

[Sign up](#)[Code](#)[Issues](#) 105[Pull requests](#) 2[Projects](#) 0[Wiki](#)[Security](#)[Pulse](#)

All your code in one place

[Dismiss](#)

GitHub makes it easy to scale back on context switching. Read rendered documentation, see the history of any file, and collaborate with contributors on projects across GitHub.

[Sign up for free](#)[See pricing for teams and enterprises](#)

Branch: master ▾

[Find file](#)[Copy path](#)[vscode-docs](#) / [docs](#) / [getstarted](#) / **[tips-and-tricks.md](#)**

gregvanl Merge pull request #2753 from Mohit-Nain/patch-12 bb72f75 Jun 13, 2019

3 contributors



897 lines (547 sloc) | 29.4 KB

[Raw](#)[Blame](#)[History](#)

Order	Area	TOCTitle	ContentId	PageTitle	DateApproved	MetaDescription
3	getstarted	Tips and Tricks	9bbbe55d-cf81-428f-8a9f-4f60280cb874	Visual Studio Code Tips and Tricks	6/5/2019	Visual Studio Code Tips and Tricks for power users.

Visual Studio Code Tips and Tricks

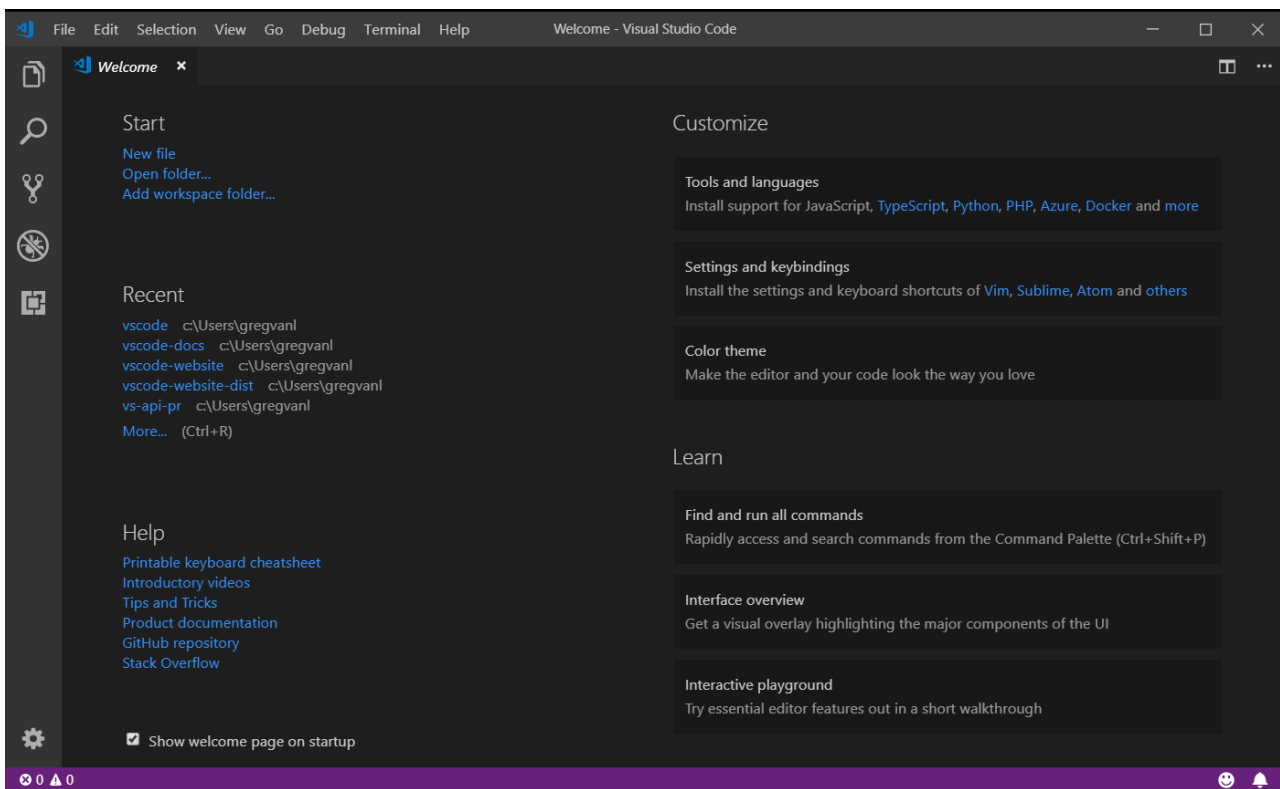
"Tips and Tricks" lets you jump right in and learn how to be productive with Visual Studio Code. You'll become familiar with its powerful editing, code intelligence, and source code control features and learn useful keyboard shortcuts. This topic goes pretty fast and provides a broad overview, so be sure to look at the other in-depth topics in [Getting Started](#) and the [User Guide](#) to learn more.

If you don't have Visual Studio Code installed, go to the [Download](#) page. You can find platform specific setup instructions at [Running VS Code on Linux](#), [macOS](#), and [Windows](#).

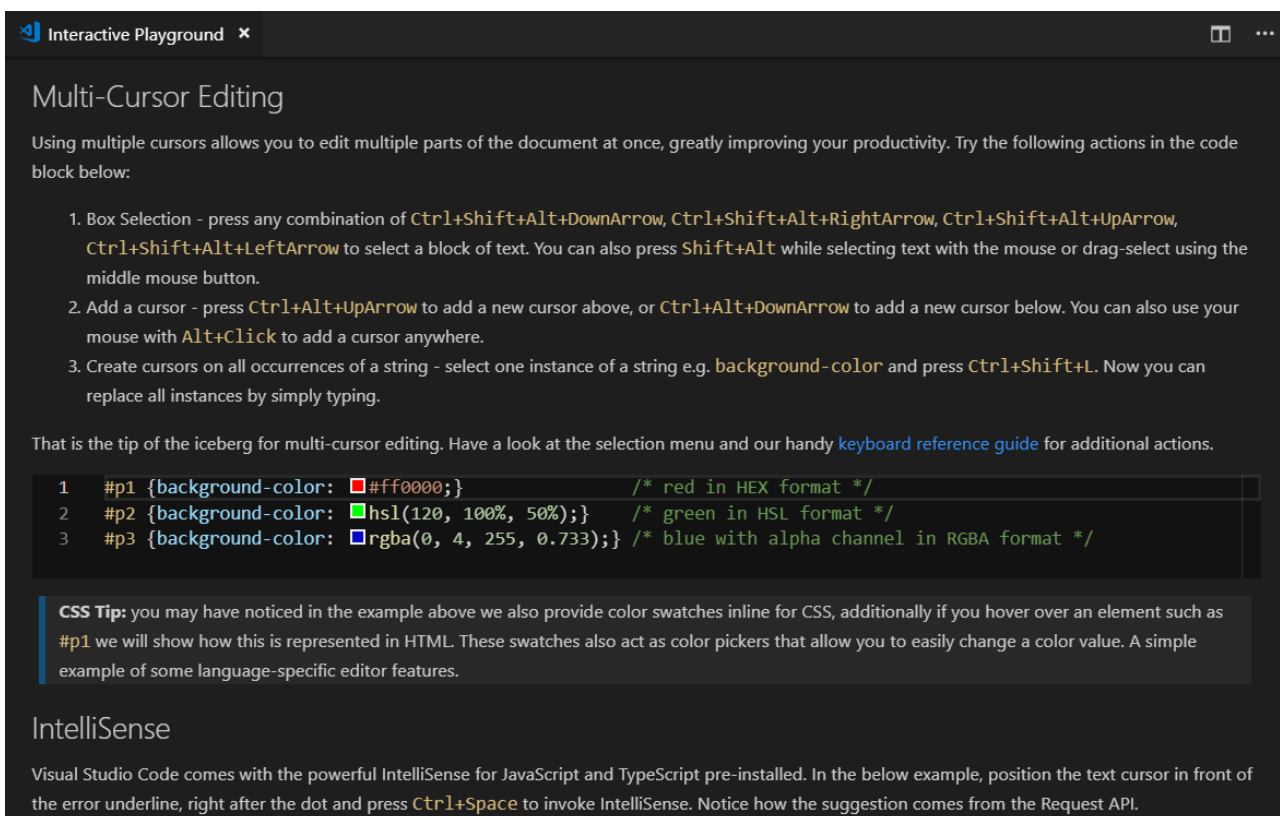
Basics

Getting started

Open the **Welcome** page to get started with the basics of VS Code. **Help > Welcome**.



In the bottom right of the **Welcome** page, there is a link to the **Interactive playground** where you can interactively try out VS Code's features. **Help > Interactive Playground**.



Command Palette

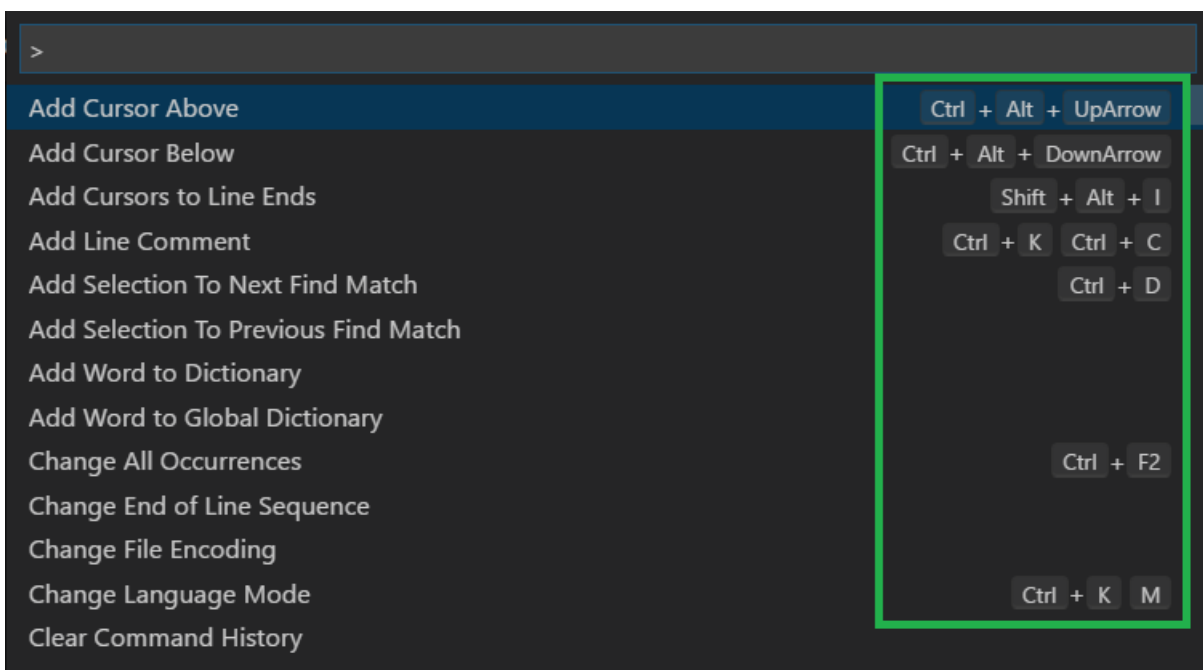
Access all available commands based on your current context.

Keyboard Shortcut: `kb(workbench.action.showCommands)`

```
App.js - sports-trivia
JS App.js x keybindings.json
1 import React from 'react';
2 import {deepOrange500} from 'material-ui/styles/colors';
3 import AppBar from 'material-ui/AppBar';
4 import getMuiTheme from 'material-ui/styles/getMuiTheme';
5 import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
6
7 import QuestionBar from './QuestionBar';
8
9 const muiTheme = getMuiTheme({
10   palette: {
11     accent1Color: deepOrange500,
12   },
13 });
14
15 export default class App extends React.Component {
16   constructor(props, context) {
17     super(props, context);
18
19     this.state = {};
20   }
21 }
```

Default keyboard shortcuts

All of the commands are in the **Command Palette** with the associated key binding (if it exists). If you forget a keyboard shortcut, use the **Command Palette** to help you out.



Keyboard reference sheets

Download the keyboard shortcut reference sheet for your platform ([macOS](#), [Windows](#), [Linux](#)).

 Visual Studio Code

Keyboard shortcuts for macOS

General

⌘P, F1	Show Command Palette
⌘P	Quick Open, Go to File...
⌘N	New window/instance
⌘W	Close window/instance
⌘,	User Settings
⌘K ⌘S	Keyboard Shortcuts







Basic editing

⌘X	Cut line (empty selection)
⌘C	Copy line (empty selection)
⇧⌘/ ⇧⇧	Move line down/up
⇧⌘- ⇧⇧	Copy line down/up
⌘K	Delete line
⇧Enter / ⇧⇧Enter	Insert line below/above
⌘⇧	Jump to matching bracket
⇧) / ⇧[Indent/outdent line
Home / End	Go to beginning/end of line
⇧1 / ⇧4	Go to beginning/end of file
⇧PgUp / ⇧PgDn	Scroll line up/down
⇧PgUp / ⇧PgDn	Scroll page up/down
⇧⌘[/ ⇧⌘]	Fold/unfold region
⇧⌘K / ⇧⌘⇧	Fold/unfold all subregions
⇧⌘K ⇧0 / ⇧⌘K ⇧⇧	Fold/unfold all regions
⇧⌘K ⇧C	Add line comment
⇧⌘K ⇧U	Remove line comment
⇧/	Toggle line comment
⇧⌘A	Toggle block comment
⇧Z	Toggle word wrap

Multi-cursor and selection

⌘ + click	Insert cursor
⌘ + ⇧ + ⌘	Insert cursor above
⌘ + ⇩ + ⌘	Insert cursor below
⌘ + U	Undo last cursor operation
⌘ + ⌥	Insert cursor at end of each line selected
⌘ + ⌵	Select current line
⌘ + L	Select all occurrences of current selection
⌘ + F2	Select all occurrences of current word
⌘ + ⇧ + ⌵	Expand / shrink selection
⌘ + drag mouse	Column (box) selection
⌘ + ⇧ + ⌵	Column (box) selection up/down
⌘ + ⇧ + ⌵	Column (box) selection left/right
⌘ + ⇧ + PgUp	Column (box) selection page up
⌘ + ⇧ + PgDn	Column (box) selection page down

Search and replace

	F	Find
	⇧F	Replace
	G / ⇧G	Find next/previous
	⇧Enter	Select all occurrences of Find match
	D	Add selection to next Find match
	⇧K D	Move last selection to next Find match

Rich languages editing

⌘Space	Trigger suggestion
⇧⌘Space	Trigger parameter hints
⌘⇧F	Format document
⌘⇧K ⌘⇧F	Format selection
F12	Go to Definition
⇧F12	Peek Definition
⌘⇧K F12	Open Definition to the side
⇧⇧	Quick Fix
⇧F12	Show References
F2	Rename Symbol
⌘⇧K ⌘⇧X	Trim trailing whitespace
⌘⇧K M	Change file language

Navigation

⌘T	Show all Symbols
⌘G	Go to Line...
⌘P	Go to File...
⌘O	Go to Symbol...
⌘M	Show Problems panel
F8 / ⌘F8	Go to next/previous error or warning
⌘Tab	Navigate editor group history
⌘_ / ⌘⇧_	Go back/forward
⌘M	Toggle Tab moves focus

Editor management

⌘W	Close editor
⌘F	Close folder
⌘\	Split editor
⌘1 / ⌘2 / ⌘3	Focus into 1 st , 2 nd , 3 rd editor group
⌘K ⌘← / ⌘K ⌘→	Focus into previous/next editor group
⌘K ⌘↶ / ⌘K ⌘↷	Move editor left/right
⌘K ⌘↵ / ⌘K ↵	Move active editor group

File management





⌘N	New File
⌘O	Open File...
⌘S	Save
⇧⌘S	Save As...
⌘⇧S	Save All
⌘W	Close
⌘K ⇧W	Close All

⇧⌘T	Reopen closed editor
⇧⌘K Enter	Keep preview mode editor open
^Tab / ⇧⌘Tab	Open next / previous
⇧⌘K P	Copy path of active file
⇧⌘K R	Reveal active file in Explorer
⇧⌘K O	Show active file in new window/instance

Display

⌘F	Toggle full screen
⌘1	Toggle editor layout (horizontal/vertical)
⌘= / ⌘-	Zoom in/out
⌘E	Toggle Sidebar visibility
⌘F	Show Explorer / Toggle focus
⌘F	Show Search
⌘G	Show Source Control
⌘D	Show Debug
⌘X	Show Extensions
⌘H	Replace in files
⌘J	Toggle Search details
⌘U	Show Output panel
⌘V	Open Markdown preview
⌘K V	Open Markdown preview to the side
⌘K Z	Zen Mode (Esc Esc to exit)

Debug

F9	Toggle breakpoint
F5	Start/Continue
F11 / 	Step into/ out
F10	Step over
	Stop
 K 	Show hover

Integrated terminal

Alt	Show integrated terminal
Alt+N	Create new terminal
Alt+C	Copy selection
Alt+V	Paste into active terminal
Alt+Up / Down	Scroll up/down
Alt+PgUp / PgDn	Scroll page up/down
Alt+Home / End	Scroll to top/bottom

Other operating systems' keyboard shortcuts and additional unassigned shortcuts available at aka.ms/vscodekeybindings

Quick Open

Quickly open files.

