Visual Studio Codeの特徴

Visual Studio Code（以降VS Code）は、元々プログラマーのためのコードエディターとして開発されました。しかし、Webページの製作や原稿の執筆にも応用できる機能を備えているため、ノンプログラマーのユーザーも増えてきています。

VS Codeでできること

VS Codeは、プログラム開発、Web制作、テキスト原稿の執筆など、さまざまな用途に使えるテキストエディターです。

本来はプログラム用のソースコード（プログラミング言語で書かれたテキストファイル）を編集するためのものなので、取っつきにくく感じる面もありますが、基本は普通のテキストエディターです。一般的な文章などを書くために使ってもまったく問題ありません（本書の原稿もVS Codeで書いています）。
従来のテキストエディターにはないメリットとしては、プログラミングができることに加えて、**VS Code本体の機能追加や、拡張機能の追加が非常に早い**ことです。流行のオープンソースソフトウェア（ソースコードが公開され、誰でも開発に参加できる）なので、世界中の開発者がVS Codeを便利にするプログラムを開発し、原則無料で提供してくれています。

原稿執筆ツールとしてのVS Code

VS Codeのテキスト編集機能は、初期状態でも他のテキストエディターと比べて不足ないものです。文字数カウントといった、テキスト編集用の拡張機能もいろいろ提供されているので、自分が使いやすい形にカスタマイズできます（第3章参照）。

VS Codeならではの特徴は、公文書などの作成にも使われはじめている**Markdown（マークダウン）**のサポートが厚いことです。Markdownはドキュメント記述言語の一種で、#や*などの記号で「見出し」「強調」「箇条書き」などの簡単な書式を指定でき、HTML（Webページのファイル）にも変換可能です。
VS Codeは標準でかなり強力なMarkdownプレビュー機能を備えています。編集中にリアルタイムで更新されるだけでなく、スクロール同期表示や、複数のMarkdownファイルを切り替えながら表示することもできます。

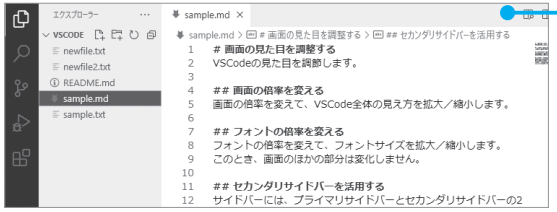
画面を見やすくカスタマイズする

VS Codeは、エディターやサイドバーなどの領域を自由に調整することができます。画面やフォントサイズを拡大／縮小し、使用していないバーは非表示にするなど工夫して、編集に集中しやすい画面を設定しましょう。

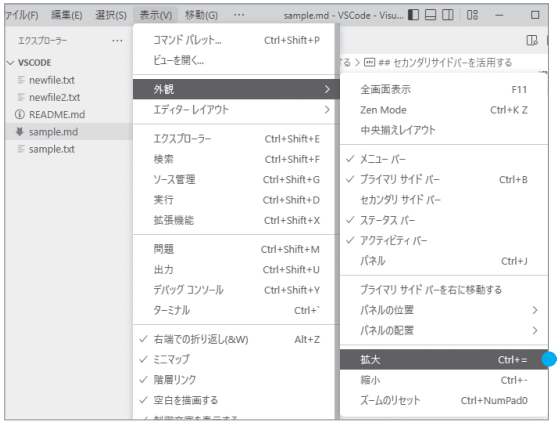
画面の倍率を変える

画面の倍率を変えると、VS Codeそのものの外観が拡大／縮小します。エディターに表示される文字だけではなく、メニューバーの文字やアクティビティバーのアイコンの大きさも変わります。

