c:5512Drive-A-SeptemberGuides2mds-combined```

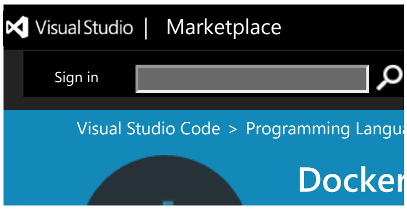
- <===============(Markdown PDF)===============>
- Markdown PDF
- Table of Contents
- Features
  - markdown-it-container
  - markdown-it-plantuml
  - markdown-it-include
  - mermaid
- Install
- Usage
  - Command Palette
  - Menu
  - Auto convert
- Extension Settings
- Options
  - List
  - Save options
  - `markdown-pdf.type`
  - `markdown-pdf.convertOnSave`
  - `markdown-pdf.convertOnSaveExclude`
  - `markdown-pdf.outputDirectory`
  - `markdown-pdf.outputDirectoryRelativePathFile`
  - Styles options
  - `markdown-pdf.styles`
  - `markdown-pdf.stylesRelativePathFile`
  - `markdown-pdf.includeDefaultStyles`
  - Syntax highlight options
  - `markdown-pdf.highlight`
  - `markdown-pdf.highlightStyle`
  - Markdown options
  - `markdown-pdf.breaks`
  - Emoji options
  - `markdown-pdf.emoji`
  - Configuration options
  - `markdown-pdf.executablePath`
  - Common Options
  - `markdown-pdf.scale`
  - PDF options
  - `markdown-pdf.displayHeaderFooter`
  - `markdown-pdf.headerTemplate`
  - `markdown-pdf.footerTemplate`
  - `markdown-pdf.printBackground`
  - `markdown-pdf.orientation`
  - `markdown-pdf.pageRanges`
  - `markdown-pdf.format`
  - `markdown-pdf.width`
  - `markdown-pdf.height`
  - `markdown-pdf.margin.top`
  - `markdown-pdf.margin.bottom`
  - `markdown-pdf.margin.right`
  - `markdown-pdf.margin.left`
  - PNG JPEG options
  - `markdown-pdf.quality`
  - `markdown-pdf.clip.x`

  - `markdown-pdf.clip.y`
  - `markdown-pdf.clip.width`
  - `markdown-pdf.clip.height`
  - `markdown-pdf.omitBackground`
  - PlantUML options
  - `markdown-pdf.plantumlOpenMarker`
  - `markdown-pdf.plantumlCloseMarker`
  - `markdown-pdf.plantumlServer`
  - markdown-it-include options
  - `markdown-pdf.markdown-it-include.enable`
  - mermaid options
  - `markdown-pdf.mermaidServer`
- FAQ
  - How can I change emoji size ?
  - Auto guess encoding of files
  - Output directory
  - Page Break
- Known Issues
  - `markdown-pdf.styles` option
- Release Notes
- <===============(Open in External App)===============>
- Open in External App
- 💡 Motivation
- 🔌 Installation
- 🔧 Configuration
- :loudspeaker: Limits
  - 1. Node package: open
  - 2. VSCode extension API: `vscode.env.openExternal(target: Uri)`
- # <===============(Change Case Extension for Visual Studio Code)===============>
- Change Case Extension for Visual Studio Code
- Install
- Commands
- Support
- <===============(vscode-js-console-utils)===============>
- vscode-js-console-utils
- Installing
- <===============(Node.js Extension Pack)===============>
- Node.js Extension Pack
- Extensions Included
- Want to see your extension added?
- <===============(Refactor CSS)===============>
- Refactor CSS
- Features
- Release Notes
- Roadmap
- <===============(Treedbox Javascript)===============>
- Treedbox JavaScript - Visual Studio Marketplace
- VS Code Javascript extension
- Screenshots:
- Install
- Pages
- Features - All Math properties and methods
- How to use
  - Example:
- <===============(pdf)===============>

- pdf
- Contribute
  - Upgrade PDF.js
- Change log
- License
- <===============(Superpowers)===============>
- Superpowers
- Preview
- Features
- Map/Sort operation usage
- Dynamic snippets / completion
- Extension Settings
- <===============(Readme Pattern)===============>
- Readme Pattern
- Features
- Logo
- <===============(Markdown Paste)===============>
- Markdown Paste
- Requirements
- Features
- Config
- Format
  - File name format
  - File link format
- <===============(Spelling Checker for Visual Studio Code)===============>
- Spelling Checker for Visual Studio Code
- Support Further Development
- Functionality
  - Example
- Suggestions
- Install
- Supported Languages
- Add-On Specialized Dictionaries
- Enabled File Types
  - Enable / Disable File Types
- How it works with camelCase
  - Special case will ALL CAPS words
- Things to note
- In Document Settings
  - Enable / Disable checking sections of code
  - Disable Checking
  - Enable Checking
  - Example
  - Ignore
  - Words
  - Enable / Disable compound words
  - Excluding and Including Text to be checked.
  - Exclude Example
  - Include Example
- Predefined RegExp expressions
  - Exclude patterns
  - Include Patterns
- Customization
  - Adding words to the Workspace Dictionary
  - cspell.json
  - Example *cspell.json* file

  - VS Code Configuration Settings
- Dictionaries
  - General Dictionaries
  - Programming Language Dictionaries
  - Miscellaneous Dictionaries
  - How to add your own Dictionaries
  - Global Dictionary
    - Define the Dictionary
  - Project / Workspace Dictionary
- FAQ
- <===============(Settings viewer)===============>
- Settings viewer
- <===============(Awesome plugin for Visual Studio Code)===============>
- Awesome plugin for Visual Studio Code :sunglasses: [Beta]
- Quickly see docs from MDN in VS Code
- Usage
  - Is the semicolon necessary?
- Examples
- Supports
- <===============(Live Server)===============>
- Live Server
- Shortcuts to Start/Stop Server
- Features
- Installation
- Settings
- FAQs
- What's new ?
- Changelog
- LICENSE
- <===============(Live Sass Compiler)===============>
- Live Sass Compiler
- Usage/Shortcuts
- Features
- Installation
- Settings
- FAQ
- Extension Dependency
- What's new ?
- Changelog
- LICENSE
- <===============(CSS PEAK)===============>
- Functionality
- Configuration
- Contributing
- <===============(jsdoc-live-preview)===============>
- jsdoc-live-preview
- Installation
- Usage
- <===============(Resource Monitor)===============>
- Resource Monitor
- Features
- Screenshots
- Requirements
- Extension Settings
- Known Issues

- <===============(JS FLOWCHART)===============>
- Features
- Quick start
- Installation in development mode
- # <===============(Visual Studio Code Remote Development)===============>
- Visual Studio Code Remote Development
- Documentation
- # <===============(Docker for Visual Studio Code)===============>
- Docker - Visual Studio Marketplace
- Docker for Visual Studio Code [Installs](https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-docker) [Build Status](https://dev.azure.com/ms-azuretools/AzCode/_build/latest?definitionId=22&branchName=master)


Visual Studio | Marketplace
Sign in
Visual Studio Code > Programming Langu
Docker

- Installation
- Overview of the extension features
  - Editing Docker files
  - Generating Docker files
  - Docker view
  - Docker commands
  - Docker Compose
  - Using image registries
  - Debugging services running inside a container
  - Azure CLI integration
- # <===============(vscode-htmlhint)===============>
- vscode-htmlhint
- Configuration
- Usage
- Rules
- .htmlhintrc
- Additional file types
  - Option 1: Treating your file like any other html file
  - Option 2: Associating HTMLHint extension with your file type
- Settings
- # <===============(Git Graph extension for Visual Studio Code)===============>
- Git Graph extension for Visual Studio Code
- Features
- Extension Settings
- Extension Commands
- Release Notes
- Visual Studio Marketplace
- Acknowledgements
- - Icons8 ([License](https://icons8.com/license))
- # <===============(MARKDOWN SHORTCUTS)===============>
- Exposed Commands
- # <===============(MarkdownConverter)===============>
- MarkdownConverter
- What's `MarkdownConverter`?
- Usage
- [VSCode]: https://code.visualstudio.com/
- # <===============(vscode-opn)===============>
- vscode-opn
- Install

  - Easiest from the Extension Gallery
  - Alternatively, with the Packaged Extension (.vsix) file
- Usage
  - Customise behaviour per language (optional)
  - forLang
  - openInApp
  - openInAppArgs
  - isUseWebServer
  - isUseFsPath
  - isWaitForAppExit
  - Web server settings (applies to all languages)
  - vscode-opn.webServerProtocol
  - vscode-opn.webServerHostname
  - vscode-opn.webServerPort
- Tested
- Release notes
- Contributions
- Dependencies
- License
- # <===============(Node.js Modules Intellisense)===============>
- Node.js Modules Intellisense
- Installation
- Issues & Contribution
- Configuration
- # <===============(VSCode format in context menus)===============>
- VSCode format in context menus
- Features
- Extension Settings
- # <===============(vscode-github README)===============>
- vscode-github README
- Features
- Setup Personal Access Token
- Usage
  - Create a new pull request
  - Create a new pull request from a forked repository
  - Checkout pull request
  - Browser pull request
  - Merge pull request
  - Telemetry data (extension usage)
- # <===========(Kite Autocomplete Plugin for Visual Studio Code)===========>
- Kite Autocomplete Plugin for Visual Studio Code
- Requirements
- Installation
  - Installing the Kite Engine
  - Installing the Kite Plugin for Visual Studio Code
- Usage
  - Autocompletions
  - Hover (Python only)
  - Documentation (Python only)
  - Definitions (Python only)
  - Function Signatures (Python only)
  - Commands
- Troubleshooting
  - About Kite
- # <===============(Code Metrics - Visual Studio Code Extension)===============>
- Code Metrics - Visual Studio Code Extension

- Complexity calculation
- It looks like this
- Install
- Customization
- Commands
  - License
- # <===============(Peacock for Visual Studio Code)===============>
- We can even add meta tags to the page! This sets the keywords meta tag.
-
- - content: vscode "visual studio code" peacock theme extension documentation docs guide help "get started"
- Peacock for Visual Studio Code
- Overview
- Install
- Quick Usage
- Features
- Settings
  - Favorite Colors
  - Preview Your Favorite
  - Save Favorite Color
  - Affected Elements
  - Element Adjustments
  - Keep Foreground Color
  - Surprise Me On Startup
  - Lighten and Darken
- Commands
- Keyboard Shortcuts
- Integrations
  - VS Live Share Integration
  - Remote Development Integration
- Input Formats
  - Alpha Support
- Roadmap
  - Issues
  - Contributions
  - Logging
  - Example 1
  - Example 2
  - Example 3
  - How does title bar coloring work
  - How are foreground colors calculated
  - Why is the foreground hard to see with my transparent color
  - Why are my affected elements not transparent
  - What are recommended favorites
  - What are mementos
- Migration
  - To Version 3+
- Try the Code
- Badges
- Resources
- # <===============(Markdown Extended Readme)===============>
- Markdown Extended Readme
- Features
  - Disable Plugins
- Q: Why You Don't Integrate Some Plugin?
- Exporter
  - Export Configurations

- Helpers
  - Editing Helpers and Keys
  - Table Editing
  - Paste as Markdown Table
  - Export & Copy
- Extended Syntaxes
  - Admonition
  - Removing Admonition Title
  - Supported Qualifiers
  - Enhanced Anchor Link
  - markdown-it-table-of-contents
  - markdown-it-footnote
  - markdown-it-abbr
  - markdown-it-deflist
  - markdown-it-sup markdown-it-sub
  - markdown-it-checkbox
  - markdown-it-attrs
  - markdown-it-kbd
  - markdown-it-underline
  - markdown-it-container
- Known Issues & Feedback
- # <===============(Search node_modules)===============>
- Search node_modules
- Features
- Settings
- Extension Packs
- Links
- # <===============(Browser Sync)===============>
- Objective
- Requirement
- Features
- Behaviour
- Server Mode
- Proxy Mode
- Configuration
- Example setting (different browser)
- Example setting (relative path)
- How it works
- Static HTML file
- Proxy for dynamic site
- Enhancement Planning
- 3. Better error handling
- # <===============(VSCode Essentials (Extension Pack))===============>
- VSCode Essentials (Extension Pack)
- VSCode Masterclass (Coming Soon)
- Introduction
- Features
- Recommended Settings
  - All Autocomplete
  - CodeStream
  - Customize UI
  - GitLens
  - Macros
  - MetaGo
  - Project Manager

  - Rewrap
  - Settings Cycler
  - Settings Sync
  - Todo Tree
- Bug Reports
- Known Issues
- Included Extensions
- # <===============(VS Code JavaScript Unit Test snippets)===============>
- JavaScript
- ## VS Code JavaScript Unit Test snippets
- Installation
- Snippets
  - Import and export
- # <===============(Path Autocomplete for Visual Studio Code)===============>
- Path Autocomplete for Visual Studio Code
- Features
- Installation
- Options
- Configure VSCode to recognize path aliases
- Tips
- # <===============(easy-snippet)===============>
- easy-snippet
- Features
- Release Notes
  - 0.1.6
  - 0.1.5
  - 0.0.1
- # <===============(VSCode Log Output Colorizer)===============>
- VSCode Log Output Colorizer
- Change Log
- Contributing
- In action
  - VSCode Git Output
  - Default Dark Theme
  - Default Light Theme
  - Night Blue Theme
- Helpful References:
- Support
- License
- Attribution
- # <===============(REST Client)===============>
- REST Client
- Main Features
- Usage
  - Select Request Text
- Install
- Making Request
  - Request Line
  - Query Strings
  - Request Headers
  - Request Body
- Making GraphQL Request
- Making cURL Request
- Copy Request As cURL
- Cancel Request
- Rerun Last Request

- Request History
- Save Full Response
- Save Response Body
- Fold and Unfold Response Body
- Authentication
    - Basic Auth
    - Digest Auth
    - SSL Client Certificates
    - Azure Active Directory(Azure AD)
    - Microsoft Identity Platform(Azure AD V2)
    - AWS Signature v4
- Generate Code Snippet
- HTTP Language
    - Auto Completion
    - Navigate to Symbols in Request File
- Environments
- Variables
    - Custom Variables
    - Environment Variables
    - File Variables
    - Request Variables
    - System Variables
- Customize Response Preview
- Settings
    - Per-request Settings
- License
- Change Log
- Special Thanks
- Feedback
- # <===============((vscode-favorites))===============>
- vscode-favorites
- Install
- Usage
    - Configuration
- Changelog
- LICENSE
- # <===============(js-beautify for VS Code)===============>
- js-beautify for VS Code
- How we determine what settings to use
    - VS Code | .jsbeautifyrc settings map
    - Keyboard Shortcut
- Contributing
- # <===============(Todo Tree)===============>
- Todo Tree
- Highlighting
- Installing
    - Source Code
- Controls
- Folder Filter Context Menu
- Commands
    - Tags
    - Export
- Configuration
    - Multiline TODOs
    - Excluding files and folders
- # <===============(Open file)===============>

- Open file
- Example
- Main Features
- Path Lookup Detail
- # <===============(Code Runner)===============>
- Code Runner
- Features
- Usages
- Configuration
- About CWD Setting (current working directory)
- Note
- Telemetry data
- Change Log
- Issues
- Contribution
- # <===============(Git Project Manager)===============>
- Git Project Manager
- Available commands
  - GPM: Open Git Project *(Defaults to: Ctrl+Alt+P)*
  - GPM: Refresh Projects
  - GPM: Refresh specific project folder
  - GPM: Open Recent Git Project *(Defaults to Ctrl+Shift+Q)*
- Available settings
- Participate
- # <===============(Open in Code)===============>
- Open in Code
- Install
- Usage
- Contributing
- License
- # <===============(Reactjs)===============>
- Reactjs
- VS Code Reactjs snippets
- Installation
- Supported languages (file extensions)
- Breaking change in version 2.0.0
- Breaking change in version 1.0.0
- Usage
- Snippets
- # <===============(jQuery Code Snippets)===============>
- jQuery Code Snippets
- Snippets
- column0
- Source
- License
- # <===============(Markdown table prettifier extension for Visual Studio Code)===============>
- Markdown table prettifier extension for Visual Studio Code
- Features
  - CLI formatting
  - Formatting with docker
- Extension Settings
- Known Issues
- - Tables with mixed character widths (eg: CJK) are not always properly formatted (issue #4).
- # <===============(vscode-goto-documentation)===============>
- vscode-goto-documentation

- Supports
- Installation
- How to use
- Edit the urls
  - The available settings are:
- # <===============(VSCode DevTools for Chrome)===============>
- VSCode DevTools for Chrome
- Attaching to a running chrome instance:
- Launching a 'debugger for chrome' project and using screencast:
- Using the extension
- Launching as a Debugger
- Launching Chrome manually
- Launching Chrome via the extension
- Known Issues
- Developing the extension itself
- # <===============(JS Refactor)===============>
- JS Refactor
- Donate to the JSR Cause
  - Supported Language Files
- Installation
  - Extensions Panel:
  - Command Pallette
- Find Me Online
- Automated Refactorings
  - Keybindings
  - Usage
  - Explanations
  - Core Refactorings
  - Other Utilities
- Snippets
  - Usage
  - Explanations
- # <===============(Npm Intellisense)===============>
- Npm Intellisense
- Sponsors
- Installation
- Contributing
- Features
  - Import command
  - Import command (ES5)
  - Scan devDependencies
  - Show build in (local) libs
  - Lookup package.json recursive
  - Experimental: Package Subfolder Intellisense
- License
- # <===============(vscode-standardjs)===============>
- vscode-standardjs
- How to use
- Plugin options
- Configuring Standard
  - Commands
  - FAQ
- How to develop
- How to package
- TODO
- # <===============(COMMENT HEADERS)===============>

- File Header Comment - Visual Studio Marketplace
- Features
- Install
- Extension Settings
- Release Notes
  - 0.0.5
  - 0.0.4
  - 0.0.3
  - 0.0.2
  - 0.0.1
- # <===============(Bookmarks)===============>
- What's new in Bookmarks 11.4
- Support
- Sponsors
- Bookmarks
- Features
- Available commands
- Manage your bookmarks
  - Toggle / Toggle Labeled
- Navigation
  - Jump to Next / Previous
  - List / List from All Files
- Selection
  - Select Lines
  - Expand Selection to the Next/Previous Bookmark or Shrink the Selection
- Available Settings
- Available Colors
- Side Bar
- Project and Session Based
- License
- # <===============(vscode-markdown-pdf)===============>
- yzane/vscode-markdown-pdf
- Usage
  - Command Palette
  - Menu
  - Auto convert
- Extension Settings
- Options
  - List
  - Save options
  - `markdown-pdf.type`
  - `markdown-pdf.convertOnSave`
  - `markdown-pdf.convertOnSaveExclude`
  - `markdown-pdf.outputDirectory`
  - `markdown-pdf.outputDirectoryRelativePathFile`
  - Styles options
  - `markdown-pdf.styles`
  - `markdown-pdf.stylesRelativePathFile`
  - `markdown-pdf.includeDefaultStyles`
  - Syntax highlight options
  - `markdown-pdf.highlight`
  - `markdown-pdf.highlightStyle`
  - Markdown options
  - `markdown-pdf.breaks`
  - Emoji options
  - `markdown-pdf.emoji`

  - Configuration options
  - `markdown-pdf.executablePath`
  - Common Options
  - `markdown-pdf.scale`
  - PDF options
  - `markdown-pdf.displayHeaderFooter`
  - `markdown-pdf.headerTemplate`
  - `markdown-pdf.footerTemplate`
  - `markdown-pdf.printBackground`
  - `markdown-pdf.orientation`
  - `markdown-pdf.pageRanges`
  - `markdown-pdf.format`
  - `markdown-pdf.width`
  - `markdown-pdf.height`
  - `markdown-pdf.margin.top`
  - `markdown-pdf.margin.bottom`
  - `markdown-pdf.margin.right`
  - `markdown-pdf.margin.left`
  - PNG JPEG options
  - `markdown-pdf.quality`
  - `markdown-pdf.clip.x`
  - `markdown-pdf.clip.y`
  - `markdown-pdf.clip.width`
  - `markdown-pdf.clip.height`
  - `markdown-pdf.omitBackground`
  - PlantUML options
  - `markdown-pdf.plantumlOpenMarker`
  - `markdown-pdf.plantumlCloseMarker`
  - `markdown-pdf.plantumlServer`
  - markdown-it-include options
  - `markdown-pdf.markdown-it-include.enable`
  - mermaid options
  - `markdown-pdf.mermaidServer`
- FAQ
  - How can I change emoji size ?
  - Auto guess encoding of files
  - Output directory
  - Page Break
- Source
- # <===============()===============>
- 194.WallabyJs.quokka-vscode
- Rapid JavaScript Prototyping in your Editor
- Quokka.js is a developer productivity tool for rapid JavaScript / TypeScript prototyping. Runtime values are updated and displayed in your IDE next to your code, as you type.
- Quick Start
- ## Live Feedback
- Live Code Coverage
- Value Explorer
  - Auto-Expand Value Explorer Objects {#auto-expand-value-explorer-objects .vsc-doc .jb-doc style="display: block;"}
- VS Code Live Share Integration {#live-share-integration .vsc-doc style="display: block;"}
  - How does it work?
- Interactive Examples
- Live Comments
  - Live comment snippet
- Live Value Display
  - Show Value

  - Copy Value
- Project Files Import
- Quick Package Install
- Live Performance Testing
- Run on Save/Run Once
- Runtime
- Browser-like Runtime
- Configuration
- Plugins
- Examples
- Questions and issues
- # <===============(Browser Preview)===============>
- Features
- # <===============(Microsoft Edge Tools)===============>
- Microsoft Edge Tools for VS Code - Visual Studio Marketplace
- Supported Features
- Using the Extension
  - Getting Started
  - Changing Extension Settings
  - Turning on Network Inspection
  - Turning on Headless Mode
  - Using the tools
  - Opening source files from the Elements tool
  - Debug Configuration
  - Other optional launch config fields
  - Sourcemaps
  - Ionic/gulp-sourcemaps note
  - Launching the browser via the side bar view
  - Launching the browser manually
  - Attaching automatically when launching the browser for debugging
- Data/Telemetry
- Source
- # <===============(Turbo Console Log)===============>
- Turbo Console Log
- Main Functionality
- Features
- # <===============(Regex Previewer - Visual Studio Marketplace)===============>
- Regex Previewer - Visual Studio Marketplace
- Features
- Source
- # <===============(JavaScript Snippet Pack)===============>
- JavaScript Snippet Pack - Visual Studio Marketplace
- JavaScript Snippet Pack for Visual Studio Code
- Usage
- Console
  - [cd] console.dir
  - [ce] console.error
  - [ci] console.info
  - [cl] console.log
  - [cw] console.warn
  - [de] debugger
- DOM
  - [ae] addEventListener
  - [ac] appendChild
  - [rc] removeChild

  - [cel] createElement
  - [cdf] createDocumentFragment
  - [ca] classList.add
  - [ct] classList.toggle
  - [cr] classList.remove
  - [gi] getElementById
  - [gc] getElementsByClassName
  - [gt] getElementsByTagName
  - [ga] getAttribute
  - [sa] setAttribute
  - [ra] removeAttribute
  - [ih] innerHTML
  - [tc] textContent
  - [qs] querySelector
  - [qsa] querySelectorAll
- Loop
  - [fe] forEach
- Function
  - [fn] function
  - [afn] anonymous function
  - [pr] prototype
  - [iife] immediately-invoked function expression
  - [call] function call
  - [apply] function apply
  - [ofn] function as a property of an object
- JSON
  - [jp] JSON.parse
  - [js] JSON.stringify
- Timer
  - [si] setInterval
  - [st] setTimeout
- Misc
  - [us] use strict
  - [al] alert
  - [co] confirm
  - [pm] prompt
- # <===============(Node.js Exec)===============>
- Node.js Exec - Visual Studio Marketplace
  - Execute the current file or your selected code with node.js.
- Usage
- Configuration
  - The folloing options need to set the legacyMode off
- How it works
- # <===============(Search Editor)===============>
- Search Editor: Apply Changes - Visual Studio Marketplace
- Known Limitations
- Keybindings
- Source
- # <===============(Node TDD)===============>
- Node TDD - Visual Studio Marketplace
- Features
- Settings
- Commands
- Limitations and known issues
- * Ironically for a TDD extension, it has very few tests of its own because I don't yet know how to test UI elements in VS Code. :/
- # <===============()===============>

- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>
- # <===============()===============>

Markdown Support for Visual Studio Code

vscode marketplace v3.3.0 downloads 11M GitHub Workflow Status github stars 1.7k contributors 44

All you need for Markdown (keyboard shortcuts, table of contents, auto preview and more).

