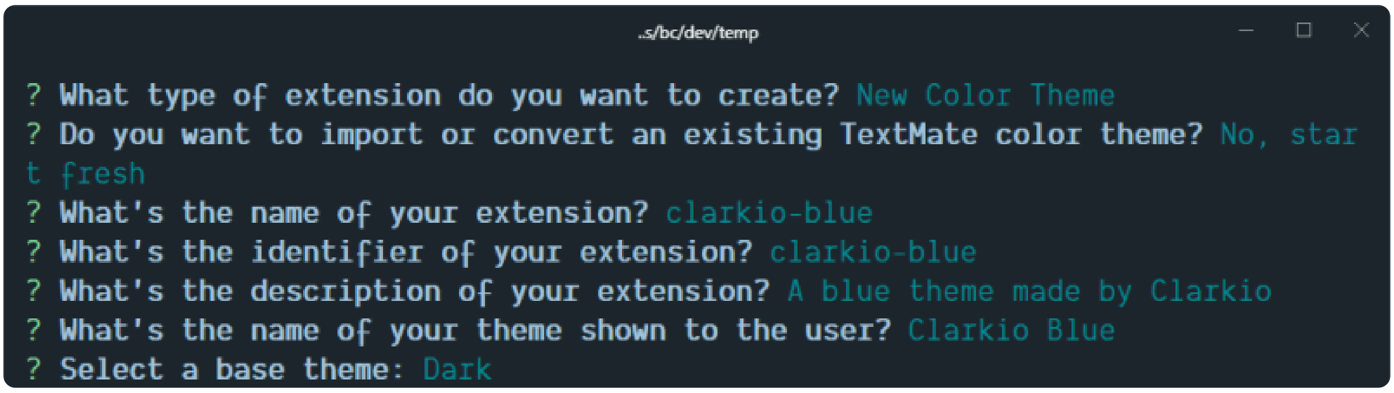


```
? What type of extension do you want to create? New Color Theme
? Do you want to import or convert an existing TextMate color theme? (Use arr
ow keys)
❯ No, start fresh
  Yes, import an existing theme but keep it as tmTheme file.
  Yes, import an existing theme and inline it in the Visual Studio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

From here, the prompt will ask you to provide details about the extension.

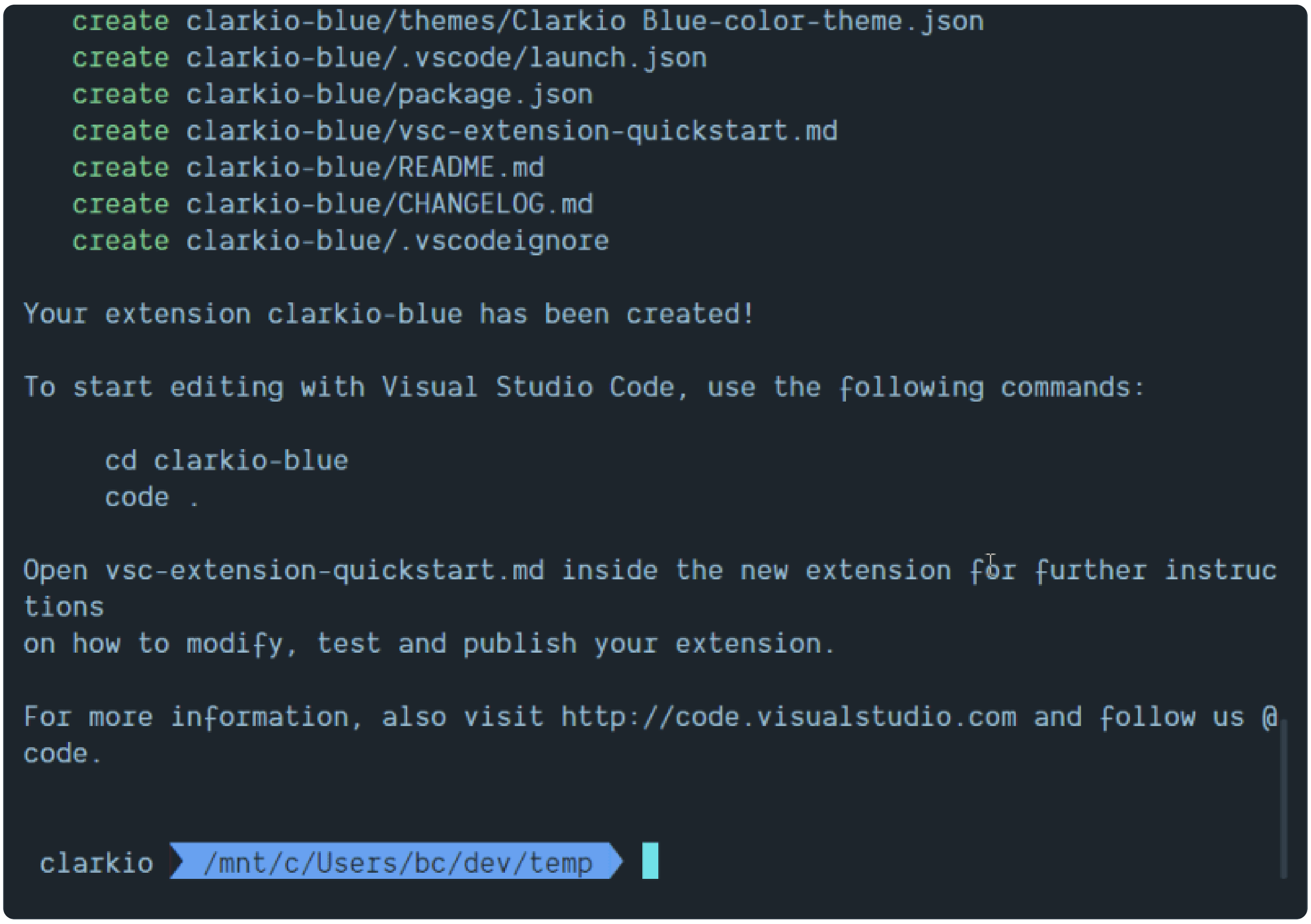
- `name`: this determines the folder it will create in your current directory.
- `identifier`: this is what will be used in the extension marketplace for others to find it so make sure this is unique if you plan to publish this later (I generally use my username handle followed by the name of the extension/theme).
- `description`: longer form text to describe your extension
- name of your theme: this is the text users will see as an option when switching themes within VS Code (after it's been installed)

- `base theme`: this give you a starting point for creating your theme instead of trying to figure out how to create one completely from scratch. Choose whichever option best suites the type of color theme you want to create: Dark, Light, High Contrast



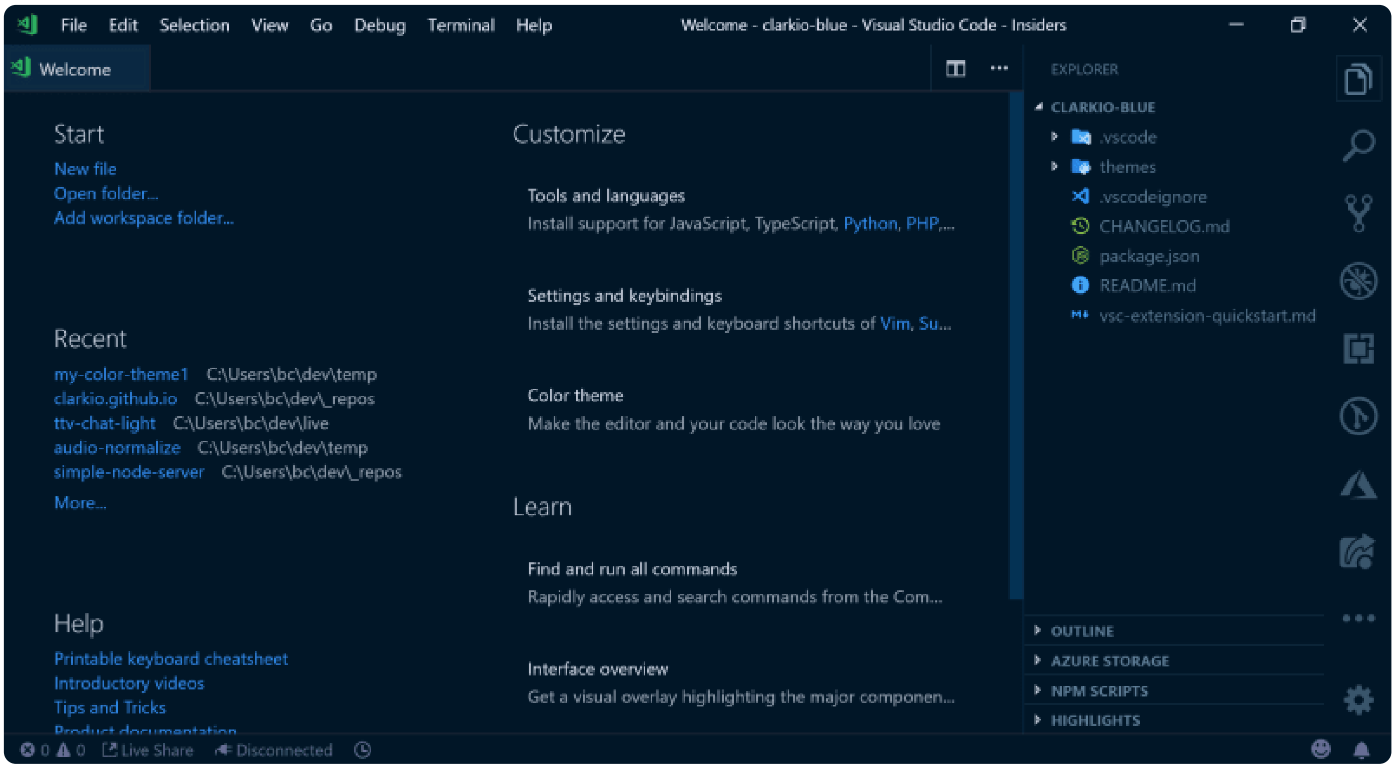
```
..s/bc/dev/temp
? What type of extension do you want to create? New Color Theme
? Do you want to import or convert an existing TextMate color theme? No, star
t fresh
? What's the name of your extension? clarkio-blue
? What's the identifier of your extension? clarkio-blue
? What's the description 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
```

When you've finished entering all the options for your color theme press the `Enter` key for the last prompt to start generatoring everything. You'll see some progress in the terminal/command prompt which shows the files and folders it is generating for you.

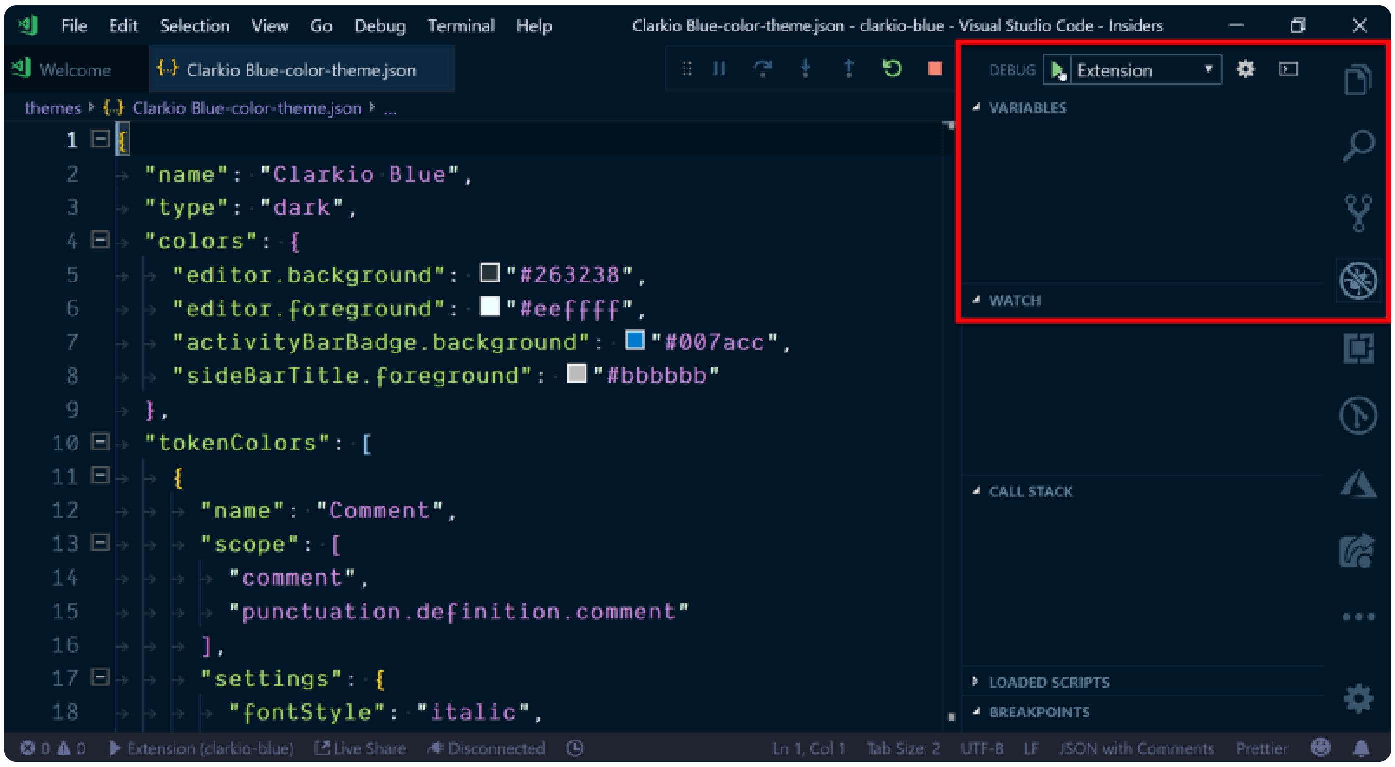