画面の倍率を拡大する

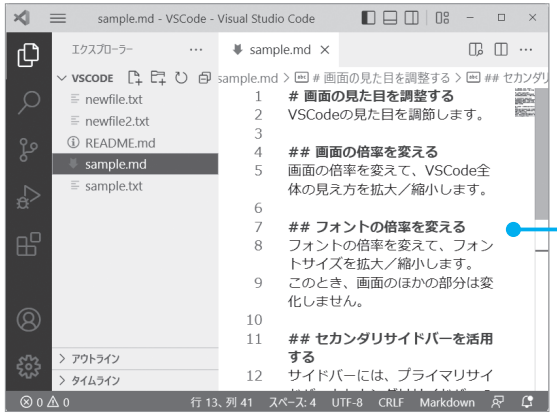


左図は初期設定の倍率です。



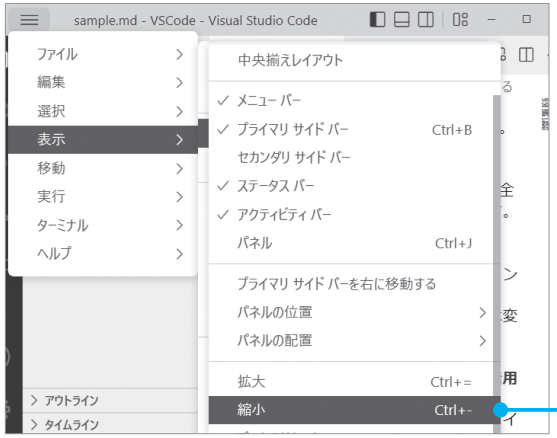
1 <表示>→<外観>→<拡大>をクリックします。

command
拡大
Win Ctrl+;
Mac command+shift+−



2 画面が拡大されました。
1回で20%拡大表示されます。
左図は40%拡大した状態です。

画面の倍率を縮小する



1 <表示>→<外観>→<縮小>をクリックします。

command
縮小
Win Ctrl+−
Mac command+−

2 画面が縮小されました。
1回で20%縮小表示されます。
左図は60%（初期設定の20%）縮小した状態です。

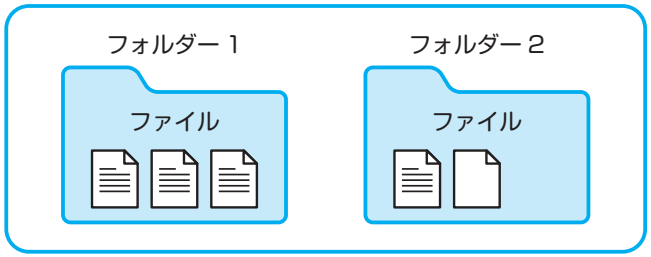
section 015

ワークスペースで複数のフォルダーをまとめる

フォルダーを開く機能では、1つのウィンドウに1つのフォルダーしか開けません。同時に複数のフォルダーを参照したい場合、ワークスペース機能を利用しましょう。ワークスペースごとにVS Codeの設定を切り替えることも可能になります。

ワークスペースとは

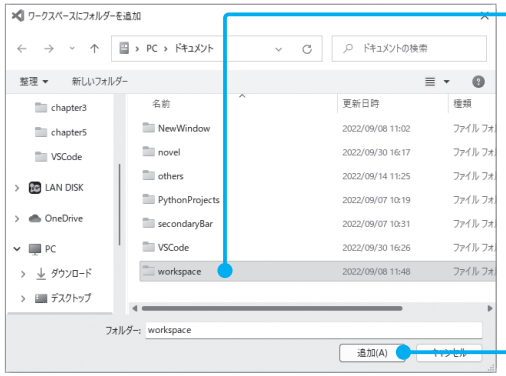
1つの案件で使用するファイルが1フォルダーにまとまっていないこともあります。その場合に役立つのがワークスペースです。離れた場所のフォルダーを登録し、エクスプローラービューにまとめて表示できます。



新しいワークスペースを作成する

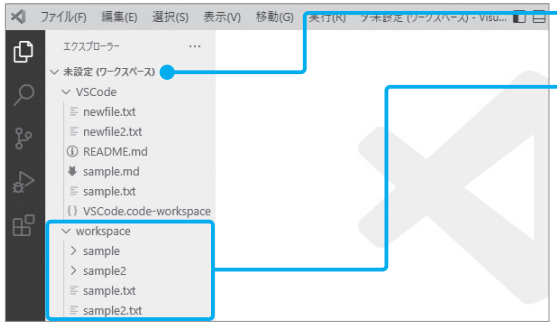
下の画像は、1つのウィンドウに1つのフォルダーを開いています。ワークスペースを作成し、複数のフォルダーを開きます。