Table of contents

- Features
- Keyboard shortcuts
- Table of contents
- List editing
- Print Markdown to HTML
- GitHub Flavored Markdown
- Math
- Auto completions
- Others
- Available Commands
- Keyboard Shortcuts
- Supported Settings
- FAQ
  - Q: Error "command 'markdown.extension.onXXXKey' not found"
  - Q: Which Markdown syntax is supported?
  - Q: This extension has overridden some of my key bindings (e.g. `Ctrl` + `B`, `Alt` + `C`)
- Changelog
- Latest Development Build
- Contributing
- Related

```
## Features

Keyboard shortcuts


<p><img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/gifs/toggle-bold.gif" alt="toggle bold gif" width="282p
<br>(Typo: multiple words)</p>


<p><img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/gifs/check-task-list.gif" alt="check task list" width="


See full key binding list in the keyboard shortcuts section
```

Table of contents

<p><img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/toc.png" alt="toc" width="305px"></p>

- The TOC is **automatically updated** on file save. To disable please change the `toc.updateOnSave` option.

- The **indentation type (tab or spaces)** of TOC can be configured per file. Find the setting in the right bottom corner of VS Code's

  Note: Be sure to also check the `list.indentationSize` option.

- To make TOC **compatible with GitHub or GitLab**, set option `slugifyMode` accordingly

- Three ways to **control which headings are present** in the TOC:

  i. Use `&lt;!-- omit in toc --&gt;` to ignore a specific heading in TOC

     (It can also be placed above a heading)

  ii. Use `toc.levels` setting.

  iii. You can also use the `toc.omittedFromToc` setting to omit some headings (and their subheadings) from TOC:

```
// In your settings.json
"markdown.extension.toc.omittedFromToc": {
  // Use a path relative to your workspace.
  "README.md": [
      "# Introduction",
      "## Also omitted",
  ],
  // Or an absolute path for standalone files.
  "/home/foo/Documents/todo-list.md": [
    "## Shame list (I'll never do these)",
  ]
}
```

     Note: Setext headings (underlined with `===` or `---`) can also be omitted, just put their `#` and `##` versions in the setting, res

- Easily add/update/remove **section numbering**

  <img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/gifs/section-numbers.gif" alt="section numbers" width="

- *In case you are seeing* ***unexpected TOC recognition****, you can add a `&lt;!-- no toc --&gt;` comment above the list.*

List editing

<p><img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/gifs/on-enter-key.gif" alt="on enter key" width="214px":

<p><img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/gifs/tab-backspace.gif" alt="on tab/backspace key" widtl

<p><img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/gifs/fix-marker.gif" alt="fix ordered list markers" widt

Note: By default, this extension tries to determine indentation size for different lists according to CommonMark Spec. If you prefe

Print Markdown to HTML

- Commands `Markdown: Print current document to HTML` and `Markdown: Print documents to HTML` (batch mode)

- **Compatible** with other installed Markdown plugins (e.g. Markdown Footnotes) The exported HTML should look the same as inside VSCode.

- Use comment `&lt;!-- title: Your Title --&gt;` to specify a title of the exported HTML.

- Plain links to `.md` files will be converted to `.html`.

- It's recommended to print the exported HTML to PDF with browser (e.g. Chrome) if you want to share your documents with others.

GitHub Flavored Markdown

- Table formatter

  <p><img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/gifs/table-formatter.gif" alt="table formatter" widt

  Note: The key binding is <kbd>Ctrl</kbd> + <kbd>Shift</kbd> + <kbd>I</kbd> on Linux. See Visual Studio Code Key Bindings.

- Task lists

Math

<p><img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/math.png" alt="math" width="544px"></p>

Please use Markdown+Math for dedicated math support. Be sure to disable `math.enabled` option of this extension.

Auto completions

Tip: also support the option `completion.root`

- Images/Files (respects option `search.exclude`)

  <p><img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/image-completions.png" alt="image completions" width

- Math functions (including option `katex.macros`)

  <p><img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/math-completions.png" alt="math completions" width="

- Reference links

  <p><img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/reference-link.png" alt="reference links" width="301

Others

- Paste link on selected text

  <p><img src="https://github.com/yzhang-gh/vscode-markdown/raw/master/images/gifs/paste-link.gif" alt="paste link" width="342px">

- Override "Open Preview" keybinding with "Toggle Preview", which means you can close preview using the same keybinding (<kbd>Ctrl

Available Commands

- Markdown All in One: Create Table of Contents
- Markdown All in One: Update Table of Contents
- Markdown All in One: Add/Update section numbers
- Markdown All in One: Remove section numbers
- Markdown All in One: Toggle code span
- Markdown All in One: Toggle code block
- Markdown All in One: Print current document to HTML
- Markdown All in One: Print documents to HTML
- Markdown All in One: Toggle math environment
- Markdown All in One: Toggle list
    - It will cycle through list markers (-, *, +, 1. and 1))

Keyboard Shortcuts

<details>
<summary>Table</summary>

Key	Command
<kbd>Ctrl</kbd>/<kbd>Cmd</kbd> + <kbd>B</kbd>	Toggle bold
<kbd>Ctrl</kbd>/<kbd>Cmd</kbd> + <kbd>I</kbd>	Toggle italic
<kbd>Ctrl</kbd>/<kbd>Cmd</kbd> + <kbd>Shift</kbd> + <kbd>]</kbd>	Toggle heading (uplevel)
<kbd>Ctrl</kbd>/<kbd>Cmd</kbd> + <kbd>Shift</kbd> + <kbd>[</kbd>	Toggle heading (downlevel)
<kbd>Ctrl</kbd>/<kbd>Cmd</kbd> + <kbd>M</kbd>	Toggle math environment
<kbd>Alt</kbd> + <kbd>C</kbd>	Check/Uncheck task list item
<kbd>Ctrl</kbd>/<kbd>Cmd</kbd> + <kbd>Shift</kbd> + <kbd>V</kbd>	Toggle preview
<kbd>Ctrl</kbd>/<kbd>Cmd</kbd> + <kbd>K</kbd> <kbd>V</kbd>	Toggle preview to side

</details>

Supported Settings

<details>
<summary>Table</summary>

Name	Default	Description
markdown.extension.completion.respectVscodeSearchExclude	true	Whether to consider search.exclude option when providing file path c
markdown.extension.completion.root		Root folder when providing file path completions (It takes effect
markdown.extension.italic.indicator	*	Use * or _ to wrap italic text
markdown.extension.katex.macros	{}	KaTeX macros e.g. { "\\name": "expansion", ... }

markdown.extension.list.indentationSize	adaptive	Use different indentation size for ordered and unordered list
markdown.extension.orderedList.autoRenumber	true	Auto fix list markers as you edits
markdown.extension.orderedList.marker	ordered	Or `one`: always use `1.` as ordered list marker
markdown.extension.preview.autoShowPreviewToSide	false	Automatically show preview when opening a Markdown file.
markdown.extension.print.absoluteImgPath	true	Convert image path to absolute path
markdown.extension.print.imgToBase64	false	Convert images to base64 when printing to HTML
markdown.extension.print.includeVscodeStylesheets	true	Whether to include VSCode's default styles
markdown.extension.print.onFileSave	false	Print to HTML on file save
markdown.extension.print.theme	light	Theme of the exported HTML
markdown.extension.print.validateUrls	true	Enable/disable URL validation when printing
markdown.extension.syntax.decorations	true	Add decorations to ~~strikethrough~~ and `code span`
markdown.extension.syntax.decorationFileSizeLimit	50000	Don't render syntax decorations if a file is larger than this size
markdown.extension.syntax.plainTheme	false	A distraction-free theme
markdown.extension.tableFormatter.enabled	true	Enable GFM table formatter
markdown.extension.toc.downcaseLink	true	Force the TOC links to be lowercase
markdown.extension.toc.slugifyMode	github	Slugify mode for TOC link generation (`vscode`, `github`, `gitlab` or `gitea`
markdown.extension.toc.omittedFromToc	{}	Lists of headings to omit by project file (e.g. `{ "README.md": ["# In`
markdown.extension.toc.levels	1..6	Control the heading levels to show in the table of contents.
markdown.extension.toc.orderedList	false	Use ordered list in the table of contents.
markdown.extension.toc.plaintext	false	Just plain text.
markdown.extension.toc.unorderedList.marker	-	Use `-`, `*` or `+` in the table of contents (for unordered list)
markdown.extension.toc.updateOnSave	true	Automatically update the table of contents on save.

</details>

FAQ

Q: Error "command 'markdown.extension.onXXXKey' not found"

In most cases, it is because VSCode needs a few seconds to load this extension when you open a Markdown file *for the first time*. (Yo

If you still see this "command not found" error after waiting for a long time, please try to restart VSCode (or reinstall this exte

<sup>1. uninstall this extension, <b>restart VSCode (important!)</b> and then reinstall</sup>

Q: Which Markdown syntax is supported?

- CommonMark
- Tables, strikethrough and task lists (from GitHub Flavored Markdown)
- Math support (from KaTeX)
- Front matter

For other Markdown syntax, you need to install the corresponding extensions from VSCode marketplace (e.g. Mermaid diagram, emoji, fo

Q: This extension has overridden some of my key bindings (e.g. <kbd>Ctrl</kbd> + <kbd>B</kbd>, <kbd>Alt</kbd> + <kbd>C</kbd>)

You can easily manage key bindings with VSCode's "Keyboard Shortcuts" page. (Commands provided by this extension have prefix `markdown`

Changelog

See CHANGELOG for more information.

Latest Development Build

Download it here, please click the latest passing event to download artifacts.

To install, execute `Extensions: Install from VSIX...` in the Command Palette (`ctrl + shift + p`)

Related

More extensions of mine

<=(Markdown PDF)=>

Markdown PDF

This extension converts Markdown files to pdf, html, png or jpeg files.

Japanese README

Table of Contents

<!-- TOC depthFrom:2 depthTo:2 updateOnSave:false -->

- Features
- Install
- Usage
- Extension Settings
- Options
- FAQ
- Known Issues
- Release Notes
- License
- Special thanks

<!-- /TOC -->

<div class="page"/>

Features

Supports the following features

- Syntax highlighting
- emoji
- markdown-it-checkbox
- markdown-it-container
- markdown-it-include
- PlantUML
  - markdown-it-plantuml
- mermaid

Sample files

- pdf
- html
- png
- jpeg

markdown-it-container

INPUT

::: warning *here be dragons* ::: ```

OUTPUT

```
<div class="warning">
<p><em>here be dragons</em></p>
</div>
```

markdown-it-plantuml

INPUT

```
@startuml
Bob -[#red]> Alice : hello
Alice -[#0000FF]->Bob : ok
@enduml
```

OUTPUT

PlantUML
PlantUML

markdown-it-include

Include markdown fragment files: `:[alternate-text](relative-path-to-file.md)`.

```
├── [plugins]
│  └── README.md
├── CHANGELOG.md
└── README.md
```

INPUT

```
README Content
:Plugins


:Changelog
```

OUTPUT

```
Content of README.md
Content of plugins/README.md


Content of CHANGELOG.md
```

mermaid

INPUT

~~~
```mermaid
stateDiagram
    [*] --> First
    state First {
        [*] --> second
        second --> [*]
    }
```
~~~

OUTPUT

mermaid
mermaid

Install

Chromium download starts automatically when Markdown PDF is installed and Markdown file is first opened with Visual Studio Code.

However, it is time-consuming depending on the environment because of its large size (~ 170Mb Mac, ~ 282Mb Linux, ~ 280Mb Win).

During downloading, the message `Installing Chromium` is displayed in the status bar.

If you are behind a proxy, set the `http.proxy` option to settings.json and restart Visual Studio Code.

If the download is not successful or you want to avoid downloading every time you upgrade Markdown PDF, please specify the installed Chrome or 'Chromium' with markdown-pdf.executablePath option.

Usage

Command Palette

1. Open the Markdown file
2. Press `F1` or `Ctrl+Shift+P`
3. Type `export` and select below

- `markdown-pdf: Export (settings.json)`
- `markdown-pdf: Export (pdf)`
- `markdown-pdf: Export (html)`
- `markdown-pdf: Export (png)`
- `markdown-pdf: Export (jpeg)`
- `markdown-pdf: Export (all: pdf, html, png, jpeg)`

usage1
usage1

Menu

1. Open the Markdown file
2. Right click and select below

- `markdown-pdf: Export (settings.json)`
- `markdown-pdf: Export (pdf)`
- `markdown-pdf: Export (html)`
- `markdown-pdf: Export (png)`
- `markdown-pdf: Export (jpeg)`
- `markdown-pdf: Export (all: pdf, html, png, jpeg)`

usage2
usage2

Auto convert

1. Add `"markdown-pdf.convertOnSave": true` option to **settings.json**
2. Restart Visual Studio Code
3. Open the Markdown file
4. Auto convert on save

Extension Settings

Visual Studio Code User and Workspace Settings

1. Select **File** > **Preferences** > **UserSettings or Workspace Settings**
2. Find markdown-pdf settings in the **Default Settings**
3. Copy `markdown-pdf.*` settings
4. Paste to the **settings.json**, and change the value

demo
demo

Options

List

Category	Option name	Configuration scope
Save options	markdown-pdf.type	
	markdown-pdf.convertOnSave	
	markdown-pdf.convertOnSaveExclude	
	markdown-pdf.outputDirectory	
	markdown-pdf.outputDirectoryRelativePathFile	
Styles options	markdown-pdf.styles	
	markdown-pdf.stylesRelativePathFile	
	markdown-pdf.includeDefaultStyles	

Category	Option name	Configuration scope
Syntax highlight options	markdown-pdf.highlight	
	markdown-pdf.highlightStyle	
Markdown options	markdown-pdf.breaks	
Emoji options	markdown-pdf.emoji	
Configuration options	markdown-pdf.executablePath	
Common Options	markdown-pdf.scale	
PDF options	markdown-pdf.displayHeaderFooter	resource
	markdown-pdf.headerTemplate	resource
	markdown-pdf.footerTemplate	resource
	markdown-pdf.printBackground	resource
	markdown-pdf.orientation	resource
	markdown-pdf.pageRanges	resource
	markdown-pdf.format	resource
	markdown-pdf.width	resource
	markdown-pdf.height	resource
	markdown-pdf.margin.top	resource
	markdown-pdf.margin.bottom	resource
	markdown-pdf.margin.right	resource
	markdown-pdf.margin.left	resource
PNG JPEG options	markdown-pdf.quality	
	markdown-pdf.clip.x	
	markdown-pdf.clip.y	
	markdown-pdf.clip.width	
	markdown-pdf.clip.height	
	markdown-pdf.omitBackground	
PlantUML options	markdown-pdf.plantumlOpenMarker	
	markdown-pdf.plantumlCloseMarker	
	markdown-pdf.plantumlServer	
markdown-it-include options	markdown-pdf.markdown-it-include.enable	
mermaid options	markdown-pdf.mermaidServer	

Save options

`markdown-pdf.type`

- Output format: pdf, html, png, jpeg
- Multiple output formats support
- Default: pdf

```
"markdown-pdf.type": [
  "pdf",
  "html",
  "png",
  "jpeg"
],
```

`markdown-pdf.convertOnSave`

- Enable Auto convert on save
- boolean. Default: false
- To apply the settings, you need to restart Visual Studio Code

`markdown-pdf.convertOnSaveExclude`

- Excluded file name of convertOnSave option

```
"markdown-pdf.convertOnSaveExclude": [
  "^work",
  "work.md$",
  "work|test",
  "[0-9][0-9][0-9][0-9]-work",
  "work\\test"  // All '\' need to be written as '\\' (Windows)
],
```

`markdown-pdf.outputDirectory`

- Output Directory
- All `\` need to be written as `\\` (Windows)

```
"markdown-pdf.outputDirectory": "C:\\work\\output",
```

- Relative path
  - If you open the `Markdown file`, it will be interpreted as a relative path from the file
  - If you open a `folder`, it will be interpreted as a relative path from the root folder
  - If you open the `workspace`, it will be interpreted as a relative path from the each root folder
  - See Multi-root Workspaces

```
"markdown-pdf.outputDirectory": "output",
```

- Relative path (home directory)
  - If path starts with `~`, it will be interpreted as a relative path from the home directory

```
"markdown-pdf.outputDirectory": "~/output",
```

- If you set a directory with a `relative path`, it will be created if the directory does not exist
- If you set a directory with an `absolute path`, an error occurs if the directory does not exist

`markdown-pdf.outputDirectoryRelativePathFile`

- If `markdown-pdf.outputDirectoryRelativePathFile` option is set to `true`, the relative path set with markdown-pdf.outputDirectory is interpreted as relative from the file
- It can be used to avoid relative paths from folders and workspaces
- boolean. Default: false

Styles options

`markdown-pdf.styles`

- A list of local paths to the stylesheets to use from the markdown-pdf
- If the file does not exist, it will be skipped
- All `\` need to be written as `\\` (Windows)

```
"markdown-pdf.styles": [
  "C:\\Users\\<USERNAME>\\Documents\\markdown-pdf.css",
  "/home/<USERNAME>/settings/markdown-pdf.css",
],
```

- Relative path
  - If you open the `Markdown file`, it will be interpreted as a relative path from the file
  - If you open a `folder`, it will be interpreted as a relative path from the root folder
  - If you open the `workspace`, it will be interpreted as a relative path from the each root folder

  - See Multi-root Workspaces

```
"markdown-pdf.styles": [
  "markdown-pdf.css",
],
```

- Relative path (home directory)
  - If path starts with `~`, it will be interpreted as a relative path from the home directory

```
"markdown-pdf.styles": [
  "~/.config/Code/User/markdown-pdf.css"
],
```

- Online CSS (https://xxx/xxx.css) is applied correctly for JPG and PNG, but problems occur with PDF #67

```
"markdown-pdf.styles": [
  "https://xxx/markdown-pdf.css"
],
```

`markdown-pdf.stylesRelativePathFile`

- If `markdown-pdf.stylesRelativePathFile` option is set to `true`, the relative path set with markdown-pdf.styles is interpreted as relative from the file
- It can be used to avoid relative paths from folders and workspaces
- boolean. Default: false

`markdown-pdf.includeDefaultStyles`

- Enable the inclusion of default Markdown styles (VSCode, markdown-pdf)
- boolean. Default: true

Syntax highlight options

`markdown-pdf.highlight`

- Enable Syntax highlighting
- boolean. Default: true

`markdown-pdf.highlightStyle`

- Set the style file name. for example: github.css, monokai.css ...
- file name list
- demo site : https://highlightjs.org/static/demo/

```
"markdown-pdf.highlightStyle": "github.css",
```

Markdown options

`markdown-pdf.breaks`

- Enable line breaks
- boolean. Default: false

Emoji options

`markdown-pdf.emoji`

- Enable emoji. EMOJI CHEAT SHEET
- boolean. Default: true

Configuration options

`markdown-pdf.executablePath`

- Path to a Chromium or Chrome executable to run instead of the bundled Chromium

- All `\` need to be written as `\\` (Windows)
- To apply the settings, you need to restart Visual Studio Code

```
"markdown-pdf.executablePath": "C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe"
```

Common Options

`markdown-pdf.scale`

- Scale of the page rendering
- number. default: 1

```
"markdown-pdf.scale": 1
```

PDF options

- pdf only. puppeteer page.pdf options

`markdown-pdf.displayHeaderFooter`

- Enable display header and footer
- boolean. Default: true

`markdown-pdf.headerTemplate`

`markdown-pdf.footerTemplate`

- HTML template for the print header and footer
- `<span class='date'></span>` : formatted print date
- `<span class='title'></span>` : markdown file name
- `<span class='url'></span>` : markdown full path name
- `<span class='pageNumber'></span>` : current page number
- `<span class='totalPages'></span>` : total pages in the document

```
"markdown-pdf.headerTemplate": "<div style=\"font-size: 9px; margin-left: 1cm;\"> <span class='title'></span></div> <div style=\"fo
```

```
"markdown-pdf.footerTemplate": "<div style=\"font-size: 9px; margin: 0 auto;\"> <span class='pageNumber'></span> / <span class='tot
```

`markdown-pdf.printBackground`

- Print background graphics
- boolean. Default: true

`markdown-pdf.orientation`

- Paper orientation
- portrait or landscape
- Default: portrait

`markdown-pdf.pageRanges`

- Paper ranges to print, e.g., '1-5, 8, 11-13'
- Default: all pages

```
"markdown-pdf.pageRanges": "1,4-",
```

`markdown-pdf.format`

- Paper format
- Letter, Legal, Tabloid, Ledger, A0, A1, A2, A3, A4, A5, A6
- Default: A4

```
"markdown-pdf.format": "A4",
```

`markdown-pdf.width`

`markdown-pdf.height`

- Paper width / height, accepts values labeled with units(mm, cm, in, px)
- If it is set, it overrides the markdown-pdf.format option

```
"markdown-pdf.width": "10cm",
"markdown-pdf.height": "20cm",
```

`markdown-pdf.margin.top`

`markdown-pdf.margin.bottom`

`markdown-pdf.margin.right`

`markdown-pdf.margin.left`

- Paper margins.units(mm, cm, in, px)

```
"markdown-pdf.margin.top": "1.5cm",
"markdown-pdf.margin.bottom": "1cm",
"markdown-pdf.margin.right": "1cm",
"markdown-pdf.margin.left": "1cm",
```

PNG JPEG options

- png and jpeg only. puppeteer page.screenshot options

`markdown-pdf.quality`

- jpeg only. The quality of the image, between 0-100. Not applicable to png images

```
"markdown-pdf.quality": 100,
```

`markdown-pdf.clip.x`

`markdown-pdf.clip.y`

`markdown-pdf.clip.width`

`markdown-pdf.clip.height`

- An object which specifies clipping region of the page
- number

```
//  x-coordinate of top-left corner of clip area
"markdown-pdf.clip.x": 0,
// y-coordinate of top-left corner of clip area
"markdown-pdf.clip.y": 0,


// width of clipping area
"markdown-pdf.clip.width": 1000,


// height of clipping area
"markdown-pdf.clip.height": 1000,
```

`markdown-pdf.omitBackground`

- Hides default white background and allows capturing screenshots with transparency
- boolean. Default: false

PlantUML options

`markdown-pdf.plantumlOpenMarker`

- Oppening delimiter used for the plantuml parser.
- Default: @startuml

`markdown-pdf.plantumlCloseMarker`

- Closing delimiter used for the plantuml parser.
- Default: @enduml

`markdown-pdf.plantumlServer`

- Plantuml server. e.g. http://localhost:8080
- Default: http://www.plantuml.com/plantuml
- For example, to run Plantuml Server locally #139 : `docker run -d -p 8080:8080 plantuml/plantuml-server:jetty` plantuml/plantuml-server - Docker Hub

markdown-it-include options

`markdown-pdf.markdown-it-include.enable`

- Enable markdown-it-include.
- boolean. Default: true

mermaid options

`markdown-pdf.mermaidServer`

- mermaid server
- Default: https://unpkg.com/mermaid/dist/mermaid.min.js

FAQ

How can I change emoji size ?

1. Add the following to your stylesheet which was specified in the markdown-pdf.styles

```
.emoji {
  height: 2em;
}
```

Auto guess encoding of files

Using `files.autoGuessEncoding` option of the Visual Studio Code is useful because it automatically guesses the character code. See files.autoGuessEncoding

```
"files.autoGuessEncoding": true,
```

Output directory

If you always want to output to the relative path directory from the Markdown file.