```
   create clarkio-blue/themes/Clarkio Blue-color-theme.json
   create clarkio-blue/.vscode/launch.json
   create clarkio-blue/package.json
   create clarkio-blue/vsc-extension-quickstart.md
   create clarkio-blue/README.md
   create clarkio-blue/CHANGELOG.md
   create clarkio-blue/.vscodeignore

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 instruc
tions
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
```

When it's done you should see a message that says the following: "To start editing with Visual Studio Code, use the following commands:" Run the commands shown below it and you'll see VS Code open the folder for your color theme extension.

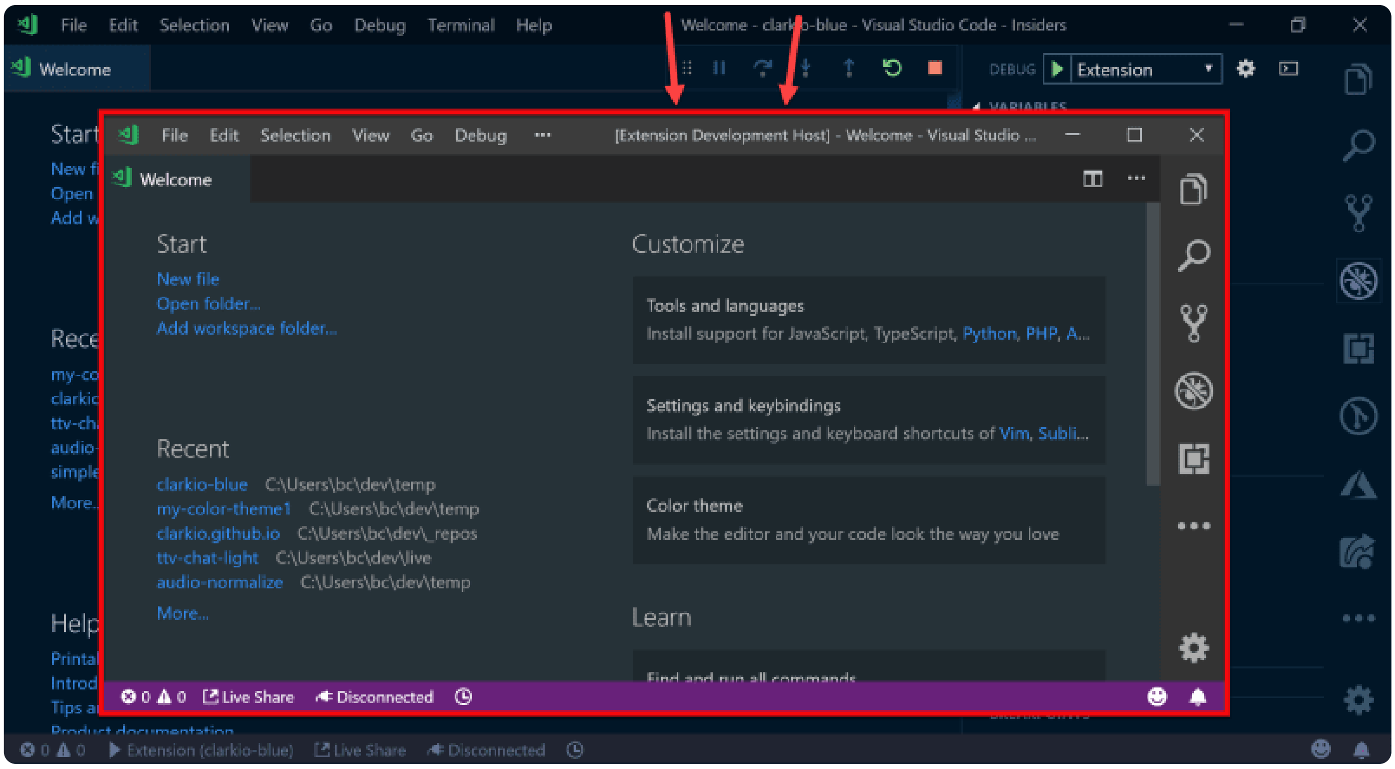

File Edit Selection View Go Debug Terminal Help
Welcome - clarkio-blue - Visual Studio Code - Insiders
Welcome
Start
New file
Open folder...
Add workspace folder...
Recent
my-color-theme1 C:\Users\bc\dev\temp
clarkio.github.io C:\Users\bc\dev_repos
ttv-chat-light C:\Users\bc\dev\live
audio-normalize C:\Users\bc\dev\temp
simple-node-server C:\Users\bc\dev_repos
More...
Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
Customize
Tools and languages
Install support for JavaScript, TypeScript, Python, PHP,...
Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Su...
Color theme
Make the editor and your code look the way you love
Learn
Find and run all commands
Rapidly access and search commands from the Com...
Interface overview
Get a visual overlay highlighting the major componen...
EXPLORER
CLARKIO-BLUE
.vscode
themes
.vscodeignore
CHANGELOG.md
package.json
README.md
vsc-extension-quickstart.md
OUTLINE
AZURE STORAGE
NPM SCRIPTS
HIGHLIGHTS
Live Share
Disconnected

Open up the debugger view in VS Code using the keyboard shortcut `CTRL/CMD + SHIFT + D` (or click on the icon with a bug in it). Click on the green "play" button (as seen in the screenshot below) to start debugging/running your color theme extension.

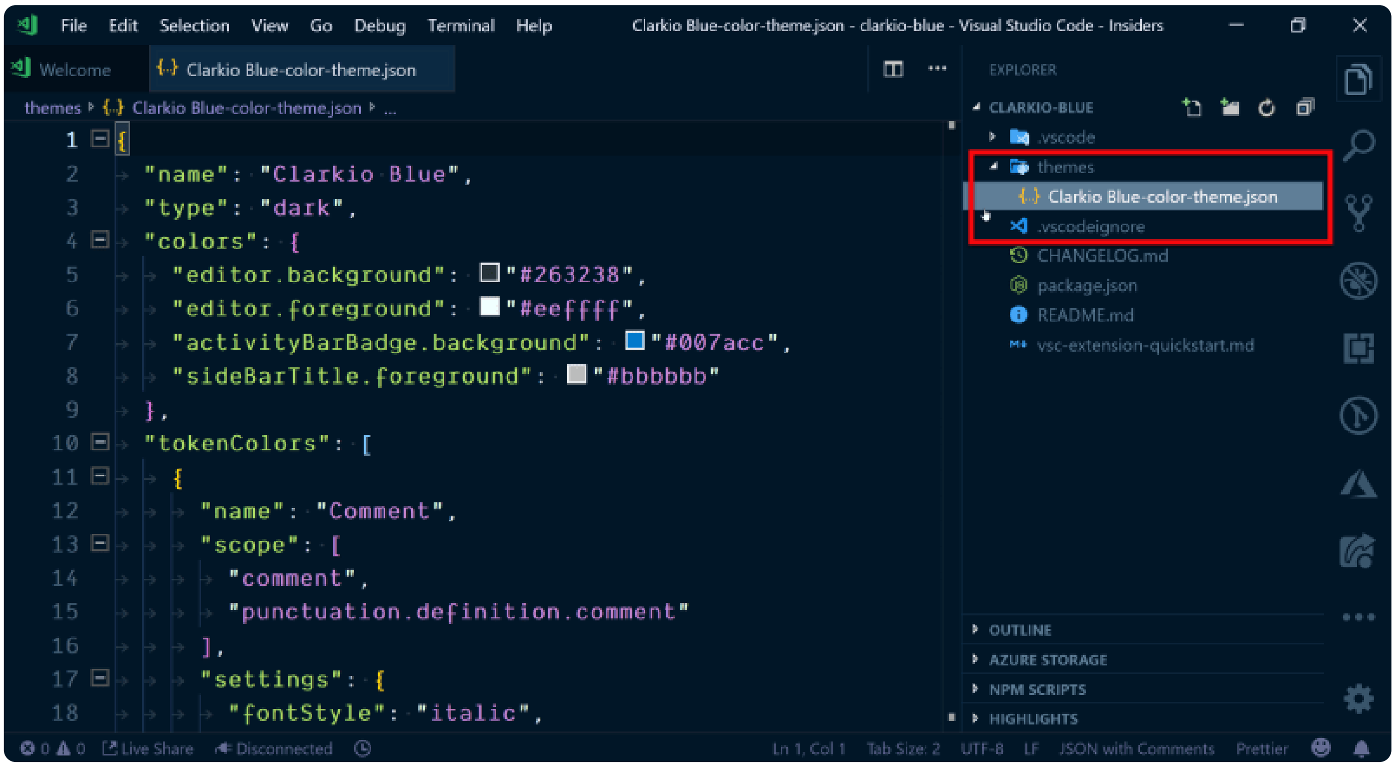

File Edit Selection View Go Debug Terminal Help
Clarkio Blue-color-theme.json - clarkio-blue - Visual Studio Code - Insiders
Welcome
Clarkio Blue-color-theme.json
themes › Clarkio Blue-color-theme.json › ...
DEBUG
Extension
VARIABLES
WATCH
CALL STACK
LOADED SCRIPTS
BREAKPOINTS
Extension (clarkio-blue)
Live Share
Disconnected
Ln 1, Col 1
Tab Size: 2
UTF-8
LF
JSON with Comments
Prettier