Keyboard Shortcut: `kb(workbench.action.quickOpen)`



Tip: Type `kbstyle(?)` to view help suggestions.

Navigate between recently opened files

Repeat the **Quick Open** keyboard shortcut to cycle quickly between recently opened files.

Open multiple files from Quick Open

You can open multiple files from **Quick Open** by pressing the Right arrow key. This will open the currently selected file in the background and you can continue selecting files from **Quick Open**.

Command line

VS Code has a powerful command line interface (CLI) which allows you to customize how the editor is launched to support various scenarios.

Make sure the VS Code binary is on your path so you can simply type 'code' to launch VS Code. See the platform specific setup topics if VS Code is added to your environment path during installation ([Running VS Code on Linux](#), [macOS](#), [Windows](#)).

```
# open code with current directory
code .

# open the current directory in the most recently used code window
code -r .

# create a new window
code -n

# change the language
code --locale=es

# open diff editor
code --diff <file1> <file2>

# open file at specific line and column <file:line[:character]>
code --goto package.json:10:5

# see help options
code --help

# disable all extensions
code --disable-extensions .
```

.vscode folder

Workspace specific files are in a `.vscode` folder at the root. For example, `tasks.json` for the Task Runner and `launch.json` for the debugger.

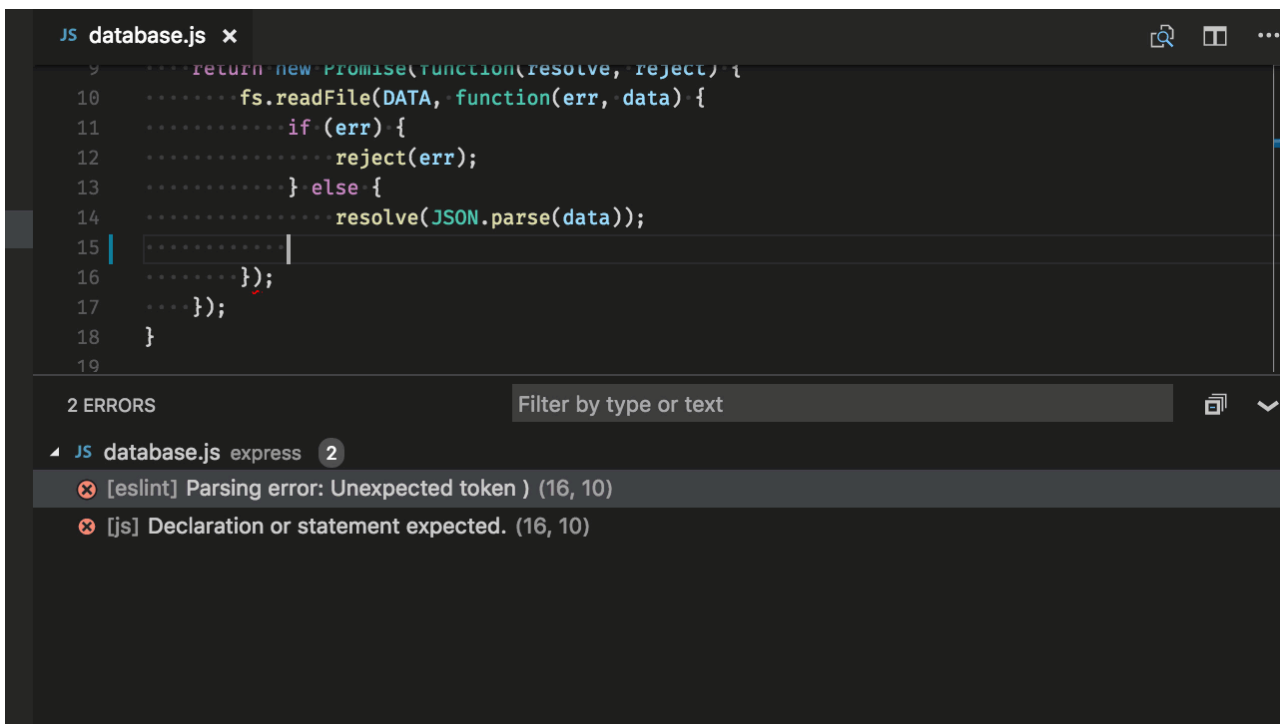
Status Bar

Errors and warnings

Keyboard Shortcut: `kb(workbench.actions.view.problems)`

Quickly jump to errors and warnings in the project.

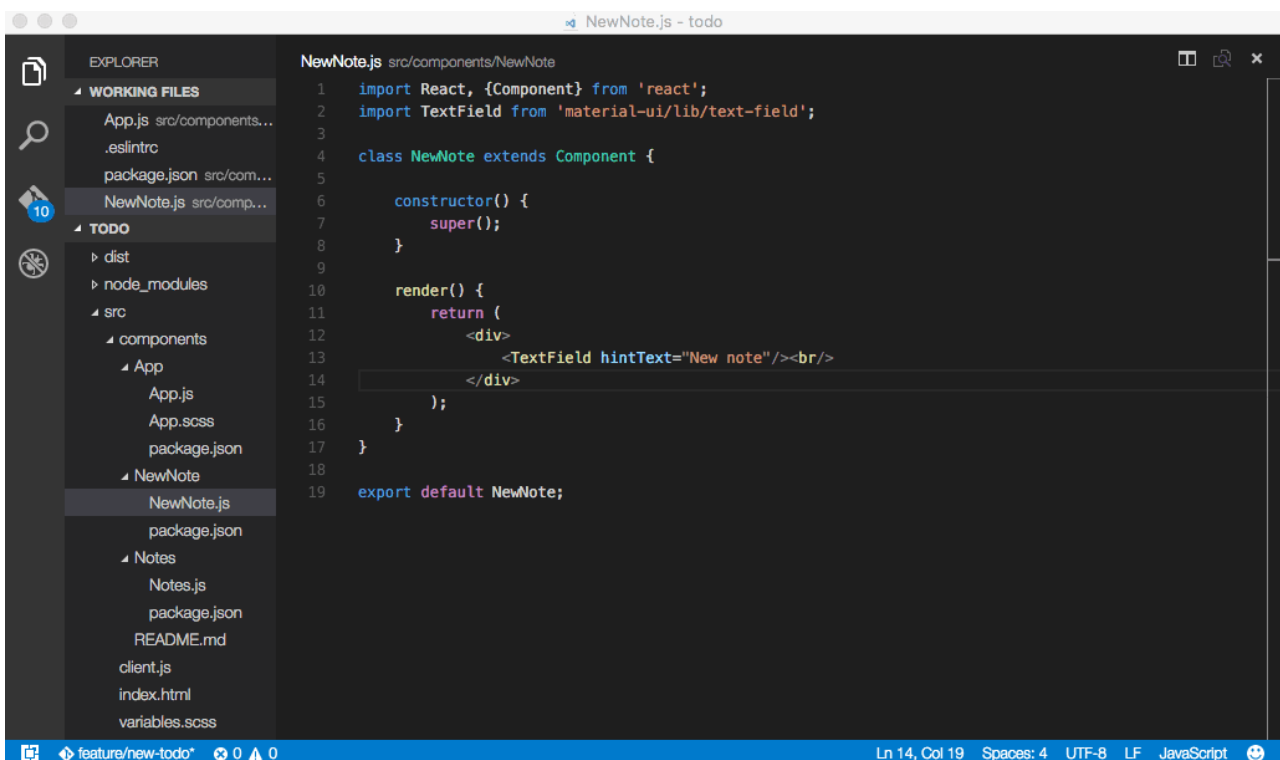
Cycle through errors with `kb(editor.action.marker.nextInFiles)` or `kb(editor.action.marker.prevInFiles)`



You can filter problems either by type ('errors', 'warnings') or text matching.

Change language mode

Keyboard Shortcut: `kb(workbench.action.editor.changeLanguageMode)`



If you want to persist the new language mode for that file type, you can use the **Configure File Association for** command to associate the current file extension with an installed language.

Customization

There are many things you can do to customize VS Code.

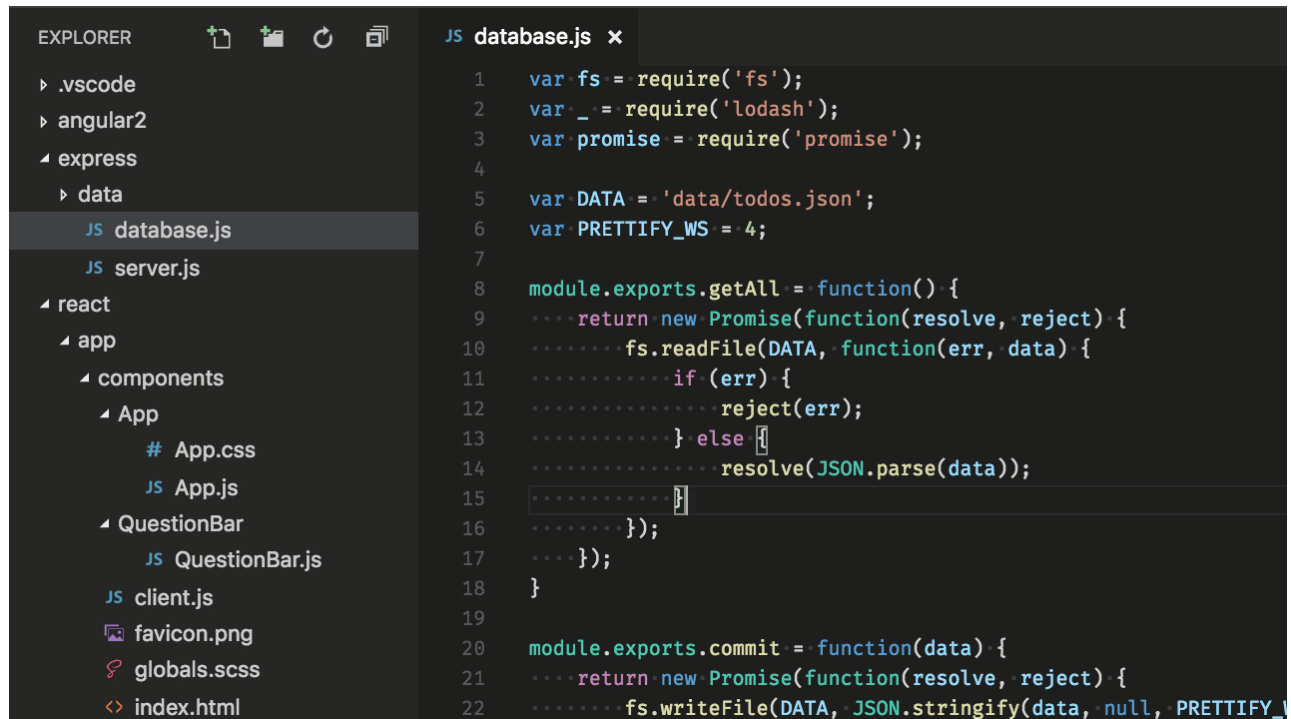
- Change your theme
- Change your keyboard shortcuts

- Tune your settings
- Add JSON validation
- Create snippets
- Install extensions

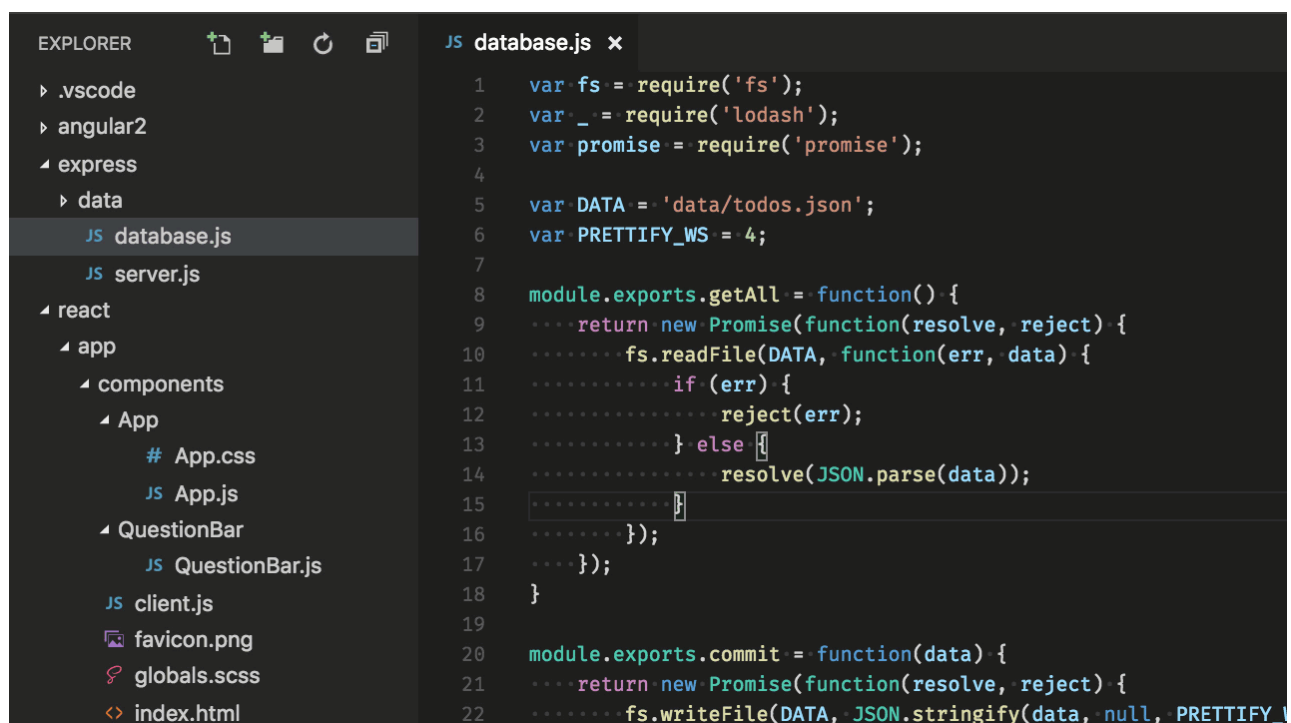
Change your theme

Keyboard Shortcut: `kb(workbench.action.selectTheme)`

You can install more themes from the VS Code extension [Marketplace](#).



Additionally, you can install and change your File Icon themes.