1 <ファイル>→<フォルダーをワークスペースに追加>をクリックします。



2 追加するフォルダーを選択して、

3 <追加>をクリックします。



4 新しくワークスペースが作成され、

5 ワークスペースにフォルダーが追加されました。

この時点ではワークスペースが保存されていないため、「未設定(ワークスペース)」と表示されます。

ワークスペースを保存する

ワークスペースの情報は、.code-workspaceという拡張子のファイルとして保存する必要があります。

section 007

ファイルを自動で保存する

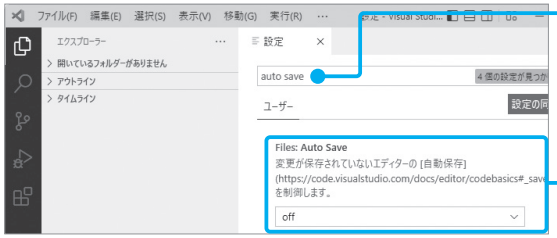
VS Codeの既定では、ファイルは自動的に保存されません。「Files:Auto Save」という設定項目を変更して編集したファイルを自動的に保存する設定にして、保存し忘れることを防止しましょう。

「Files:Auto Save」の設定変更

「Files:Auto Save」を設定することで、編集したファイルの保存し忘れを防ぐことができます。設定値によって自動保存のタイミングが変わります。

「Files:Auto Save」の設定値

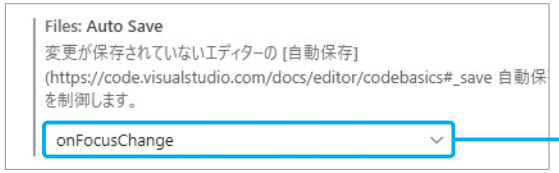
設定値	説明
off	ファイルを自動保存しない（既定）
afterDelay	「File: Auto Save Delay」で指定した時間が経過してから自動保存する
onFocusChange	エディターで操作しているファイルを切り替えると、自動保存する
onWindowsChange	VS Codeからフォーカスが外れると自動保存する



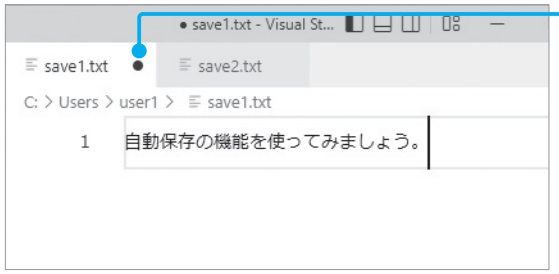
1 検索欄に「auto save」と入力し、

2 「Files: Auto Save」を探します。

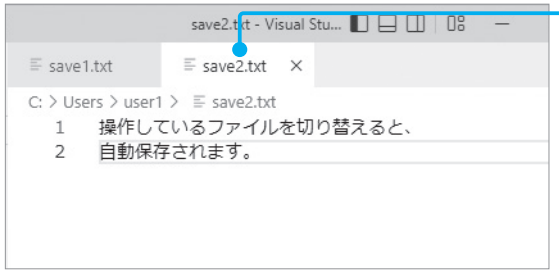
3 プルダウンリストをクリックします。



4 onFocusChange を選択します。



5 ファイルを編集します。未保存ではファイル名の右側に「●」が表示されます。

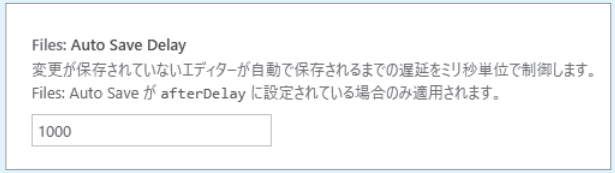


6 別のファイルに切り替えると、作業中だった「save.txt」の「●」がなくなり、自動保存されました。

column

「afterDelay」について

「Files:Auto Save」で「afterDelay」を選択した場合、ファイルを編集してから「File:Auto Save Delay」という項目で設定した時間が経過したあとに自動で保存されます。時間の単位はミリ秒（1ミリ秒は1秒の1,000分の1）なので、既定の「1000」では1秒後に自動保存されます。