```
{
  "name": "Clarkio Blue",
  "type": "dark",
  "colors": {
    "editor.background": "#263238",
    "editor.foreground": "#eeffff",
    "activityBarBadge.background": "#007acc",
    "sideBarTitle.foreground": "#bbbbbb"
  },
  "tokenColors": [
    {
      "name": "Comment",
      "scope": [
        "comment",
        "punctuation.definition.comment"
      ],
      "settings": {
        "fontStyle": "italic",
```

Once running you'll see a new instance of VS Code open up with a title of "Extension Development Host". This is a separate instance of VS Code running with your extension loaded into it. Back in a previous step when generating this color theme extensions in the terminal/command prompt I chose the "Dark" option for my base theme so I see this as the default color in the Extension Development Host instance of VS Code. You may see different colors based on the base theme you chose during that step. Let's test out changing a color to the theme.


File Edit Selection View Go Debug Terminal Help
Welcome - clarkio-blue - Visual Studio Code - Insiders
DEBUG Extension
[Extension Development Host] - Welcome - Visual Studio ...
Welcome
Start
New file
Open folder...
Add workspace folder...
Recent
clarkio-blue C:\Users\bc\dev\temp
my-color-theme1 C:\Users\bc\dev\temp
clarkio.github.io C:\Users\bc\dev_repos
ttv-chat-light C:\Users\bc\dev\live
audio-normalize C:\Users\bc\dev\temp
More...
Customize
Tools and languages
Install support for JavaScript, TypeScript, Python, PHP, A...
Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Subli...
Color theme
Make the editor and your code look the way you love
Learn
Find and run all commands
Live Share
Disconnected
Extension (clarkio-blue)

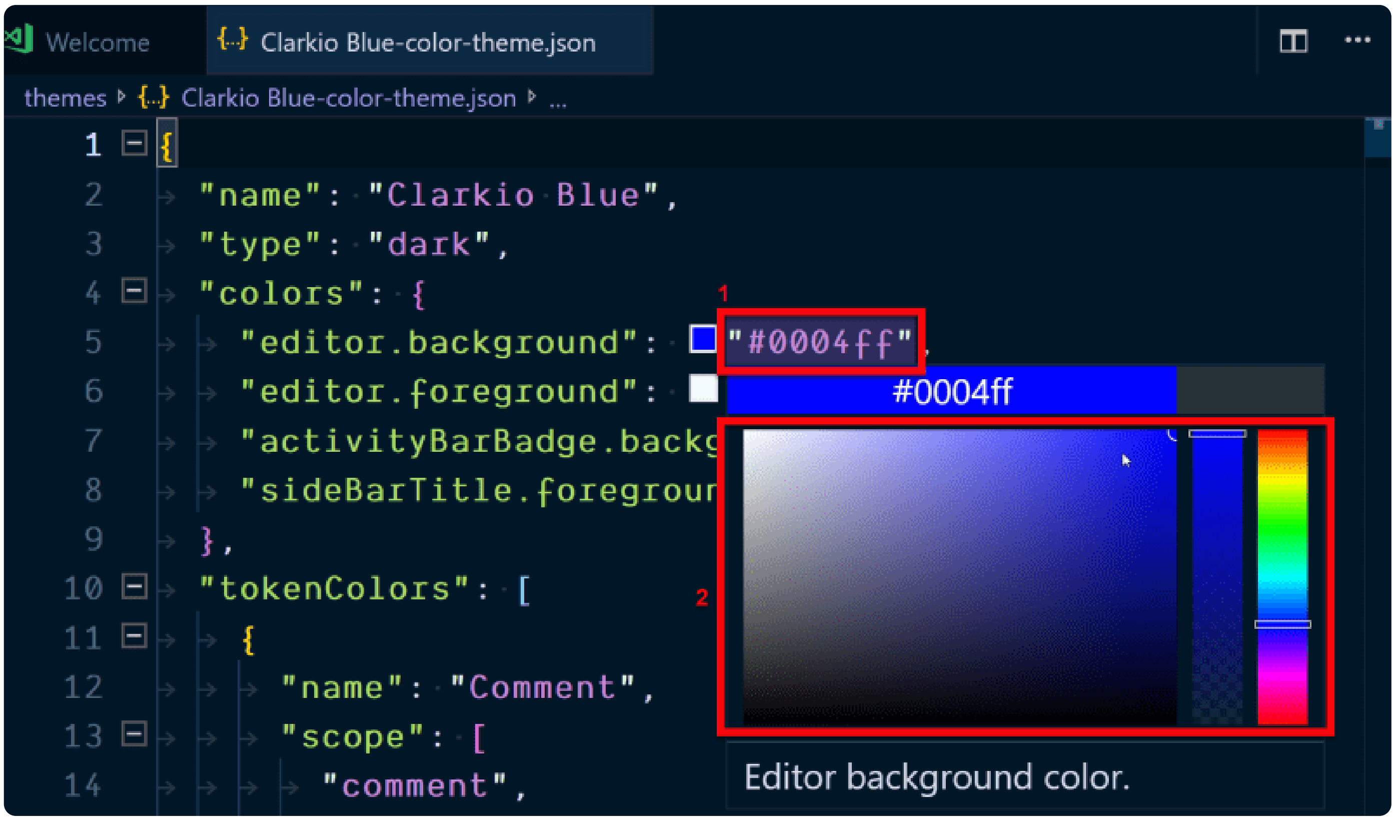
Go back to your other instance of VS Code that has the extension folder opened. Expand the "themes" folder and double click on the `.json` file to open/view the contents.


File Edit Selection View Go Debug Terminal Help
Clarkio Blue-color-theme.json - clarkio-blue - Visual Studio Code - Insiders
Welcome
Clarkio Blue-color-theme.json
themes › Clarkio Blue-color-theme.json › ...
EXPLORER
CLARKIO-BLUE
.vscode
themes
Clarkio Blue-color-theme.json
.vscodeignore
CHANGELOG.md
package.json
README.md
vsc-extension-quickstart.md
OUTLINE
AZURE STORAGE
NPM SCRIPTS
HIGHLIGHTS
Ln 1, Col 1 Tab Size: 2 UTF-8 LF JSON with Comments Prettier