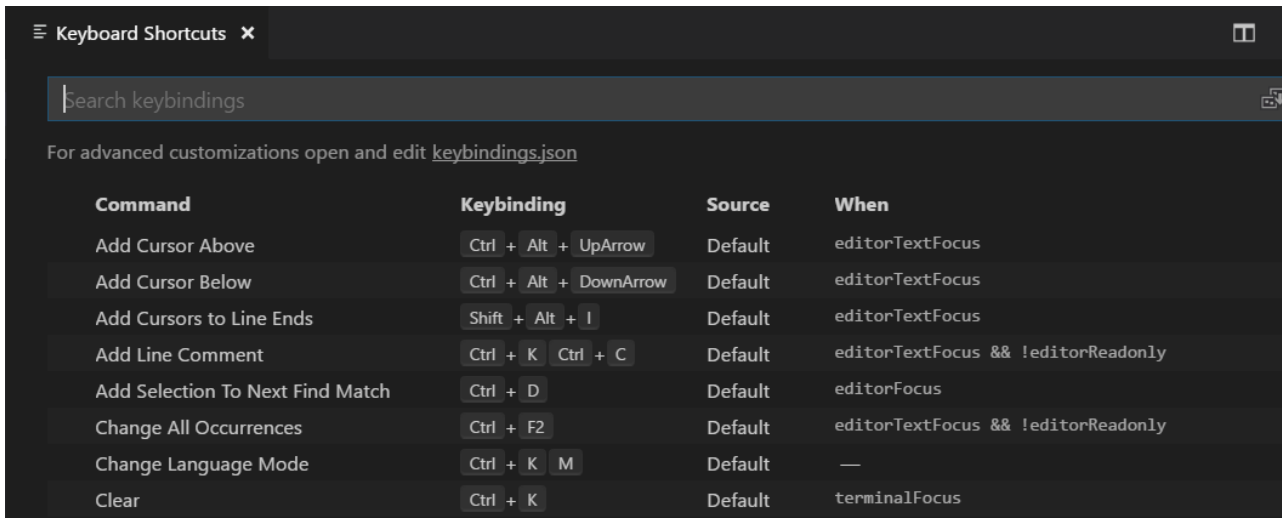
Keymaps

Are you used to keyboard shortcuts from another editor? You can install a Keymap extension that brings the keyboard shortcuts from your favorite editor to VS Code. Go to **Preferences > Keymap Extensions** to see the current list on the [Marketplace](#). Some of the more popular ones:

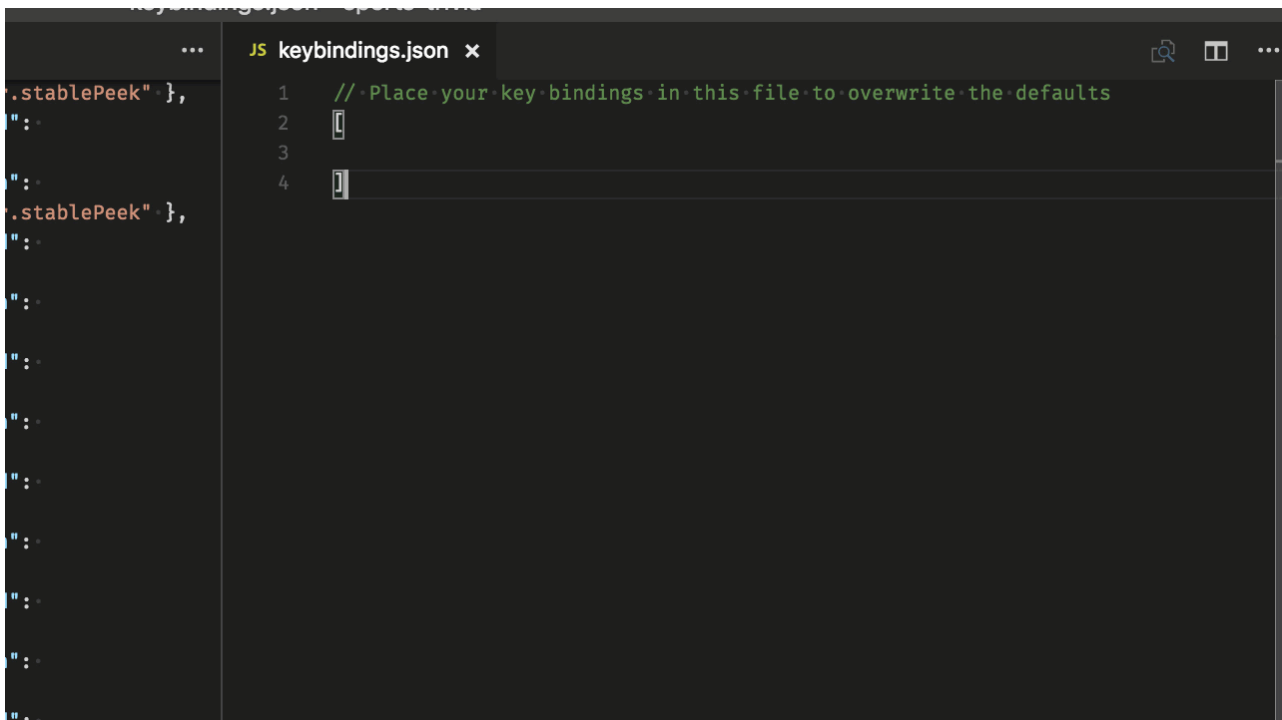
- [Vim](#)
- [Sublime Text Keymap](#)
- [Emacs Keymap](#)
- [Atom Keymap](#)
- [Eclipse Keymap](#)

Customize your keyboard shortcuts

Keyboard Shortcut: `kb(workbench.action.openGlobalKeybindings)`



You can search for shortcuts and add your own keybindings to the `keybindings.json` file.



See more in [Key Bindings for Visual Studio Code](#).

Tune your settings

By default VS Code shows the Settings editor, you can find settings listed below in a search bar, but you can still

edit the underlying `settings.json` file by using the **Open Settings (JSON)** command or by changing your default settings editor with the `workbench.settings.editor` setting.

Open User Settings `settings.json`

Keyboard Shortcut: `kb(workbench.action.openSettings)`

Format on paste

```
"editor.formatOnPaste": true
```

Change the font size of various UI elements

```
// Main editor
"editor.fontSize": 18,
// Terminal panel
"terminal.integrated.fontSize": 14,
// Output panel
"[Log]": {
  "editor.fontSize": 15
}
```

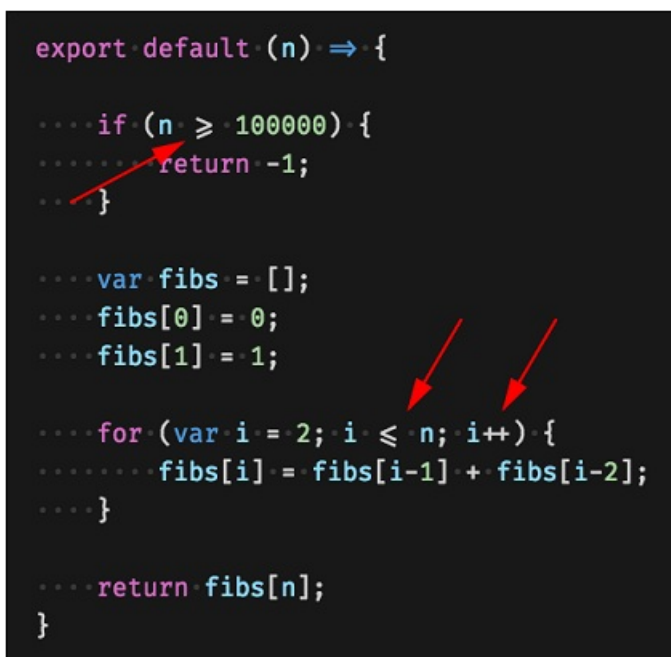
Change the zoom level

```
"window.zoomLevel": 5
```

Font ligatures

```
"editor.fontFamily": "Fira Code",
"editor.fontLigatures": true
```

Tip: You will need to have a font installed that supports font ligatures. [FiraCode](#) is a popular font on the VS Code team.



```
export default (n) => {
  ...if (n ≥ 100000) {
  ...return -1;
  ...}

  ...var fibs = [];
  ...fibs[0] = 0;
  ...fibs[1] = 1;

  ...for (var i = 2; i ≤ n; i++) {
  ...  ...fibs[i] = fibs[i-1] + fibs[i-2];
  ...}

  ...return fibs[n];
}
```

The image shows a code editor with a dark background. The code is a JavaScript function that calculates the nth Fibonacci number. It uses various characters that are rendered as ligatures in the Fira Code font, such as ≥, ≤, and →. Three red arrows point to specific characters: the first arrow points to the 'n' in the first 'if' statement, the second arrow points to the '≤' symbol in the 'for' loop, and the third arrow points to the '-' sign in the 'fibs[i-1]' array access.

Auto Save

```
"files.autoSave": "afterDelay"
```

You can also toggle Auto Save from the top-level menu with the **File > Auto Save**.

Format on save

```
"editor.formatOnSave": true
```

Change the size of Tab characters

```
"editor.tabSize": 4
```

Spaces or Tabs

```
"editor.insertSpaces": true
```

Render whitespace

```
"editor.renderWhitespace": "all"
```

Ignore files / folders

Removes these files / folders from your editor window.

```
"files.exclude": {  
  "somefolder/": true,  
  "somefile": true  
}
```

Remove these files / folders from search results.

```
"search.exclude": {  
  "someFolder/": true,  
  "somefile": true  
}
```

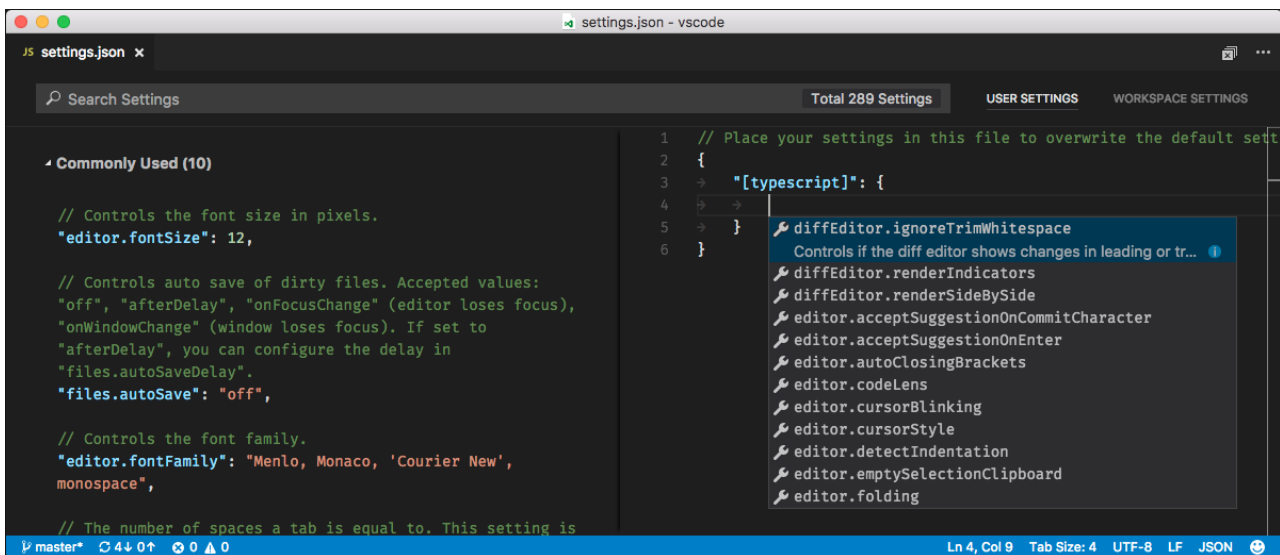
And many, many [other customizations](#).

Language specific settings

For the settings, which you only want for specific languages, you can scope the settings by the language identifier. You can find a list of commonly used language ids in the [Language Identifiers](#) reference.

```
"[languageid]": {  
  
}
```

Tip: You can also create language specific settings with the **Configure Language Specific Settings** command.



Add JSON validation

Enabled by default for many file types. Create your own schema and validation in `settings.json`

```
"json.schemas": [  
  {  
    "fileMatch": [  
      "/bower.json"  
    ],  
    "url": "http://json.schemastore.org/bower"  
  }  
]
```

or for a schema defined in your workspace

```
"json.schemas": [  
  {  
    "fileMatch": [  
      "/foo.json"  
    ],  
    "url": "./myschema.json"  
  }  
]
```

or a custom schema

```
"json.schemas": [  
  {  
    "fileMatch": [  
      "/.myconfig"  
    ],  
    "schema": {  
      "type": "object",  
      "properties": {  
        "name": {  
          "type": "string",  
          "description": "The name of the entry"  
        }  
      }  
    }  
  }  
],
```

See more in the [JSON](#) documentation.

Extensions

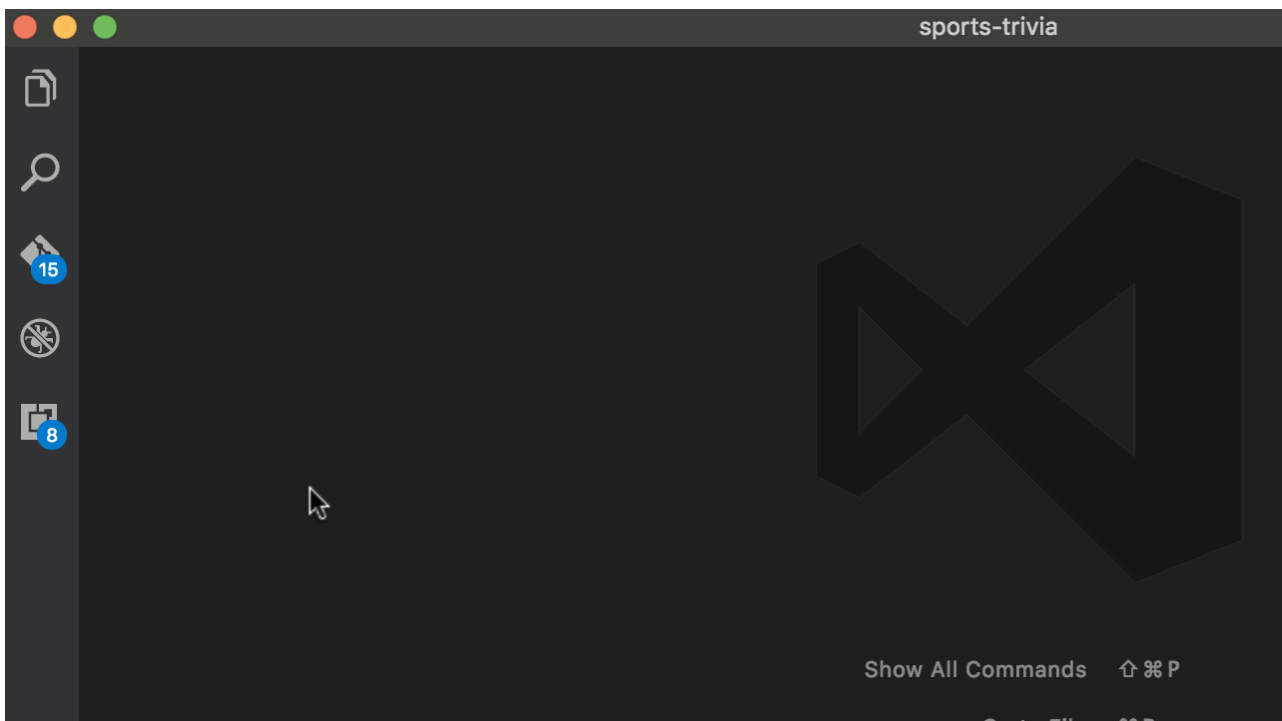
Keyboard Shortcut: `kb(workbench.view.extensions)`

Find extensions

1. In the VS Code [Marketplace](#).
2. Search inside VS Code in the **Extensions** view.
3. View extension recommendations
4. Community curated extension lists, such as [awesome-vscode](#).