settings.jsonを開く

P.71で説明したVS Codeの設定を変更する方法のうち、2つ目のsettings.jsonを編集する方法を紹介します。まずは、JSONというファイル形式について理解することから始めましょう。

settings.jsonについて

JSONとは「JavaScript Object Notation」の略です。データのやり取りに適したファイル形式で、「ジェイソン」と読みます。settings.jsonはJSON形式で記述されたVS Codeの設定ファイルで、このファイルを編集することでVS Codeのすべての設定を変更できます。

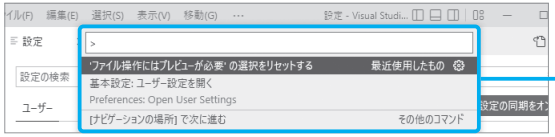
JSONの書き方

基本的なJSONの書き方は、{}（波かっこ）の中にダブルクォート（"）で囲った「キー名」を書き、コロン（:）で区切ってキー名に対応する「値」を書くというものです。複数のキーと値を書く場合は、カンマ（,）による区切りが必要です。
settings.jsonでは、「キー名」の部分には設定ID（P.73参照）を書きます。

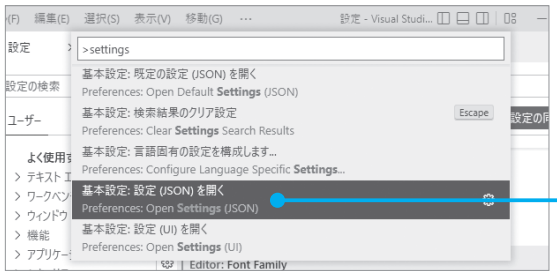
```
{
  "キー名": 値,
  "キー名": 値,
  "キー名": 値
}
```

設定画面とsettings.json

settings.jsonの開き方

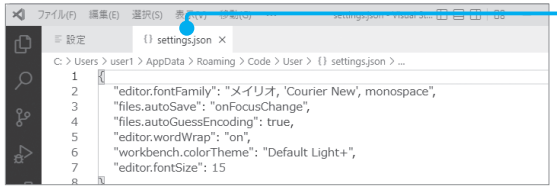


1 コマンドパレットを表示します。



2「settings」と検索し、「Preferences: Open Settings(JSON)」コマンドを実行します。

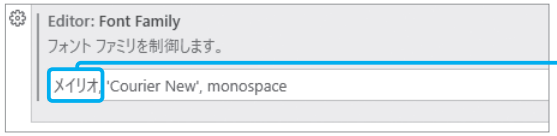
「Default Settings」のほうではないので注意してください（P.94参照）。



3 settings.jsonファイルが開きます（ファイルの内容は環境によって異なります）。

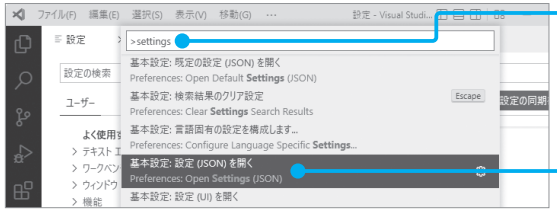
設定画面とsettings.jsonの比較

P.71では、設定画面から設定を変更する方法を説明しました。実は設定画面はsettings.jsonと連動していて、設定画面から設定を変えるとsettings.jsonの内容も自動的に書き換えられます。