```json
{
  "name": "Clarkio Blue",
  "type": "dark",
  "colors": {
    "editor.background": "#263238",
    "editor.foreground": "#eeffff",
    "activityBarBadge.background": "#007acc",
    "sideBarTitle.foreground": "#bbbbbb"
  },
  "tokenColors": [
    {
      "name": "Comment",
      "scope": [
        "comment",
        "punctuation.definition.comment"
      ],
      "settings": {
        "fontStyle": "italic",
```

Now find the `editor.background` key in the JSON file under the `colors` key. Hover your mouse over the `string` value to bring up the color picker view in VS Code.

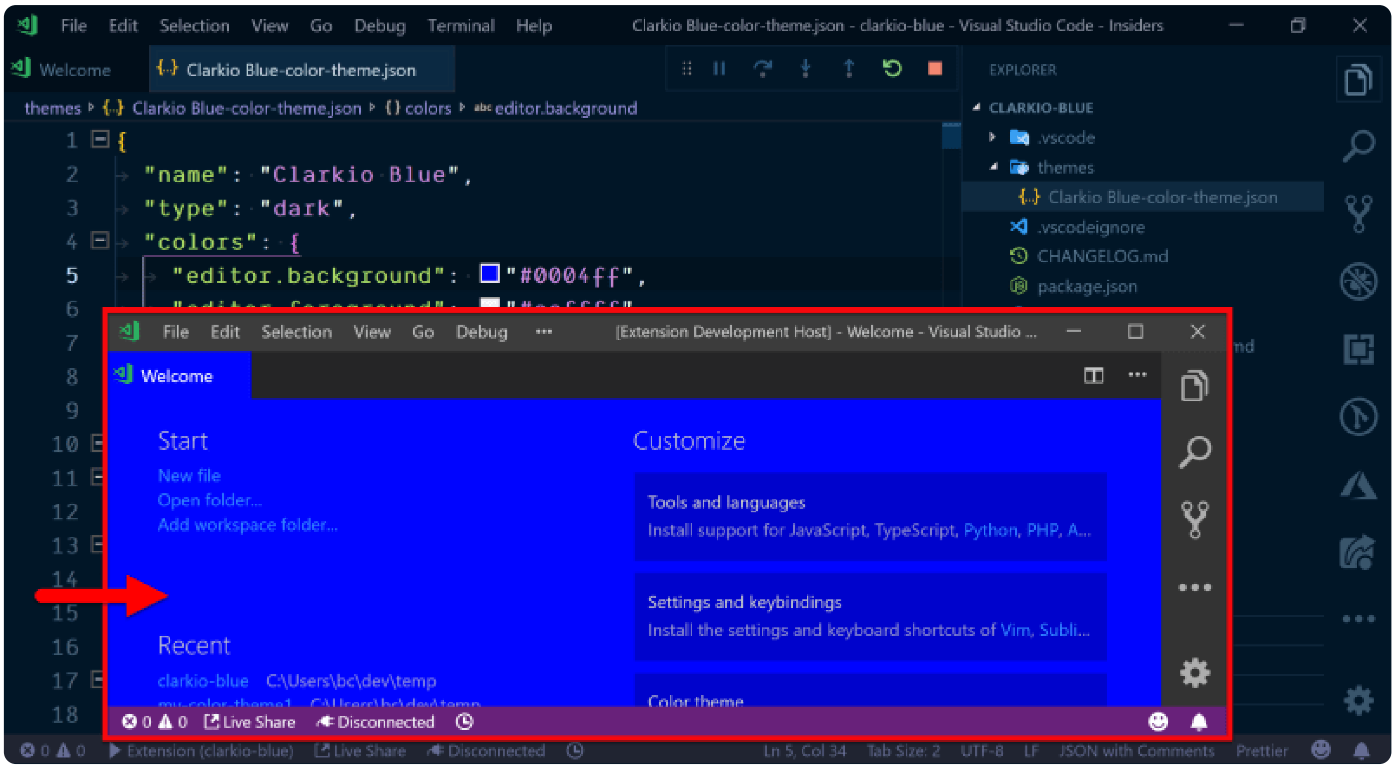
Carefully move your mouse into the color picker pop up view and change the color to something different. You'll see the `string` value change.


Welcome
Clarkio Blue-color-theme.json
themes › Clarkio Blue-color-theme.json › ...
1
2
#0004ff

```
{
  "name": "Clarkio Blue",
  "type": "dark",
  "colors": {
    "editor.background": "#0004ff",
    "editor.foreground": 
    "activityBarBadge.backg
    "sideBarTitle.foregrour
  },
  "tokenColors": [
    {
      "name": "Comment",
      "scope": [
        "comment",
```

Editor background color.

Once you save your changes you'll see the "Extension Development Host" of VS Code will automatically update and display the color change.


File Edit Selection View Go Debug Terminal Help
Clarkio Blue-color-theme.json - clarkio-blue - Visual Studio Code - Insiders
Welcome
Clarkio Blue-color-theme.json
themes › Clarkio Blue-color-theme.json › colors › editor.background
EXPLORER
CLARKIO-BLUE
.vscode
themes
Clarkio Blue-color-theme.json
.vscodeignore
CHANGELOG.md
package.json
File Edit Selection View Go Debug ...
[Extension Development Host] - Welcome - Visual Studio ...
Welcome
0 0 Live Share Disconnected
0 0 Extension (clarkio-blue) Live Share Disconnected Ln 5, Col 34 Tab Size: 2 UTF-8 LF JSON with Comments Prettier

```
{
  "name": "Clarkio Blue",
  "type": "dark",
  "colors": {
    "editor.background": "#0004ff",
```

Start
New file
Open folder...
Add workspace folder...

Customize
Tools and languages
Install support for JavaScript, TypeScript, Python, PHP, A...
Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Subli...
Color theme

Recent
clarkio-blue C:\Users\bc\dev\temp

You've done it! Congratulations on creating your first extension for Visual Studio Code! Feel free to continue tweaking color options in the `.json` file to further customize your color theme extension.

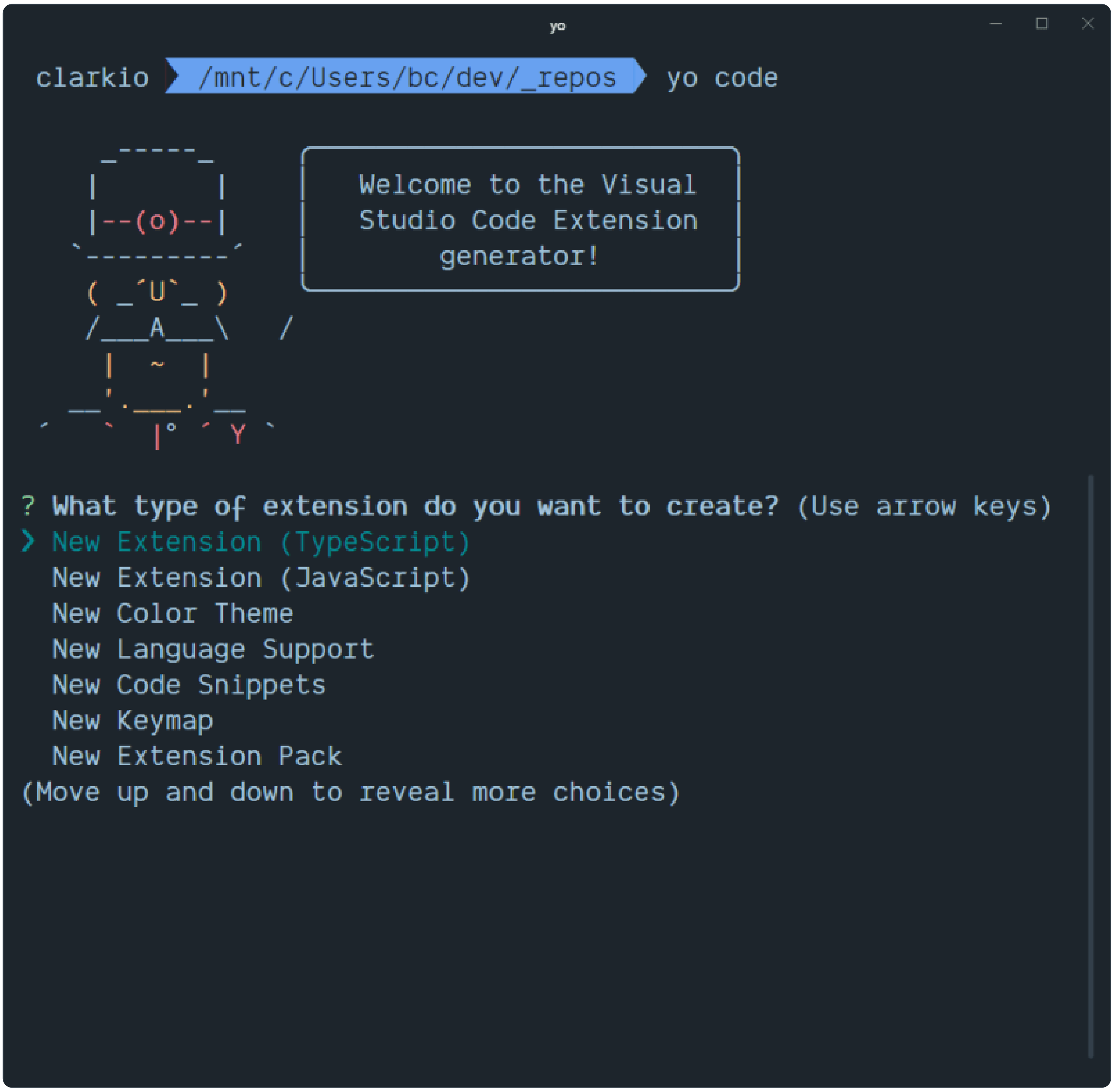
Building Something More

Creating a color theme extension is great and all but it doesn't dive deep into enhancing or creating features within VS Code. Let's walk through creating an extension that results in creating a file that lists all the extensions we're using in VS Code.

Go to your main folder/directory that you'd like to use for development (example: `cd /my/dev/folder`) and run the following command to get started:

```
yo code
```

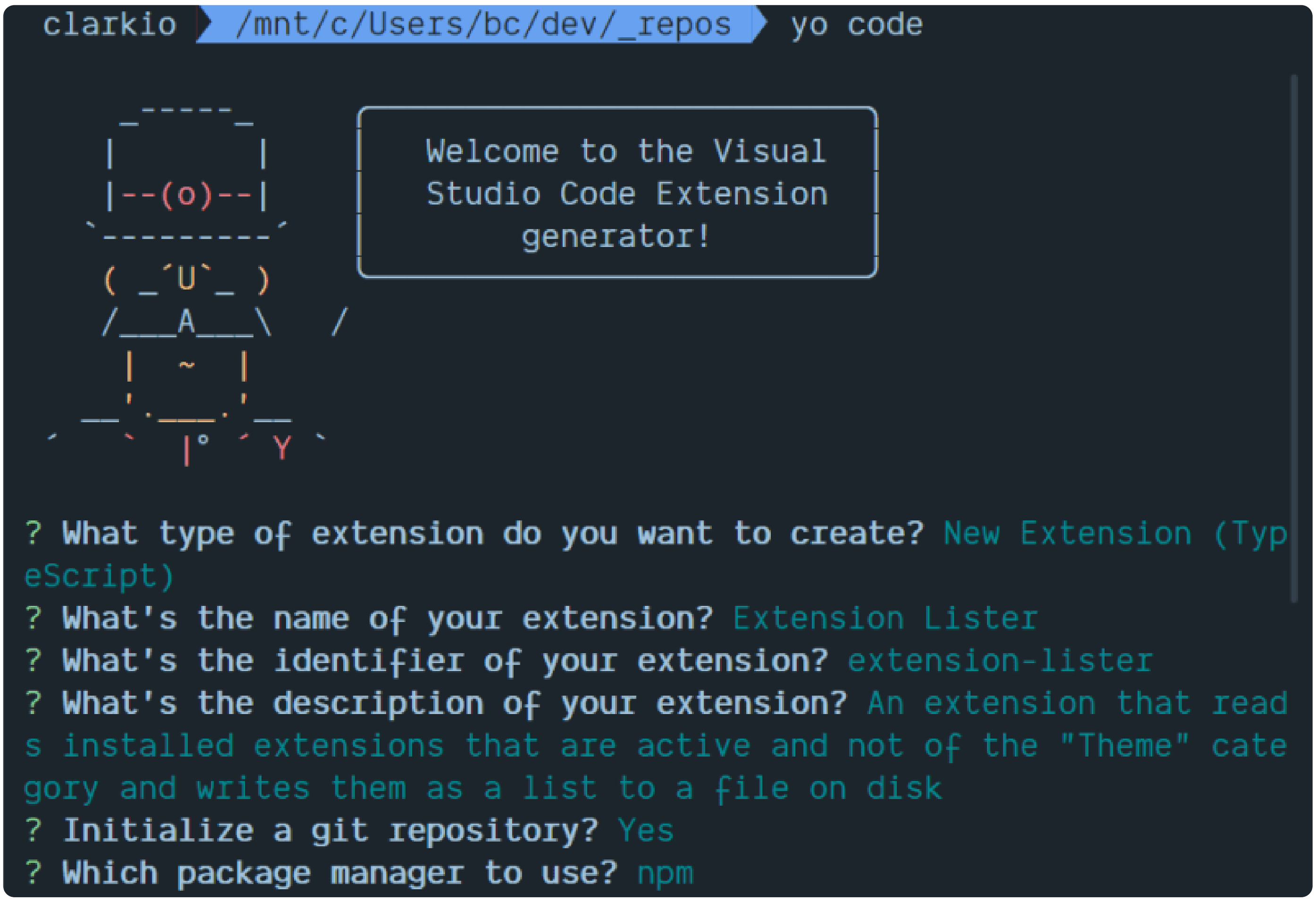
You'll be prompted to choose the type of extension you want to create. Use your keyboard arrow keys to move to the "New Extension (TypeScript)" option and press your `Enter` key.



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

     _-----_     ╭──────────────────────────╮
    |       |    │   Welcome to the Visual  │
    |--(o)--|    │   Studio Code Extension  │
   `---------´   │        generator!        │
    ( _´U`_ )    ╰──────────────────────────╯
    /___A___\   /
     |  ~  |
   __'.___.'__
 ´   `  |° ´ Y `

? 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)
```

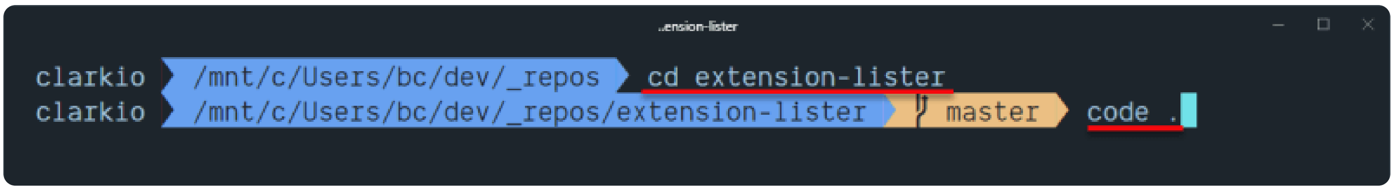
From here, the prompt will ask you to provide details about the extension.

- `name` : this determines the folder it will create in your current directory.
- `identifier` : this is what will be used in the extension marketplace for others to find it so make sure this is unique if you plan to publish it later (I generally use my username handle followed by the name of the extension/theme).
- `description` : longer form text to describe your extension
- name of your theme: this is the text users will see as an option when switching themes within VS Code (after it's been installed)
- `git repository` : this give you the option to initialize the project as a new git repository or not
- `package manager` : choose whichever you prefer between npm and yarn



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

     _-----_     ╭──────────────────────────╮
    |       |    │   Welcome to the Visual  │
    |--(o)--|    │   Studio Code Extension  │
   `---------´   │        generator!        │
    ( _´U`_ )    ╰──────────────────────────╯
    /___A___\   /
     |  ~  |
   __'.___.'__
 ´   `  |° ´ Y `