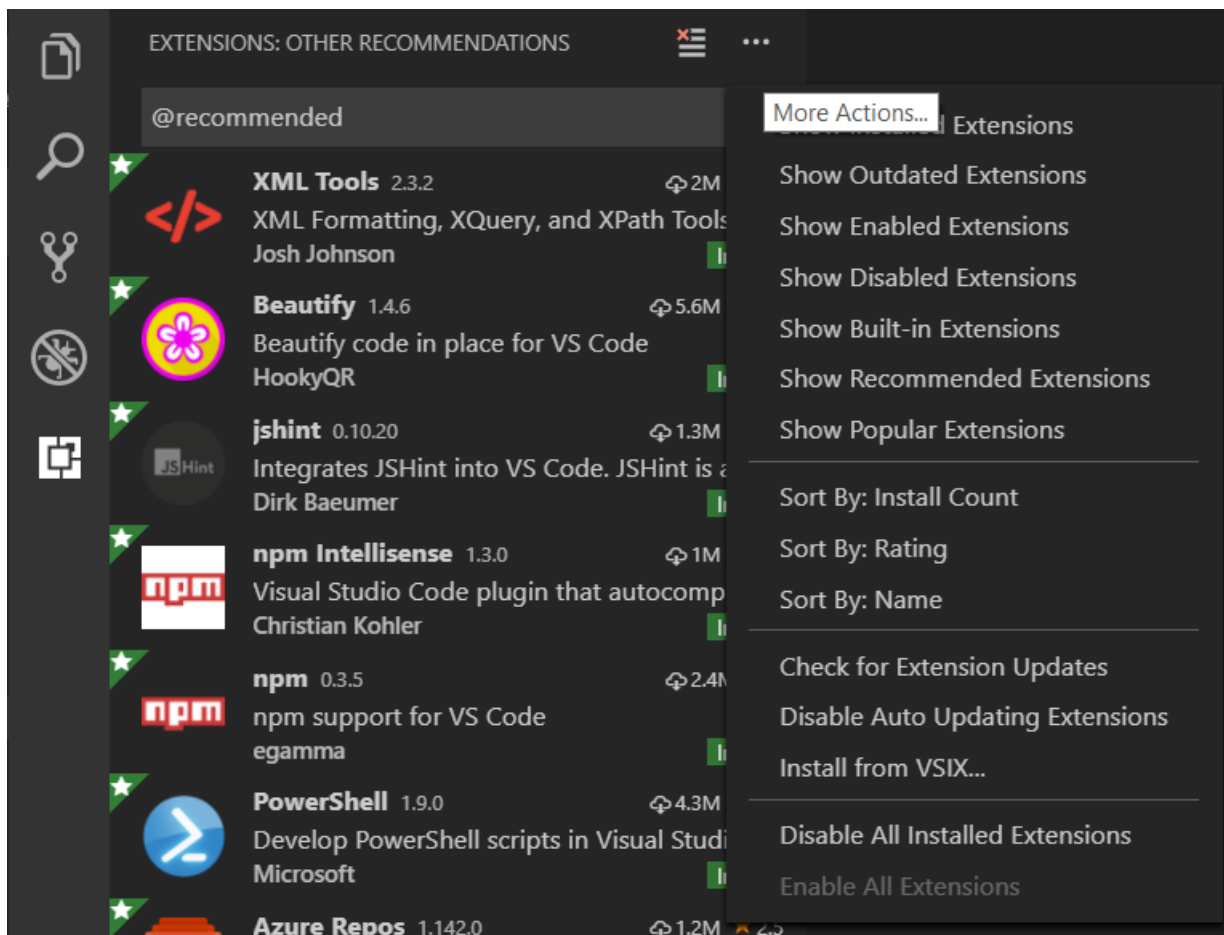
Install extensions

In the **Extensions** view, you can search via the search bar or click the **More Actions** (...) button to filter and sort by install count.



Extension recommendations

In the **Extensions** view, click **Show Recommended Extensions** in the **More Actions** (...) button menu.



Creating my own extension

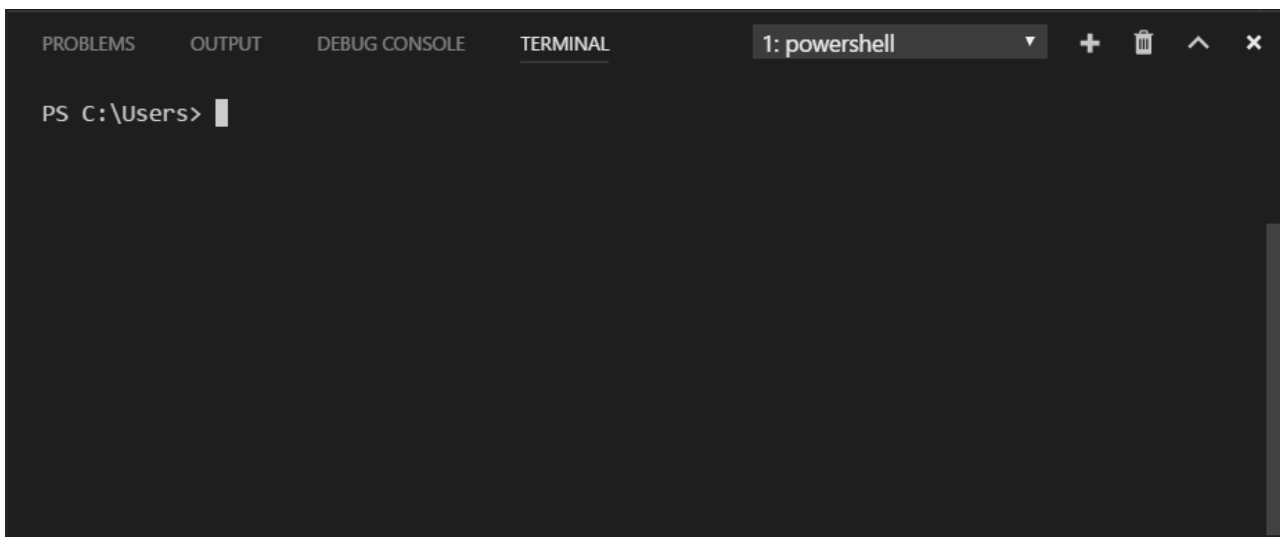
Are you interested in creating your own extension? You can learn how to do this in the [Extension API documentation](#), specifically check out the [documentation on contribution points](#).

- configuration
- commands
- keybindings
- languages
- debuggers
- grammars
- themes
- snippets
- jsonValidation

Files and folders

Integrated Terminal

Keyboard Shortcut: `kb(workbench.action.terminal.toggleTerminal)`



Further reading:

- [Integrated Terminal](#) documentation
- [Mastering VS Code's Terminal](#) article

Auto Save

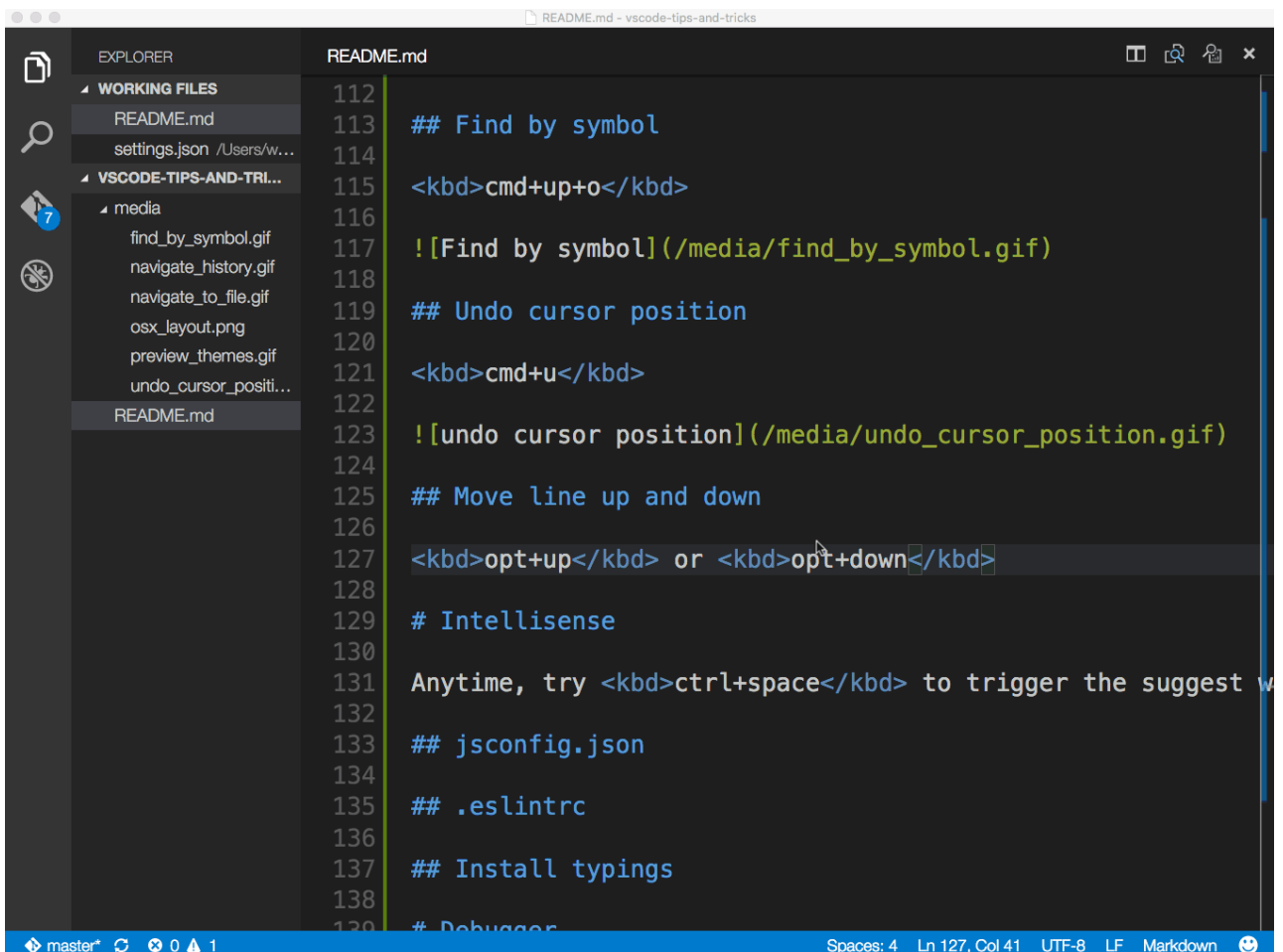
Open User Settings `settings.json` with `kb(workbench.action.openSettings)`

```
"files.autoSave": "afterDelay"
```

You can also toggle Auto Save from the top-level menu with the **File > Auto Save**.

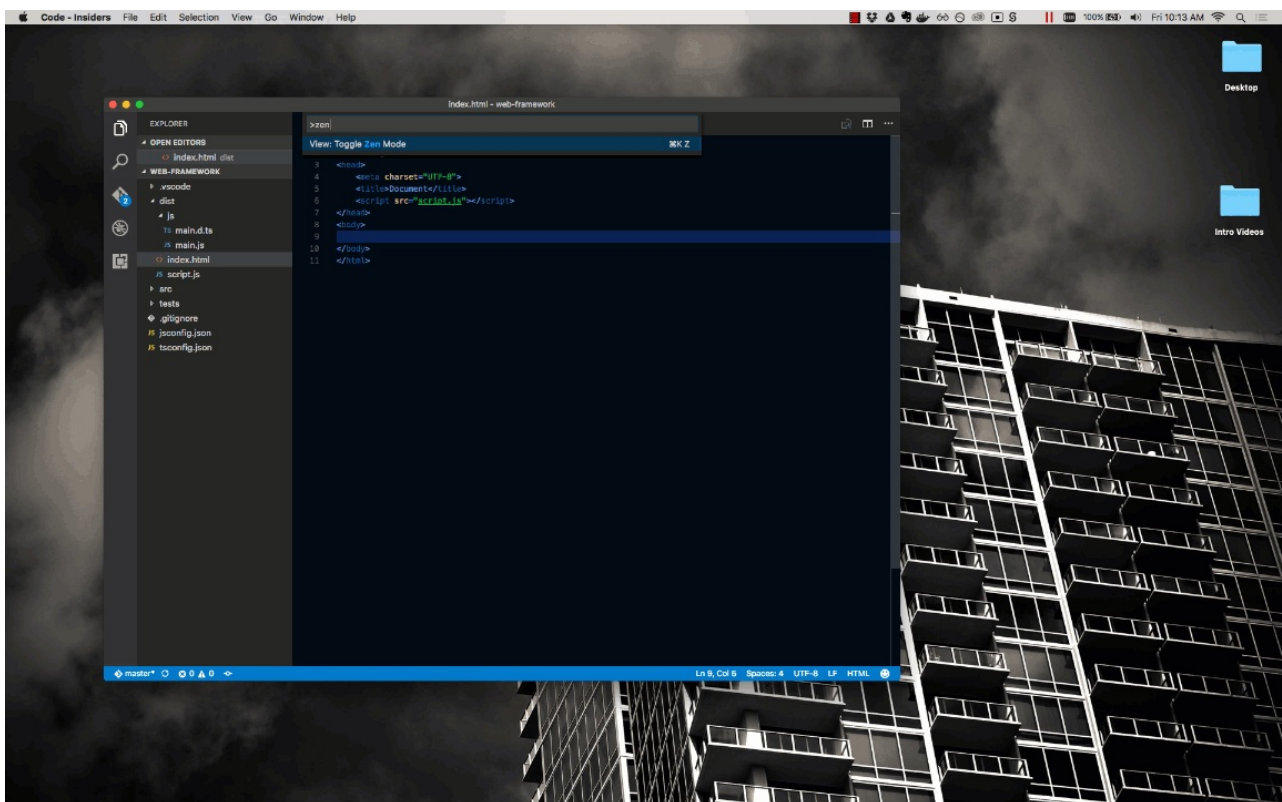
Toggle Sidebar

Keyboard Shortcut: `kb(workbench.action.toggleSidebarVisibility)`



Zen mode

Keyboard Shortcut: `kb(workbench.action.toggleZenMode)`



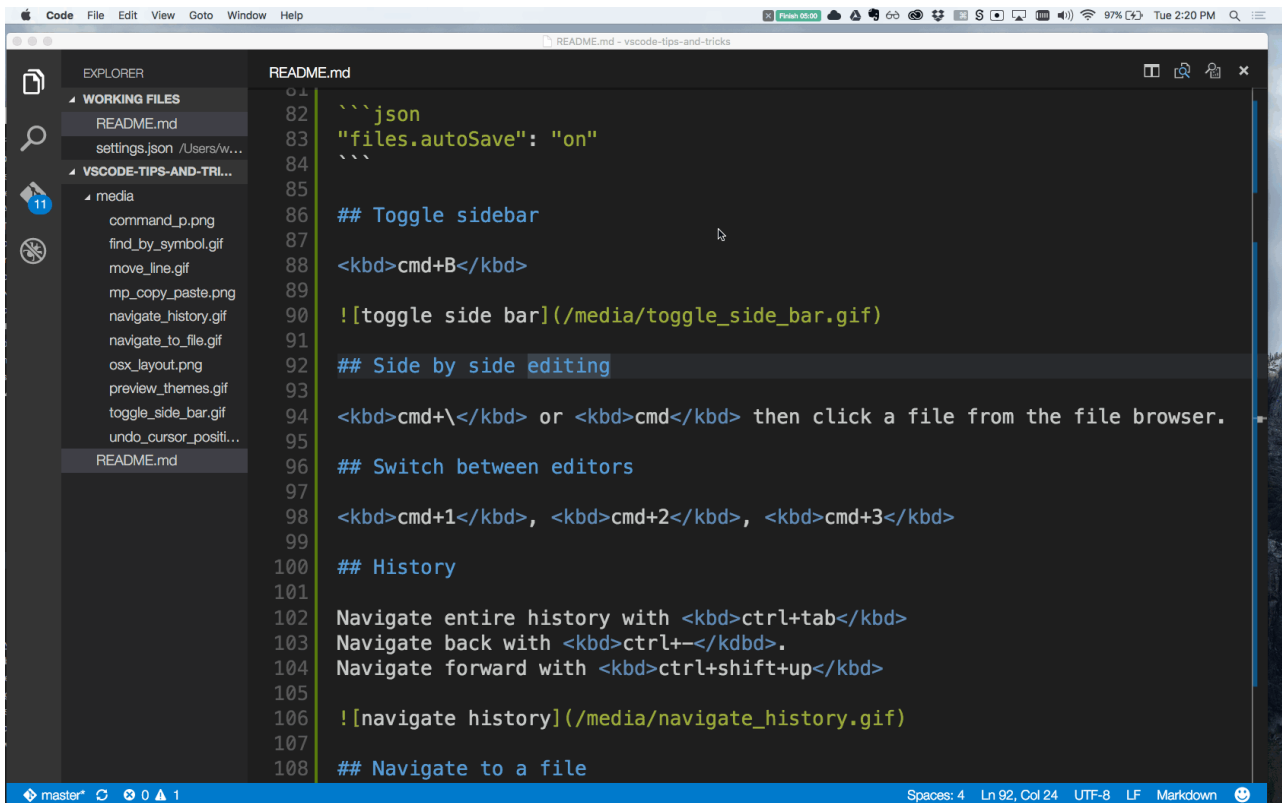
Enter distraction free Zen mode.