For example, to output to the "output" directory in the same directory as the Markdown file, set it as follows.

```
"markdown-pdf.outputDirectory" : "output",
"markdown-pdf.outputDirectoryRelativePathFile": true,
```

Page Break

Please use the following to insert a page break.

```
<div class="page"/>
```

Known Issues

`markdown-pdf.styles` option

- Online CSS (https://xxx/xxx.css) is applied correctly for JPG and PNG, but problems occur with PDF. #67

Release Notes

<================(Open in External App)================>

Open in External App

Open file with external application in VSCode.

Version Installs Downloads Rating Star Trending Monthly Percentage of issues still open

build passing dependencies Status devDependencies Status

Motivation

VSCode is a very excellent editor, but sometime I prefer to use external application to work with some files. For example, I like to use typora to edit the markdown files. Usually, I will right click to the file, and select `Reveal in File Explorer` , then open the file using external application.

But, with this extension, you can do it more simply. Just right click to the file, and select `Open in External App`, that file would be opened by system default application. You can also use this way to open `.psd` files with photoshop, `.html` files with browser, and so on...

Installation

1. Execute `Extensions: Install Extensions` command from Command Palette.
2. Type `YuTengjing.open-in-external-app` into the search form and install.

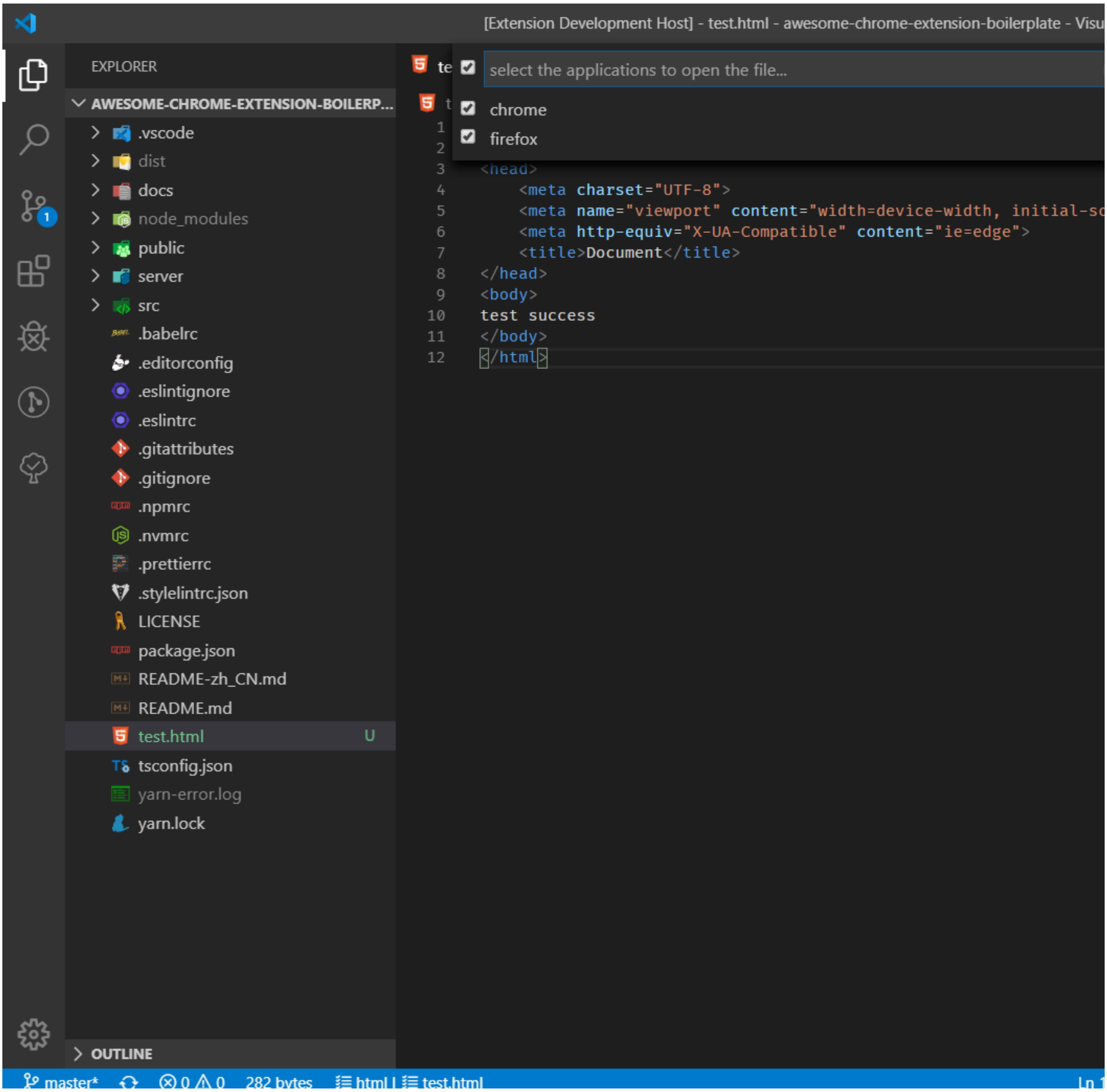
Read the extension installation guide for more details.

Configuration

Via custom configuration, you can make extensions more powerful. For example, to see the rendering differences, You can open one HTML in chrome and Firefox at the same time.

Example configuration:

```
{
    "openInExternalApp.openMapper": [
        {
            // represent file extension name
            "extensionName": "html",
            // the external applications to open the file which extension name is html
            "apps": [
                // openCommand can be shell command or the complete executable application path
                // title will be shown in the drop list if there are several apps
                {
                    "title": "chrome",
                    "openCommand": "C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe"
                },
                {
                    "title": "firefox",
                    "openCommand": "C:\\Program Files\\Firefox Developer Edition\\firefox.exe",
                    // open in firefox under private mode
                    "args": ["-private-window"]
                }
            ]
        },
        {
            "extensionName": "tsx",
            // apps can be Object array or just is openCommand
            // the code is command you can access from shell
            "apps": "code"
        },
        {
            "extensionName": "psd",
            "apps": "/path/to/photoshop.exe"
        }
    ]
}
```



☑ select the applications to open the file...

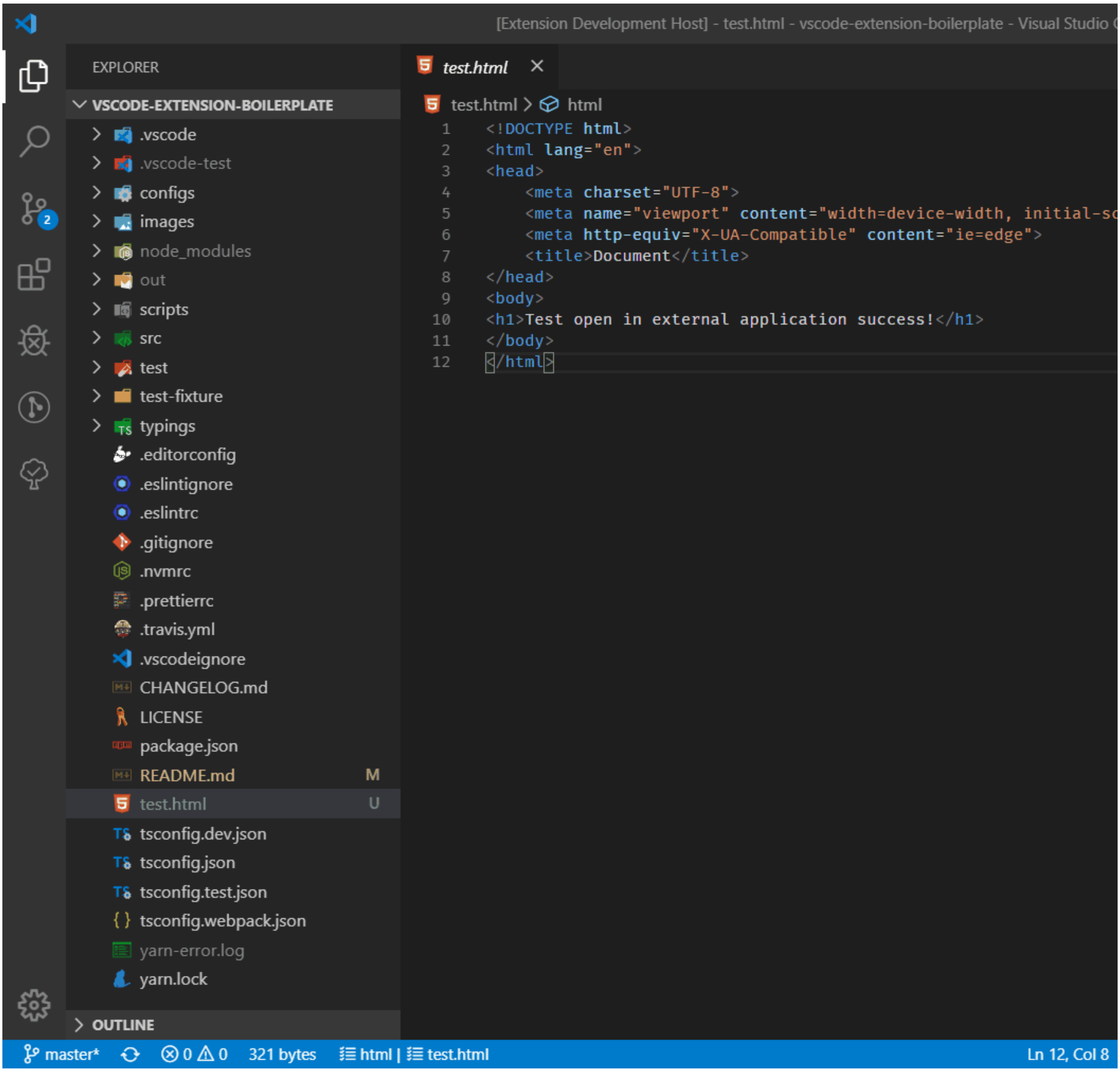
☑ chrome

☑ firefox

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-sc
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
test success
</body>
</html>
```

open multiple

In VSCode, Right-clicking is different from right-clicking while holding `alt` key. If you just right click the file, you will see the command `Open in External App`, but if you right click file while holding `alt` key, you will see the command `Open in Multiple External Apps`.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-sc
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
<h1>Test open in external application success!</h1>
</body>
</html>
```

usage

:loudspeaker: Limits

This extension use two ways to open file in external applications.

1. Node package: open

This package has one limit that can't open a file which is also made by electron. For example, you can't open `md` file in `typora` using this package. The `openCommand` , `args` configuration item is also supported by this package. When `isElectronApp: false` (by default), extension will use this way.

2. VSCode extension API: `vscode.env.openExternal(target: Uri)`

This API has one limit that can't open file path which includes `Non-ascii` characters, but support open file in application which is made by electron. This API can only pass one argument `target` , so `openCommand` and `args` configuration item is not work.

If you want to open file in application which is made by electron, you can choose one of two ways:

1. don not config it in VSCode settings, and set the default application of your operation system to open that file format.
2. using `isElectronApp` option:

`javascript { "extensionName": "md", "isElectronApp": true, }`

multiple apps example:

`javascript { "extensionName": "md", "apps": [ { "title": "typora", "isElectronApp": true, // following config item is not work // "openCommand": "/path/to/typora.exe", // "args": ["--slient"] }, { "title": "idea", "extensionName": "md", "openCommand": "/path/to/idea.exe", "args": ["--slient"], } ] }`

Change Case Extension for Visual Studio Code

A wrapper around node-change-case for Visual Studio Code. Quickly change the case of the current selection or current word.

If only one word is selected, the `extension.changeCase.commands` command gives you a preview of each option:

change-case-preview
change-case-preview

`change-case` also works with multiple cursors:

change-case-multi
change-case-multi

Note: Please read the documentation on how to use multiple cursors in Visual Studio Code.

Install

Launch VS Code Quick Open (Ctrl/Cmd+P), paste the following command, and press enter.

```
ext install change-case
```

Commands

- `extension.changeCase.commands` : List all Change Case commands, with preview if only one word is selected
- `extension.changeCase.camel` : Change Case 'camel': Convert to a string with the separators denoted by having the next letter capitalised
- `extension.changeCase.constant` : Change Case 'constant': Convert to an upper case, underscore separated string
- `extension.changeCase.dot` : Change Case 'dot': Convert to a lower case, period separated string
- `extension.changeCase.kebab` : Change Case 'kebab': Convert to a lower case, dash separated string (alias for param case)
- `extension.changeCase.lower` : Change Case 'lower': Convert to a string in lower case
- `extension.changeCase.lowerFirst` : Change Case 'lowerFirst': Convert to a string with the first character lower cased
- `extension.changeCase.no` : Convert the string without any casing (lower case, space separated)
- `extension.changeCase.param` : Change Case 'param': Convert to a lower case, dash separated string
- `extension.changeCase.pascal` : Change Case 'pascal': Convert to a string denoted in the same fashion as camelCase, but with the first letter also capitalised
- `extension.changeCase.path` : Change Case 'path': Convert to a lower case, slash separated string
- `extension.changeCase.sentence` : Change Case 'sentence': Convert to a lower case, space separated string
- `extension.changeCase.snake` : Change Case 'snake': Convert to a lower case, underscore separated string
- `extension.changeCase.swap` : Change Case 'swap': Convert to a string with every character case reversed
- `extension.changeCase.title` : Change Case 'title': Convert to a space separated string with the first character of every word upper cased
- `extension.changeCase.upper` : Change Case 'upper': Convert to a string in upper case
- `extension.changeCase.upperFirst` : Change Case 'upperFirst': Convert to a string with the first character upper cased

Support

Create an issue, or ping [@waynemaurer](https://twitter.com/waynemaurer) on Twitter.

<================(vscode-js-console-utils)================>

vscode-js-console-utils

Easily insert and remove console.log statements, by [@whtouche](https://twitter.com/whtouche)

Installing

This extension is available for free in the Visual Studio Code Marketplace

MIT License # <===============(Node.js Extension Pack)===============> # Node.js Extension Pack

VS Code comes with a ton of features for Node.js development out of the box. This extension pack adds more!

These are some of my favorite extensions to make Node.js development easier and fun.

Extensions Included

- ES Lint - Integrates ESLint into VS Code.
- npm - Run npm scripts from the command palatte and validate the installed modules defined in `package.json`.
- JavaScript (ES6) Snippets - Adds code snippets for JavaScript development in ES6 syntax.
- Search node_modules - Quickly search for node modules in your project.
- NPM IntelliSense - Adds IntelliSense for npm modules in your code.
- Path IntelliSense - Autocompletes filenames in your code.

Want to see your extension added?

Open a PR and I'd be happy to take a look.

Enjoy! # <===============(Refactor CSS)===============> # Refactor CSS

Install via VS Code Marketplace

Helps you identify reoccurring CSS class name combinations in your markup. This is especially useful if you are working with an utility-first CSS framework like TailwindCSS, Tachyons,...

Hovering over CSS classes reveals infos.

Features

Class names are highlighted if they have more than 3 unique classes and this combination of classes appears more than 3 times in the current document. These numbers can be changed in the settings.

Hovering over classes highlights all other elements with the same combination of classes.

The order of the class names does not matter.

Release Notes

See CHANGELOG.

Roadmap

- [x] Parse whole workspace, not only current document.
- [ ] Provide text selection of all occurrences for easy refactoring
- [x] Add settings for the user (limits)

<===============(Treedbox Javascript)===============>

Treedbox JavaScript - Visual Studio Marketplace

> Extension for Visual Studio Code - JavaScript Snippets for fast development

How fast do you want to be to transform an idea into code?

VS Code Javascript extension

> It's not an extension that forces you to remember some "custom prefix" to call the snippets. **Treedbox Javascript** was designed to serve snippets in the most logical way, delivering the most commons code structures and combinations to make your combo's code speedy and powerful.

Javascript snippets Visual Studio Code Extension for fast development.

By: Jonimar Marques Policarpo at Treedbox

Screenshots:

Screenshot 1: Using snippet for fetch DELETE:

Screenshot 1
Screenshot 1

Screenshot 2: Accessing all fetch snippets:

Screenshot 2
Screenshot 2

Install

Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter

`ext install treedbox.treedboxjavascript`

Or search for `treedbox` in the VS Code Extensions tab.

Pages

Treedbox
Treedbox

GitHub: https://github.com/treedbox/treedboxjavascript

VS Code extension: https://marketplace.visualstudio.com/items?itemName=treedbox.treedboxjavascript

Features

Version 1.0.0: **501** javascript snippets!

- Snippets for `fetchGET`, `fetchPOST`, `fetchPUT`, `fetchDELETE`;
- Include: URL interface https://developer.mozilla.org/docs/Web/API/URL `URLcreateObjectURL`, `URLrevokeObjectURL`;
- Snippets without semicolons; [YouTube: Semicolons cannot save you! - FunFunFunction #9](https://www.youtube.com/watch?v=Qlr-FGbhKaI), Article: Semicolons in JavaScript: A preference;
- Fixed: typo with `parseInt`.

All Math properties and methods

Including an extra complete functions as `MathRandomCompleteFunc` or `randomCompleteFunc`:

```
const random = (min,max) => Math.floor(Math.random() * (max - min + 1)) + min
```

and many others **Extras** that you will find out when you need it :)

How to use

Example:

in a `.js` file or in a `.html` file, between the HTML tag `<script></script>`, type: `fetchblob` and press "**Tab/Enter**" to generate:

```
fetch(url)
  .then(response => response.blob())
  .then(data =>{
      console.log('data:',data)
  }).catch(error => console.log('ERROR:',error))
```

Like `fetchblob`, you have a lot of snippets to help you with `import`, `forEach`, `map`, `generator` and so on.

https://github.com/treedbox/treedboxjavascript/

<================(pdf)================>

pdf

Display pdf in VSCode.

screenshot
screenshot

Contribute

Upgrade PDF.js

1. Download latest Prebuilt.
2. Extract the ZIP file.
3. Overwrite ./lib/* by extracted directories.

- If lib/web/viewer.html has changes, apply these changes to HTML template at pdfPreview.ts.

1. To not use sample pdf.

- Remove sample pdf called `compressed.tracemonkey-pldi-09.pdf`.
- Remove code about using sample pdf from lib/web/viewer.js. `js defaultUrl: { value: "", // "compressed.tracemonkey-pldi-09.pdf" kind: OptionKind.VIEWER },`

Change log

See CHANGELOG.md.

License

Please see LICENSE

<================(Superpowers)================>

Superpowers

This VSCode extension gives you superpowers in the form of JavaScript expressions, that can be used to map or sort multi-selections.

Preview

Superpowers Superpresets

Features

- Perform map operations on a selection or multiple selections
- Perform sort operations on a multiple selection
- Generate text via JavaScript expressions
- Save your expressions as presets for easy access

- Support for dynamic snippets / completions

Map/Sort operation usage

1. Select some text. Optionally, use `Cmd` to select multiple regions.
2. Press `Cmd` + `Shift` + `P` to open the command palette.
3. Type "super" to show a list of available commands;

   > ### Custom map function

   Type a JavaScript map callback function and press enter. Your function will get applied to each selection.

   > ### Map Presets

   Pick a preset to use as the map function.

   > ### Custom sort function

   Type a JavaScript sort comparison function and press enter to sort the selections.

   > ### Sort Presets

   Pick a preset to use as the sort function.

   > ### Custom reduce function

   Type a JavaScript reduce function and press enter to output the result after the selections.

Dynamic snippets / completion

Currently, only plaintext and markdown documents are supported, and only two snippets are available;

- `RT` - Inserts the current time, rounded to 15 minutes
- `TD` - Calculates the time delta between two times

Type a snippet, and the autocompletion menu should appear.

Extension Settings

This extension contributes the following settings:

- `superpowers.mapPresets` : List of map presets as an array of objects.

Example: `json { "name": "replace with index", "function": "(_, i) => i + 1" }` * `superpowers.sortPresets` : List of sort presets as an array of objects.

Example: `json { "name": "sort by codepoint", "function": "(a, b) => a.charCodeAt(0) - b.charCodeAt(0)" }`

<================(Readme Pattern)================>

Readme Pattern

> A VSCode extension that generates README.md files

version v1.3.0 downloads 16k installs 11k

Screenshot
Screenshot

Features

- Includes 4 readme templates: Bot, Hackathon, Minimal, Standard, based on The-Documentation-Compendium
- Creates README.md with context menu
- Supports package.json and composer.json
- Creates project name by reading config

Logo

Created my free logo at LogoMakr.com

<===============(Markdown Paste)===============>

Markdown Paste

Smartly paste for Markdown.

Support Mac/Windows/Linux!.

markdown paste demo
markdown paste demo

Requirements

- 'xclip' command be required (Linux)
- 'powershell' command be required (Win32)
- 'pbpaste' command be required (Mac)

Features

- Paste smart

  Smartly paste in markdown by pressing 'Ctrl+Alt+V' ('Cmd+Alt+V' on Mac)

  - If you paste an image, the extension will create an new file for the image and insert link code to Markdown.
  - If you paste a text, it will test the text with customize regex, and replace matched content by regex.
  - If you paste a text contain HTML tag, it will try to convert the HTML content to Markdown.
  - If you paste a rich text, it will try to convert the rich text to Markdown.(Linux only)

- Download file

  Use `Markdown Download` command (Linux or Windows: `Ctrl+Alt+D`, Mac: `Cmd+Alt+D`) to download file and insert link code into Markdown.

- Ruby tag

  Also if you want to write article for learning asian language like Chinese or Japanese, ruby tag(for example:聡明(Cōngming)) may be useful. Now a ruby tag snippet are prepare for you, select some text and press 'Ctrl+Alt+T'.

  ```
  <ruby>聡明<rp>(</rp><rt>pronunciation</rt><rp>)</rp></ruby>
  ```

  This extension will not get the pronunciation for you in this version. You have to replace 'pronunciation' by yourself.

- Insert latex math symbol and emoji

  You can insert latex math symbol and emoji to any text file, such as Julia source file.

  press 'Ctrl+Alt+\' or input "Insert latex math symbol" in vscode command panel, then input latex symbol name and choose symbol you want.

Config

- `MarkdownPaste.path`

  The folder path that image will be saved. Support absolute path and relative path and the following predefined variables

  - ${workspaceRoot} - the path of the folder opened in VS Code
  - ${fileBasename} - the current opened file's basename
  - ${fileBasenameNoExtension} - the current opened file's basename with no file extension

  - ${fileExtname} - the current opened file's extension
  - ${fileDirname} - the current opened file's dirname

  Default value is `./`, mean save image in the folder contains current file.

- `MarkdownPaste.silence`

  enable/disable showing confirm box while paste image. Set this config option to `true`, filename confirm box will not be shown while paste image.

  Default value is `false`

- `MarkdownPaste.enableImgTag`

  enable/disable using HTML img tag with width and height for pasting image. If this option be enabled, you can input width and height by using `<filepath>[,width,height]` in filename confirm input box. for example input `\abc\filename.png,200,100`, then `<img src='\abc\filename.png' width='200' height='100' />` will be inserted. Note that if `MarkdownPaste.silence` be enabled, this option will be not work.

  Default value is `true`

- `MarkdownPaste.rules`

  If you want to define your own regex to parse and replace content for pasting text. You can fill the following JSON, and set it to this option.

  ```
  [{
      // rule 1
      "regex": "(https?:\/\/.*)", // your javascript style regex
      "options": "ig",            // regex option
      "replace": "[]($1)"         // replace string
  },
  {
      // rule 2
      "regex": "(https?:\/\/.*)", // your javascript style regex
      "options": "ig",            // regex option
      "replace": "[]($1)"         // replace string
  },
  ...
  ]
  ```

  The extension will try to test text content by regex defined in this option, if matched it whill replace content by using the TypeScript function string.replace().