? 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
```

Change to the directory for the newly generated extension and open it up in VS Code



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

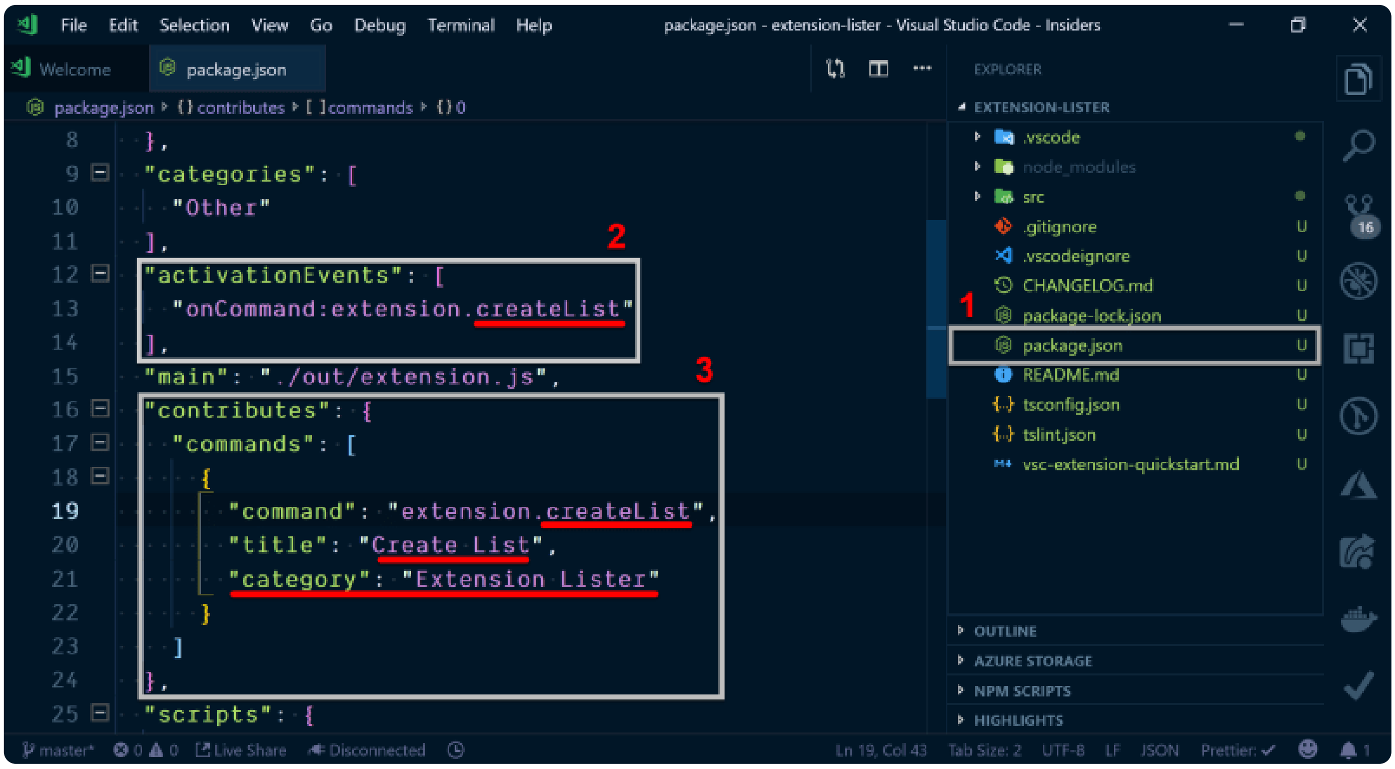
VS Code extensions leverage `package.json` as their manifest and extend it with some further options beyond the conventional options. We won't go into detail for all of them but we'll need to change a few for the purposes of this extension.

1. Open up the `package.json` file and find the `activationEvents` key. This defines what action will activate your extension. At the moment it's defined to activate your extension when the command "helloWorld" is triggered. You'll see this as string value `"onCommand:extension.helloWorld"`. To make sure we keep things consistent with the intent of this new extension replace `helloWorld` with `createList` as that will be the name of the command we'll create.

2. Shortly below the `activationEvents` key you'll find another key `contributes`. This defines the many ways your extension can "contribute" to VS Code. At the

moment it has one option defined and that is via the `commands` option. This is an array defining various commands the extension will make available to VS Code via the command palette. You should see one command defined as `extension.helloWorld`. The "extension" part of that command acts as a namespace when wiring up the command within VS Code and the "helloWorld" part creates a unique name for the command within your extension. We need to make sure this gets changed to match what we updated in the `activationEvents` so replace `helloWorld` here with `createList`.

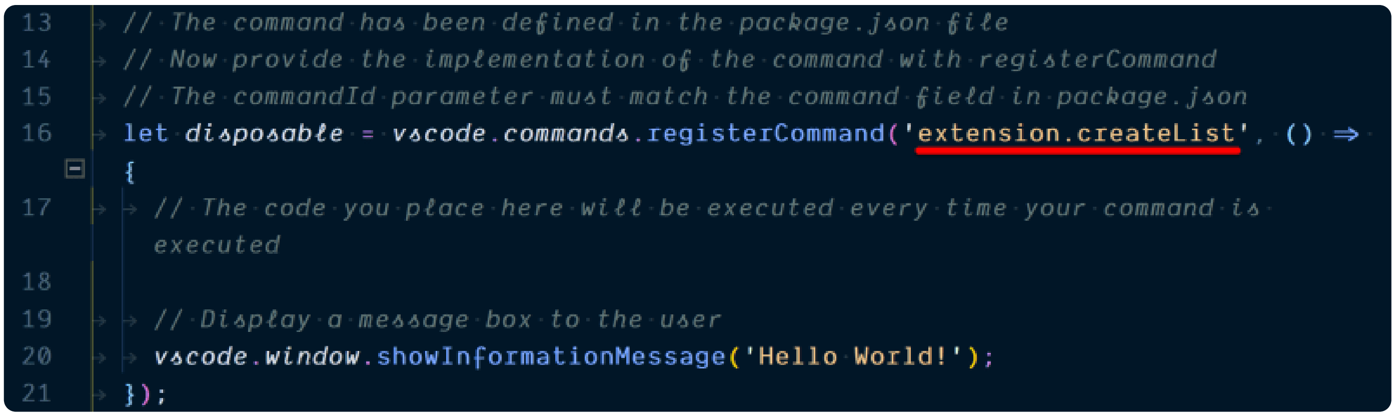
3. The `title` key within the command definition provides the text that will be shown to users within the command palette. Change the value of this key to be `Create List`. The last thing we'll change in this file is add another key below `title` so put a comma at the end of the `title` value and press your "Enter" key for a new line. Add a key `"category"` (intellisense should pop up to help here too) and give it a value `"Extension Lister"`. This `category` helps group commands visually such that they show up as the following in the command palette ": " (Example: "Extension Listener: Create List")


File Edit Selection View Go Debug Terminal Help
package.json - extension-lister - Visual Studio Code - Insiders
Welcome
package.json
package.json › {} contributes › [ ] commands › {} 0
EXPLORER
EXTENSION-LISTER
.vscode
node_modules
src
.gitignore
.vscodeignore
CHANGELOG.md
package-lock.json
package.json
README.md
tsconfig.json
tslint.json
vsc-extension-quickstart.md
OUTLINE
AZURE STORAGE
NPM SCRIPTS
HIGHLIGHTS
1
2
3
master* ⊗ 0 ⚠ 0 Live Share Disconnected
Ln 19, Col 43 Tab Size: 2 UTF-8 LF JSON Prettier: ✓

```
  },
  "categories": [
    "Other"
  ],
  "activationEvents": [
    "onCommand:extension.createList"
  ],
  "main": "./out/extension.js",
  "contributes": {
    "commands": [
      {
        "command": "extension.createList",
        "title": "Create List",
        "category": "Extension Lister"
      }
    ]
  },
  "scripts": {
```

Let's get into building out the functionality for this extension. Within the `src` folder you'll see a file `extension.ts`. This is the main file that's used to load up your extension and corresponding functionality. Take a moment to read the comments within the code that helps describe some of the parts of code that were generated for you... Seriously go read it and come back.

Now that you're a bit more familiar with the code let's get our first command set up. We first need to update our command code to match the changes we made in the `package.json`. Find the code `registerCommand('extension.helloWorld'` and replace `helloWorld` with `createList`.



```
13  // The command has been defined in the package.json file
14  // Now provide the implementation of the command with registerCommand
15  // The commandId parameter must match the command field in package.json
16  let disposable = vscode.commands.registerCommand('extension.createList', () =>
    {
17      // The code you place here will be executed every time your command is
        executed
18
19      // Display a message box to the user
20      vscode.window.showInformationMessage('Hello World!');
21  });
```

Ok the command is wired up correctly and we're ready to start gathering the list of extensions. The VS Code API provides a namespace `extensions` which contains a list of all extensions available in the instance under the `all` array. We access the API via the `import * as vscode from 'vscode';` statement. Since that was already done within the `registerCommand` (on line 17) let's grabbed the array and assign it to a constant. Replace the line of code `vscode.window.showInformationMessage('Hello World!');` (and the comment above it) with the following code:

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

If you run and debug the extension with a breakpoint on this line you'll notice the array contains literally all extensions installed and available for the instance of VS Code the extension is running within. Feel free to try this yourself now but it is not required. This is a good start but if you inspect collection of extensions you'll notice it includes extensions that are built-in for VS Code, ones that are not currently active and extensions that are themes.

Being that we're getting extensions outside the scope of what we want to include in the list (active, non-theme type, extensions) we need to filter some of these results first. Add the following array method, `filter`, to capture the extensions within scope.

```
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'
      )
    );
  }
);
```

Next we want to format the content of what will be written to the file to be as follows: `<extension name>: <extension marketplace URL>`. This will make it easy to share and allow others to quickly install the extensions. To do this let's iterate over the array of extensions to create a string of the content to write to the file. Use the following code to accomplish that and add it after the `activeExtensions` array is defined and assigned:

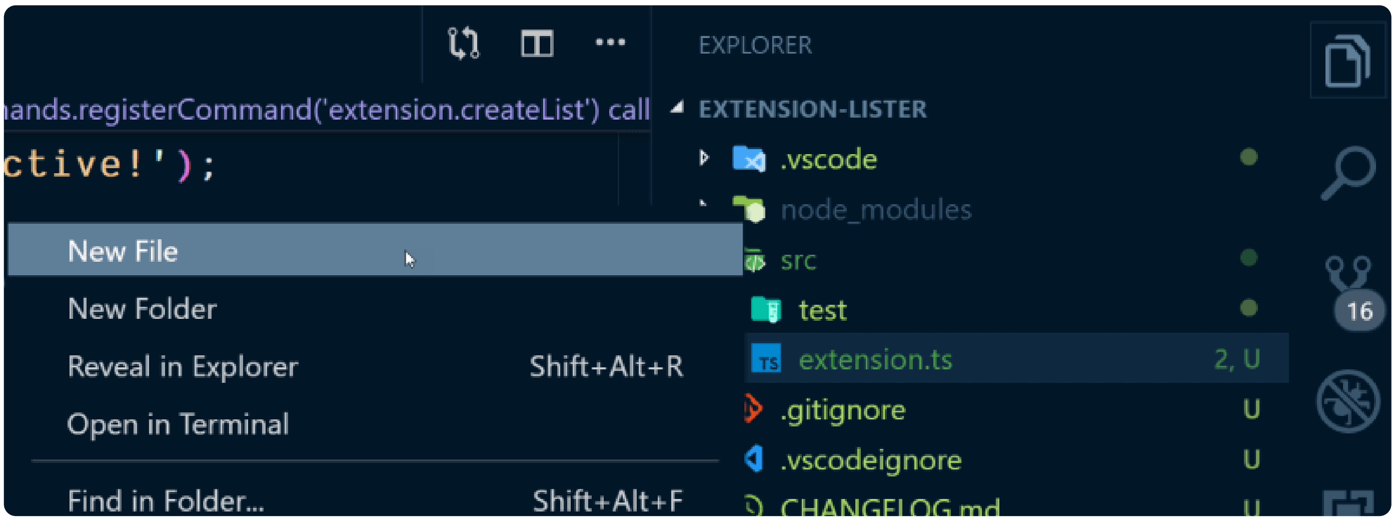
```
let extensionListData = ''; // variable to hold the file contents as a string
activeExtensions.forEach((extension: vscode.Extension<any>) => {
  // thanks to TypeScript and the exposed Extension type from the VS Code API we get
  // In particular we want to read the display name property found in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

At this point we have the data we need and in a human readable format so we're ready to ask the user where they want to save this file. For visual prompts and queues VS Code provides options via its `window` namespace. In particular we want the `showSaveDialog` function. There are some options/arguments we can provide to it such that it helps create a better user experience in this dialog. We know this will just be a plaintext file so we'll provide a filter option that says to limit the file type to `*.txt`.