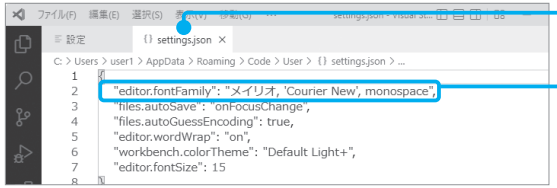
1 設定画面を開きます。

2「Editor:Font Family」の最初の値を「メイリオ」に書き換えます。



3 コマンドパレットで「settings」と検索し、

4「Preferences: Open Settings (JSON)」コマンドを実行します。



5 settings.jsonファイルが開きます。

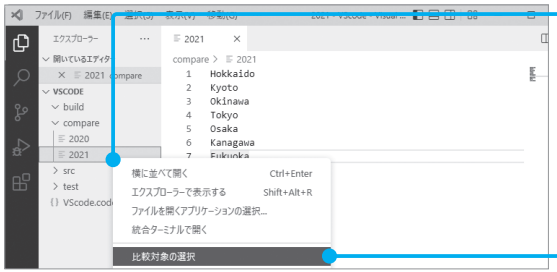
6 設定画面の「Editor:Font Family」の内容と、settings.jsonの「editor.fontFamily」の値が同じになっています。

section 005

2つのファイルの内容を比較する

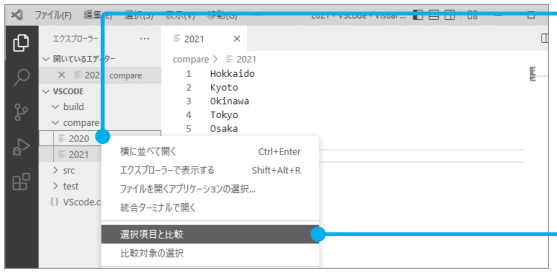
ファイルの最新の内容と古い内容を比較したいときなど、ファイル同士の内容を比べたい場合もあるでしょう。VS Codeにはそのようなときに利用できるファイル比較の機能があります。

エクスプローラービューから2つのファイルを比較する



1 エクスプローラービューのファイル名を右クリックします。

2 <比較対象の選択>をクリックします。



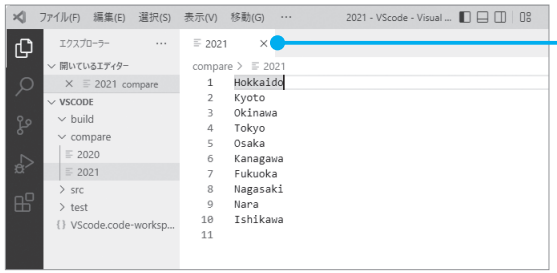
3 比較するもう1つのファイル名を右クリックします。

4 <選択項目と比較>をクリックします。

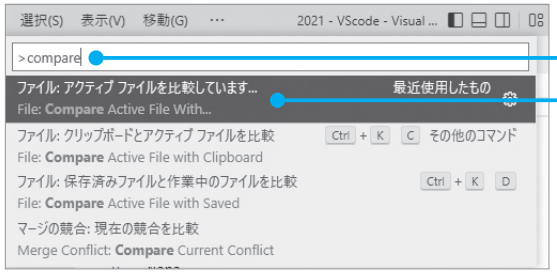
5 2つのファイルがエディターの左右に並び、異なる部分が強調表示されました。

コマンドパレットから2つのファイルを比較する

ファイルの比較はエクスプローラービューのクリック操作ではなく、コマンドパレットからでも行えます。

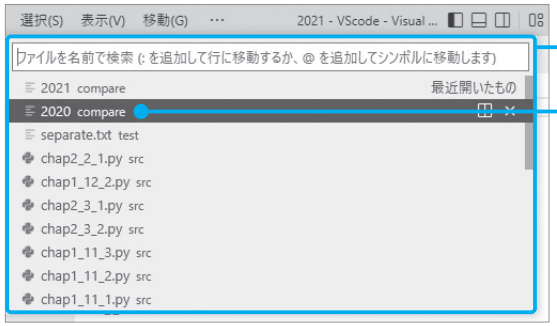


1 比較するファイルを開きます。



2 コマンドパレットを開き、入力欄に「compare」と入力します。

3 < File: Compare Active File With >をクリックします。



4 最近開いたファイルの一覧が表示されます。

5 クイックオープン（P.62 参照）と同じように、この一覧から比較したいファイルを選択するか、ファイル名を入力して検索します。

section 005

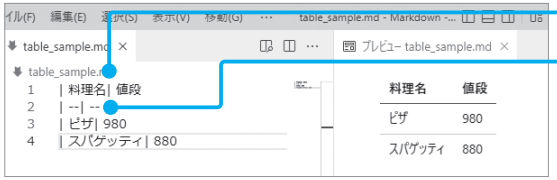
Markdownファイルで表を作成する

Markdown記法では、表（テーブル）を表現できます。テキストで見ても表のような見た目になります。ここでは基本的な表の作りかたと、拡張機能を使った表の整形方法を紹介します。