  Default value is

  ```
  [{
      "regex": "^(?:https?:\/\/)?(?:(?:(?:www\\.?)?youtube\\.com(?:\/(?:(?:watch\\?.*?v=([^&\\s]+).*)|))?))",
      "options": "g",
      "replace": "[![](https://img.youtube.com/vi/$1/0.jpg)](https://www.youtube.com/watch?v=$1)"
  },
  {
      "regex": "^(https?:\/\/.*)",
      "options": "ig",
      "replace": "[]($1)"
  }]
  ```

Format

File name format

If you selected some text in editor, then extension will use it as the image file name. If not the image will be saved in this format: "Y-MM-DD-HH-mm-ss.png".

File link format

When you editing a markdown, it will pasted as markdown image link format `![](imagePath)`, the imagePath will be resolve to relative path of current markdown file. In other file, it just paste the image's path.

<===============(Spelling Checker for Visual Studio Code)===============>

Spelling Checker for Visual Studio Code

A basic spell checker that works well with camelCase code.

The goal of this spell checker is to help catch common spelling errors while keeping the number of false positives low.

Support Further Development

PayPal Donations

Functionality

Load a TypeScript, JavaScript, Text, etc. file. Words not in the dictionary files will have a squiggly underline.

Example

Example
Example

Suggestions

Example
Example

To see the list of suggestions:

After positioning the cursor in the word, any of the following should display the list of suggestions: - Click on the 💡 (lightbulb) in the left hand margin. - `Quick Fix` Editor action command: - Mac: `⌘` + `.` or `Cmd` + `.` - PC: `Ctrl` + `.`

Install

Open up VS Code and hit `F1` and type `ext` select install and type `code-spell-checker` hit enter and reload window to enable.

Supported Languages

- English (US)
- English (GB) - turn on by changing `"cSpell.language": "en"` to `"cSpell.language": "en-GB"`

Add-On Specialized Dictionaries

- Medical Terms
- Fullstack

Enabled File Types

- AsciiDoc
- C, C++
- C#
- css, less, scss
- Elixir
- Go
- Html
- Java
- JavaScript
- JSON / JSONC
- LaTex

- Markdown
- PHP
- PowerShell
- Pug / Jade
- Python
- reStructuredText
- Rust
- Scala
- Text
- TypeScript
- YAML

Enable / Disable File Types

To *Enable* or *Disable* spell checking for a file type:

1. Click on the Spell Checker status in the status bar:

Spell Checker Status Bar

2. On the Info screen, click the ***Enable*** link.

Spell Checker Information Window

How it works with camelCase

The concept is simple, split camelCase words before checking them against a list of known English words. * camelCase -> camel case * HTMLInput -> html input -- Notice that the `I` is associated with `Input` and not `HTML` * snake_case_words -> snake case words * camel2snake -> camel snake -- (the 2 is ignored)

Special case will ALL CAPS words

There are a few special cases to help will common spelling practices for ALL CAPS words.

Trailing `s`, `ing`, `ies`, `es`, `ed` are kept with the previous word.

- CURLs -> curls -- trailing `s`
- CURLedRequest -> curled request -- trailing `ed`

Things to note

- This spellchecker is case insensitive. It will not catch errors like english which should be English.
- The spellchecker uses a local word dictionary. It does not send anything outside your machine.
- The words in the dictionary can and do contain errors.
- There are missing words.
- Only words longer than 3 characters are checked. "jsj" is ok, while "jsja" is not.
- All symbols and punctuation are ignored.

In Document Settings

It is possible to add spell check settings into your source code. This is to help with file specific issues that may not be applicable to the entire project.

All settings are prefixed with `cSpell:` or `spell-checker:`.

- `disable` -- turn off the spell checker for a section of code.
- `enable` -- turn the spell checker back on after it has been turned off.
- `ignore` -- specify a list of words to be ignored.
- `words` -- specify a list of words to be considered correct and will appear in the suggestions list.
- `ignoreRegExp` -- Any text matching the regular expression will NOT be checked for spelling.
- `includeRegExp` -- Only text matching the collection of includeRegExp will be checked.
- `enableCompoundWords` / `disableCompoundWords` -- Allow / disallow words like: "stringlength".

Enable / Disable checking sections of code

It is possible to disable / enable the spell checker by adding comments to your code.

Disable Checking

- `/* cSpell:disable */`
- `/* spell-checker: disable */`
- `/* spellchecker: disable */`
- `/* cspell: disable-line */`
- `/* cspell: disable-next-line */`

Enable Checking

- `/* cSpell:enable */`
- `/* spell-checker: enable */`
- `/* spellchecker: enable */`

Example

```
// cSpell:disable
const wackyWord = ['zaallano', 'wooorrdd', 'zzooommmmmmmm'];
/* cSpell:enable */
// Nest disable / enable is not Supported


// spell-checker:disable
// It is now disabled.


var liep = 1;


/* cspell:disable */
// It is still disabled


// cSpell:enable
// It is now enabled


const str = "goededag";  // <- will be flagged as an error.


// spell-checker:enable <- doesn't do anything


// cSPELL:DISABLE <-- also works.


// if there isn't an enable, spelling is disabled till the end of the file.
const str = "goedemorgen";  // <- will NOT be flagged as an error.
```

Ignore

Ignore allows you the specify a list of words you want to ignore within the document.

```
// cSpell:ignore zaallano, wooorrdd
// cSpell:ignore zzooommmmmmmm
const wackyWord = ['zaallano', 'wooorrdd', 'zzooommmmmmmm'];
```

Note: words defined with `ignore` will be ignored for the entire file.

Words

The words list allows you to add words that will be considered correct and will be used as suggestions.

```
// cSpell:words woorxs sweeetbeat
const companyName = 'woorxs sweeetbeat';
```

Note: words defined with `words` will be used for the entire file.

Enable / Disable compound words

In some programing language it is common to glue words together.

```
// cSpell:enableCompoundWords
char * errormessage;  // Is ok with cSpell:enableCompoundWords
int    errornumber;   // Is also ok.
```

Note: Compound word checking cannot be turned on / off in the same file. The last setting in the file determines the value for the entire file.

Excluding and Including Text to be checked.

By default, the entire document is checked for spelling. `cSpell:disable` / `cSpell:enable` above allows you to block off sections of the document. `ignoreRegExp` and `includeRegExp` give you the ability to ignore or include patterns of text. By default the flags `gim` are added if no flags are given.

The spell checker works in the following way: 1. Find all text matching `includeRegExp` 2. Remove any text matching `excludeRegExp` 3. Check the remaining text.

Exclude Example

```
// cSpell:ignoreRegExp 0x[0-9a-f]+     -- will ignore c style hex numbers
// cSpell:ignoreRegExp /0x[0-9A-F]+/g  -- will ignore upper case c style hex numbers.
// cSpell:ignoreRegExp g{5} h{5}       -- will only match ggggg, but not hhhhh or 'ggggg hhhhh'
// cSpell:ignoreRegExp g{5}|h{5}       -- will match both ggggg and hhhhh
// cSpell:ignoreRegExp /g{5} h{5}/     -- will match 'ggggg hhhhh'
/* cSpell:ignoreRegExp /n{5}/          -- will NOT work as expected because of the ending comment -> */
/*
   cSpell:ignoreRegExp /q{5}/          -- will match qqqqq just fine but NOT QQQQQ
*/
// cSpell:ignoreRegExp /[^\s]{40,}/    -- will ignore long strings with no spaces.
// cSpell:ignoreRegExp Email           -- this will ignore email like patterns -- see Predefined RegExp expressions
var encodedImage = 'HR+cPzr7XGAOJNurPL0G8I2kU0UhKcqFssoKvFTR7z0T3VJfK37vS025uKroHfJ9nA6WWbHZ/ASn...';
var email1 = 'emailaddress@myfancynewcompany.com';
var email2 = '<emailaddress@myfancynewcompany.com>';
```

Note: ignoreRegExp and includeRegExp are applied to the entire file. They do not start and stop.

Include Example

In general you should not need to use `includeRegExp`. But if you are mixing languages then it could come in helpful.

```
# cSpell:includeRegExp #.*
# cSpell:includeRegExp ("""|''')[^\1]*\1
# only comments and block strings will be checked for spelling.
def sum_it(self, seq):
    """This is checked for spelling"""
    variabele = 0
    alinea = 'this is not checked'
    for num in seq:
        # The local state of 'value' will be retained between iterations
        variabele += num
        yield variabele
```

Predefined RegExp expressions

Exclude patterns

- `Urls` [1] -- Matches urls
- `HexDigits` -- Matches hex digits: `/^x?[0-1a-f]+$/i`
- `HexValues` -- Matches common hex format like #aaa, 0xfeef, \u0134
- `EscapeCharacters` [1] -- matches special characters: '\n', '\t' etc.
- `Base64` [1] -- matches base64 blocks of text longer than 40 characters.
- `Email` -- matches most email addresses.

Include Patterns

- `Everything` [1] -- By default we match an entire document and remove the excludes.
- `string` -- This matches common string formats like '...', "...", and `...`
- `CStyleComment` -- These are C Style comments /* */ and //
- `PhpHereDoc` -- This matches PHPHereDoc strings.

[1] These patterns are part of the default include/exclude list for every file.

Customization

The spell checker configuration can be controlled via VS Code preferences or `cspell.json` configuration file.

Order of precedence: 1. Workspace Folder `cspell.json` 1. Workspace Folder `.vscode/cspell.json` 1. VS Code Preferences `cSpell` section.

Adding words to the Workspace Dictionary

You have the option to add you own words to the workspace dictionary. The easiest, is to put your cursor on the word you wish to add, when you lightbulb shows up, hit `Ctrl+.` (windows) / `Cmd+.` (Mac). You will get a list of suggestions and the option to add the word.

You can also type in a word you want to add to the dictionary: `F1` `add word` -- select `Add Word to Dictionary` and type in the word you wish to add.

cspell.json

Words added to the dictionary are placed in the `cspell.json` file in the *workspace* folder. Note, the settings in `cspell.json` will override the equivalent cSpell settings in VS Code's `settings.json`.

Example *cspell.json* file

```
// cSpell Settings
{
    // Version of the setting file.  Always 0.1
    "version": "0.1",
    // language - current active spelling language
    "language": "en",
    // words - list of words to be always considered correct
    "words": [
        "mkdirp",
        "tsmerge",
        "githubusercontent",
        "streetsidesoftware",
        "vsmarketplacebadge",
        "visualstudio"
    ],
    // flagWords - list of words to be always considered incorrect
    // This is useful for offensive words and common spelling errors.
    // For example "hte" should be "the"
    "flagWords": [
        "hte"
    ]
}
```

VS Code Configuration Settings

```
    //-------- Code Spell Checker Configuration --------
    // The Language local to use when spell checking. "en", "en-US" and "en-GB" are currently supported by default.
    "cSpell.language": "en",

  <span class="co">// Controls the maximum number of spelling errors per document.</span>
  <span class="st">"cSpell.maxNumberOfProblems"</span>: <span class="dv">100</span>,

  <span class="co">// Controls the number of suggestions shown.</span>
  <span class="st">"cSpell.numSuggestions"</span>: <span class="dv">8</span>,

  <span class="co">// The minimum length of a word before checking it against a dictionary.</span>
  <span class="st">"cSpell.minWordLength"</span>: <span class="dv">4</span>,

  <span class="co">// Specify file types to spell check.</span>
  <span class="st">"cSpell.enabledLanguageIds"</span>: [
      <span class="st">"csharp"</span>,
      <span class="st">"go"</span>,
      <span class="st">"javascript"</span>,
      <span class="st">"javascriptreact"</span>,
      <span class="st">"markdown"</span>,
      <span class="st">"php"</span>,
      <span class="st">"plaintext"</span>,
      <span class="st">"typescript"</span>,
      <span class="st">"typescriptreact"</span>,
      <span class="st">"yml"</span>
  ],

  <span class="co">// Enable / Disable the spell checker.</span>
  <span class="st">"cSpell.enabled"</span>: <span class="kw">true</span>,

  <span class="co">// Display the spell checker status on the status bar.</span>
  <span class="st">"cSpell.showStatus"</span>: <span class="kw">true</span>,

  <span class="co">// Words to add to dictionary for a workspace.</span>
  <span class="st">"cSpell.words"</span>: [],

  <span class="co">// Enable / Disable compound words like 'errormessage'</span>
  <span class="st">"cSpell.allowCompoundWords"</span>: <span class="kw">false</span>,

  <span class="co">// Words to be ignored and not suggested.</span>
  <span class="st">"cSpell.ignoreWords"</span>: [<span class="st">"behaviour"</span>],

  <span class="co">// User words to add to dictionary.  Should only be in the user settings.</span>
  <span class="st">"cSpell.userWords"</span>: [],

  <span class="co">// Specify paths/files to ignore.</span>
  <span class="st">"cSpell.ignorePaths"</span>: [
      <span class="st">"node_modules"</span>,        <span class="co">// this will ignore anything the node_modules directory</span>
      <span class="st">"**/node_modules"</span>,     <span class="co">// the same for this one</span>
      <span class="st">"**/node_modules/**"</span>,  <span class="co">// the same for this one</span>
      <span class="st">"node_modules/**"</span>,     <span class="co">// Doesn't currently work due to how the current working directory is determined.<
      <span class="st">"vscode-extension"</span>,    <span class="co">//</span>
      <span class="st">".git"</span>,                <span class="co">// Ignore the .git directory</span>
      <span class="st">"*.dll"</span>,               <span class="co">// Ignore all .dll files.</span>
      <span class="st">"**/*.dll"</span>             <span class="co">// Ignore all .dll files</span>
  ],

  <span class="co">// flagWords - list of words to be always considered incorrect</span>
  <span class="co">// This is useful for offensive words and common spelling errors.</span>
  <span class="co">// For example "hte" should be "the"`</span>
  <span class="st">"cSpell.flagWords"</span>: [<span class="st">"hte"</span>],

  <span class="co">// Set the delay before spell checking the document. Default is 50.</span>
  <span class="st">"cSpell.spellCheckDelayMs"</span>: <span class="dv">50</span>,</code></pre>
```

Dictionaries

The spell checker includes a set of default dictionaries.

General Dictionaries

- **wordsEn** - Derived from Hunspell US English words.
- **wordsEnGb** - Derived from Hunspell GB English words.
- **companies** - List of well known companies
- **softwareTerms** - Software Terms and concepts like "coroutine", "debounce", "tree", etc.
- **misc** - Terms that do not belong in the other dictionaries.

Programming Language Dictionaries

- **typescript** - keywords for Typescript and Javascript
- **node** - terms related to using nodejs.
- **php** - *php* keywords and library methods
- **go** - *go* keywords and library methods
- **python** - *python* keywords

- **powershell** - *powershell* keywords
- **html** - *html* related keywords
- **css** - *css*, *less*, and *scss* related keywords

Miscellaneous Dictionaries

- **fonts** - long list of fonts - to assist with *css*

Based upon the programming language, different dictionaries will be loaded.

Here are the default rules: "*" matches any language. `"local"` is used to filter based upon the `"cSpell.language"` setting.

```
{
"cSpell.languageSettings": [
    { "languageId": '*',      "local": 'en',               "dictionaries": ['wordsEn'] },
    { "languageId": '*',      "local": 'en-US',            "dictionaries": ['wordsEn'] },
    { "languageId": '*',      "local": 'en-GB',            "dictionaries": ['wordsEnGb'] },
    { "languageId": '*',                                   "dictionaries": ['companies', 'softwareTerms', 'misc'] },
    { "languageId": "python", "allowCompoundWords": true,  "dictionaries": ["python"]},
    { "languageId": "go",     "allowCompoundWords": true,  "dictionaries": ["go"] },
    { "languageId": "javascript",                          "dictionaries": ["typescript", "node"] },
    { "languageId": "javascriptreact",                     "dictionaries": ["typescript", "node"] },
    { "languageId": "typescript",                          "dictionaries": ["typescript", "node"] },
    { "languageId": "typescriptreact",                     "dictionaries": ["typescript", "node"] },
    { "languageId": "html",                                "dictionaries": ["html", "fonts", "typescript", "css"] },
    { "languageId": "php",                                 "dictionaries": ["php", "html", "fonts", "css", "typescript"] },
    { "languageId": "css",                                 "dictionaries": ["fonts", "css"] },
    { "languageId": "less",                                "dictionaries": ["fonts", "css"] },
    { "languageId": "scss",                                "dictionaries": ["fonts", "css"] },
];
}
```

How to add your own Dictionaries

Global Dictionary

To add a global dictionary, you will need change your user settings.

Define the Dictionary

In your user settings, you will need to tell the spell checker where to find your word list.

Example adding medical terms, so words like *acanthopterygious* can be found.

```
// A List of Dictionary Definitions.
"cSpell.dictionaryDefinitions": [
    { "name": "medicalTerms", "path": "/Users/guest/projects/cSpell-WordLists/dictionaries/medicalterms-en.txt"},
    // To specify a path relative to the workspace folder use ${workspaceFolder} or ${workspaceFolder:Name}
    { "name": "companyTerms", "path": "${workspaceFolder}/../company/terms.txt"}
],
// List of dictionaries to use when checking files.
"cSpell.dictionaries": [
    "medicalTerms",
    "companyTerms"
]
```

Explained: In this example, we have told the spell checker where to find the word list file. Since it is in the user settings, we ha

Once the dictionary is defined. We need to tell the spell checker when to use it. Adding it to `cSpell.dictionaries` advises the spell cl

Note: Adding large dictionary files to be always used will slow down the generation of suggestions.

Project / Workspace Dictionary

To add a dictionary at the project level, it needs to be in the `cspell.json` file. This file can be either at the project root or in t

Example adding medical terms, where the terms are checked into the project and we only want to use it for .md files.

```
{
    "dictionaryDefinitions": [
        { "name": "medicalTerms", "path": "./dictionaries/medicalterms-en.txt"},
        { "name": "cities", "path": "./dictionaries/cities.txt"}
    ],
    "dictionaries": [
        "cities"
    ],
    "languageSettings": [
        { "languageId": "markdown", "dictionaries": ["medicalTerms"] },
        { "languageId": "plaintext", "dictionaries": ["medicalTerms"] }
    ]
}
```

Explained: In this example, two dictionaries were defined: *cities* and *medicalTerms*. The paths are relative to the location of the c

The *cities* dictionary is used for every file type, because it was added to the list to *dictionaries*. The *medicalTerms* dictionary is

FAQ

See: FAQ

Visual Studio Code Remote Development

Visual Studio Code logo	**Visual Studio Code Remote Development** *Open any folder in a container, on a remote machine, or in WSL and take advantage of VS Code's full* ***Download now!***

This repository is for providing feedback on the **Visual Studio Remote Development** extension pack and its related extensions. You ca

- **Remote - SSH**
  - Remote - SSH: Editing Configuration Files
- **Remote - Containers**
- **Remote - WSL**

If you are running into an issue with **another extension** you'd like to use with the Remote Development extensions, please raise an i

Documentation

Running into trouble? Wondering what you can do? These articles can help.

- Overview
- Remote Development using SSH
- Developing inside a Container
- Developing in WSL
- Tips, Tricks, and Troubleshooting

<===============(Docker for Visual Studio Code)===============>

Docker - Visual Studio Marketplace
> Extension for Visual Studio Code - Makes it easy to create, manage, and debug containerized applications.
Docker for Visual Studio Code Version Installs Build Status

The Docker extension makes it easy to build, manage, and deploy containerized applications from Visual Studio Code. It also provides

Docker extension overview
Docker extension overview

Check out the Working with containers topic on the Visual Studio Code documentation site to get started.

The Docker extension wiki has troubleshooting tips and additional technical information.

Installation

Install Docker on your machine and add it to the system path.

On Linux, you should also enable Docker CLI for the non-root user account that will be used to run VS Code.

To install the extension, open the Extensions view, search for `docker` to filter results and select Docker extension authored by Micro

Overview of the extension features

Editing Docker files

You can get IntelliSense when editing your `Dockerfile` and `docker-compose.yml` files, with completions and syntax help for common commands

IntelliSense for Dockerfiles
IntelliSense for Dockerfiles

In addition, you can use the Problems panel (Ctrl+Shift+M on Windows/Linux, Shift+Command+M on Mac) to view common errors for `Docker`

Generating Docker files

You can add Docker files to your workspace by opening the Command Palette (F1) and using **Docker: Add Docker Files to Workspace** comma

The extension recognizes workspaces that use most popular development languages (C#, Node.js, Python, Ruby, Go, and Java) and custor

Docker view

The Docker extension contributes a Docker view to VS Code. The Docker view lets you examine and manage Docker assets: containers, ir

The right-click menu provides access to commonly used commands for each type of asset.

Docker view context menu
Docker view context menu

You can rearrange the Docker view panes by dragging them up or down with a mouse and use the context menu to hide or show them.

Customize Docker view
Customize Docker view

Docker commands

Many of the most common Docker commands are built right into the Command Palette:

Docker commands
Docker commands

You can run Docker commands to manage images, networks, volumes, image registries, and Docker Compose. In addition, the **Docker: Prur**

Docker Compose

Docker Compose lets you define and run multi-container applications with Docker. Visual Studio Code's experience for authoring `docke`

Docker Compose IntelliSense
Docker Compose IntelliSense

For the `image` directive, you can press ctrl+space and VS Code will query the Docker Hub index for public images:

Docker Compose image suggestions
Docker Compose image suggestions

VS Code will first show a list of popular images along with metadata such as the number of stars and description. If you continue ty

Docker Compose Microsoft image suggestions
Docker Compose Microsoft image suggestions

Using image registries

You can display the content and push/pull/delete images from Docker Hub and Azure Container Registry:

Azure Container Registry content
Azure Container Registry content

An image in an Azure Container Registry can be deployed to Azure App Service directly from VS Code; see Deploy images to Azure App S

Debugging services running inside a container

You can debug services built using Node.js, Python, or .NET (C#) that are running inside a container. The extension offers custom ta

Azure CLI integration

You can start Azure CLI (command-line interface) in a standalone, Linux-based container with **Docker Images: Run Azure CLI** command. T

vscode-htmlhint

Integrates the HTMLHint static analysis tool into Visual Studio Code.

hero
hero

Configuration

The HTMLHint extension will attempt to use the locally installed HTMLHint module (the project-specific module if present, or a globa

To install a version to the local project folder, run `npm install --save-dev htmlhint`. To install a global version on the current machin

Usage

The HTMLHint extension will run HTMLHint on your open HTML files and report the number of errors on the Status Bar with details in t

status bar
status bar

Errors in HTML files are highlighted with squiggles and you can hover over the squiggles to see the error message.

hover
hover

> **Note:** HTMLHint will only analyze open HTML files and does not search for HTML files in your project folder.

Rules

The HTMLHint extension uses the default rules provided by HTMLHint.

```
{
    "tagname-lowercase": true,
    "attr-lowercase": true,
    "attr-value-double-quotes": true,
    "doctype-first": true,
    "tag-pair": true,
    "spec-char-escape": true,
    "id-unique": true,
    "src-not-empty": true,
    "attr-no-duplication": true,
    "title-require": true
}
```

.htmlhintrc

If you'd like to modify the rules, you can provide a `.htmlhintrc` file in the root of your project folder with a reduced ruleset or mo

You can learn more about rule configuration at the HTMLHint Usage page.

Additional file types

By default, HTMLHint will run on any files associated with the "html" language service (i.e., ".html" and ".htm" files). If you'd l

Option 1: Treating your file like any other html file

If you would like the file type to be treated as any other html file (including syntax highlighting, as well as HTMLHint linting), y

```
{
  "files.associations": {
    "*.ext": "html",
  }
}
```

Option 2: Associating HTMLHint extension with your file type

If your file type already has an associated language service other than "html", and you'd like HTMLHint to process those file types

```
{
  "htmlhint.documentSelector": [
    "html",
    "mylang"
  ]
}
```

Settings

The HTMLHint extension provides these settings:

- `htmlhint.enable` - disable the HTMLHint extension globally or per workspace.
- `htmlhint.documentSelector` - specify additional language services to be linted
- `htmlhint.options` - provide a rule set to override on disk `.htmlhintrc` or HTMLHint defaults.
- `htmlhint.configFile` - specify a custom HTMLHint configuration file. Please specify either 'htmlhint.configFile' or 'htmlhint.optio

You can change settings globally (**File** > **Preferences** > **User Settings**) or per workspace (**File** > **Preferences** > **Workspace Settings**). Th

Here's an example using the `htmlhint.documentSelector` and `htmlhint.options` settings:

```
"htmlhint.documentSelector: [
    "html",
    "htm",
    "twig"
],
"htmlhint.options": {
    "tagname-lowercase": false,
    "attr-lowercase": true,
    "attr-value-double-quotes":  true,
    "doctype-first": true
}
```

Note that in order to have the linter apply to addi

Git Graph extension for Visual Studio Code

View a Git Graph of your repository, and easily perform Git actions from the graph. Configurable to look the way you want!