When the user has completed their actions in the save dialog the resulting URI (universal resource indicator) for the file will be returned to us in the promise `.then()` resolve function. We'll need to make sure a URI was indeed provided first. If it wasn't we can show an error dialog with `vscode.window.showErrorMessage()`. Once we know we at least have a value we can write to the file but let's hold on that just for a moment. See the following code which accomplishes what we've discussed in this section:

```
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...
});
```

We want to implement writing to a file on the Operating System's file system, however, we shouldn't pollute our main extension code with that code. Let's create a separate file to handle writing a file to the file system. Open VS Code's file explorer view by clicking on the icon or using the keyboard shortcut `CTRL/CMD + SHIFT + E`. Right-click on the `src` folder and choose "New File"



Give it the name `file-manager` and press enter to complete the file creation. VS Code should automatically open up that file but if not double-click on it. We won't go through the following code as it's not really pertinent to the creation of VS Code extensions but know that it's handling writing the contents to a file. Feel free to review the code if you'd like. Add this to the `file-manager.ts` file and save:

```
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);
}
```

We can now import the `writeExtensionListFile()` function from the `file-manager.ts` file we created to handle the last step. Open up the `extension.ts` file again and add an import statement to the top of it after the `vscode` import (found on line 3):

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

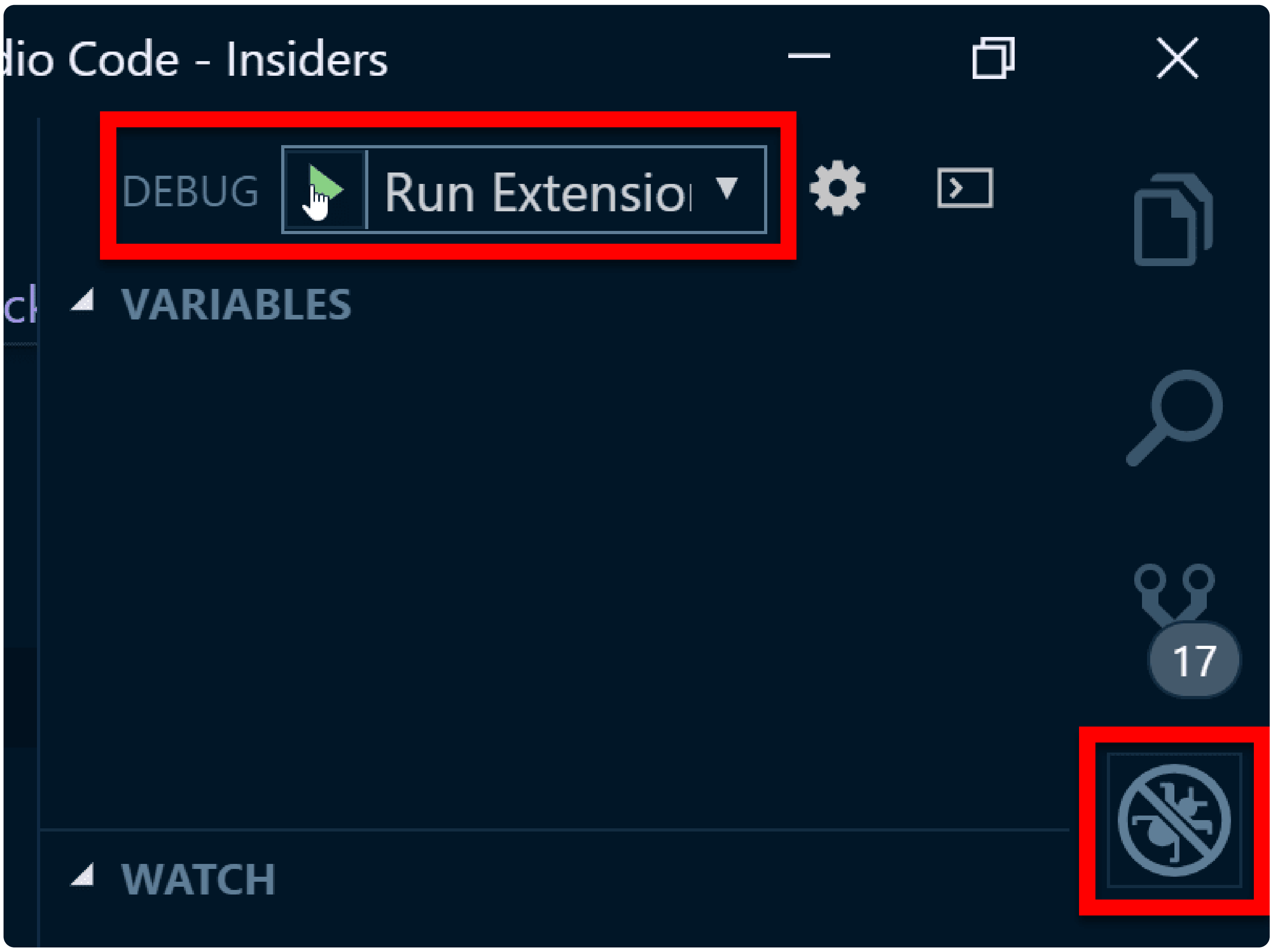
Sweet, let's use it. Go down to where we have the `showSaveDialog` and after the part that's checking for a `uri` value add the following code:

```
// 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);
  });
```

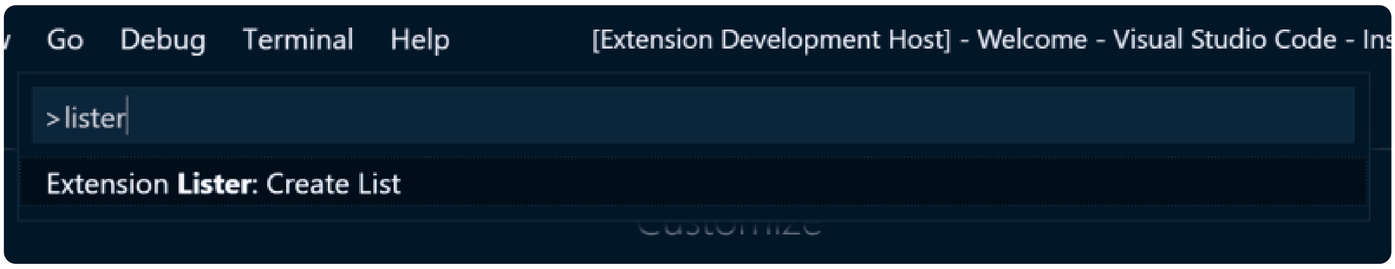
What does this code do? It's calling the function `writeExtensionListFile` and passing in the full file system path value (a property on the `uri` object) and the contents we want to be written to it. If that goes through successfully we let the user know through the `vscode.window.showInformationMessage()` function otherwise we show an error message with the `vscode.window.showErrorMessage()` function.

> In case something isn't working correctly for you use the following GitHub repository as a reference for the completed code: [VS Code Extension Lister]

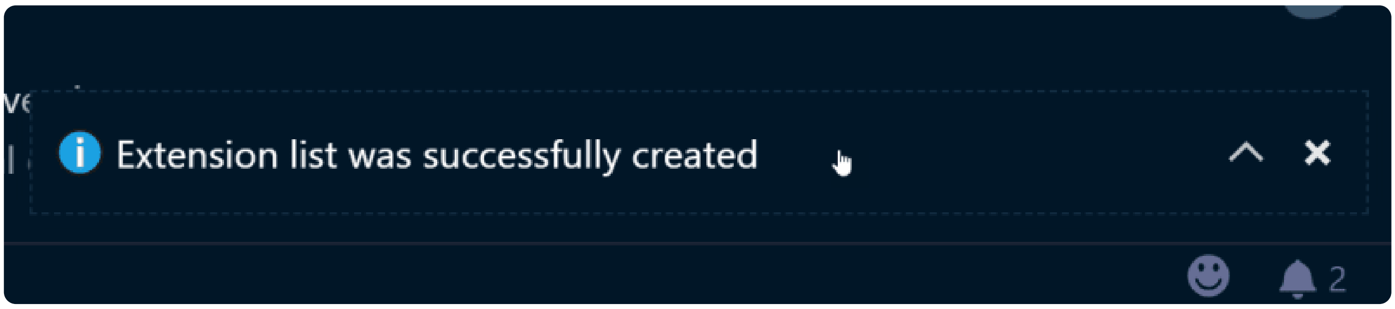
How about we give it a run and test it out? Press `F5` to start debugging or open the debug view in VS Code and press the green "Play" button.



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.

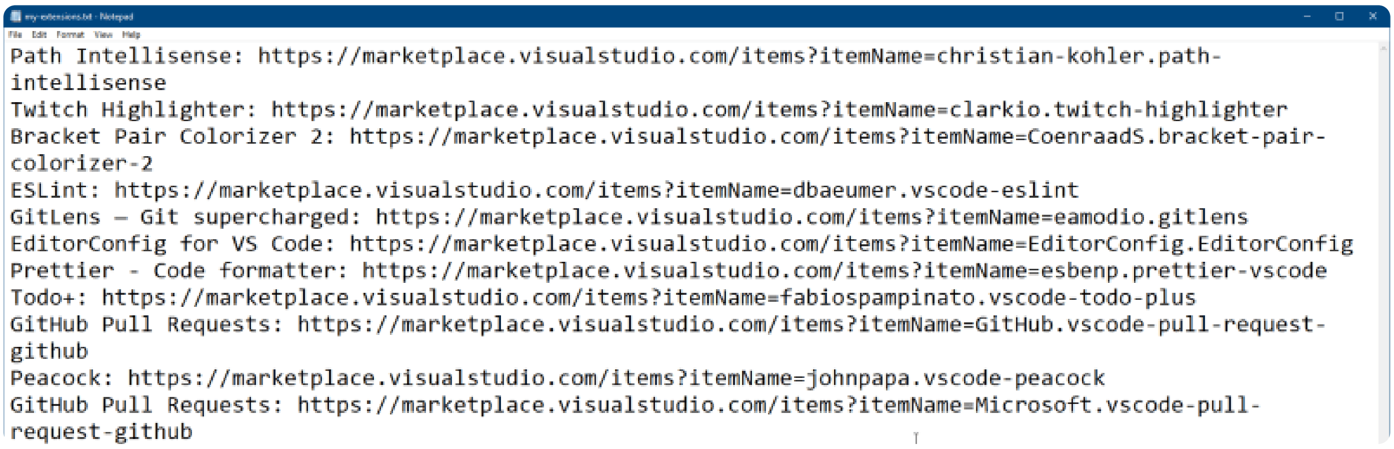

Go Debug Terminal Help
[Extension Development Host] - Welcome - Visual Studio Code - Ins
>lister
Extension Lister: Create List

Press the "Enter" key on your keyboard and you should be prompted to choose a location and name for your file. Pick a location and name for your file and click the "Save" button. You should see a success notification pop up in the bottom right-hand corner of VS Code.