Press `kbstyle(Esc)` twice to exit Zen Mode.

Side by side editing

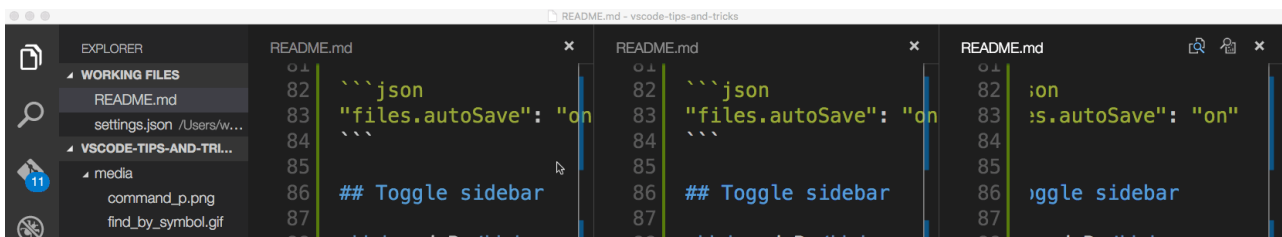
Keyboard Shortcut: `kb(workbench.action.splitEditor)`

You can also drag and drop editors to create new editor groups and move editors between groups.



Switch between editors

Keyboard Shortcut: `kb(workbench.action.focusFirstEditorGroup)` ,
`kb(workbench.action.focusSecondEditorGroup)` , `kb(workbench.action.focusThirdEditorGroup)`



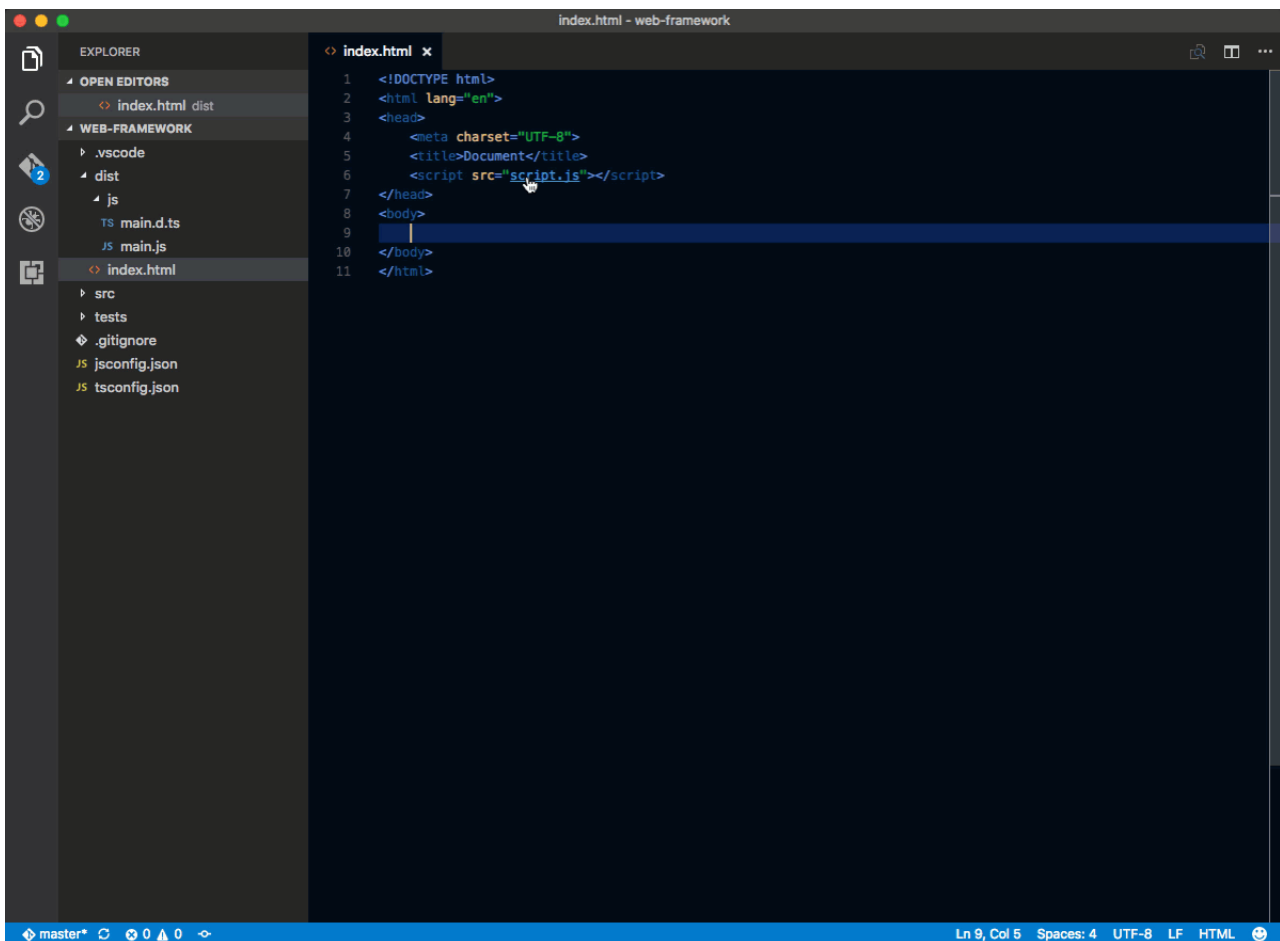
Move to Explorer window

Keyboard Shortcut: `kb(workbench.view.explorer)`

Create or open a file

Keyboard Shortcut: `kbstyle(Ctrl+click)` (`kbstyle(Cmd+click)` on macOS)

You can quickly open a file or image or create a new file by moving the cursor to the file link and using `kbstyle(Ctrl+click)` .



Close the currently opened folder

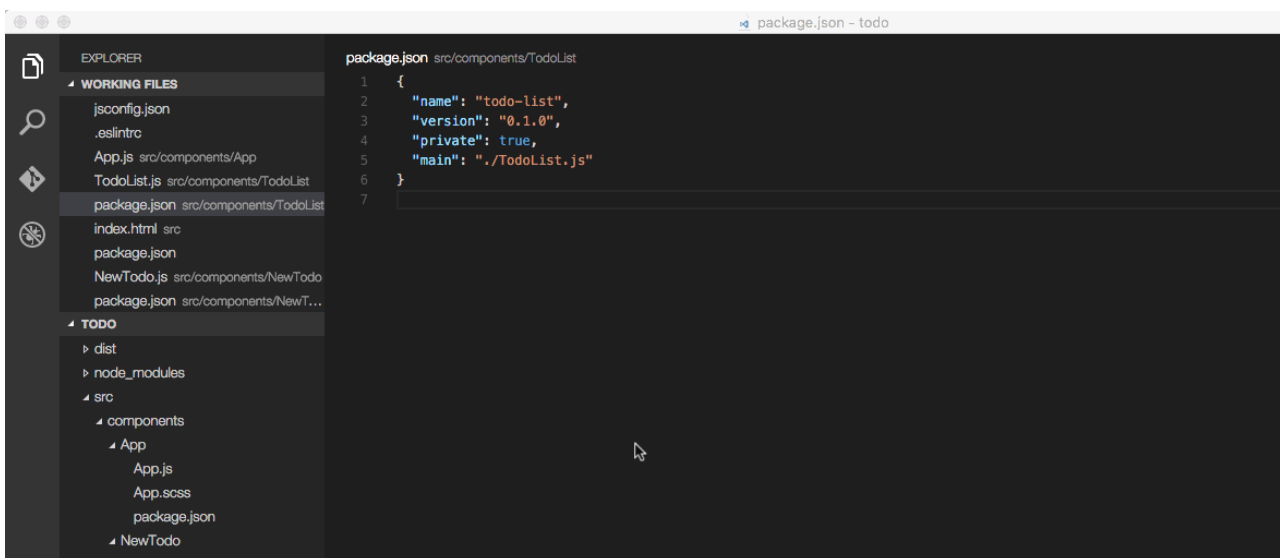
Keyboard Shortcut: `kb(workbench.action.closeActiveEditor)`

Navigation history

Navigate entire history: `kb(workbench.action.openNextRecentlyUsedEditorInGroup)`

Navigate back: `kb(workbench.action.navigateBack)`

Navigate forward: `kb(workbench.action.navigateForward)`



File associations

Create language associations for files that aren't detected correctly. For example, many configuration files with

custom file extensions are actually JSON.

```
"files.associations": {
  ".database": "json"
}
```

Preventing dirty writes

VS Code will show you an error message when you try to save a file that cannot be saved because it has changed on disk. VS Code blocks saving the file to prevent overwriting changes that have been made outside of the editor.

In order to resolve the save conflict, click the **Compare** action in the error message to open a diff editor that will show you the contents of the file on disk (to the left) compared to the contents in VS Code (on the right):



Use the actions in the editor toolbar to resolve the save conflict. You can either **Accept** your changes and thereby overwriting any changes on disk, or **Revert** to the version on disk. Reverting means that your changes will be lost.

Note: The file will remain dirty and cannot be saved until you pick one of the two actions to resolve the conflict.

Editing hacks

Here is a selection of common features for editing code. If the keyboard shortcuts aren't comfortable for you, consider installing a [keymap extension](#) for your old editor.

Tip: You can see recommended keymap extensions in the **Extensions** view with

```
kb(workbench.extensions.action.showRecommendedKeymapExtensions) which filters the search to
@recommended:keymaps .
```

Multi cursor selection

To add cursors at arbitrary positions, select a position with your mouse and use `kbstyle(Alt+Click)` (`kbstyle(Option+click)` on macOS).

To set cursors above or below the current position use:

Keyboard Shortcut: `kb(editor.action.insertCursorAbove)` or `kb(editor.action.insertCursorBelow)`

```

31 .global-message-list.transition {
32 → -webkit-transition: top 200ms linear;
33 → -ms-transition:      top 200ms linear;
34 → -moz-transition:     top 200ms linear;
35 → -khtml-transition:   top 200ms linear;
36 → -o-transition:       top 200ms linear;
37 → transition:          top 200ms linear;
38 }

```

You can add additional cursors to all occurrences of the current selection with `kb(editor.action.selectHighlights)`.

```

render() {
  return (
    <div className={s.app}>
      <AppBar
        title="Todo"
        iconClassNameRight="muidocs-icon-navigation-expand-more"
      />
      <div style={{
        marginTop: 20,
        marginLeft: 20
      }}>
        <NewNote />
        <Notes items={this.state.notes}/>
      </div>
    </div>
  );
}

```

Note: You can also change the modifier to `kbstyle(Ctrl/Cmd)` for applying multiple cursors with the `editor.multiCursorModifier` [setting](#). See [Multi-cursor Modifier](#) for details.

If you do not want to add all occurrences of the current selection, you can use `kb(editor.action.addSelectionToNextFindMatch)` instead. This only selects the next occurrence after the one you selected so you can add selections one by one.

```

<body>
  <span>Should be a div</span>
  <span>Should also be a div</span>
  <div><span>This is ok</span></div>
</body>

```

Column (box) selection

You can select blocks of text by holding `kbstyle(Shift+Alt)` (`kbstyle(Shift+Option)` on macOS) while you drag your mouse. A separate cursor will be added to the end of each selected line.

```

208
209 // Key          Character          Virtual Key Code
210 // Semicolon    ';'                    186
211 // Colon        ':'                    186
212 // Equals sign  '='                    187
213 // Plus         '+'                    187
214 // Comma        ','                    188
215 // Less than sign '<'                188
216 // Minus        '-'                    189
217 // Underscore   '_'                    189
218 // Period       '.'                    190
219 // Greater than sign '>'                190
220 // Forward slash '/'                  191
221

```

You can also use [keyboard shortcuts](#) to trigger column selection.

Fast scrolling

Pressing the `kbstyle(Alt)` key enables fast scrolling in the editor and Explorers. By default, fast scrolling uses a 5X speed multiplier but you can control the multiplier with the **Editor: Fast Scroll Sensitivity** (`editor.fastScrollSensitivity`) setting.

Copy line up / down

Keyboard Shortcut: `kb(editor.action.copyLinesUpAction)` or `kb(editor.action.copyLinesDownAction)`

The commands **Copy Line Up/Down** are unbound on Linux because the VS Code default keybindings would conflict with Ubuntu keybindings, see [Issue #509](#). You can still set the commands `editor.action.copyLinesUpAction` and `editor.action.copyLinesDownAction` to your own preferred keyboard shortcuts.

```

server.js
1  var express = require('express');
2  var bodyParser = require('body-parser')
3
4  var database = require('./database');
5

```

Move line up and down

Keyboard Shortcut: `kb(editor.action.moveLinesUpAction)` or `kb(editor.action.moveLinesDownAction)`

```
README.md
112
113 ## Find by symbol
114
115 <kbd>cmd+up+o</kbd>
116
117 ![Find by symbol](/media/find_by_symbol.gif)
118
119 ## Undo cursor position
120
121 <kbd>cmd+u</kbd>
122
123 ![undo cursor position](/media/undo_cursor_position.gif)
124
125 ## Move line up and down
126
127 <kbd>opt+up</kbd> or <kbd>opt+down</kbd>
128
129 # Intellisense
130
131 Anytime, try <kbd>ctrl+space</kbd> to trigger the suggest w
132
133 ## jsconfig.json
134
135 ## .eslintrc
136
137 ## Install typings
138
139 # Debugger
```

Shrink / expand selection

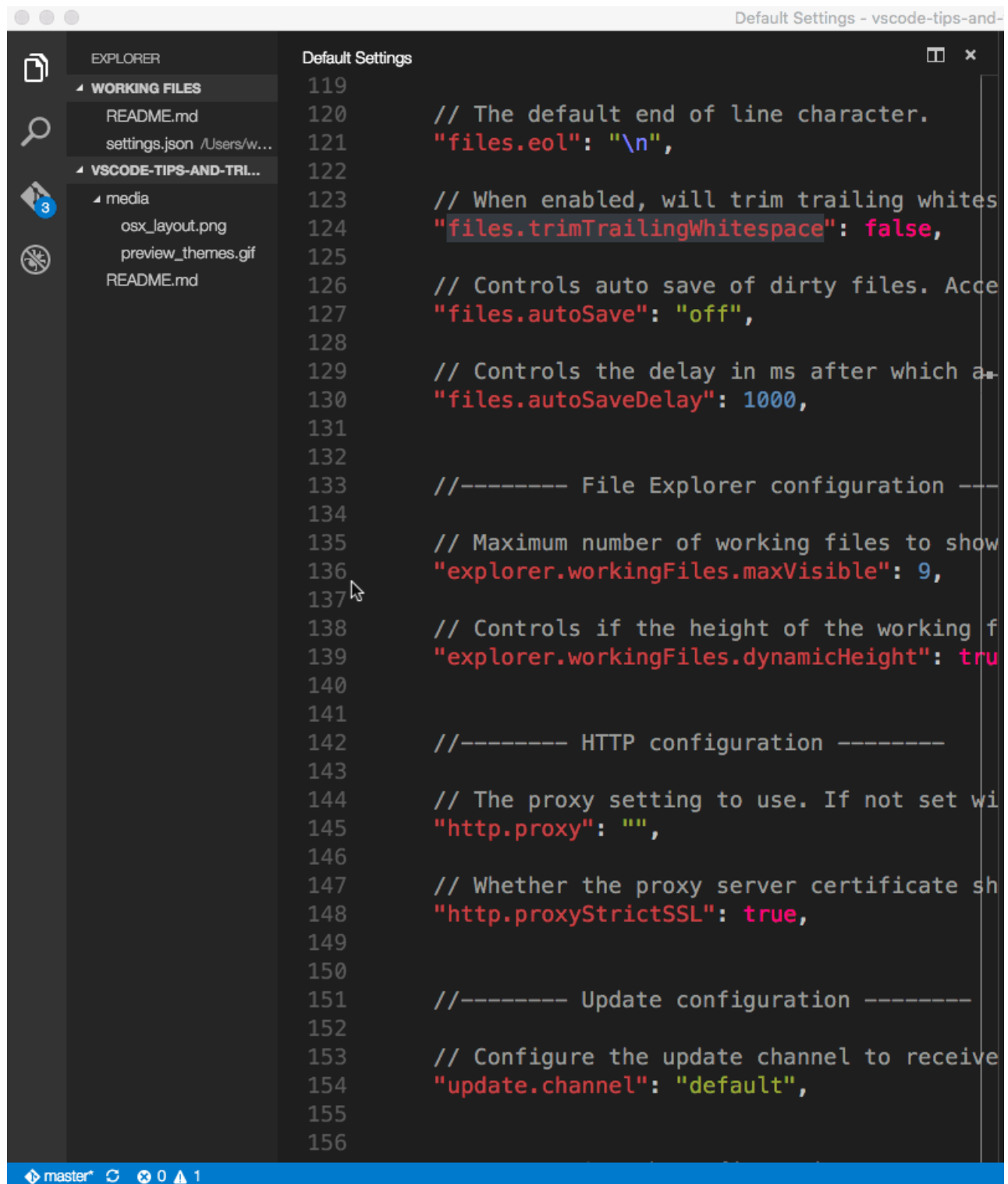
Keyboard Shortcut: `kb(editor.action.smartSelect.shrink)` or `kb(editor.action.smartSelect.expand)`

```
constructor() {
  super();
  this.state = {
    notes: [
      {
        text: 'Book flight for build'
      },
      {
        text: 'Write a cool, new app'
      },
      {
        text: 'build a react app'
      }
    ]
  };
}
```