テーブルを作成する

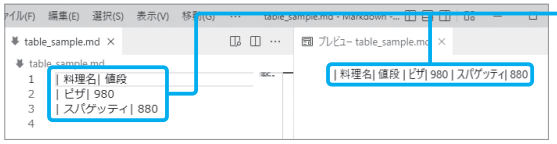
表を作るには「|（パイプ）」でセルを区切り、2つ以上の「-（ハイフン）」でヘッダー行とデータ行を区切ります。セル数が合わないと表にならないため、注意しましょう。また、テキスト上の見た目を整えたいときは、半角スペースを入れてそろえます。

```
| 料理名| 値段
| --| --
| ピザ| 980
| スパゲッティ| 880
```



1 項目名を「|」で区切ります。

2 次の行に「-」を２つずつ入力します。



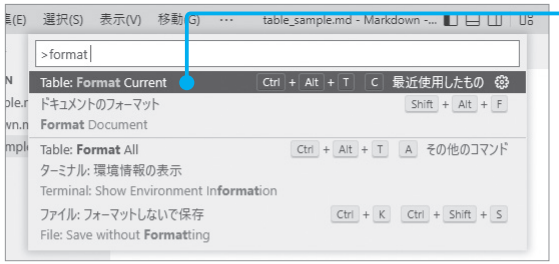
「-」の行がない場合、表になりません。

「-」の行の数が項目数と異なる場合も、表になりません。

テーブルの見た目を整える拡張機能

Markdownのエディター画面で見ても表に見えるように整形する、拡張機能の「Table Formatter」を紹介します。

1 拡張機能ビューで「Table Formatter」を検索してインストールします。



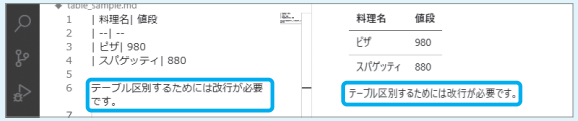
2 コマンドパレットを表示し、< Table:Format Current >コマンドを実行します。

3 < Table:Format Current >コマンドによって見た目が調整されました。

column

うまく表にならないときは

記法に間違いがないのにうまく表にならないときは、表の上下を1行空けると解決することがあります。



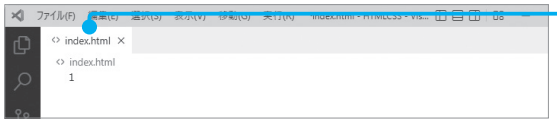
section 001

HTMLファイルのひな型を一瞬で作成する

Emmetと呼ばれるHTML／CSSの入力支援ツールを使うことで、省略記法を使って入力の手間を省くことができます。VS CodeにはEmmetが標準搭載されているので、どんどん活用しましょう。

EmmetでHTMLのひな型を作る

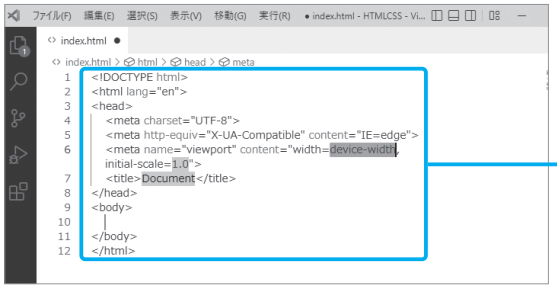
Emmet（エメット）とは、HTMLやCSSを省略記法で自動入力するためのテキストエディター用のプラグインです。VS CodeにはEmmetが標準搭載されているので、拡張機能のようにインストールする必要はありません。
Emmetを使うとHTMLのひな型や複数のタグを一瞬で作成でき、コーディングの手間を大幅に削減できます。
まずはEmmetの省略記法を使って、一瞬でHTMLファイルのひな型を作成しましょう。