Extension list was successfully created

Just to make sure go to where you saved the file and open it to see the contents. You should see the full list of active extensions you're using with their marketplace URLs.



```
Path Intellisense: https://marketplace.visualstudio.com/items?itemName=christian-kohler.path-
intellisense
Twitch Highlighter: https://marketplace.visualstudio.com/items?itemName=clarkio.twitch-highlighter
Bracket Pair Colorizer 2: https://marketplace.visualstudio.com/items?itemName=CoenraadS.bracket-pair-
colorizer-2
ESLint: https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint
GitLens — Git supercharged: https://marketplace.visualstudio.com/items?itemName=eamodio.gitlens
EditorConfig for VS Code: https://marketplace.visualstudio.com/items?itemName=EditorConfig.EditorConfig
Prettier - Code formatter: https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode
Todo+: https://marketplace.visualstudio.com/items?itemName=fabiospampinato.vscode-todo-plus
GitHub Pull Requests: https://marketplace.visualstudio.com/items?itemName=GitHub.vscode-pull-request-
github
Peacock: https://marketplace.visualstudio.com/items?itemName=johnpapa.vscode-peacock
GitHub Pull Requests: https://marketplace.visualstudio.com/items?itemName=Microsoft.vscode-pull-
request-github
```

Congrats you've created another VS Code extension and one that adds new functionality!

How do I publish the extension?

Rather than write that same information again here take a look the Publishing Extension documentation. The documentation provided on the VS Code website is pretty straightforward and easy to follow so check that out for further details.

Where do I go to find out more?

The best place to read up more on VS Code extension development is at the official VS Code API Docs site. There you'll find a table of contents on the left-hand side that is expandable and breaks out important topics around extension development.

Closing

If there are more questions you have that haven't been answered here let me know in the comments. I'll check in regularly and follow up with a response or a post if it's more appropriate in that format.

Thanks for reading this post and I hope you found it helpful.

Get connected on:

- Twitter
- Twitch
- Instagram
- GitHub
- YouTube
- Website

Top comments (7)

James Allen • 10 мая 19 г.

Hi Brian, your write up here has me want to ask... could an extension read the user's code and render something in-line? Like if I have a comment in my code such as...

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

I'd like an extension to replace that with a link to my Azure DevOps ABC project's ticket 255127.

I've found the restrictions page in Code's API docs and I'm assuming I cannot have an extension do this. Is there anything possible you're aware of?

Tanner B. Hess Webber • 12 сент. 19 г.

At the very worst, you should be able to use something like Acorns to generate an AST and then walk the tree to find comments that match that pattern, query your DevOps source of truth, and add text to the second line. That's a lot of overhead, so hopefully you can find a better way.

On a single instance basis, you should be able to do this easily just by highlighting the comment, running the command, parsing the comment, querying your source of truth, and then using the "editBuilder" API to populate the ticket info. You may have to wrestle with some auth stuff, but that shouldn't be terrible.

Brian Clark 💡 • 12 сент. 19 г.

Hi James I know this is very delayed but I don't believe those restrictions will hinder you from what you're trying to accomplish. I'm not entirely sure but I'd look into the Programmatic Language features and specifically the refactoring capabilities to see if that would work for your scenario.

link -> code.visualstudio.com/api/language...

Hope this helps

Tanner B. Hess Webber • 12 мая 19 г.

Hey Brian,

Great post! I'm curious if you know of (or have written) any good resources for testing extensions as you build them. So far, this has been my biggest hurdle with extension development

Thanks in advance for any help!

Brian Clark 💡 • 12 сент. 19 г.

Hey Tanner,

My bad for the very delayed response. I haven't dug deep into doing this just yet but when I do I'll be using John Papa's VS Code extension, Peacock, as a reference: github.com/johnpapa/vscode-peacock

Tanner B. Hess Webber • 12 сент. 19 г.

Oh, that's a great repo for really exploring what you can do! I've now built a few internal extensions for my company and have yet to add tests to them (TDD is my jam, so this is a sad thing for me). Time to go back and add in tests! Thanks a lot!

Ashwini Manoj • 26 окт. 19 г.

Wow, exactly what I was looking for as well. I wasn't sure how to go about testing an extension! Thanks!

Trending in JavaScript

The JavaScript community is currently discussing the process of creating and releasing side-projects, exploring advanced Node.js concepts, and experimenting with CSS-only syntax highlighting. There's also interest in understanding React props visually and using ExpressoTS for server-side applications.

Creating a more than minor side-project: From planning to release

Lucas Lima do Nascimento ▪ Aug 24
#svelte #javascript #webdev #tutorial

📦 React Props: A Visual Guide

Reed Barger ▪ Aug 18
#react #webdev #beginners #javascript

Impossible⁉️ CSS only syntax highlighting 😱 ...with a single element and GRADIENTS 🤯

GrahamTheDev ▪ Aug 16
#css #html #javascript #webdev

Notes: Advanced Node.js Concepts by Stephen Grider

Muhammad Bin Zafar ▪ Aug 19
#javascript #node #webdev #beginners

A first project with ExpressoTS

Daniel Boll for ExpressoTS ▪ Aug 27
#javascript #typescript #expressots #restapi

Microsoft Azure

Invent with purpose

Any language. Any platform.

Try Azure for Free

More from Microsoft Azure

Do More using GitHub Copilot with AI and VS Code
#javascript #githubcopilot #vscode #ai

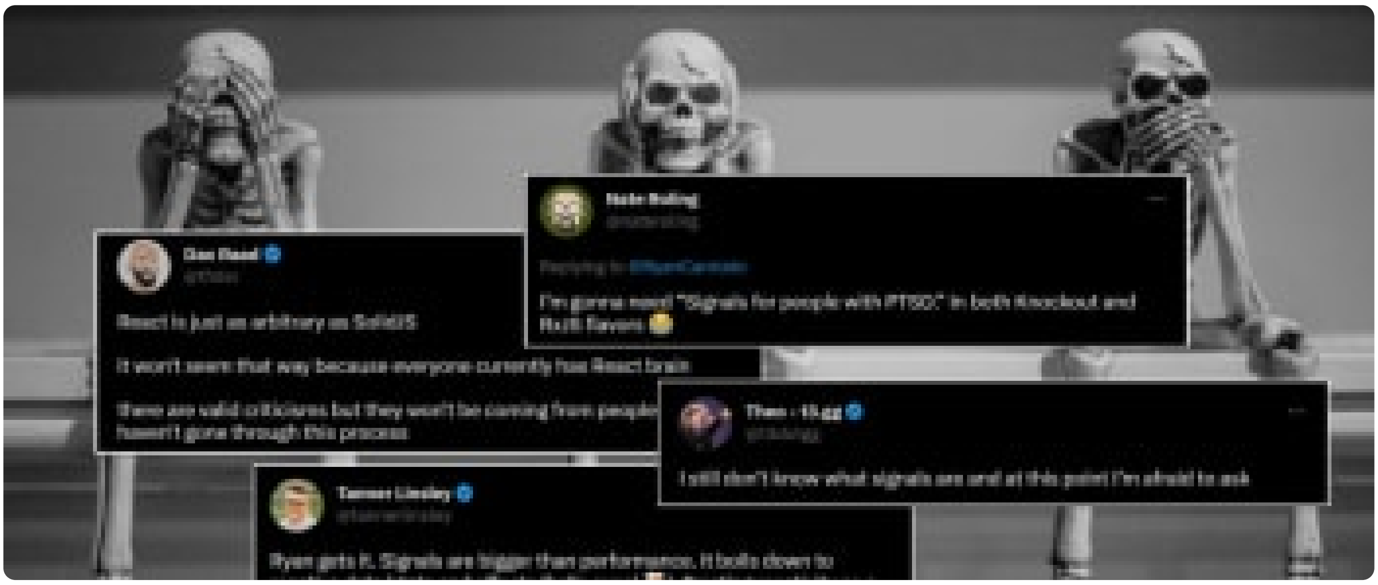
Lets #HackTogether: Javascript On Azure Keynote
#contosorealestate #javascript #azure #hacktogether

Step by Step Guide: Migrating v3 to v4 programming model for Azure Functions for Node.Js Application
#javascript #hacktogether #contosorealestate #webdev

DEV Community •••

The most essential JavaScript conversations happen on DEV

["Those who fail to learn from history are doomed to repeat it"]



Dan Reed [illegible]

React is just as arbitrary as SolidJS

It won't seem that way because everyone currently has React brain

there are valid criticisms but they won't be coming from people [illegible] haven't gone through this process

Nate Roling [illegible]

Replying to @RyanCarniato

I'm gonna need "Signals for people with PTSD" in both Knockout and RxJS flavors

Theo [illegible]

I still don't know what signals are and at this point I'm afraid to ask

Tanner Linsley [illegible]

Ryan gets it. Signals are bigger than performance. It boils down to [illegible]

☝️ **Ryan Carniato** and **Dan Abramov** discuss React fundamentals!