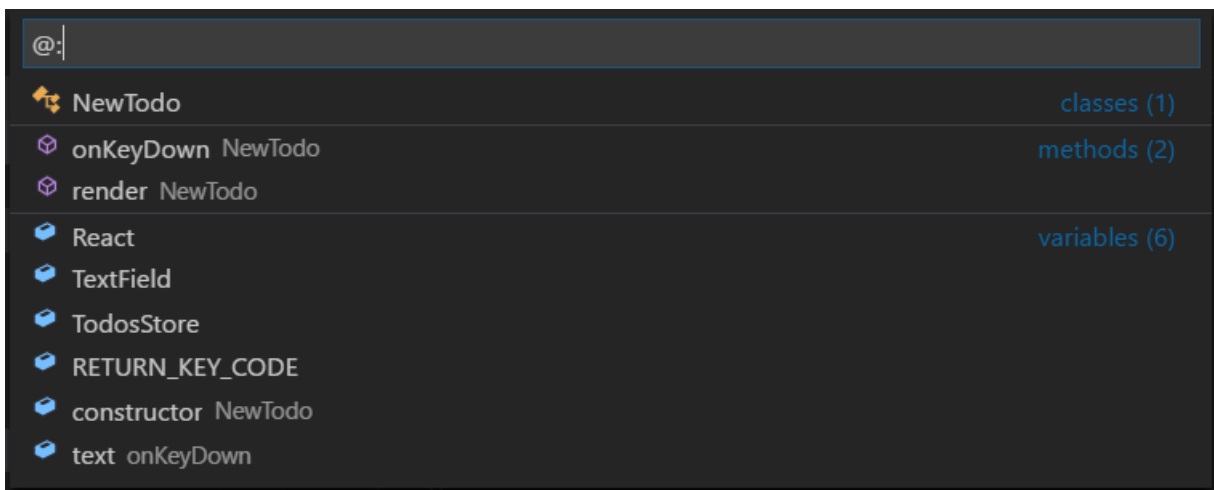
You can learn more in the [Basic Editing](#) documentation.

Go to Symbol in File

Keyboard Shortcut: `kb(workbench.action.gotoSymbol)`

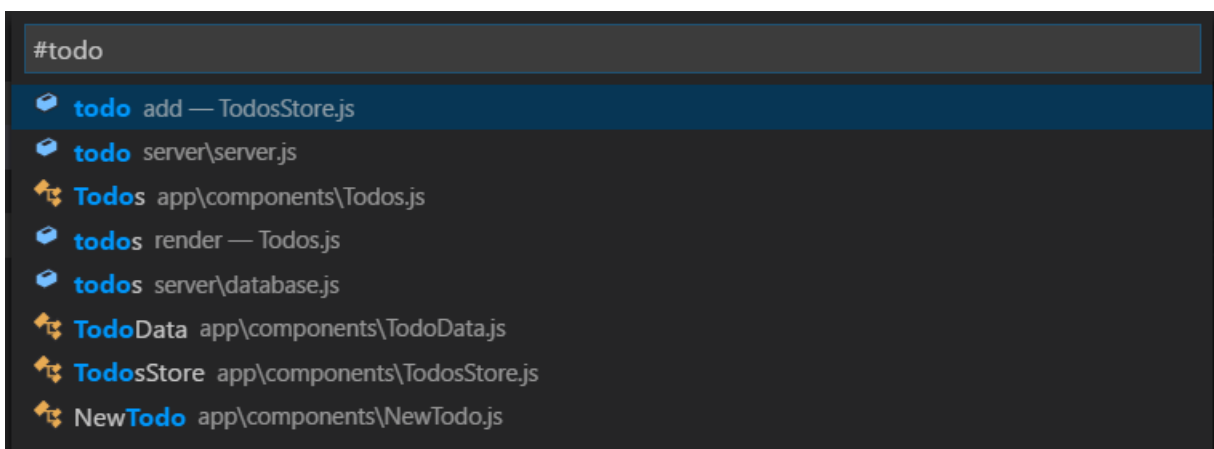


You can group the symbols by kind by adding a colon, `@:`.



Go to Symbol in Workspace

Keyboard Shortcut: `kb(workbench.action.showAllSymbols)`



Navigate to a specific line

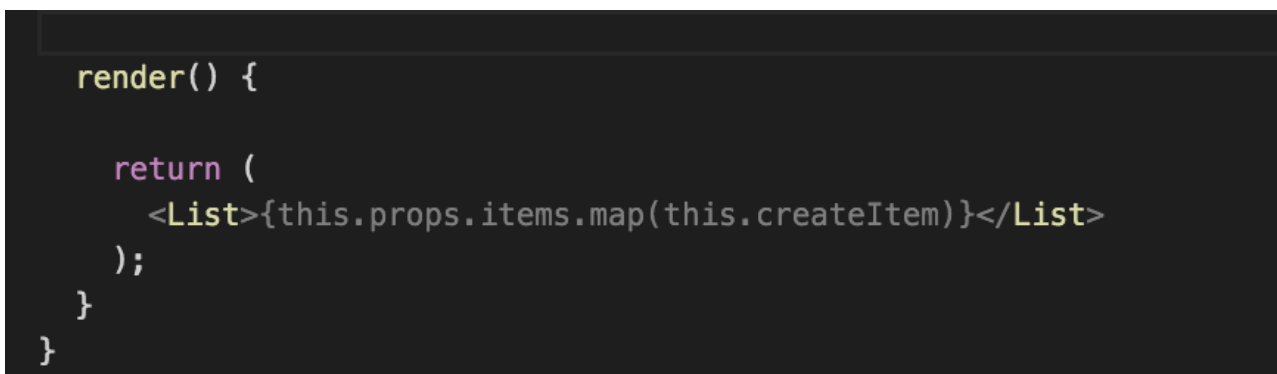
Keyboard Shortcut: `kb(workbench.action.gotoLine)`

Undo cursor position

Keyboard Shortcut: `kb(cursorUndo)`

Trim trailing whitespace

Keyboard Shortcut: `kb(editor.action.trimTrailingWhitespace)`



Code formatting

Currently selected source code: `kb(editor.action.formatSelection)`

Whole document format: `kb(editor.action.formatDocument)`

```
render() {
  return (
    <div className={s.app}>
      <AppBar
        title="Notes"
        iconClassNameRight="muidocs-icon-navigation-expand-more"
      />
      <div style={{
        marginTop: 20,
        marginLeft: 20
      }}>
        <NewNote />
        <Notes items={this.state.notes}/>
      </div>
    </div>
  );
}
```

Code folding

Keyboard Shortcut: `kb(editor.fold)` and `kb(editor.unfold)`


```
class App extends Component {  
  
  constructor() {  
    super();  
    this.state = {  
      notes: [  
        {  
          text: "go to the grocery store"  
        },  
        {  
          text: 'read medium article about engineering'  
        },  
        {  
          text: 'create build session'  
        },  
        {  
          text: 'fix bug #232'  
        }  
      ]  
    };  
  }  
}
```

Select current line

Keyboard Shortcut: `kb(expandLineSelection)`

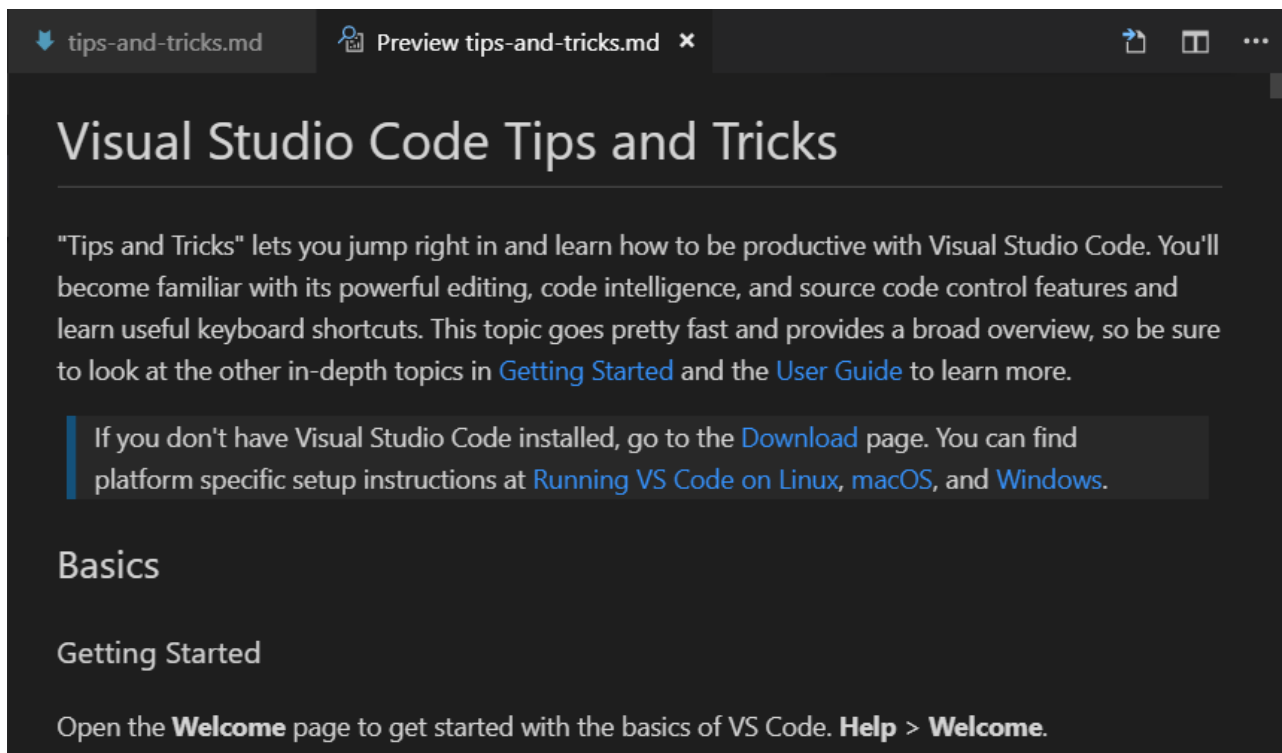
Navigate to beginning and end of file

Keyboard Shortcut: `kb(cursorTop)` and `kb(cursorBottom)`

Open Markdown preview

In a Markdown file, use

Keyboard Shortcut: `kb(markdown.showPreview)`

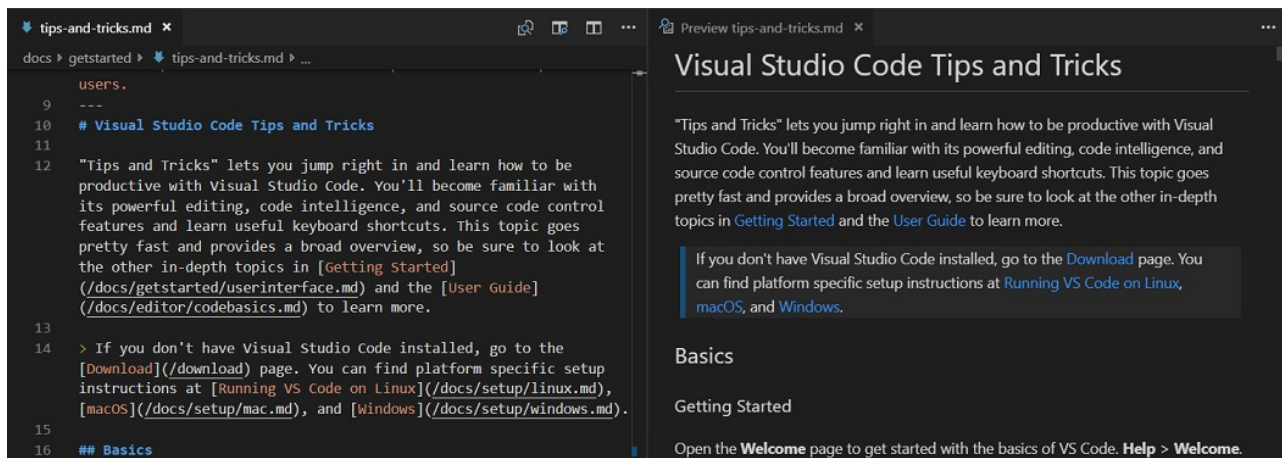


Side by side Markdown edit and preview

In a Markdown file, use

Keyboard Shortcut: `kb(markdown.showPreviewToSide)`

The preview and editor will synchronize with your scrolling in either view.



IntelliSense

`kb(editor.action.triggerSuggest)` to trigger the Suggestions widget.

```

18 // Routes
19
20 // Retrieve all todos
21 server.get('/todos', function(req, res, next){
22   database.getAll().then(function(todos) {
23     ....
24   });
25 });
26
27 // Add a new todo
28 server.post('/todos', function(req, res, next){
29   var todo = req.body;
30   database.add(todo).then(function(todos) {
31     res.send(todos);
32     next();
33   });

```

You can view available methods, parameter hints, short documentation, etc.

Peek

Select a symbol then type `kb(editor.action.peekDefinition)`. Alternatively, you can use the context menu.

```

22   });
23 }
24
25 handleClick(note) {
26   NotesStore.remove(note);
27 }
28
29 create(note) {
30   return (<ListItem onMouseDown={this.handleClick.bind(null, note)}
31     {note.text}
32   </ListItem>
33   );
34 }
35
36 render() {
37   const todos = this.state.notes.map(this.create.bind(this));
38   return (
39     <List style={{width: 400}}>

```

Go to Definition

Select a symbol then type `kb(editor.action.revealDefinition)`. Alternatively, you can use the context menu or `kbstyle(Ctrl+click)` (`kbstyle(Cmd+click)` on macOS).

```
Notes.js src/components
1  import React, {Component} from 'react';
2  import List from 'material-ui/lib/lists/list';
3  import ListItem from 'material-ui/lib/lists/list-item';
4  import DoneIcon from 'material-ui/lib/svg-icons/action/done';
5
6  import NotesStore from './NotesStore';
7
8  class Notes extends Component {
9
10     constructor() {
11         super();
12         this.state = {
13             notes: NotesStore.getAll()
14         };
15     }
16
17     componentDidMount() {
18         NotesStore.subscribe((action) => {
19             this.setState({
20                 notes: action.notes
21             });
22         });
23     }
24
25     handleClick(note) {
26         NotesStore.remove(note);
27     }
28 }
```

DO:s Ln 12, Col 21 Spaces: 2 UTF-8 LF JavaScript

You can go back to your previous location with the **Go > Back** command or

```
kb(workbench.action.navigateBack) .
```

You can also see the type definition if you press `kbstyle(Ctrl)` (`kbstyle(Cmd)` on macOS) when you are hovering over the type.