Recording of Git Graph
Recording of Git Graph

Features

- Git Graph View:
  - Display:
    - Local & Remote Branches
    - Local Refs: Heads, Tags & Remotes
    - Uncommitted Changes
  - Perform Git Actions (available by right clicking on a commit / branch / tag):
    - Create, Checkout, Delete, Fetch, Merge, Pull, Push, Rebase, Rename & Reset Branches
    - Add, Delete & Push Tags
    - Checkout, Cherry Pick, Drop, Merge & Revert Commits
    - Clean, Reset & Stash Uncommitted Changes
    - Apply, Create Branch From, Drop & Pop Stashes
    - View annotated tag details (name, email, date and message)
    - Copy commit hashes, and branch, stash & tag names to the clipboard
  - View commit details and file changes by clicking on a commit. On the Commit Details View you can:
    - View the Visual Studio Code Diff of any file change by clicking on it.
    - Open the current version of any file that was affected in the commit.
    - Copy the path of any file that was affected in the commit to the clipboard.

  - Click on any HTTP/HTTPS url in the commit body to open it in your default web browser.

- Compare any two commits by clicking on a commit, and then CTRL/CMD clicking on another commit. On the Commit Comparison View
  - View the Visual Studio Code Diff of any file change between the selected commits by clicking on it.
  - Open the current version of any file that was affected between the selected commits.
  - Copy the path of any file that was affected between the selected commits to the clipboard.

- Code Review - Keep track of which files you have reviewed in the Commit Details & Comparison Views.
  - Code Review's can be performed on any commit, or between any two commits (not on Uncommitted Changes).
  - When a Code Review is started, all files needing to be reviewed are bolded. When you view the diff / open a file, it will
  - Code Reviews persist across Visual Studio Code sessions. They are automatically closed after 90 days of inactivity.

- View uncommitted changes, and compare the uncommitted changes with any commit.
- Hover over any commit vertex on the graph to see a tooltip indicating:
  - Whether the commit is included in the HEAD.
  - Which branches, tags and stashes include the commit.

- Filter the branches shown in Git Graph using the 'Branches' dropdown menu. The options for filtering the branches are:
  - Show All branches
  - Select one or more branches to be viewed
  - Select from a user predefined array of custom glob patterns (by setting `git-graph.customBranchGlobPatterns`)

- Fetch from Remote(s) *(available on the top control bar)*
- Find Widget allows you to quickly find one or more commits containing a specific phrase (in the commit message / date / auth
- Repository Settings Widget:
  - Allows you to view, add, edit, delete, fetch & prune remotes of the repository.
  - Configure "Issue Linking" - Converts issue numbers in commit messages into hyperlinks, that open the issue in your issue
  - Configure "Pull Request Creation" - Automates the opening and pre-filling of a Pull Request form, directly from a branche
    - Support for the publicly hosted Bitbucket, GitHub and GitLab Pull Request providers is built-in.
    - Custom Pull Request providers can be configured using the Extension Setting `git-graph.customPullRequestProviders` (e.g. for

- Keyboard Shortcuts (available in the Git Graph View):
  - `CTRL/CMD + F`: Open the Find Widget.
  - `CTRL/CMD + H`: Scrolls the Git Graph View to be centered on the commit referenced by HEAD.
  - `CTRL/CMD + R`: Refresh the Git Graph View.
  - `CTRL/CMD + S`: Scrolls the Git Graph View to the first (or next) stash in the loaded commits.
  - `CTRL/CMD + SHIFT + S`: Scrolls the Git Graph View to the last (or previous) stash in the loaded commits.
  - When the Commit Details View is open on a commit:
    - `Up` / `Down`: The Commit Details View will be opened on the commit directly above or below it on the Git Graph View.
    - `CTRL/CMD + Up` / `CTRL/CMD + Down`: The Commit Details View will be opened on its child or parent commit on the same branch.

        - `Enter`: If a dialog is open, pressing enter submits the dialog, taking the primary (left) action.
        - `Escape`: Closes the active dialog, context menu or the Commit Details View.
    - Resize the width of each column, and show/hide the Date, Author & Commit columns.
    - Common Emoji Shortcodes are automatically replaced with the corresponding emoji in commit messages (including all gitmoji).
- A broad range of configurable settings (e.g. graph style, branch colours, and more...). See the 'Extension Settings' section bel
- "Git Graph" launch button in the Status Bar
- "Git Graph: View Git Graph" launch command in the Command Palette

Extension Settings

Detailed information of all Git Graph settings is available here, including: descriptions, screenshots, default values and types.

A summary of the Git Graph extension settings are: * **Commit Details View**: * **Auto Center**: Automatically center the Commit Details Vie

This extension consumes the following settings:

- `git.path`: Specifies the path and filename of a portable Git installation.

Extension Commands

This extension contributes the following commands:

- `git-graph.view`: Git Graph: View Git Graph
- `git-graph.addGitRepository`: Git Graph: Add Git Repository... *(used to add sub-repos to Git Graph)*
- `git-graph.clearAvatarCache`: Git Graph: Clear Avatar Cache
- `git-graph.endAllWorkspaceCodeReviews`: Git Graph: End All Code Reviews in Workspace
- `git-graph.endSpecificWorkspaceCodeReview`: Git Graph: End a specific Code Review in Workspace... *(used to end a specific Code Review wi*
- `git-graph.removeGitRepository`: Git Graph: Remove Git Repository... *(used to remove repositories from Git Graph)*
- `git-graph.resumeWorkspaceCodeReview`: Git Graph: Resume a specific Code Review in Workspace... *(used to open the Git Graph View to a C*

Release Notes

Detailed Release Notes are available here.

Visual Studio Marketplace

This extension is available on the Visual Studio Marketplace for Visual Studio Code.

Acknowledgements

Thank you to all of the contributors that help with the development of Git Graph!

Some of the icons used in Git Graph are from the following sources, please support them for their excellent work! - GitHub Octicons

Handy shortcuts for editing Markdown (.md, .markdown) files. You can also use markdown formats in any other file (see configuration s

Quickly toggle bullet points

Easily generate URLs

Convert tabular data to tables

Context and title menu integration

You can show and hide icons in the title bar with the markdownShortcuts.icons.* config settings.

Exposed Commands

Name	Description	Default key binding
md-shortcut.showCommandPalette	Display all commands	ctrl+M ctrl+M
md-shortcut.toggleBold	Make **bold**	ctrl+B
md-shortcut.toggleItalic	Make _italic_	ctrl+I
md-shortcut.toggleStrikethrough	Make ~~strikethrough~~	
md-shortcut.toggleLink	Make [a hyperlink](www.example.org)	ctrl+L
md-shortcut.toggleImage	Make an image [](image_url.png)	ctrl+shift+L
md-shortcut.toggleCodeBlock	Make ```a code block```	ctrl+M ctrl+C
md-shortcut.toggleInlineCode	Make `inline code`	ctrl+M ctrl+I
md-shortcut.toggleBullets	Make * bullet point	ctrl+M ctrl+B
md-shortcut.toggleNumbers	Make 1. numbered list	ctrl+M ctrl+1
md-shortcut.toggleCheckboxes	Make - [ ] check list (Github flavored markdown)	ctrl+M ctrl+X
md-shortcut.toggleTitleH1	Toggle # H1 title	
md-shortcut.toggleTitleH2	Toggle ## H2 title	
md-shortcut.toggleTitleH3	Toggle ### H3 title	
md-shortcut.toggleTitleH4	Toggle #### H4 title	
md-shortcut.toggleTitleH5	Toggle ##### H5 title	
md-shortcut.toggleTitleH6	Toggle ###### H6 title	
md-shortcut.addTable	Add Tabular values	
md-shortcut.addTableWithHeader	Add Tabular values with header	

MarkdownConverter

A markdown-converter for [Visual Studio Code][VSCode]

What's `MarkdownConverter`?

> MarkdownConverter is a Visual Studio Code-extension which allows you to export your Markdown-file as PDF-, HTML or Image-files.
> It provides many features, such as DateTime-Formatting, configuring your own CSS-styles, setting Headers and Footers, FrontMatter

Usage

1. Set your desired conversion-types or skip this step to convert your markdown-file to `PDF`:
   - Open up your Visual Studio Code-Settings and set `markdownConverter.ConversionType` to an array of types: json { "markdownConv
2. Optionally set the pattern for resolving the destination-path: json { "markdownConverter.DestinationPattern": "${workspaceFolder}
3. Open up the command pallet (`Ctrl`, `Shift`+`P`) and search one of these commands:
   - `Markdown: Convert Document` (`Markdown: Dokument Konvertieren` in German) or `mco` (`mk` in German) for short
   - `Markdown: Convert all Documents` (`Markdown: Alle Dokumente konvertieren`) or `mcd` (`madk` in German) for short
   - `Markdown: Chain all Documents` (`Markdown: Alle Dokumente verketten`) or `mcad` (`madv` in German) for short
4. Press enter and wait for the process to finish

[VSCode]: https://code.visualstudio.com/

<===============(vscode-opn)===============>

vscode-opn

vscode-opn
vscode-opn

> Microsoft Visual Studio Code extension integrating node.js module: opn

> Opens files directly by default, or on your local web server using the default app set by the OS, or an app set by you.

> Great for rapid prototyping!

Install

Easiest from the Extension Gallery

1. Start VS Code.

2. From within VS Code press `F1`, then type `ext install`.

3. Select `Extensions: Install Extension` and click or press `Enter`.

4. Wait a few seconds for the list to populate and type `Open File in App`.

5. Click the install icon next to the `Open File in App` extension.

6. Restart VS Code to complete installing the extension.

Alternatively, with the Packaged Extension (.vsix) file

Download the latest `vscode-opn.vsix` from GitHub Releases.

You can manually install the VS Code extension packaged in a .vsix file.

Option 1)

Execute the VS Code command line below providing the path to the .vsix file;

```
code myExtensionFolder\vscode-opn.vsix
```

Depending on your platform replace `myExtensionFolder\` with;

- Windows:- `%USERPROFILE%\.vscode\extensions\`
- Mac:- `$HOME/.vscode/extensions/`
- Linux:- `$HOME/.vscode/extensions/`

Option 2)

Start VS Code.

From within VS Code open the 'File' menu, select 'Open File...' or press Ctrl+O, navigate to the .vsix file location and select it t

The extension will be installed under your user .vscode/extensions folder.

Usage

Execute the extension with the keyboard shortcut;

- **Mac**: `Command + Alt + O`
- **Windows/Linux**: `Ctrl + Alt + O`

Customise behaviour per language (optional)

By default plain text based files will open directly in the app set by your OS.

To change this default behaviour you will need to edit 'Workspace Settings' or 'User Settings'.

From within VS Code open the 'File' menu, select 'Preferences', and then select either 'Workspace Settings' or 'User Settings'.

Options are set per 'language', not per file type, so for example setting options for the language 'plaintext', will change the beha

The minimum JSON to change the behaviour of a single language, e.g. html is;

```
  "vscode-opn.perLang": {
      "opnOptions": [
        {
          "forLang": "html",
          "openInApp": "chrome"
        }
      ]
    }
```

forLang

Required
Type: `string`
Examples include; - "html" - "plaintext" - "css" - "json" - "javascript"

openInApp

Required
Type: `string`
Examples include; - "chrome" - "firefox" - "iexplore" - "notepad" - "notepad++" - "wordpad" - "winword" - "winmerge"

Notes: The app name examples above are platform dependent and valid for an MS Windows environment. For Google's Chrome web browser t

The following example sets all available options for `html` and `plaintext` languages;

```
  "vscode-opn.perLang": {
      "opnOptions": [
        {
          "forLang": "html",
          "openInApp": "chrome",
          "openInAppArgs": ["--incognito"],
          "isUseWebServer": true,
          "isUseFsPath": false,
          "isWaitForAppExit": true
        },
        {
          "forLang": "plaintext",
          "openInApp": "notepad++",
          "openInAppArgs": [],
          "isUseWebServer": false,
          "isUseFsPath": true,
          "isWaitForAppExit": true
        }
      ]
    },
    "vscode-opn.webServerProtocol": "http",
    "vscode-opn.webServerHostname": "localhost",
    "vscode-opn.webServerPort": 8080
```

openInAppArgs

Optional
Type: `array`
Examples include: ["--incognito", "-private-window"]

Notes: A string array of arguments to pass to the app. For example, `--incognito` will tell Google Chrome to open a new window in "Priv

isUseWebServer

Optional
Type: `boolean`
Default: `false`

Notes: If you set up a site on your local web server such as MS Windows IIS and your VS Code project folder is within the site root,

isUseFsPath

Optional
Type: `boolean`
Default: `false`

Notes: During testing in an MS Win 10 environment both Notepad++ and MS WordPad did not support file URIs. This option will use the

isWaitForAppExit

Optional
Type: `boolean`
Default: `true`

Notes: See node.js module **opn option: wait** for more info.

Web server settings (applies to all languages)

Required by the per language option: `isUseWebServer`.

vscode-opn.webServerProtocol

Optional
Type: `string`
Default: `"http"`

Notes: Enter a protocol, e.g. "http" or "https".

vscode-opn.webServerHostname

Optional
Type: `string`
Default: `"localhost"`

Notes: Enter the hostname or IP of your local web server.

vscode-opn.webServerPort

Optional
Type: `number`
Default: `8080`

Notes: Enter the port of your local web server.

Tested

- Tested in MS Win 10 environment.
- Works with plain text based file types including;
- `.html`
- `.htm`
- `.xml`
- `.json`
- `.log`
- `.txt`
- Does not work with other file types including;
- `.png`
- `.gif`
- `.jpg`
- `.docx`
- `.xlsx`
- `.pdf`

Release notes

Version 1.1.2
Date: 25 Feb 2016

- Set original feature as the default behaviour.
- NEW FEATURE: Setting per language for which app to use

- NEW FEATURE: Setting per language to pass argument(s) to app
- NEW FEATURE: Setting per language to open file URL in local web server
- NEW FEATURE: Setting per language to use file system path, instead of URI
- NEW FEATURE: Setting per language to wait for the opened app to exit before calling the callback

Version 1.0.2
Date: 17 Feb 2016

- FEATURE: Open file URIs directly in the default application for the file type set in the OS.

Contributions

- Please use Github Issues, for feedback, feature suggestions, comments and reporting bugs.
- Feel free to fork this project and create pull requests with new features and/or bug fixes.
- Help with bugs/issues specific to other platforms such as OSX and Linux is particularly welcome.

Dependencies

- opn

License

MIT

Marketplace Version Installs Rating license MIT Build Status dependencies none

Node.js Modules Intellisense

Greenkeeper badge

Visual Studio Code plugin that autocompletes JavaScript / TypeScript modules in import statements.

This plugin was inspired by Npm Intellisense and AutoFileName.

auto complete
auto complete

Installation

Launch VS Code Quick Open (⌘+P), paste the following command, and press enter.

```
ext install node-module-intellisense
```

View detail on Visual Studio Code Marketplace

Issues & Contribution

If there is any bug, create a pull request or an issue please. Github

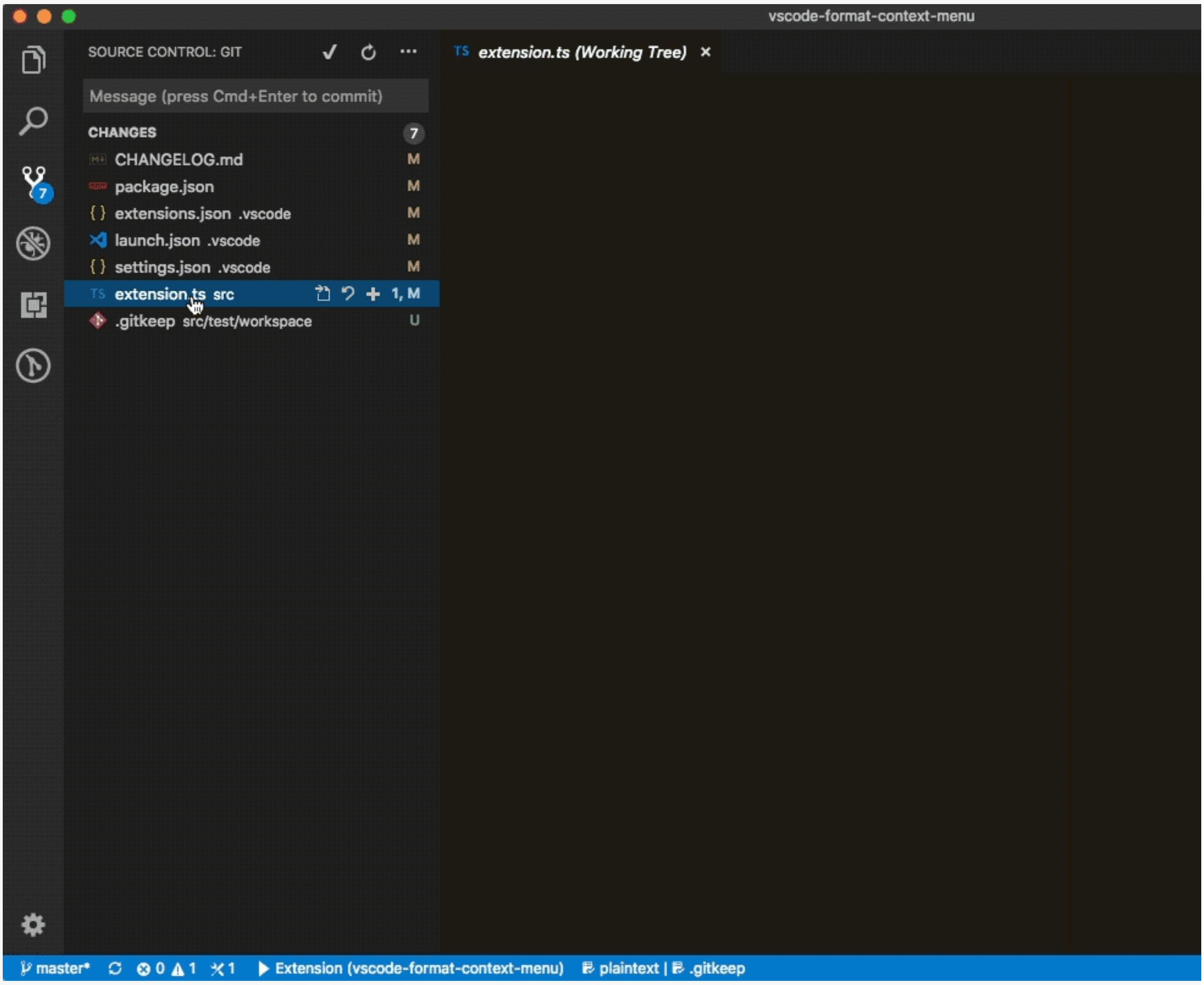
Configuration

Node.js Module Intellisense scans builtin modules, dependencies, devDependencies and file modules by default. Set scanBuiltinModules

```
{
    // Scans builtin modules as well
  "node-module-intellisense.scanBuiltinModules": true,
// Scans devDependencies as well
"node-module-intellisense.scanDevDependencies": true,


// Scans file modules as well
"node-module-intellisense.scanFileModules": true,


/**
   * Scans alternative module paths (eg. Search on ${workspaceFolder}/lib).
   * Useful when using packages like (https://www.npmjs.com/package/app-module-path) to manage require paths folder.
   **/
"node-module-intellisense.modulePaths": [],


// Auto strip module extensions
"node-module-intellisense.autoStripExtensions": [
".js",
".jsx",
".ts",
".d.ts",
".tsx"
],
}
```

VSCode format in context menus

This VSCode extension allows the user to format one or multiple files with right-click context menu.


vscode-format-context-menu
SOURCE CONTROL: GIT
Message (press Cmd+Enter to commit)
CHANGES 7
CHANGELOG.md M
package.json M
extensions.json .vscode M
launch.json .vscode M
settings.json .vscode M
extension.ts src 1, M
.gitkeep src/test/workspace U
extension.ts (Working Tree)
master* 0 1 1 Extension (vscode-format-context-menu) plaintext | .gitkeep

Demo

Features

- Format one or multiple files from Explorer Context Menu
- Format one or multiple files from SCM Context Menu
- Format one file from Editor File Tile Context Menu

Extension Settings

This extension contributes the following settings:

- `formatContextMenu.saveAfterFormat`: enable/disable saving after formatting (default: `true`).
- `formatContextMenu.closeAfterSave`: enable/disable closing closing after saving (default: `false`). This does nothing unless you have enal

vscode-github README

Marketplace Version Installs Travis renovate badge

This vscode extension integrates with GitHub.

Note: I recommend to use GitHub Pull Requests instead of this, because most usecases are supported and there is a team at Microsoft,

Features

Currently it is possible to do the following:

- Checkout one of the open pull requests
- Open github page for the current project in your default browser
- Browse one of the open pull requests in your default browser
- Browse the pull requests of your current branch
- Display pull request and current status (e.g. mergeable, travis build done, ...) in the StatusBar (if enabled)
- Create a new pull request based on the current branch and the last commit The current branch will be requested to merge into mas
- Create a pull request in forked repositories
- Allow to select a branch to create a pull request for
- Merge current pull request with either of 'merge', 'squash' or 'rebase' method.
- Configure default branch, merge method and refresh interval.
- Allow to manage assignees for pull requests
- Assign or unassign a user (currently only one)
- Allow to create and cancel pull request reviews
- Support for GitHub Enterprise (on-premise installations)
- Browse open issues
- Browse the current open file (including current cursor position)
- Configure the statusbar behaviour by setting the `github.statusBarCommand` configuration value.
- Specify a GitLab private access token and connect to a GitLab server
- Support multi folder setup

Create pull request
Create pull request

Setup Personal Access Token

To use this extension one needs to create a new GitHub Personal Access Token and registers it in the extension. The 'GitHub: Set Per

GitHub Personal Access Token
GitHub Personal Access Token

GitHub Personal Access Token
GitHub Personal Access Token

Set GitHub Personal Access Token
Set GitHub Personal Access Token

Additionally, by default this extension assumes your remote for a checked out repo is named "origin". If you wish to use a remote wi

There are additional settings for this extension as well, enter `github.` in the User Settings pane of VS Code to see them all.

Usage

Create a new pull request

1. Create a new local branch from the commit you wanted to start developing with
2. Do you code changes
3. Commit your changes
4. Push your changes to your remote
5. Then execute `Create pull request from current branch in current repository (quick)`
6. In the status bar you can then see the status of your created pull request and if you'd like to open it

Create a new pull request from a forked repository

1. Fork a repository and clone it afterwards
2. Create a new local branch from the commit you wanted to start developing with
3. Do you code changes
4. Commit your changes
5. Push your changes to your remote
6. Then execute `Create pull request...`
7. Select the upstream repository you want to create the pull requests for **Note**: The status bar will not reflect the pull request s

Checkout pull request

1. Execute `Checkout open pull request...`
2. Select a pull request from the list
3. The pull request is checked out and your working copy switches to that branch

Browser pull request

1. Execute `Browse open pull request...`
2. Select a pull request from the list
3. Your default browser opens the pull request on github

Merge pull request

1. Execute `Merge pull request (current branch)...`
2. Select your merge strategy from the shown list (merge, squash, rebase)
3. The pull request associated with your current branch is then merged

Telemetry data (extension usage)

This extension collects telemetry data to track and improve usage. The collection of data could be disabled as described here https

Kite Autocomplete Plugin for Visual Studio Code

Kite is an AI-powered programming assistant that helps you write Python & JavaScript code inside Visual Studio Code. Kite helps you

At a high level, Kite provides you with: * 🧠 **Line-of-Code Completions** powered by machine learning models trained on over 25 millio

Requirements

- macOS 10.11+, Windows 7+ or Linux
- Visual Studio Code v1.28.0+
- Kite Engine

Use another editor? Check out Kite's other editor integrations.