1 拡張子「.html」を付けてHTMLファイルを作成します。

2 「!」と入力します。

3 右に「Emmet Abbreviation」と表示されている「!」を選択して、Tab キーを押します。

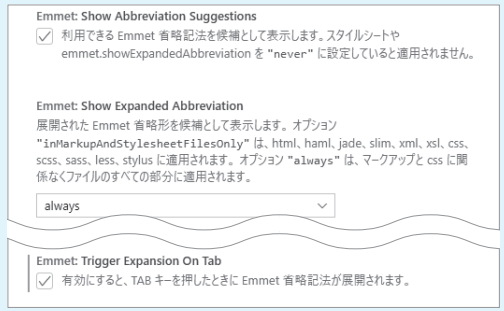


4 HTMLファイルのひな型が作成されました。

column

Emmetが使えない場合は設定を確認する

Emmetは標準で有効ですが、使えない場合は以下の設定を確認してください。



Emmetが使えないときに確認する設定項目

設定項目名	説明
Emmet:Show Abbreviation Suggestions	入力した文字列からEmmet省略記法の候補を表示します
Emmet:Show Expanded Abbreviation	Emmet省略記法の候補を表示するタイミングを制御します
Emmet:Trigger Expansion On Tab	Tab キーを押したときにEmmet省略記法を展開するか制御します

上の2つはEmmetに関係する候補リストを表示する設定です。「Emmet:Trigger Expansion On Tab」を無効にしていても、Emmetの候補リストが表示されるのでTab キーで省略記法を展開できます。
「Emmet:Show Abbreviation Suggestions」と「Emmet:Show Expanded Abbreviation」が無効になっていても、「Emmet:Trigger Expansion On Tab」が有効になっている場合は、候補リストは表示されないものの、Emmetの省略記法を正しく入力してTab キーを押せば展開できます。
つまり、3つの設定がすべて無効になっている場合、Emmetを展開できません。
また、Emmetは拡張子がhtmlやhtm、css以外のファイルでは使えません。上記を設定しても使えない場合は、ファイルの拡張子を確認しましょう。

section 003

ソース管理ビューで ファイルの変更点を確認する

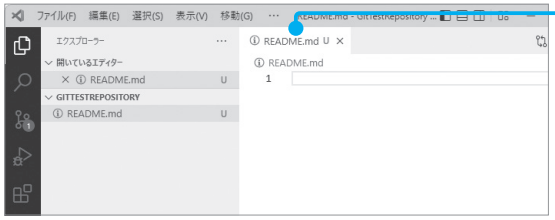
ローカルリポジトリ内にファイルを作成したり、ファイルを修正したりすると、ソース管理ビューなどに「変更が生じたこと」が表示されます。区切りがいいところでコミットしていきます。

ローカルリポジトリ内にファイルを作成する

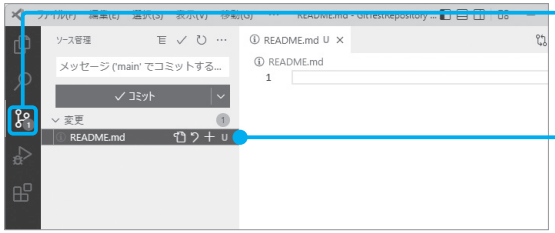
ローカルリポジトリへの操作といっても、普通のフォルダー内で行うこととまったく変わりません。ただし、何が変更されたかはローカルリポジトリ内の非表示領域に記録されており、VS Codeのソース管理ビューで確認できます。