Peek References

Select a symbol then type `kb(editor.action.referenceSearch.trigger)` . Alternatively, you can use the context menu.

```
5
6 import NotesStore from './NotesStore';
7
8 class Todo extends Component {
9
10   constructor() {
11     super();
12     this.state = {
13       notes: NotesStore.getAll()
14     };
15   }
16
17   componentDidMount() {
18     NotesStore.subscribe((action) => {
19       this.setState({
20         notes: action.notes
21       });
22     });
23   }
24
25   handleClick(note) {
26     NotesStore.remove(note);
27   }
28
29   create(note) {
30     return (<ListItem onMouseDown={this.handleClick.bind(null, note)} key={note.id}>
31       {note.text}
32     </ListItem>
33   )
34 }
```

O:s Ln 8, Col 10 Spaces: 2 UTF-8 LF JavaScript

Find All References view

Select a symbol then type `kb(references-view.find)` to open the References view showing all your file's symbols in a dedicated view.

Rename Symbol

Select a symbol then type `kb(editor.action.rename)`. Alternatively, you can use the context menu.

```

7
8  class Notes extends Component {
9
10     constructor() {
11         super();
12         this.state = {
13             notes: NotesStore.getAll()
14         };
15     }
16
17     componentDidMount() {
18         NotesStore.subscribe((action) => {
19             this.setState({
20                 notes: action.notes
21             });
22         });
23     }
24
25     handleClick(note) {
26         NotesStore.remove(note);
27     }
28
29     create(note) {
30         return (<ListItem onMouseDown={this.handleClick.bind(null, note)} key={note.id}
31                     {note.text}
32                 </ListItem>

```

O:s

Ln 16, Col 1 Spaces: 2 UTF-8 LF JavaScript

Search and modify

Besides searching and replacing expressions, you can also search and reuse parts of what was matched, using regular expressions with capturing groups. Enable regular expressions in the search box by clicking the **Use Regular Expression** `.*` button (`kb(toggleSearchRegex)`) and then write a regular expression and use parenthesis to define groups. You can then reuse the content matched in each group by using `$1` , `$2` , etc. in the Replace field.

```

0 - root
1 - bin
2 - daemon
3 - adm
4 - lp
5 - sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:/:/sbin/nologin

```

.eslintrc.json

Install the [ESLint extension](#). Configure your linter however you'd like. Consult the [ESLint specification](#) for details

on its linting rules and options.

Here is configuration to use ES6.

```
{
  "env": {
    "browser": true,
    "commonjs": true,
    "es6": true,
    "node": true
  },
  "parserOptions": {
    "ecmaVersion": 6,
    "sourceType": "module",
    "ecmaFeatures": {
      "jsx": true,
      "classes": true,
      "defaultParams": true
    }
  },
  "rules": {
    "no-const-assign": 1,
    "no-extra-semi": 0,
    "semi": 0,
    "no-fallthrough": 0,
    "no-empty": 0,
    "no-mixed-spaces-and-tabs": 0,
    "no-redeclare": 0,
    "no-this-before-super": 1,
    "no-undef": 1,
    "no-unreachable": 1,
    "no-use-before-define": 0,
    "constructor-super": 1,
    "curly": 0,
    "eqeqeq": 0,
    "func-names": 0,
    "valid-typeof": 1
  }
}
```

package.json

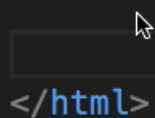
See IntelliSense for your `package.json` file.

package.json

```
1  {
2    "name": "react-todo",
3    "version": "1.0.0",
4    "description": "Simple todo application to demo react and es
5    "main": "webpack.config.js",
6    "scripts": {
7      "build": "webpack --config webpack.config.js",
8      "dev": "webpack-dev-server --config webpack.config.js --op
9    },
10   "author": "waderyan",
11   "license": "ISC",
12   "devDependencies": {
13     "babel-core": "^6.5.2",
14     "babel-loader": "^6.2.3",
15     "babel-preset-es2015": "^6.5.0",
16     "babel-preset-react": "^6.5.0",
17     "babel-preset-stage-0": "^6.5.0",
18     "html-webpack-plugin": "^2.8.1",
19     "node-sass": "^3.4.2",
20     "sass-loader": "^3.1.2",
21     "style-loader": "^0.13.0",
22     "webpack": "^1.12.13",
23     "webpack-dev-server": "^1.14.1"
24   }
```

Emmet syntax

[Support for Emmet syntax.](#)



```
</html>
```

Snippets

Create custom snippets

File > Preferences > User Snippets (**Code > Preferences > User Snippets** on macOS), select the language, and create a snippet.


```
"create component": {
  "prefix": "component",
  "body": [
    "class $1 extends React.Component {",
    "",
    "  render() {",
    "    return ($2);",
    "  }",
    "",
    "}"
  ]
},
```

See more details in [Creating your own Snippets](#).

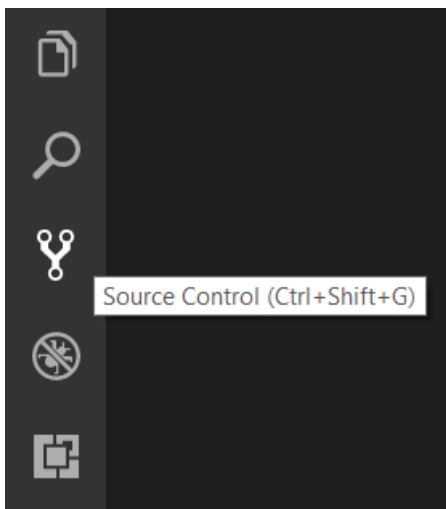
Git integration

Keyboard Shortcut: `kb(workbench.view.scm)`

Git integration comes with VS Code "out-of-the-box". You can install other SCM providers from the extension Marketplace. This section describes the Git integration but much of the UI and gestures are shared by other SCM providers.

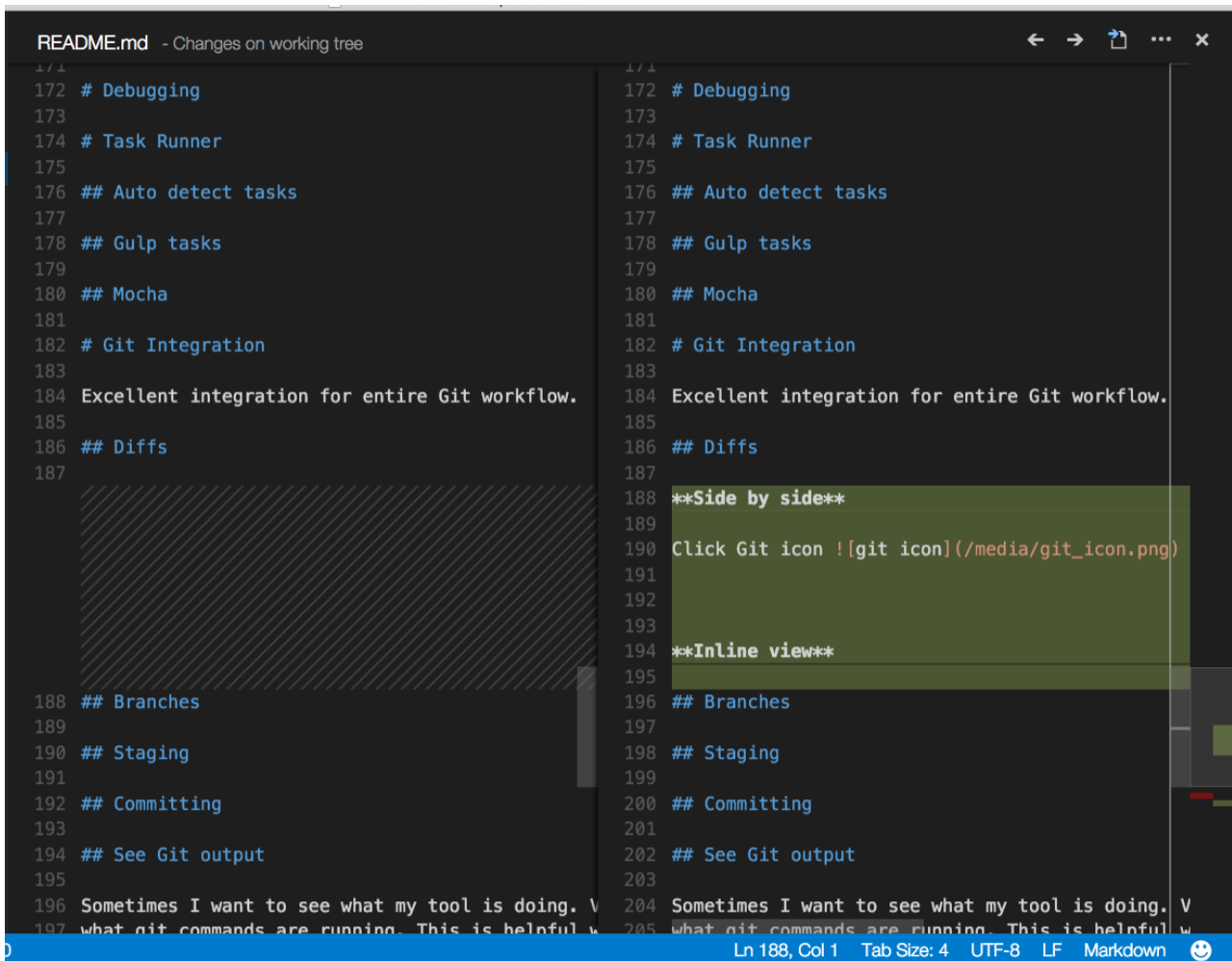
Diffs

From the **Source Control** view, select the file to diff.



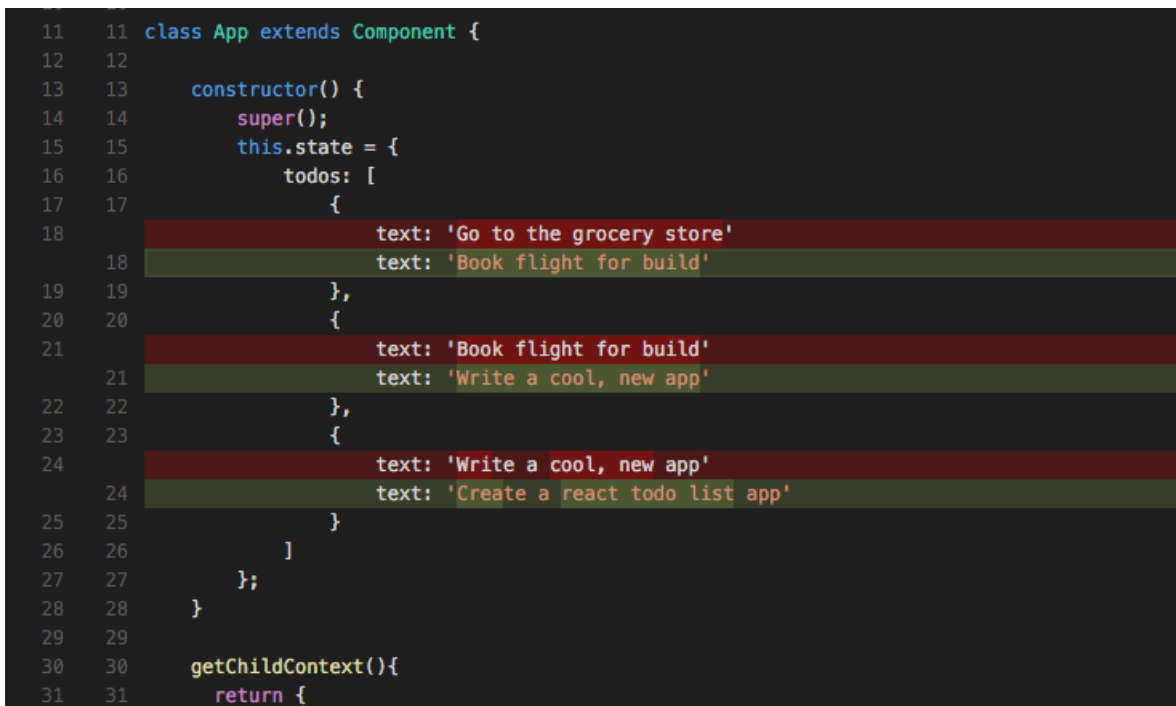
Side by side

Default is side by side diff.



Inline view

Toggle inline view by clicking the **More Actions** (...) button in the top right and selecting **Switch to Inline View**.



If you prefer the inline view, you can set `"diffEditor.renderSideBySide": false`.

Review pane

Navigate through diffs with `kb(editor.action.diffReview.next)` and `kb(editor.action.diffReview.prev)`.

This will present them in a unified patch format. Lines can be navigated with arrow keys and pressing `kbstyle(Enter)` will jump back in the diff editor and the selected line.

```
JS diff.js (Index) x
1/2: @@ -7,10 +7,13 @@
7      7  Version: 0.5
8      8  */
9      9
10     10  +// Here are some inserted lines
11     11  +// with some extra comments
12     12  +...
10     13  (function (global, undefined) {
11     14    .."use strict";
12     15    -...undefinedVariable = {};
13     16    -...undefinedVariable.prop = 5;
15     17    +...var definedVariable = {};
16     18    +...definedVariable.prop = 5;
14     19
15     20    ..function initializeProperties(target, members) {
16     21    .....var keys = Object.keys(members);
```

Edit pending changes

You can make edits directly in the pending changes of the diff view.

Branches

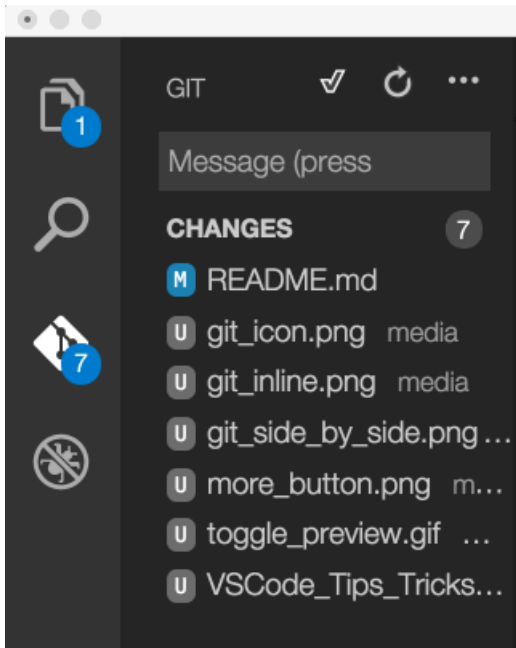
Easily switch between Git branches via the Status Bar.

```
App.js - todo
App.js src/components/App - Changes on index
1  1  import React, {Component, PropTypes} from 'react';
2  2  import AppBar from 'material-ui/lib/app-bar';
3  3  import getMuiTheme from 'material-ui/lib/styles/getMuiTheme';
4  4
5  5  import TodoList from '../TodoList';
6  6  import NewTodo from '../NewTodo';
7  7
8  8  import s from './App.scss';
9  9
10 10
11 11  class App extends Component {
12 12
13 13    constructor() {
14 14      super();
15 15      this.state = {
16 16        todos: [
17 17          {
18 18            text: 'Book flight for build'
19 19          },
20 20          {
21 21            text: 'Write a cool, new app'
22 22          },
23 23          {
24 24            text: 'Create a react todo list app'
25 25          }
        ]
      }
    }
  }
```

Staging

Stage all

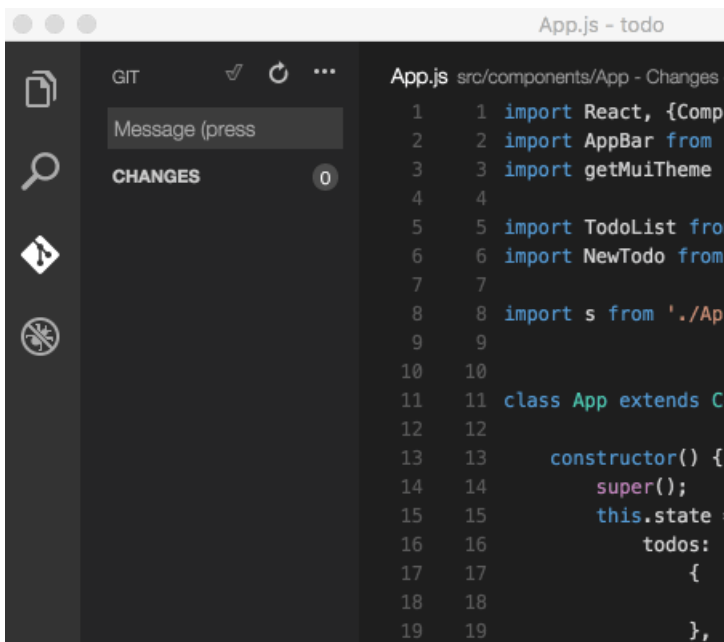
Hover over the number of files and click the plus button.