Installation

Installing the Kite Engine

The Kite Engine needs to be installed in order for the package to work properly. The package itself provides the frontend that inter

macOS Instructions 1. Download the installer and open the downloaded `.dmg` file. 2. Drag the Kite icon into the `Applications` folder. 3.

Windows Instructions 1. Download the installer and run the downloaded `.exe` file. 2. The installer should run the Kite Engine automat

Linux Instructions 1. Visit https://kite.com/linux/ to install Kite. 2. The installer should run the Kite Engine automatically afte

Installing the Kite Plugin for Visual Studio Code

When running the Kite Engine for the first time, you'll be guided through a setup process which will allow you to install the VS Coc

Alternatively, you have 2 options to manually install the package: 1. Search for "Kite" in VS Code's built-in extension marketplace

Learn about the capabilities Kite adds to VS Code.

Usage

The following is a brief guide to using Kite in its default configuration.

Autocompletions

Simply start typing in a saved Python or JavaScript file and Kite will automatically suggest completions for what you're typing. Kit

completions

completions

Hover (Python only)

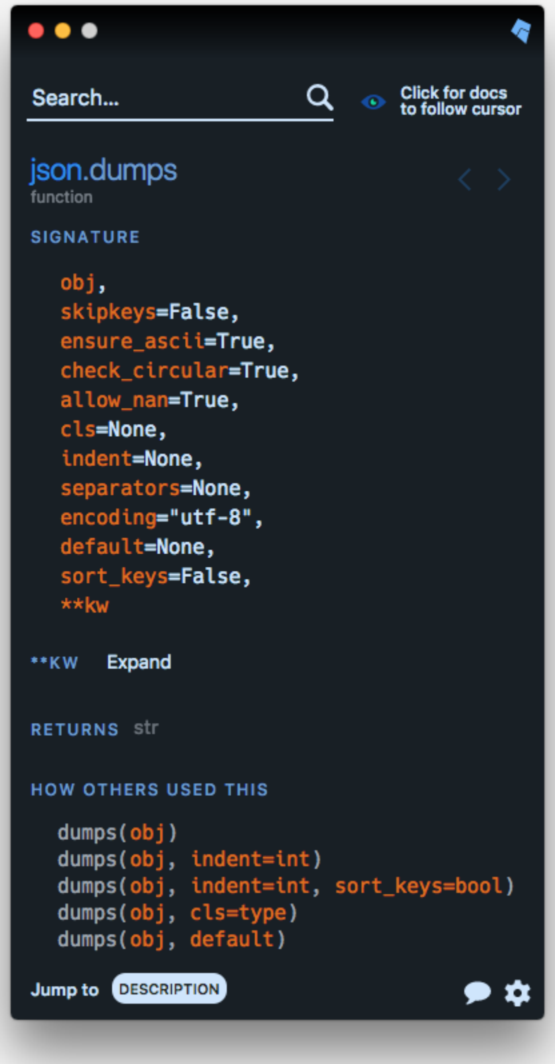
Hover your mouse cursor over a symbol to view a short summary of what the symbol represents.

hover

hover

Documentation (Python only)

Click on the `Docs` link in the hover popup to open the documentation for the symbol inside the Copilot, Kite's standalone reference t



json.dumps
function

SIGNATURE

```
obj,
skipkeys=False,
ensure_ascii=True,
check_circular=True,
allow_nan=True,
cls=None,
indent=None,
separators=None,
encoding="utf-8",
default=None,
sort_keys=False,
**kw
```

**KW Expand

RETURNS str

HOW OTHERS USED THIS

```
dumps(obj)
dumps(obj, indent=int)
dumps(obj, indent=int, sort_keys=bool)
dumps(obj, cls=type)
dumps(obj, default)
```

copilot

Definitions (Python only)

If a `Def` link is available in the hover popup, clicking on it will jump to the definition of the symbol.

Function Signatures (Python only)

When you call a function, Kite will show you the arguments required to call it. Kite's function signatures are also all labeled wit

signature
signature

> **Note:** If you have the Microsoft Python extension installed, Kite will *not* be able to show you information on function signatures.

Commands

Kite comes with sevaral commands that you can run from VS Code's command palette.

Command	Description
kite.open-copilot	Open the Copilot
kite.docs-at-cursor	Show documentation of the symbol underneath your cursor in the Copilot
kite.engine-settings	Open the settings for the Kite Engine
kite.python-tutorial	Open the Kite Python tutorial file
kite.javascript-tutorial	Open the Kite JavaScript tutorial file
kite.go-tutorial	Open the Kite Go tutorial file
kite.help	Open Kite's help website in the browser

Troubleshooting

Visit our help docs for FAQs and troubleshooting support.

Happy coding!

About Kite

Kite is built by a team in San Francisco devoted to making programming easier and more enjoyable for all. Follow Kite on Twitter anc

Code Metrics - Visual Studio Code Extension

Computes complexity in TypeScript / JavaScript / Lua files.

Complexity calculation

The steps of the calculation:

- create an AST from the input source file
- walk through each and every node of it
- depending on the type of the node and the configuration associated with it create a new entry about the node. This entry contair
- show the sum of complexity of child nodes for methods and the maximum of child nodes for classes

Please note that it is not a standard metric, but it is a close approximation of Cyclomatic complexity.

Please also note that it is possible to balance the complexity calculation for the project / team / personal taste by adjusting the

For example if one prefers guard clauses, and is ok with all the branches in switch statements then the following could be applied:

```
"codemetrics.nodeconfiguration.ReturnStatement": 0,
"codemetrics.nodeconfiguration.CaseClause": 0,
"codemetrics.nodeconfiguration.DefaultClause": 0
```

If You want to know the causes You can click on the code lens to list all the entries for a given method or class. (This also allows

Metric details example, showing how one might check the overall complexity for a method by clicking on the codelens
Metric details example, showing how one might check the overall complexity for a method by clicking on the codelens

It looks like this

First sample, demonstrating a constructor with overall complexity of 21
First sample, demonstrating a constructor with overall complexity of 21

Second sample, demonstrating a constructor with overall complexity of 1
Second sample, demonstrating a constructor with overall complexity of 1

Third sample, demonstrating a method with overall complexity of 5
Third sample, demonstrating a method with overall complexity of 5

Install

How to install Visual Studio Code extensions

Direct link to Visual Studio Code Marketplace

Customization

In the workspace settings one can override the defaults. For a complete list please check the configuration section in the package.j

```
{
  // highest complexity level will be set when it exceeds 15
  "codemetrics.basics.ComplexityLevelExtreme" : 15,
// Hides code lenses with complexity lesser than the given value
"codemetrics.basics.CodeLensHiddenUnder" : 5,


// Description for the highest complexity level
"codemetrics.basics.ComplexityLevelExtremeDescription" : "OMG split this up!",


// someone uses 'any', it must be punished
"codemetrics.nodeconfiguration.AnyKeyword": 100
}
```

Commands

- Toggle code lenses for arrow functions
- Toggle code metrics

They can be bound in the keybindings.json (File -> Preferences -> Keyboard Shortcuts)

```
  { "key": "f4",                  "command": "codemetrics.toggleCodeMetricsForArrowFunctions",
                                     "when": "editorTextFocus" },
  { "key": "f5",                  "command": "codemetrics.toggleCodeMetricsDisplayed",
                                     "when": "editorTextFocus" }
```

License

Licensed under MIT

Peacock for Visual Studio Code

Overview

Subtly change the color of your Visual Studio Code workspace. Ideal when you have multiple VS Code instances, use VS Live Share, or

> Peacock docs are hosted on Azure -> Get a Free Azure Trial

Install

1. Open **Extensions** sideBar panel in Visual Studio Code via the menu item `View → Extensions`
2. Search for **Peacock**
3. Click **Install**
4. Click **Reload**, if required

> You can also install Peacock from the marketplace here

Quick Usage

Let's see Peacock in action!

1. Create/Open a VSCode Workspace (Peacock only works in a Workspace)
2. Press `F1` to open the command palette
3. Type `Peacock`
4. Choose `Peacock: Change to a favorite color`
5. Choose one of the pre-defined colors and see how it changes your editor

Now enjoy exploring the rest of the features explained in the docs, here!

Peacock Windows
Peacock Windows

Features

Commands can be found in the command palette. Look for commands beginning with "Peacock:"

- Change the color of Affected Elements (see `peacock.affect*` in the Settings section) to
- user defined color
- a random color
- Select a user-defined color from your Favorite Colors
- Save a user-defined color with the Save Favorite Color
- Adjust the coloring of affected elements by making them slightly darker or lighter to provide a subtle visual contrast between t
- Saves colors to your workspace in the `.vscode/settings.json` file
- Integrates with Live Share.
- Integrates with VS Code Remote.

Settings

Property	Description
peacock.affectActivityBar	Specifies whether Peacock should affect the activity bar
peacock.affectStatusBar	Specifies whether Peacock should affect the status bar
peacock.affectTitleBar	Specifies whether Peacock should affect the title bar (see title bar coloring)
peacock.affectAccentBorders	Specifies whether Peacock should affect accent borders (panel, sideBar, editorGroup) Defa
peacock.affectStatusAndTitleBorders	Specifies whether Peacock should affect the status or title borders. Defaults to false.
peacock.affectTabActiveBorder	Specifies whether Peacock should affect the active tab's border. Defaults to false
peacock.elementAdjustments	fine tune coloring of affected elements
peacock.favoriteColors	array of objects for color names and hex values
peacock.keepForegroundColor	Specifies whether Peacock should change affect colors
peacock.surpriseMeOnStartup	Specifies whether Peacock apply a random color on startup
peacock.darkForeground	override for the dark foreground
peacock.lightForeground	override for the light foreground
peacock.darkenLightenPercentage	the percentage to darken or lighten the color
peacock.surpriseMeFromFavoritesOnly	Specifies whether Peacock should choose a random color from the favorites list or a purel
peacock.showColorInStatusBar	Show the Peacock color in the status bar
peacock.remoteColor	The Peacock color that will be applied to remote workspaces
peacock.color	The Peacock color that will be applied to workspaces
peacock.vslsShareColor	Peacock color for Live Share Color when acting as a Guest
peacock.vslsJoinColor	Peacock color for Live Share color when acting as the Host

Favorite Colors

After setting 1 or more colors (hex or named) in the user setting for `peacock.favoriteColors`, you can select **Peacock: Change to a Favor**

```
Gatsby Purple -> #123456
Auth0 Orange -> #eb5424
Azure Blue -> #007fff
```

Favorite colors require a user-defined name (`name`) and a value (`value`), as shown in the example below.

```
  "peacock.favoriteColors": [
    { "name": "Gatsby Purple", "value": "#639" },
    { "name": "Auth0 Orange", "value": "#eb5424" },
    { "name": "Azure Blue", "value": "#007fff" }
  ]
```

> You can find brand color hex codes from https://brandcolors.net

Preview Your Favorite

When opening the Favorites command in the command palette, Peacock now previews (applies) the color as you cycle through them. If yo

Animated GIF
Animated GIF

Save Favorite Color

When you apply a color you enjoy, you can go to the workspace `settings.json` and copy the color's hex code, then create your own favor

The `Peacock: Save Current Color as Favorite Color` feature allows you to save the currently set color as a favorite color, and prompts you

Affected Elements

You can tell peacock which parts of VS Code will be affected by when you select a color. You can do this by checking the appropriate

Animated GIF
Animated GIF

Element Adjustments

You can fine tune the coloring of affected elements by making them slightly darker or lighter to provide a subtle visual contrast be

- `"darken"`: reduces the value of the selected color to make it slightly darker
- `"lighten"`: increases the value of the selected color to make it slightly lighter
- `"none"`: no adjustment will be made to the selected color

An example of using this might be to make the Activity Bar slightly lighter than the Status Bar and Title Bar to better visually dis

```
  "peacock.affectActivityBar": true,
  "peacock.affectStatusBar": true,
  "peacock.affectTitleBar": true,
  "peacock.elementAdjustments": {
    "activityBar": "lighten"
  }
```

This results in the Activity Bar being slightly lighter than the Status Bar and Title Bar (see below).

Animated GIF
Animated GIF

Keep Foreground Color

Recommended to remain `false` (the default value).

When set to true Peacock will not colorize the foreground of any of the affected elements and will only alter the background. Some u

Surprise Me On Startup

Recommended to remain `false` (the default value).

When set to true Peacock will automatically apply a random color when opening a workspace that does not define color customizations

If this setting is `true` and there is no peacock color set, then Peacock will choose a new color. If there is already a color set, Pe

Lighten and Darken

You may like a color but want to lighten or darken it. You can do this through the corresponding commands. When you choose one of th

There are key bindings for the lighten command `alt+cmd+=` and for darken command `alt+cmd+-`, to make it easier to adjust the colors.

Commands

Command	Description
Peacock: Reset Workspace Colors	Removes any of the color settings from the `.vscode/settings.json` file. If colors
Peacock: Remove All Global and Workspace Colors	Removes all of the color settings from both the Workspace `.vscode/settings.json`
Peacock: Enter a Color	Prompts you to enter a color (see input formats)
Peacock: Color to Peacock Green	Sets the color to Peacock main color, #42b883
Peacock: Surprise me with a Random Color	Sets the color to a random color
Peacock: Change to a Favorite Color	Prompts user to select from their Favorites
Peacock: Save Current Color to Favorites	Save Current Color to their Favorites
Peacock: Add Recommended Favorites	Add the recommended favorites to user settings (override same names)
Peacock: Darken	Darkens the current color by `darkenLightenPercentage`
Peacock: Lighten	Lightens the current color by `darkenLightenPercentage`
Peacock: Show and Copy Current Color	Shows the current color and copies it to the clipboard
Peacock: Show the Documentation	Opens the Peacock documentation web site in a browser

Keyboard Shortcuts

description	key binding	command
Darken the colors	alt+cmd+-	peacock.darken
Lighten the colors	alt+cmd+=	peacock.lighten
Surprise Me with a Random Color	cmd+shift+k	peacock.changeColorToRandom

Integrations

Peacock integrates with other extensions, as described in this section.

VS Live Share Integration

Animated GIF
Animated GIF

Peacock detects when the Live Share extension is installed and automatically adds two commands that allow the user to change color c

Command	Description
Peacock: Change Live Share Color (Host)	Prompts user to select a color for Live Share Host session from the Favorites
Peacock: Change Live Share Color (Guest)	Prompts user to select a color for Live Share Guest session from the Favorites

When a Live Share session is started, the selected workspace color will be applied. When the session is finished, the workspace colo

- Learn more about Live Share
- Get the Live Share extension
- Get the Live Share extension pack, which now includes Peacock

Remote Development Integration

Peacock integrates with the Remote Development features of VS Code.

- Learn more about VS Code Remote Development
- Get the VS Code Remote Development Extensions

Peacock detects when the VS Code Remote extension is installed and adds commands that allow the user to change color when in a remot

When a workspace is opened in a remote context, if a `peacock.remoteColor` is set, it will be applied. Otherwise, the regular `peacock.colo`

Remote Integration with Peacock
Remote Integration with Peacock

VS Code distinguishes two classes of extensions: UI Extensions and Workspace Extensions. Peacock is classified as a UI extension as

In version 2.1.2 Peacock enabled integration with the Remote Development by adding `"extensionKind": "ui"` in the extension's `package.json`

Input Formats

When entering a color in Peacock several formats are acceptable. These include

Format	Examples
Named HTML colors	purple, blanchedalmond
Short Hex	#8b2, f00
Short Hex8 (RGBA)	#8b2c, f00c
Hex	#88bb22, ff0000
Hex8 (RGBA)	#88bb22cc, ff0000cc
RGB	rgb (136, 187, 34), rgb 255 0 0
RGBA	rgba (136, 187, 34, .8), rgba 255 0 0 .8
HSL	hsl (80, 69%, 43%), hsl (0 1 .5)
HSLA	hsla (80, 69%, 43%, 0.8), hsla (0 1 .5 .8)
HSV	hsv (80, 82%, 73%), hsv (0 1 1)
HSVA	hsva (80, 82%, 73%, 0.8), hsva (0,100%,100%,0.8)

All formats offer flexible data validation:

- For named HTML colors, case is insensitive
- For any hex value, the `#` is optional.
- For any color formula value all parentheses and commas are optional and any number can be a decimal or percentage (with the exce

Alpha Support

Peacock allows for control of the alpha channel through a variety of input formats listed above. In general, it is recommended to av

Roadmap

There are many features in the roadmap.

Issues

Please refer to the issues list and feel free to grab one and contribute!

Contributions

See these pages for details on contributions and our code of conduct.

Logging

Peacock writes to VS Code's log output. You can open the output panel and select "Peacock" to see the log. This can be helpful when

Example 1

User selects a color, then later changes which elements are affected.

1. User chooses "surprise me" and sets the color to #ff0000
2. Peacock saves #ff0000 in memory as the most recently used color
3. User goes to settings and unchecks the "Peacock: Affect StatusBar"
4. Peacock listens to this change, clears all colors and reapplies the #ff0000

Example 2

User opens VS Code, already has colors in their workspace, and immediately changes which elements are affected.

1. User opens VS Code
2. Workspace colors are set to #369
3. User goes to settings and unchecks the "Peacock: Affect StatusBar"
4. Peacock listens to this change, clears all colors and reapplies the #369

Example 3

User opens VS Code, has no colors in workspace, and immediately changes which elements are affected.

1. User opens VS Code
2. No workspace colors are set
3. Peacock's most recently used color is not set
4. User goes to settings and unchecks the "Peacock: Affect StatusBar"
5. Peacock listens to this change, however no colors are applied

How does title bar coloring work

The VS Code Title Bar style can be configured to be custom or native with the `window.titleBarStyle` setting. When operating in native mo

On macOS there are additional settings that can impact the Title Bar style and force it into native mode regardless of the `window.tit`

- `window.nativeTabs` should be set to **false**. If using native tabs, the rendering of the title bar is deferred to the OS and native mo
- `window.nativeFullScreen` should be set to **true**. If not using native full screen mode, the custom title bar rendering presents issues

A successful and recommended settings configuration to colorize the Title Bar is:

Title Bar Settings
Title Bar Settings

How are foreground colors calculated

Peacock is using tinycolor which provides some basic color theory mechanisms to determine whether or not to show a light or dark fo

Brightness is measured on a scale of 0-255 where a value of 127.5 is perfectly 50%.

Example:

```
const lightForeground = '#e7e7e7';
const darkForegound = '#15202b';
const background = '#498aff';
const perceivedBrightness = tinycolor(background).getBrightness(); // 131.903, so 51.7%
const isDark = tinycolor(background).isDark(); // false, since brightness is above 50%
const textColor = isDark ? lightForeground : darkForeground; // We end up using dark text
```

This particular color (`#498aff`) is very near 50% on the perceived brightness, but the determination is binary so the color is either

```
const readability = tinycolor.readability(darkForeground, background); // 4.996713
const isReadable = tinycolor.isReadable(darkForeground, background); // true
```

The readability calculations and metrics are based on Web Content Accessibility Guidelines (Version 2.0) and, in general, a ratio c

```
const readability = tinycolor.readability(lightForeground, background); // 2.669008
const isReadable = tinycolor.isReadable(lightForeground, background); // false
```

Why is the foreground hard to see with my transparent color

The readability calculations that Peacock uses to determine an appropriate foreground color are based only on the color information

Why are my affected elements not transparent

Peacock allows you to enter colors that can be transparent, but the VS Code window itself is not transparent. If the entered color h

What are recommended favorites

Recommended favorites are a list of constants found in `favorites.ts`. These are alphabetized.

Recommended favorites are a starting point for favorites. They will be installed whenever a new version is installed. They will exte

This list may change from version to version depending on the Peacock authoring team.

What are mementos

Peacock takes advantage of a memento (a value stored between sessions and not in settings).

Name	Type	Description
peacockMementos.favoritesVersion	Global	The version of Peacock. Helps identify when the list of favorites should be writte

Migration

Migration notes between versions are documented here.

> Note: If at any time Peacock is writing colors unexpectedly, it may be due to a migration (see migration notes below). However, a

To Version 3+

Version 3+ of Peacock stores the color in the settings `peacock.color`. When migrating from version 2.5, the peacock color was in a meme

Once the color is determined, peacock removes the memento, and writes the color to the settings `peacock.color`. Fixes #230 and addresse

This is an aggressive approach, as it is possible to have a color customization that peacock uses, and if it sees this, it will set

This logic is marked as deprecated but will not be removed until version 4.0 is released and enough time has passed reasonably for p

Examples:

1. If this is detected at startup with `#ff0`, then the `peacock.color` will bet set to match it.

```
// .vscode/settings.json
{
  "workbench.colorCustomizations": {
    "activityBar.background": "#ff0"
  }
}
```

2. If this is detected at startup and there is a peacock memento, then the `peacock.color` will set set to match the memento color.

```
// .vscode/settings.json
{}
```

3. If this is detected at startup and there is no peacock memento, no migration will occur.

```
// .vscode/settings.json
{}
```

4. If there already is a `peacock.color`, no migration will occur.

```
// .vscode/settings.json
{}
```

Try the Code

If you want to try the extension out start by cloning this repo, `cd` into the folder, and then run `npm install`.

Then you can run the debugger for the launch configuration `Run Extension`. Set breakpoints, step through the code, and enjoy!

Badges

Visual Studio Marketplace v3.8.0 installs 886.95K downloads 4.52M rating average: 4.67/5 (101 ratings) Live Share enabled

license MIT all contributors 15

Build Status Greenkeeper badge

Resources

- Get VS Code
- Create your first VS Code extension
- VS Code Extension API
- Learn how to add WebPack bundles to your favorite extensions
- Try Azure Free

Sketchnote
Sketchnote

Markdown Extended Readme

vscode marketplace v1.0.18 downloads 93k

Markdown Extended is an extension extends syntaxes and abilities to VSCode built-in markdown function.

Markdown Extended includes lots of editing helpers and a `what you see is what you get` exporter, which means export files are consistent

Features

- Exporter (View Detail)
  - Export to Self Contained HTML / PDF / PNG / JPEG
  - Export current document / workspace
  - Copy exported HTML to clipboard

- Editing Helpers (View Detail):
  - Paste, format table.
  - Add, delete and move table columns & rows.
  - Toggle various formates, eg.: bold, italics, underline, strikethrough, code inline, code block, block quote, superscript, su
- Extended Language Features (View Detail):
  - Admonition (built-in), View Document
  - Enhanced Anchor Link (built-in), View Document
  - markdown-it-table-of-contents
  - markdown-it-footnote
  - markdown-it-abbr
  - markdown-it-deflist
  - markdown-it-sup
  - markdown-it-sub
  - markdown-it-checkbox
  - markdown-it-attrs
  - markdown-it-kbd
  - markdown-it-underline
  - markdown-it-multimd-table
  - markdown-it-emoji
  - markdown-it-html5-embed
  - markdown-it-toc
  - markdown-it-container
  - markdown-it-mark

> Post an issue on GitHub if you want other plugins.

Disable Plugins

To disable integrated plugins, put their names separated with ,:

```
"markdownExtended.disabledPlugins": "underline, toc"
```

Available names: toc, container, admonition, footnote, abbr, sup, sub, checkbox, attrs, kbd, underline, mark, deflist, emoji, multir

Q: Why You Don't Integrate Some Plugin?

The extension works with other markdown plugin extensions (those who contribute to built-in Markdown engine) well, **Both Preview and**

The extension does not tend to do all the work, so just use them, those plugins could be deeper developed, with better experience.

Exporter

Find in command palette, or right click on an editor / workspace folder, and execute:

- `Markdown: Export to File`
- `Markdown: Export Markdown to File`

The export files are organized in `out` directory in the root of workspace folder by default.