1 エクスプローラービューの<新しいファイル>をクリックし、

2 ファイル名を入力して Enter キーを押します。



3 新しいファイルが作成されます。

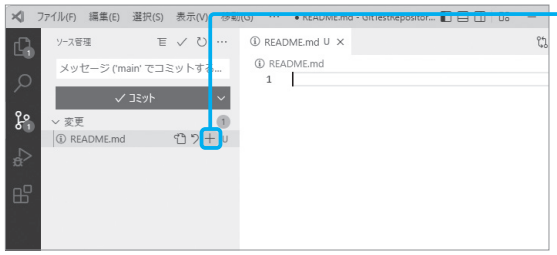


4 <ソース管理>をクリックすると、

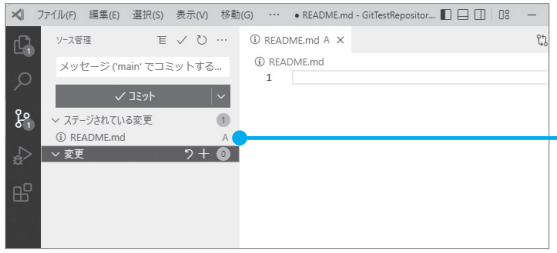
5 <変更>の下に新規作成したファイルの名前が表示されます。

変更をコミットする

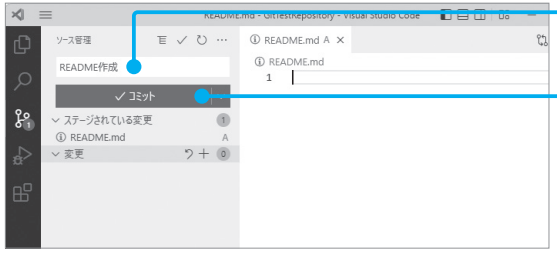
変更履歴を記録することをコミットといいます。コミットするには、まず変更をステージし、コミットメッセージを付けて登録します。



1 <変更をステージ>をクリックすると、

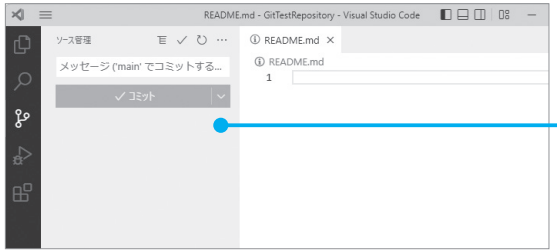


2 <ステージされている変更>の下に移ります。



3 コミットメッセージを入力して、

4 <コミット>をクリックします。



5 コミットされていない変更がない状態になりました。

section 001

主な設定ID一覧

本書には多くの設定ID（P.73参照）が登場しました。VS Codeには、紹介しきれないほど膨大な設定IDがあります。ここでは、紹介しきれなかったものを含めて、よく使われる設定IDをまとめました。

editor.autoClosingBrackets

エディタで左角かっこを追加したときに、自動的に右角かっこを挿入するか制御

設定値	説明
always	常に自動でかっこを閉じる
languageDefined	言語設定を利用し、いつ自動でかっこを閉じるか制御
beforeWhitespace	カーソルが空白文字の左にあるときのみ、かっこを自動で閉じる
never	自動でかっこを閉じない

editor.autoSave

自動保存の制御

設定値	説明
off	自動保存しない
afterDelay	files.autoSaveDelay値を経過後に自動保存
onForcusChange	エディタがフォーカスを失うと自動保存
onWindowChange	ウィンドウが変わると自動保存

editor.bracketPairColorization

角かっこのペアを彩色するか制御

editor.cursorStyle

カーソルのスタイル設定

設定値	説明
line	縦線
block	ブロック
underline	下線
line-htin	細い縦線
block-outline	ブロックの枠線
underline-thin	細い下線

editor.fontFamily

フォントの制御

editor.fontSize

フォントサイズをピクセル単位で制御

editor.formatOnPaste

（フォーマッタが有効の場合）貼り付けたときに内容のフォーマットを制御

editor.formatOnSave

（フォーマッタが有効の場合）ファイルを保存するときに内容のフォーマットを制御

editor.incertSpaces

Tabキーを押したときにスペースを挿入する制御

editor.lineHeight

行の高さを制御

設定値	説明
0	フォントサイズから行の高さを自動計算
1~7	フォントサイズの乗算として使用
8以上	その値で高さを制御

editor.minimap.enable

ミニマップの表示を制御