Stage selected

Stage a portion of a file by selecting that file (using the arrows) and then choosing **Stage Selected Ranges** from the **Command Palette**.

Undo last commit



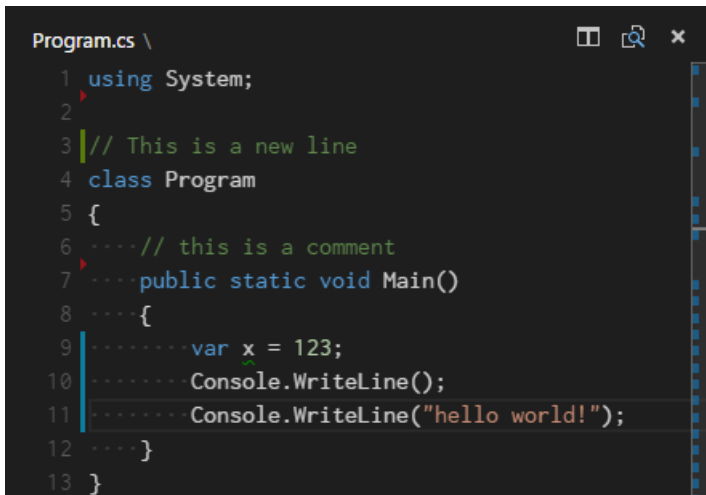
See Git output

VS Code makes it easy to see what Git commands are actually running. This is helpful when learning Git or debugging a difficult source control issue.

Use the **Toggle Output** command (`kb(workbench.action.output.toggleOutput)`) and select **Git** in the dropdown.

Gutter indicators

View diff decorations in editor. See [documentation](#) for more details.



```
Program.cs \
1 using System;
2
3 // This is a new line
4 class Program
5 {
6     // this is a comment
7     public static void Main()
8     {
9         var x = 123;
10        Console.WriteLine();
11        Console.WriteLine("hello world!");
12    }
13 }
```

Resolve merge conflicts

During a merge, go to the **Source Control** view (`kb(workbench.view.scm)`) and make changes in the diff view.

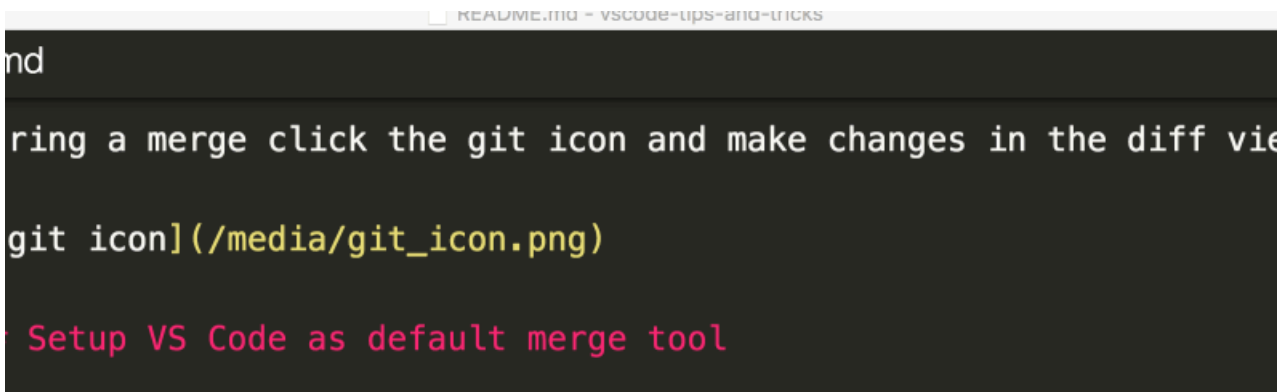
Set VS Code as default merge tool

```
git config --global merge.tool code
```

Debugging

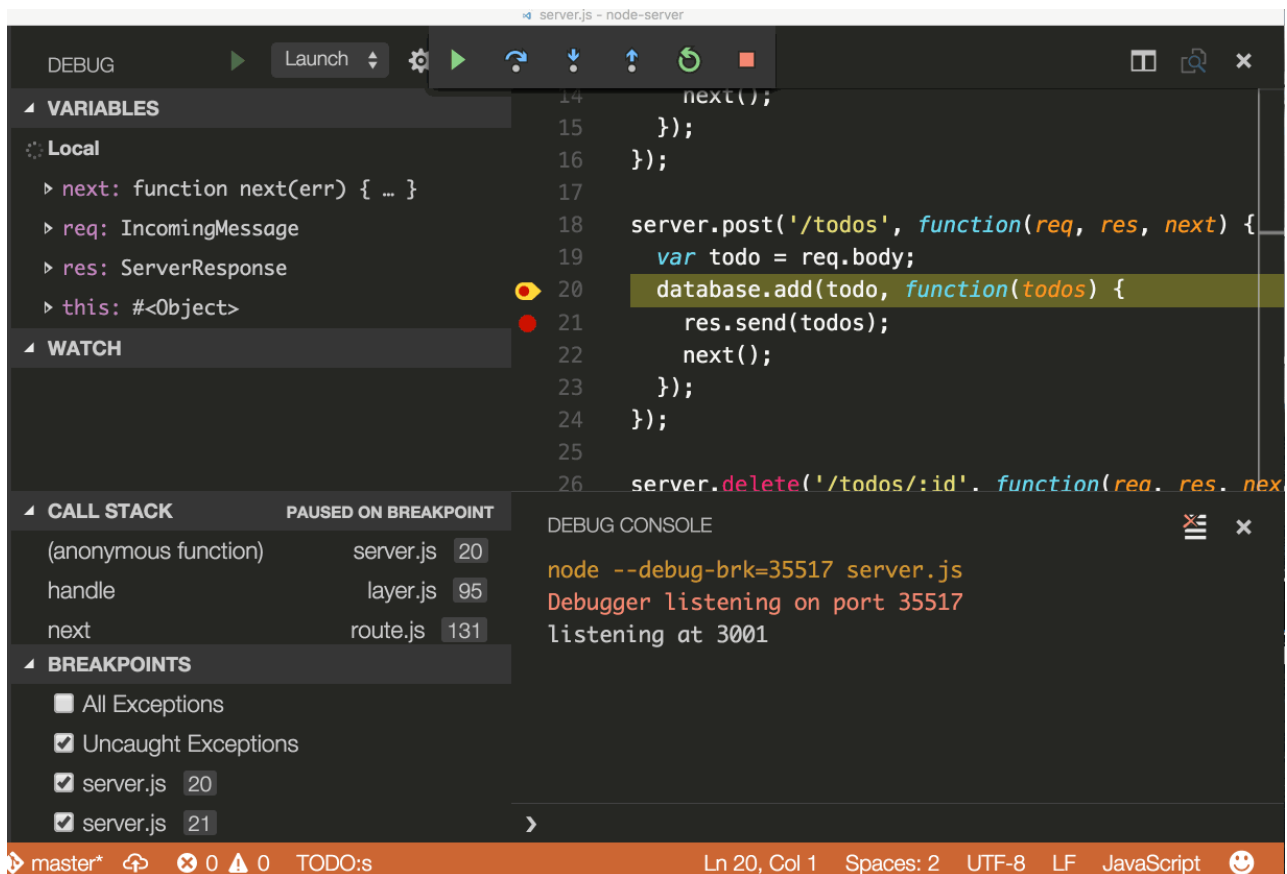
Configure debugger

Open the **Command Palette** (`kb(workbench.action.showCommands)`) and select **Debug: Open launch.json**, which will prompt you to select the environment that matches your project (Node.js, Python, C++, etc). This will generate a `launch.json` file. Node.js support is built-in and other environments require installing the appropriate language extensions. See the debugging [documentation](#) for more details.



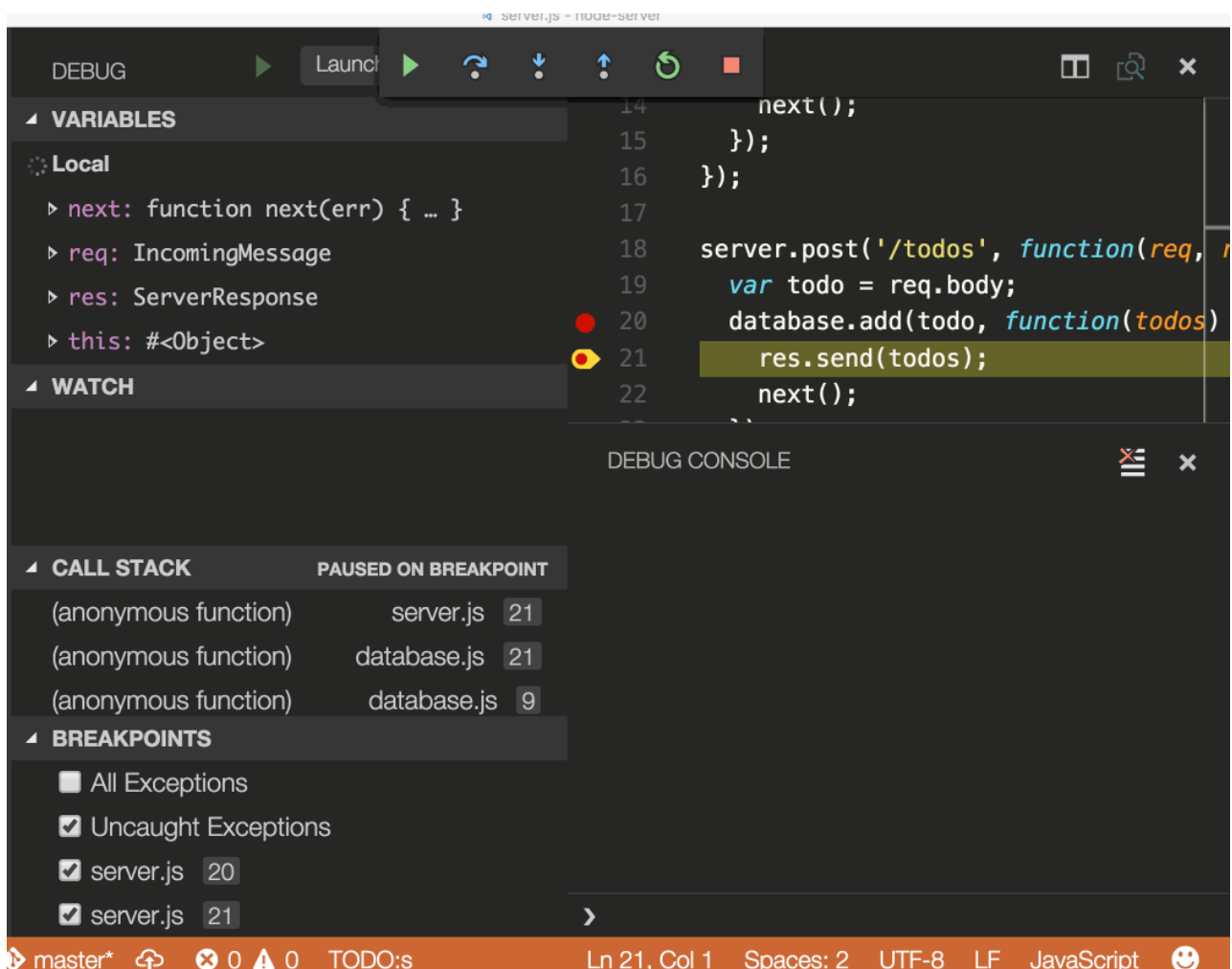
Breakpoints and stepping through

Place breakpoints next to the line number. Navigate forward with the Debug widget.



Data inspection

Inspect variables in the **Debug** panels and in the console.



Inline values

You can set `"debug.inlineValues": true` to see variable values inline in the debugger. This feature can be expensive and may slow down stepping, so it is disabled by default.

Task runner

Auto detect tasks

Select **Terminal** from the top-level menu, run the command **Configure Tasks**, then select the type of task you'd like to run. This will generate a `tasks.json` file with content like the following. See the [Tasks](#) documentation for more details.

```
{
  // See https://go.microsoft.com/fwlink/?LinkId=733558
  // for the documentation about the tasks.json format
  "version": "2.0.0",
  "tasks": [
    {
      "type": "npm",
      "script": "install",
      "group": {
        "kind": "build",
        "isDefault": true
      }
    }
  ]
}
```

There are occasionally issues with auto generation. Check out the documentation for getting things to work properly.

Run tasks from the Terminal menu

Select **Terminal** from the top-level menu, run the command **Run Task**, and select the task you want to run. Terminate the running task by running the command **Terminate Task**

```
tasks.json .vscode
1  {
2      // See http://go.microsoft.com/fwlink/?LinkId=733558
3      // for the documentation about the tasks.json format
4      "version": "0.1.0",
5      "command": "npm",
6      "isShellCommand": true,
7      "showOutput": "always",
8      "suppressTaskName": true,
9      "tasks": [
10         {
11             "taskName": "install",
12             "args": ["install"]
13         },
14         {
15             "taskName": "build",
16             "args": ["run", "dev"]
17         }
18     ]
19 }
```

master* 0 0 TODO:s Ln 9, Col 15 Tab Size: 4 UTF-8 LF JSON

Define keyboard shortcuts for tasks

You can define a keyboard shortcut for any task. From the **Command Palette**

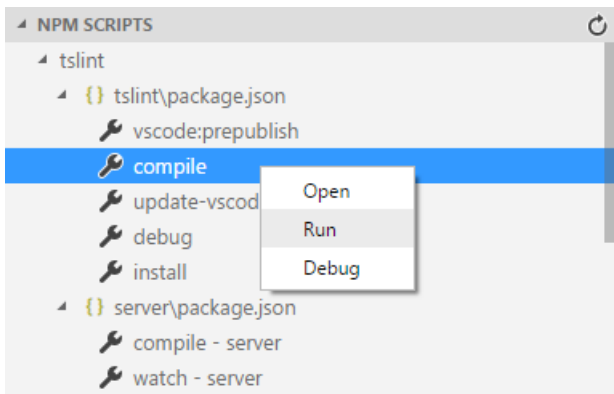
(`kb(workbench.action.showCommands)`), select **Preferences: Open Keyboard Shortcuts File**, bind the desired shortcut to the `workbench.action.tasks.runTask` command, and define the **Task** as `args`.

For example, to bind `kbstyle(Ctrl+H)` to the `Run tests` task, add the following:

```
{
  "key": "ctrl+h",
  "command": "workbench.action.tasks.runTask",
  "args": "Run tests"
}
```

Run npm scripts as tasks from the explorer

With the setting `npm.enableScriptExplorer`, you can enable an explorer that shows the scripts defined in your workspace.



From the explorer you can open a script in the editor, run it as a task, and launch it with the node debugger (when the script defines a debug option like `--inspect-brk`). The default action on click is to open the script. To run a script on a single click, set `npm.scriptExplorerAction` to "run". Use the setting `npm.exclude` to exclude scripts in `package.json` files contained in particular folders.

Portable mode

VS Code has a [Portable mode](#) which lets you keep settings and data in the same location as your installation, for example, on a USB drive.

Insiders builds

The Visual Studio Code team uses the Insiders version to test the latest features and bug fixes of VS Code. You can also use the Insiders version by [downloading it here](#).

- For Early Adopters - Insiders has the most recent code changes for users and extension authors to try out.
- Frequent Builds - New builds every day with the latest bug fixes and features.
- Side-by-side install - Insiders installs next to the Stable build allowing you to use either independently.