Export Configurations

You can configure exporting for multiple documents with user settings.

Further, you can add per-file settings inside markdown to override user settings, it has the highest priority:

```
---
puppeteer:
    pdf:
        format: A4
        displayHeaderFooter: true
        margin:
            top: 1cm
            right: 1cm
            bottom: 1cm
            left: 1cm
    image:
        quality: 90
        fullPage: true
---
contents goes here...
```

See all available settings for puppeteer.pdf, and puppeteer.image

Helpers

Editing Helpers and Keys

> Inspired by joshbax.mdhelper, but totally new implements.

Command	Keyboard Shortcut
Format: Toggle Bold	Ctrl+B
Format: Toggle Italics	Ctrl+I
Format: Toggle Underline	Ctrl+U
Format: Toggle Mark	Ctrl+M

Format: Toggle Strikethrough	Alt+S
Format: Toggle Code Inline	Alt+| | Format: Toggle Code Block | Alt+Shift+
Format: Toggle Block Quote	Ctrl+Shift+Q
Format: Toggle Superscript	Ctrl+Shift+U
Format: Toggle Subscript	Ctrl+Shift+L
Format: Toggle Unordered List	Ctrl+L, Ctrl+U
Format: Toggle Ordered List	Ctrl+L, Ctrl+O
Table: Paste as Table	Ctrl+Shift+T, Ctrl+Shift+P
Table: Format Table	Ctrl+Shift+T, Ctrl+Shift+F
Table: Add Columns to Left	Ctrl+Shift+T, Ctrl+Shift+L
Table: Add Columns to Right	Ctrl+Shift+T, Ctrl+Shift+R
Table: Add Rows Above	Ctrl+Shift+T, Ctrl+Shift+A
Table: Add Row Below	Ctrl+Shift+T, Ctrl+Shift+B
Table: Move Columns Left	Ctrl+Shift+T Ctrl+Shift+Left
Table: Move Columns Right	Ctrl+Shift+T Ctrl+Shift+Right
Table: Delete Rows	Ctrl+Shift+D, Ctrl+Shift+R
Table: Delete Columns	Ctrl+Shift+D, Ctrl+Shift+C

> Looking for `Move Rows Up / Down`?
> You can use vscode built-in `Move Line Up / Down`, shortcuts are `alt+↑` and `alt+↓`

Table Editing

tableEdit
tableEdit

moveCols
moveCols

> Move columns key bindings has been changed to `ctrl+shift+t ctrl+shift+left/right`, due to #57, #68

Paste as Markdown Table

Copy a table from Excel, Web and other applications which support the format of Comma-Separated Values (CSV), then run the command

pasteTable
pasteTable

Export & Copy

command
command

Extended Syntaxes

Admonition

> Inspired by MkDocs

Nesting supported (by indent) admonition, the following shows a danger admonition nested by a note admonition.

```
!!! note
    This is the **note** admonition body

    !!! danger Danger Title
        This is the **danger** admonition body
```

admonition-demo
admonition-demo

Removing Admonition Title

```
!!! danger ""
    This is the danger admonition body
```

admonition-demo
admonition-demo

Supported Qualifiers

note | summary, abstract, tldr | info, todo | tip, hint | success, check, done | question, help, faq | warning, attention, caution | failure, fail, mis

Enhanced Anchor Link

Now, you're able to write anchor links consistent to heading texts.

```
Go to
[简体中文](#简体中文),
[Español Título](#Español-Título).
## 简体中文

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Aenean euismod bibendum laoreet.

## Español Título

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Aenean euismod bibendum laoreet.
```

markdown-it-table-of-contents

[[TOC]]

- Markdown Extended Readme
  - Features
  - Requirements
  - Demos

markdown-it-footnote

Here is a footnote reference,[^1] and another.[^longnote]

Here is a footnote reference,[1] and another.[2]

markdown-it-abbr

```
*[HTML]: Hyper Text Markup Language
*[W3C]:  World Wide Web Consortium
The HTML specification
is maintained by the W3C.
```

The HTML specification is maintained by the W3C.

markdown-it-deflist

```
Apple
:   Pomaceous fruit of plants of the genus Malus in the family Rosaceae.
```

Apple

Pomaceous fruit of plants of the genus Malus in the family Rosaceae.

markdown-it-sup markdown-it-sub

```
29^th^, H~2~O
```

29th, H_2O

markdown-it-checkbox

```
[ ] unchecked
[x] checked
```

☐unchecked ☑checked

markdown-it-attrs

```
item **bold red**{style="color:red"}
```

item **bold red**

markdown-it-kbd

```
[[Ctrl+Esc]]
```

Ctrl+Esc

markdown-it-underline

```
_underline_
```

underline

markdown-it-container

```
::::: container
:::: row
::: col-xs-6 alert alert-success
success text
:::
::: col-xs-6 alert alert-warning
warning text
:::
::::
:::::
```

container-demo.png
container-demo.png

(Rendered with style bootstrap, to see the same result, you need the follow config)

```
"markdown.styles": [
    "https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
]
```

Known Issues & Feedback

Please post and view issues on GitHub

Enjoy!

Search node_modules

Simple plugin for VS Code that allows you to quickly navigate the file inside your project's `node_modules` directory.

Chances are you have the `node_modules` folder excluded from the built-in search in VS Code, which means if you want to open and/or edit a file inside `noc`

Features

- Quickly navigate and open files in `node_modules` by traversing the folder tree.

Settings

- `search-node-modules.useLastFolder`: Default to folder of last opened file when searching (defaults to `false`).
- `search-node-modules.path`: Relative path to node_modules folder (defaults to `node_modules`).

Extension Packs

- Node.js Extension Pack

Links

- Visual Studio Marketplace: https://marketplace.visualstudio.com/items?itemName=jasonnutter.search-node-modules
- Repo: https://github.com/jasonnutter/vscode-search-node-modules
- Known Issues: https://github.com/jasonnutter/vscode-search-node-modules/issues
- Change Log: https://github.com/jasonnutter/vscode-search-node-modules/blob/master/CHANGELOG.md

Objective

Some of my friends ask me for similar requrest so I create this extension. It aims to integrate browser sync with VSCode to provide live preview of we

Requirement

- Just *VSCode*
- A web server supports proxy, php -S seems to has some issues with localhost

It can run without installing browser-sync and `Node.js` runtime becasues the extension containing the codes of **browser-sync** is running on a separated e

Features

Although many enhancements can be done, the basic functions are finished. For any issue, you can leave a comment below or leave a issue at github.

Behaviour

The preview can be opened at the right panel and in the default browser in different command, and there are two modes: Server mode and Proxy mode.

- Server mode: static `html` file
- Proxy mode: dynamic website that has its own web server, such as `php`, `jsp`, `asp` etc ...

With opening or Without opening folder

- With opening folder: watch the files you input in inputbox relative to the root folder
- Without opening folder: watch all files with the same extension at the parent folder of opening document

Server Mode

Without opening a folder

server open at panel
server open at panel

1. Type command on a `html` file: `Browser Sync: Server mode at side panel`
2. Right panel will be opened with a web page enabling browser sync
3. Once you save your change, the right panel will be changed. This feature also works if you enable auto save feature in VSCode.

You can also try the browser mode by command: `Browser Sync: Server mode in browser`

server open at browser
server open at browser

With opening a folder

server open at panel
server open at panel

1. Type command on a `html` file: `Browser Sync: Server mode at side panel`
2. Type the path of files you want to watch relative to the root folder

Proxy Mode

Without opening a folder

The image below is a **Laravel** web application hosted on a docker machine, the guideline don't just applies for **Laravel**, this also applied for other wel

proxy open at browser
proxy open at browser

1. Type command on any kind of file: `Browser Sync: Proxy mode at side panel`
2. Type in the URL of the original website, e.g. `http://localhost:8080`, or `8080`

With opening a folder

1. Type command on any kind of file: `Browser Sync: Proxy mode at side panel`
2. Type in the URL of the original website, e.g. `http://localhost:8080`, or `8080`
3. Type the path of files you want to watch relative to the root folder

Refresh Side Panel

Run command `Browser Sync: Refresh Side Panel` which acts like F5 in browser to refresh the side panel

Sometimes the page may crash and no `<body>` tag is returned, then the script of the browser sync cannot be injected and the codeSync will stop. In this

Configuration

Add setting as JSON format into user setting or workspace setting to override default behaviour. The setting options come from here, please use the op

Example setting (different browser)

- Open in different browser: chrome and firefox
- Without code sync.

Note from issue: Use "google chrome" under window, "chrome" under Mac

```
{
    "browserSync.config" : {
        "browser" : ["chrome", "firefox"],
        "codeSync" : false
    }
}
```

Example setting (relative path)

- Relative path of the watching files to the workspace root
- Can mix with absolute path

```
{
    "browserSync.config": {
        "files": ["*.html", "*.log", "C:\\Users\\jason\\Desktop\\db.txt"]
    }
}
```

How it works

Static HTML file

1. Once the command is typed, a `Node.js` server will be started, it handle request of the HTML file, and listen to the changes of files.
2. The right panel is a static HTML page but it looks like an embedded browser by containing an iframe with `src` pointing to the URL of the HTML file.
3. Once the file is saved, the server notifies the website embedded inside the iframe.

Proxy for dynamic site

1. Once the command is typed, a `Node.js` proxy will be started, it forward your request to the target URL with injecting the script in the returned HTM
2. The right panel is a static HTML page but it looks like an embedded browser by containing an iframe with `src` pointing to the URL of the proxy serv
3. Once the file is saved, the server notifies the proxy server embedded inside the iframe.

Enhancement Planning

1. Better resource management
2. Provide Better message to end user
3. ### Better error handling

<===============(VSCode Essentials (Extension Pack))===============>

VSCode Essentials (Extension Pack)

VS Marketplace v1.5.0 installs 21.5K rating 5/5 (2) license MIT

VSCode Masterclass (Coming Soon)

Follow on Twitter and subscribe at DevCast to get priority access.

Introduction

VSCode Essentials is a collection of many individual extensions that includes all of the functionality needed to turn VSCode into a ⚡ supercharged ⚡

This extension pack is language-agnostic, and does not include functionality for specific programming languages or frameworks. Instead, this extensio

Features

- Enhanced comment highlights, styling, and wrapping
- Todo manager
- Snippet manager and generator
- Run code
- Spell check
- Real-time collaboration with edits, calls, chats, and comments
- Formatting of multiple files
- Advanced git tooling
- Multiple clipboards (clipboard ring)
- Code screenshots
- Project manager
- Detection of non-printing characters

- Smart backspacing (delete indentation/whitespace)
- Cursor navigation with the ability to go to any character with a few keystrokes
- Macros for executing multiple commands with a single keyboard shortcut
- Synced and shared VSCode settings
- Extended multi-cursor functionality (incrementing, inserting numeric sequences, etc.)
- Hack VSCode's stylesheets for UI elements that aren't yet included in the settings or official theme API
- Semantic indent and scope guides
- Automatic and manually added (tagged commits) local edit history
- Bookmarks for marking lines of code and quickly jumping back to them
- Giving workspaces unique colors to help identify the editor when working with multiple instances, using Live Share, or taking advantage of remote
- Add syntax highlighting to log files and to VSCode's output/debug panels
- Toggle the display of hidden files in the explorer
- GitHub Gist management with interactive playgrounds, code snippets, and notes
- Synchronize edits you make in a Search Editor back to source files.

Recommended Settings

All Autocomplete

```
{
  // // TODO: Add files that you want to be included in IntelliSense at all times.
  // // Enable to get autocomplete for the entire project (as opposed to just open files). It's a
  // // good idea to read the All Autocomplete extension's documentation to understand the performance
  // // impact of specific settings.
  // "AllAutocomplete.wordListFiles": ["/src/"],
}
```

CodeStream

```
{
  "codestream.autoSignIn": false,
}
```

Customize UI

```
{
  "customizeUI.stylesheet": {
    // NOTE: Only hide icons if you're already familiar with their functionality and shortcuts
    // Hides specific editor action icons
    ".editor-actions a[title^=\"Toggle File Blame Annotations\"]": "display: none !important;",
    ".editor-actions a[title^=\"Open Changes\"]": "display: none !important;",
    ".editor-actions a[title^=\"More Actions...\"]": "display: none !important;",
    ".editor-actions a[title^=\"Split Editor Right (⌘\\\\)\"]": "display: none !important;",
    ".editor-actions a[title^=\"Run Code (^⌥N)\"]": "display: none !important;",

   // Adds a border below the sidebar title.
   // TODO: Update `19252B` in the setting below with the hex color of your choice.
   ".sidebar .composite.title": "border-bottom: 1px solid #19252B;",

   // Leaves only the bottom border on matching bracket border.
   ".monaco-editor .bracket-match": "border-top: none; border-right: none; border-left: none;",

   // Changes color of the circle that appears in a dirty file's editor tab.
   // TODO: Update `00bcd480` in the setting below with the hex color of your choice.
   ".monaco-workbench .part.editor&gt;.content .editor-group-container.active&gt;.title .tabs-container&gt;.tab.dirty&gt;.tab-close .action-label:not(:hover):before, .monaco-workbench .part.editor



  },
}
```

GitLens

```
{
  // Toggle from command palette as needed.
  "gitlens.codeLens.enabled": false,
  "gitlens.currentLine.enabled": false,
// Keep source control tools in same view.
"gitlens.views.compare.location": "scm",
"gitlens.views.fileHistory.location": "scm",
"gitlens.views.lineHistory.location": "scm",
"gitlens.views.repositories.location": "scm",
"gitlens.views.search.location": "scm",
}
```

Macros

```
{
  // TODO: Add your own macros. Below are some examples to get you started. Execute these commands with the command palette, or assign them to keyboard shortcuts.
  // NOTE: Use in combination with the Settings Cycler settings listed further down this page.
  "macros.list": {
    // Toggle zoom setting and panel position when moving from laptop to external monitor and vice versa.
    "toggleDesktop": [
      "workbench.action.togglePanelPosition",
      "settings.cycle.desktopZoom",
    ],
    // Copy line and paste it below, comment out the original line, and move cursor down to the pasted line. Great for debugging/experimenting with code while keeping the
    "commentDown": [
      "editor.action.copyLinesDownAction",
      "cursorUp",
      "editor.action.addCommentLine",
      "cursorDown"
    ],
  },
}
```

MetaGo

```
{
  // TODO: Update styling to fit your theme. The following works well with Material Theme.
  "metaGo.decoration.borderColor": "#253036",
  "metaGo.decoration.backgroundColor": "red, blue",
  "metaGo.decoration.backgroundOpacity": "1",
  "metaGo.decoration.color": "white",
// TODO: Update font settings to fit your preference.
"metaGo.decoration.fontFamily": "'Operator Mono SSm Lig', 'Fira Code', Menlo, Monaco, 'Courier New', monospace",
"metaGo.decoration.fontWeight": "normal",


"metaGo.jumper.timeout": 60,
}
```

Project Manager

```
{
  // TODO: Add the folders where you keep your code projects at.
  "projectManager.any.baseFolders": [
    "$home/Code",
  ],
"projectManager.any.maxDepthRecursion": 1,
"projectManager.sortList": "Recent",
"projectManager.removeCurrentProjectFromList": false,
}
```

Rewrap

```
{
  "rewrap.autoWrap.enabled": true,
  "rewrap.reformat": true,
  "rewrap.wrappingColumn": 100,
}
```

Settings Cycler

```
{
  // Use in combination with the Macro settings listed further up this page.
  "settings.cycle": [
    {
      "id": "desktopZoom",
      "values": [
        {
          "window.zoomLevel": 1,
        },
        {
          "window.zoomLevel": 1.4,
        }
      ]
    }
  ],
}
```

Settings Sync

```
{
  "sync.quietSync": true,
// TODO: Add your gist ID
"sync.gist": "00000000000000000000000000000000",
}
```

Todo Tree

```
{
  "todo-tree.highlights.customHighlight": {
      "TODO": {
        "foreground": "#FFEB95",
      },
      "NOTE": {
        "foreground": "#FFEB95",
        "icon": "note",
      },
      "FIXME": {
        "foreground": "#FFEB95",
        "icon": "alert",
      },

     // Clearly mark comments that belong to disabled (commented-out) code.
     "// //": {
       "foreground": "#5d7783",
       "textDecoration": "line-through",
       "type": "text",
       "hideFromTree": true,
     },



},


"todo-tree.tree.grouped": true,
"todo-tree.tree.hideIconsWhenGroupedByTag": true,
"todo-tree.tree.labelFormat": "∙ ${after}",
"todo-tree.tree.showCountsInTree": true,
"todo-tree.tree.showInExplorer": false,
"todo-tree.tree.showScanOpenFilesOrWorkspaceButton": true,
"todo-tree.tree.tagsOnly": true,
"todo-tree.highlights.highlightDelay": 0,
"todo-tree.general.tags": [
"TODO",
"FIXME",
"NOTE",
"// //",
],


// Allow for matches inside of VSCode files (e.g. settings.json).
"todo-tree.highlights.schemes": [
"file",
"untitled",
"vscode-userdata",
],


// Allows for matches inside of JSDoc comments.
"todo-tree.regex.regex": "((\|//|#|<!--|;|/\|)\s*($TAGS)|\s*- \[ \])",
}
```

Bug Reports

Bug reports should be filed at the repository belonging to the individual extension that is causing the issue (click on the extension's marketplace li

Known Issues

Some extensions will prevent the Output Colorizer extension from adding syntax highlighting in the output/debug panels. This is a VSCode limitation wa

Included Extensions

Extension	Link
All Autocomplete	VS Marketplace v0.0.23
Bookmarks	VS Marketplace v11.4.0
Checkpoints	VS Marketplace v1.3.1
Code Runner	VS Marketplace v0.11.1
Code Spell Checker	VS Marketplace v1.9.2
CodeStream	VS Marketplace v10.0.1
Control Snippets	VS Marketplace v1.9.1
Customize UI	VS Marketplace v0.1.49
DotENV	VS Marketplace v1.0.1
Easy Snippet	VS Marketplace v0.6.2
Explorer Exclude	VS Marketplace v1.2.0
Format All	VS Marketplace v1.0.4
GistPad	VS Marketplace v0.1.8
Git Graph	VS Marketplace v1.26.0
GitHub Pull Requests	VS Marketplace v0.20.1
Gitignore	VS Marketplace v0.6.0
GitLens	VS Marketplace v10.2.2
Gremlins	VS Marketplace v0.25.0
Hungry Delete	VS Marketplace v1.6.0
IntelliCode	VS Marketplace v1.2.10
Live Share	VS Marketplace v1.0.3046
Live Share Audio	VS Marketplace v0.1.91
Live Share Chat	VS Marketplace v0.35.0
Live Share Whiteboard	VS Marketplace v0.0.11
Local History	VS Marketplace v1.8.1
Macros	VS Marketplace v0.0.4
MetaGo	VS Marketplace v4.2.0
Monkey Patch	VS Marketplace v0.1.11
Multiple Clipboards	VS Marketplace v0.1.5
Output Colorizer	VS Marketplace v0.1.2
PDF	VS Marketplace v1.1.0
Peacock	VS Marketplace v3.8.0
Polacode	VS Marketplace v0.3.4
Project Manager	

	VS Marketplace v11.3.1
Rewrap	VS Marketplace v1.13.0
Search Editor: Apply Changes	VS Marketplace v1.13.0
Settings Cycler	VS Marketplace v1.0.1
Settings Sync	VS Marketplace v3.4.3
Text Pastry	VS Marketplace v1.2.0
Todo Tree	VS Marketplace v0.0.183

JavaScript

## VS Code JavaScript Unit Test snippets

This extension contains code snippets for Unit Test use describe in JavaScript for Vs Code editor (supports both JavaScript and TypeScript).

Installation

In order to install an extension you need to launch the Command Pallete (Ctrl + Shift + P or Cmd + Shift + P) and type Extensions. There you have eith

Snippets

Below is a list of all available snippets and the triggers of each one. The ⇥ means the TAB key.

Import and export

Trigger	Content
det→	add describe with it describe('{description}',() => { it('{1}', () => {}) });
deb→	add describe describe('{description}',() => {});
dit→	add it it('{1}',() => {});
ete→	add expect expect({object}).toExist();
ene→	add expect expect({object}).toNotExist()
etb→	add expect expect({object}).toBe({value})
enb→	add expect expect({object}).toNotBe({value})
etq→	add expect expect({object}).toEqual({value})
enq→	add expect expect({object}).toNotEqual({value})

Path Autocomplete for Visual Studio Code

Provides path completion for visual studio code.

demo gif

Features

- it supports relative paths (starting with ./)

- it supports absolute path to the workspace (starting with /)
- it supports absolute path to the file system (starts with: C:)
- it supports paths relative to the user folder (starts with ~)
- it supports items exclusions via the `path-autocomplete.excludedItems` option
- it supports npm packages (starting with a-z and not relative to disk)
- it supports automatic suggestion after selecting a folder
- it supports custom mappings via the `path-autocomplete.pathMappings` option
- it supports custom transformations to the inserted text via the `path-autocomplete.transformations`
- it supports windows paths with the `path-autocomplete.useBackslash`

Installation

You can install it from the marketplace. `ext install path-autocomplete`

Options

- `path-autocomplete.extensionOnImport` - boolean If true it will append the extension as well when inserting the file name on `import` or `require` statements.
- `path-autocomplete.includeExtension` - boolean If true it will append the extension as well when inserting the file name.
- `path-autocomplete.excludedItems` This option allows you to exclude certain files from the suggestions. `"path-autocomplete.excludedItems": {         "**/*.js": {`

  minimatch is used to check if the files match the pattern.
- `path-autocomplete.pathMappings` Useful for defining aliases for absolute or relative paths. `"path-autocomplete.pathMappings": {         "/test": "${folder}/src/Ac`
- `path-autocomplete.pathSeparators` - string Lists the separators used for extracting the inserted path when used outside strings. The default value is: \
- `path-autocomplete.transformations` List of custom transformation applied to the inserted text. Example: replace _ with an empty string when selecting a

  Supported transformation:
  - `replace` - Performs a string replace on the selected item text. Parameters:
    - `regex` - a regex pattern
    - `replaceString` - the replacement string
- `path-autocomplete.triggerOutsideStrings` boolean - if true it will trigger the autocomplete outside of quotes
- `path-autocomplete.enableFolderTrailingSlash` boolean - if true it will add a slash after the insertion of a folder path that will trigger the autocomplet
- `path-autocomplete.useBackslash` boolean - if true it will use \\ when iserting the paths.
- `path-autocomplete.ignoredFilesPattern` - string - Glob patterns for disabling the path completion in the specified file types. Example: "**/*.{css,scss}'
- `path-autocomplete.ignoredPrefixes` array - list of ignored prefixes to disable suggestions on certain preceeding words/characters. Example: `js        "pa`

Configure VSCode to recognize path aliases

VSCode doesn't automatically recognize path aliases so you cannot `alt`+`click` to open files. To fix this you need to create `jsconfig.json` or `tsconfig.js`

```
{
  "compilerOptions": {
    "target": "esnext", // define to your liking
```

```
      "baseUrl": "./",
      "paths": {
        "test/*": ["src/actions/test"],
        "assets/*": ["src/assets"]
      }
    },
    "exclude": ["node_modules"] // Optional
  }
```

Tips

- if you want to use this in markdown or simple text files you need to enable `path-autocomplete.triggerOutsideStrings`

- ./ for relative paths > If ./ doesn't work properly, add this to `keybindings.json`: `{ "key": ".", "command": "" }`. Refer to https://github.com/ChristianKol

- When I use aliases I can't jump to imported file on Ctrl + Click > This is controlled by the compiler options in jsconfig.json. You can create the

- if you have issues with duplicate suggestions please use the `path-autocomplete.ignoredFilesPattern` option to disable the path autocomplete in certain fi

- if you need more fine grained control for handing duplicate items you can use the `path-autocomplete.excludedItems` option as follows: ``` // disable all

// for js and typescript you can disable the vscode suggestions using the following options "javascript.suggest.paths": false, "typescript.suggest.pat

easy-snippet

another easy way to manage snippet

Features

- Auto-detects language
- Easily edit and delete snippets once created
- View, edit, and remove snippet groups from a dedicated snippet manager

1 2

default keymaps `cmd+k shift+cmd+s`

Release Notes

0.1.6

change default keymaps `cmd+. cmd+s` to `cmd+k shift+cmd+s`, issue#5

0.1.5

support insiders version, improve the marketplace listing

0.0.1

first release

VSCode Log Output Colorizer

license MIT Version Installs Ratings

Language extension for VSCode/Bluemix Code that adds syntax colorization for both the output/debug/extensions panel and `*.log` files.

Note: If you are using other extensions that colorize the output panel, it could override and disable this extension.

Colorization should work with most themes because it uses common theme token style names. It also works with most instances of the output panel. Initi

Change Log

- **0.1.2** - Updated for compliance with upcoming VS Code marketplace changes
- **0.1.1** - Regex updates for color coding support

Contributing

You can contribute to the project by reading the Contribution guidelines

In action

VSCode Git Output

Colorized Git Output
Colorized Git Output

Default Dark Theme

Default Dark Theme
Default Dark Theme

Default Light Theme

Default Light Theme
Default Light Theme

Night Blue Theme

Night Blue Theme
Night Blue Theme

Helpful References:

- https://code.visualstudio.com/docs/customization/colorizer
- http://stackoverflow.com/questions/33403324/how-to-create-a-simple-custom-language-colorization-to-vs-code

- http://manual.macromates.com/en/language_grammars

Support

You can open an issue on the GitHub repo

License

MIT

Attribution

Portions of the language grammar are based off of a StackOverflow question, asked by user emilast and answered by user Wosi, availble under Creative (

REST Client

Join the chat at https://gitter.im/Huachao/vscode-restclient Marketplace Version Downloads
Rating

REST Client allows you to send HTTP request and view the response in Visual Studio Code directly.

Main Features

- Send/Cancel/Rerun **HTTP request** in editor and view response in a separate pane with syntax highlight
- Send **GraphQL query** and author **GraphQL variables** in editor
- Send **cURL command** in editor and copy HTTP request as `cURL command`
- Auto save and view/clear request history
- Organize *MULTIPLE* requests in the same file (separated by `###` delimiter)
- View image response directly in pane
- Save raw response and response body only to local disk
- Fold and unfold response body
- Customize font(size/family/weight) in response preview
- Preview response with expected parts(*headers only*, *body only*, *full response* and *both request and response*)
- Authentication support for:
  - Basic Auth
  - Digest Auth
  - SSL Client Certificates
  - Azure Active Directory
  - Microsoft Identity Platform
  - AWS Signature v4
- Environments and custom/system variables support

  - Use variables in any place of request(*URL*, *Headers*, *Body*)
  - Support both **environment**, **file** and **request** custom variables
  - Auto completion and hover support for both **environment**, **file** and **request** custom variables
  - Diagnostic support for **request** and **file** custom variables
  - Go to definition support for **request** and **file** custom variables
  - Find all references support *ONLY* for **file** custom variables
  - Provide system dynamic variables
  - `{{$guid}}`
  - `{{$randomInt min max}}`
  - `{{$timestamp [offset option]}}`
  - `{{$datetime rfc1123|iso8601 [offset option]}}`
  - `{{$localDatetime rfc1123|iso8601 [offset option]}}`
  - `{{$processEnv [%]envVarName}}`
  - `{{$dotenv [%]variableName}}`
  - `{{$aadToken [new] [public|cn|de|us|ppe] [<domain|tenantId>] [aud:<domain|tenantId>]}}`
  - Easily create/update/delete environments and environment variables in setting file
  - File variables can reference both custom and system variables
  - Support environment switch
  - Support shared environment to provide variables that available in all environments

- Generate code snippets for **HTTP request** in languages like `Python`, `JavaScript` and more!
- Remember Cookies for subsequent requests
- Proxy support
- Send SOAP requests, as well as snippet support to build SOAP envelope easily
- `HTTP` language support
  - `.http` and `.rest` file extensions support
  - Syntax highlight (Request and Response)
  - Auto completion for method, url, header, custom/system variables, mime types and so on
  - Comments (line starts with `#` or `//`) support
  - Support `json` and `xml` body indentation, comment shortcut and auto closing brackets
  - Code snippets for operations like `GET` and `POST`
  - Support navigate to symbol definitions(request and file level custom variable) in open `http` file
  - CodeLens support to add an actionable link to send request
  - Fold/Unfold for request block

Usage

In editor, type an HTTP request as simple as below:

```
https://example.com/comments/1
```

Or, you can follow the standard RFC 2616 that including request method, headers, and body.

```
POST https://example.com/comments HTTP/1.1
content-type: application/json
{
"name": "sample",
"time": "Wed, 21 Oct 2015 18:27:50 GMT"
}
```

Once you prepared a request, click the `Send Request` link above the request (this will appear if the file's language mode is `HTTP`, by default `.http` files

You can view the breakdown of the response time when hovering over the total duration in status bar, you could view the duration details of *Socket*, *DN*

When hovering over the response size in status bar, you could view the breakdown response size details of *headers* and *body*.

> All the shortcuts in REST Client Extension are **ONLY** available for file language mode `http` and `plaintext`.

> **Send Request** link above each request will only be visible when the request file is in `http` mode, more details can be found in http language section.

Select Request Text

You may even want to save numerous requests in the same file and execute any of them as you wish easily. REST Client extension could recognize request

```
GET https://example.com/comments/1 HTTP/1.1
```

```
GET https://example.com/topics/1 HTTP/1.1
```

```
POST https://example.com/comments HTTP/1.1
content-type: application/json

{
"name": "sample",
"time": "Wed, 21 Oct 2015 18:27:50 GMT"
}
```

REST Client extension also provides the flexibility that you can send the request with your selected text in editor.

Install

Press `F1`, type `ext install` then search for `rest-client`.

Making Request

rest-client ### Request Line The first non-empty line of the selection (or document if nothing is selected) is the *Request Line*. Below are some exa

```
GET https://example.com/comments/1 HTTP/1.1
```

```
GET https://example.com/comments/1
```

```
https://example.com/comments/1
```

If request method is omitted, request will be treated as **GET**, so above requests are the same after parsing.

Query Strings

You can always write query strings in the request line, like:

```
GET https://example.com/comments?page=2&pageSize=10
```

Sometimes there may be several query parameters in a single request, putting all the query parameters in *Request Line* is difficult to read and modify.

```
GET https://example.com/comments
    ?page=2
    &pageSize=10
```

Request Headers

The lines immediately after the *request line* to first empty line are parsed as *Request Headers*. Please provide headers with the standard `field-name: fie`

```
User-Agent: rest-client
Accept-Language: en-GB,en-US;q=0.8,en;q=0.6,zh-CN;q=0.4
Content-Type: application/json
```

Request Body

If you want to provide the request body, please add a blank line after the request headers like the POST example in usage, and all content after it wi

```
POST https://example.com/comments HTTP/1.1
Content-Type: application/xml
Authorization: token xxx
<request>
<name>sample</name>
<time>Wed, 21 Oct 2015 18:27:50 GMT</time>
</request>
```

You can also specify file path to use as a body, which starts with `<`, the file path(*whitespaces* should be preserved) can be either in absolute or rela

```
POST https://example.com/comments HTTP/1.1
Content-Type: application/xml
Authorization: token xxx
< C:\Users\Default\Desktop\demo.xml
```

```
POST https://example.com/comments HTTP/1.1
Content-Type: application/xml
Authorization: token xxx

< ./demo.xml
```

If you want to use variables in that file, you'll have to use an `@` to ensure variables are processed when referencing a file (UTF-8 is assumed as the

```
POST https://example.com/comments HTTP/1.1
Content-Type: application/xml
Authorization: token xxx
<@ ./demo.xml
```

to override the default encoding, simply type it next to the `@` like the below example

```
POST https://example.com/comments HTTP/1.1
Content-Type: application/xml
Authorization: token xxx
<@latin1 ./demo.xml
```

When content type of request body is `multipart/form-data`, you may have the mixed format of the request body as follows:

```
POST https://api.example.com/user/upload
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="text"


title
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="image"; filename="1.png"
Content-Type: image/png


< ./1.png
------WebKitFormBoundary7MA4YWxkTrZu0gW--
```

When content type of request body is `application/x-www-form-urlencoded`, you may even divide the request body into multiple lines. And each key and value pa

```
POST https://api.example.com/login HTTP/1.1
Content-Type: application/x-www-form-urlencoded
name=foo
&password=bar
```

> When your mouse is over the document link, you can `Ctrl+Click`(`Cmd+Click` for macOS) to open the file in a new tab.

Making GraphQL Request

With GraphQL support in REST Client extension, you can author and send `GraphQL` query using the request body. Besides that you can also author GraphQL v

You can specify a request as `GraphQL Request` by adding a custom request header `X-Request-Type: GraphQL` in your headers. The following code illustrates this

```
POST https://api.github.com/graphql
Content-Type: application/json
Authorization: Bearer xxx
X-REQUEST-TYPE: GraphQL
query ($name: String!, $owner: String!) {
repository(name: $name, owner: $owner) {
name
fullName: nameWithOwner
description
diskUsage
forkCount
stargazers(first: 5) {
totalCount
nodes {
login
name
}
}
watchers {
totalCount
}
}
}


{
"name": "vscode-restclient",
"owner": "Huachao"
}
```

Making cURL Request

cURL Request We add the capability to directly run curl request in REST Client extension. The issuing request command is the same as raw HTTP one.

`REST Client` doesn't fully support all the options of `cURL`, since underneath we use `request` library to send request which doesn't accept all the `cURL` optic

Copy Request As cURL

Sometimes you may want to get the curl format of an http request quickly and save it to clipboard, just pressing `F1` and then selecting/typing `Rest Clie`

Cancel Request

Once you want to cancel a processing request, click the waiting spin icon or use shortcut `Ctrl+Alt+K`(`Cmd+Alt+K` for macOS), or press `F1` and then select/ty

Rerun Last Request

Sometimes you may want to refresh the API response, now you could do it simply using shortcut `Ctrl+Alt+L`(`Cmd+Alt+L` for macOS), or press `F1` and then selec

Request History

request-history Each time we sent an http request, the request details(method, url, headers, and body) would be persisted into file. By using short

You can also clear request history by pressing `F1` and then selecting/typing `Rest Client: Clear Request History`.

Save Full Response

Save Response In the upper right corner of the response preview tab, we add a new icon to save the latest response to local file system. After you

Save Response Body

Another icon in the upper right corner of the response preview tab is the `Save Response Body` button, it will only save the response body **ONLY** to local f

```
"rest-client.mimeAndFileExtensionMapping": {
    "application/atom+xml": "xml"
}
```

Fold and Unfold Response Body

In the response webview panel, there are two options `Fold Response` and `Unfold Response` after clicking the `More Actions...` button. Sometimes you may want to

Authentication

We have supported some most common authentication schemes like *Basic Auth*, *Digest Auth*, *SSL Client Certificates*, *Azure Active Directory(Azure AD)* and

Basic Auth

HTTP Basic Auth is a widely used protocol for simple username/password authentication. We support **three** formats of Authorization header to use Basic A

The corresponding examples are as follows, they are equivalent:

```
GET https://httpbin.org/basic-auth/user/passwd HTTP/1.1
Authorization: Basic user:passwd
```

and

```
GET https://httpbin.org/basic-auth/user/passwd HTTP/1.1
Authorization: Basic dXNlcjpwYXNzd2Q=
```

and

```
GET https://httpbin.org/basic-auth/user/passwd HTTP/1.1
Authorization: Basic user passwd
```

Digest Auth

HTTP Digest Auth is also a username/password authentication protocol that aims to be slightly safer than Basic Auth. The format of Authorization heade

```
GET https://httpbin.org/digest-auth/auth/user/passwd
Authorization: Digest user passwd
```

SSL Client Certificates

We support `PFX`, `PKCS12`, and `PEM` certificates. Before using your certificates, you need to set the certificates paths(absolute/relative to workspace/rel

```
"rest-client.certificates": {
    "localhost:8081": {
        "cert": "/Users/demo/Certificates/client.crt",
        "key": "/Users/demo/Keys/client.key"
    },
    "example.com": {
        "cert": "/Users/demo/Certificates/client.crt",
        "key": "/Users/demo/Keys/client.key"
    }
}
```

Or if you have certificate in `PFX` or `PKCS12` format, setting code can be like this:

```
"rest-client.certificates": {
    "localhost:8081": {
        "pfx": "/Users/demo/Certificates/clientcert.p12",
        "passphrase": "123456"
    }
}
```

Azure Active Directory(Azure AD)

Azure AD is Microsoft's multi-tenant, cloud-based directory and identity management service, you can refer to the System Variables section for more de

Microsoft Identity Platform(Azure AD V2)

Microsoft identity platform is an evolution of the Azure Active Directory (Azure AD) developer platform. It allows developers to build applications th

AWS Signature v4

AWS Signature version 4 authenticates requests to AWS services. To use it you need to set the Authorization header schema to `AWS` and provide your AWS

```
GET https://httpbin.org/aws-auth HTTP/1.1
Authorization: AWS <accessId> <accessKey> [token:<sessionToken>] [region:<regionName>] [service:<serviceName>]
```

Generate Code Snippet

Generate Code Snippet Once you've finalized your request in REST Client extension, you might want to make the same request from your source code. W

HTTP Language

Add language support for HTTP request, with features like **syntax highlight**, **auto completion**, **code lens** and **comment support**, when writing HTTP request

1. File with extension `.http` or `.rest`

2. First line of file follows standard request line in RFC 2616, with `Method SP Request-URI SP HTTP-Version` format

If you want to enable language association in other cases, just change the language mode in the right bottom of `Visual Studio Code` to `HTTP`.

HTTP Language ### Auto Completion Currently, auto completion will be enabled for following seven categories:

1. HTTP Method
2. HTTP URL from request history
3. HTTP Header
4. System variables
5. Custom variables in current environment/file/request
6. MIME Types for `Accept` and `Content-Type` headers
7. Authentication scheme for `Basic` and `Digest`

Navigate to Symbols in Request File

A single `http` file may define lots of requests and file level custom variables, it will be difficult to find the request/variable you want. We leverag
Goto Symbols

Environments

Environments give you the ability to customize requests using variables, and you can easily switch environment without changing requests in `http` file.

Environments and including variables are defined directly in `Visual Studio Code` setting file, so you can create/update/delete environments and variables

Variables

We support two types of variables, one is **Custom Variables** which is defined by user and can be further divided into **Environment Variables, File Variab**

The reference syntax of system and custom variables types has a subtle difference, for the former the syntax is `{{$SystemVariableName}}`, while for the la

Custom Variables

Custom variables can cover different user scenarios with the benefit of environment variables, file variables, and request variables. Environment vari

Environment Variables

For environment variables, each environment comprises a set of key value pairs defined in setting file, key and value are variable name and value resp

```
"rest-client.environmentVariables": {
    "$shared": {
        "version": "v1",
        "prodToken": "foo",
        "nonProdToken": "bar"
    },
    "local": {
        "version": "v2",
        "host": "localhost",
        "token": "{{$shared nonProdToken}}",
        "secretKey": "devSecret"
    },
    "production": {
        "host": "example.com",
        "token": "{{$shared prodToken}}",
        "secretKey" : "prodSecret"
    }
}
```

A sample usage in `http` file for above environment variables is listed below, note that if you switch to *local* environment, the `version` would be *v2*, if

```
GET https://{{host}}/api/{{version}}comments/1 HTTP/1.1
Authorization: {{token}}
```

File Variables

For file variables, the definition follows syntax `@variableName = variableValue` which occupies a complete line. And variable name **MUST NOT** contain any spa

File variables can be defined in a separate request block only filled with variable definitions, as well as define request variables before any reques

```
@hostname = api.example.com
@port = 8080
@host = {{hostname}}:{{port}}
@contentType = application/json
@createdAt = {{$datetime iso8601}}
@modifiedBy = {{$processEnv USERNAME}}


@name = hello


GET https://{{host}}/authors/{{name}} HTTP/1.1




PATCH https://{{host}}/authors/{{name}} HTTP/1.1
Content-Type: {{contentType}}


{
"content": "foo bar",
"created_at": "{{createdAt}}",
"modified_by": "{{modifiedBy}}"
}
```

Request Variables

Request variables are similar to file variables in some aspects like scope and definition location. However, they have some obvious differences. The d

The reference syntax of a request variable is a bit more complex than other kinds of custom variables. The request variable reference syntax follows {

> If the *JSONPath* or *XPath* of body, or *Header Name* of headers can't be resolved, the plain text of variable reference will be sent instead. And in th

Below is a sample of request variable definitions and references in an `http` file.

```
@baseUrl = https://example.com/api
```

@name login

```
POST {{baseUrl}}/api/login HTTP/1.1
Content-Type: application/x-www-form-urlencoded


name=foo&password=bar




@authToken = {{login.response.headers.X-AuthToken}}
```

@name createComment

```
POST {{baseUrl}}/comments HTTP/1.1
Authorization: {{authToken}}
Content-Type: application/json


{
"content": "fake content"
}
```

```
@commentId = {{createComment.response.body.$.id}}
```

@name getCreatedComment

```
GET {{baseUrl}}/comments/{{commentId}} HTTP/1.1
Authorization: {{authToken}}
```

@name getReplies

```
GET {{baseUrl}}/comments/{{commentId}}/replies HTTP/1.1
Accept: application/xml
```

@name getFirstReply

```
GET {{baseUrl}}/comments/{{commentId}}/replies/{{getReplies.response.body.//reply[1]/@id}}
```

System Variables

System variables provide a pre-defined set of variables that can be used in any part of the request(Url/Headers/Body) in the format `{{$variableName}}`. C

`new`: Optional. Specify `new` to force re-authentication and get a new token for the specified directory. Default: Reuse previous token for the specified

`public|cn|de|us|ppe`: Optional. Specify top-level domain (TLD) to get a token for the specified government cloud, `public` for the public cloud, or `ppe` for i

`<domain|tenantId>`: Optional. Domain or tenant id for the directory to sign in to. Default: Pick a directory from a drop-down or press `Esc` to use the home

`aud:<domain|tenantId>`: Optional. Target Azure AD app id (aka client id) or domain the token should be created for (aka audience or resource). Default: D

`new`: Optional. Specify `new` to force re-authentication and get a new token for the specified directory. Default: Reuse previous token for the specified

`appOnly`: Optional. Specify `appOnly` to use make to use a client credentials flow to obtain a token. `aadV2ClientSecret` and `aadV2AppUri`must be provided as REST

`scopes:<scope[,]>`: Optional. Comma delimited list of scopes that must have consent to allow the call to be successful. Not applicable for `appOnly` calls.

`tenantId:<domain|tenantId>`: Optional. Domain or tenant id for the tenant to sign in to. (`common` to determine tenant from sign in).

`clientId:<clientid>`: Optional. Identifier of the application registration to use to obtain the token. Default uses an application registration created s

- `{{$guid}}`: Add a RFC 4122 v4 UUID
- `{{$processEnv [%]envVarName}}`: Allows the resolution of a local machine environment variable to a string value. A typical use case is for secret keys

You can refer directly to the key (e.g. `PRODSECRET`) in the script, for example if running in the production environment `http  # Lookup PRODSECRET from local`

`%`: Optional. If specified, treats envVarName as an extension setting environment variable, and uses the value of that for the lookup.

- `{{$dotenv [%]variableName}}`: Returns the environment value stored in the `.env` file which exists in the same directory of your `.http` file.
- `{{$randomInt min max}}`: Returns a random integer between min (included) and max (excluded)
- `{{$timestamp [offset option]}}`: Add UTC timestamp of now. You can even specify any date time based on current time in the format `{{$timestamp number optio`
- `{{$datetime rfc1123|iso8601|"custom format"|'custom format' [offset option]}}`: Add a datetime string in either *ISO8601*, *RFC1123* or a custom display format. Y

- `{{$localDatetime rfc1123|iso8601|"custom format"|'custom format' [offset option]}}`: Similar to `$datetime` except that `$localDatetime` returns a time string in your

The offset options you can specify in `timestamp` and `datetime` are:

Option	Description
y	Year
M	Month
w	Week
d	Day
h	Hour
m	Minute
s	Second
ms	Millisecond

Below is a example using system variables:

```
POST https://api.example.com/comments HTTP/1.1
Content-Type: application/xml
Date: {{$datetime rfc1123}}
{
"user_name": "{{dotenv USERNAME}}",
    "request_id": "{{guid}}",
"updated_at": "{{timestamp}}",
    "created_at": "{{timestamp -1 d}}",
"review_count": "{{randomInt 5 200}}",
    "custom_date": "{{datetime 'YYYY-MM-DD'}}",
"local_custom_date": "{{$localDatetime 'YYYY-MM-DD'}}"
}
```

> More details about `aadToken` (Azure Active Directory Token) can be found on Wiki

Customize Response Preview

REST Client Extension adds the ability to control the font family, size and weight used in the response preview.

By default, REST Client Extension only previews the full response in preview panel(*status line*, *headers* and *body*). You can control which part should b

Option	Description
full	Default. Full response is previewed
headers	Only the response headers(including *status line*) are previewed
body	Only the response body is previewed
exchange	Preview the whole HTTP exchange(request and response)

Settings

- `rest-client.followredirect`: Follow HTTP 3xx responses as redirects. (Default is **true**)
- `rest-client.defaultHeaders`: If particular headers are omitted in request header, these will be added as headers for each request. (Default is `{ "User-A`
- `rest-client.timeoutinmilliseconds`: Timeout in milliseconds. 0 for infinity. (Default is **0**)
- `rest-client.showResponseInDifferentTab`: Show response in different tab. (Default is **false**)
- `rest-client.requestNameAsResponseTabTitle`: Show request name as the response tab title. Only valid when using html view, if no request name is specified
- `rest-client.rememberCookiesForSubsequentRequests`: Save cookies from `Set-Cookie` header in response and use for subsequent requests. (Default is **true**)

- `rest-client.enableTelemetry`: Send out anonymous usage data. (Default is **true**)
- `rest-client.excludeHostsForProxy`: Excluded hosts when using proxy settings. (Default is **[]**)
- `rest-client.fontSize`: Controls the font size in pixels used in the response preview. (Default is **13**)
- `rest-client.fontFamily`: Controls the font family used in the response preview. (Default is **Menlo, Monaco, Consolas, "Droid Sans Mono", "Courier New",**
- `rest-client.fontWeight`: Controls the font weight used in the response preview. (Default is **normal**)
- `rest-client.environmentVariables`: Sets the environments and custom variables belongs to it (e.g., `{"production": {"host": "api.example.com"}, "sandbox":{"host":`
- `rest-client.mimeAndFileExtensionMapping`: Sets the custom mapping of mime type and file extension of saved response body. (Default is **{}**)
- `rest-client.previewResponseInUntitledDocument`: Preview response in untitled document if set to true, otherwise displayed in html view. (Default is **false**)
- `rest-client.certificates`: Certificate paths for different hosts. The path can be absolute path or relative path(relative to workspace or current http
- `rest-client.suppressResponseBodyContentTypeValidationWarning`: Suppress response body content type validation. (Default is **false**)
- `rest-client.previewOption`: Response preview output option. Option details is described above. (Default is **full**)
- `rest-client.disableHighlightResonseBodyForLargeResponse`: Controls whether to highlight response body for response whose size is larger than limit specifie
- `rest-client.disableAddingHrefLinkForLargeResponse`: Controls whether to add href link in previewed response for response whose size is larger than limit s
- `rest-client.largeResponseBodySizeLimitInMB`: Set the response body size threshold of MB to identify whether a response is a so-called 'large response', o
- `rest-client.previewColumn`: Response preview column option. 'current' for previewing in the column of current request file. 'beside' for previewing at
- `rest-client.previewResponsePanelTakeFocus`: Preview response panel will take focus after receiving response. (Default is **True**)
- `rest-client.formParamEncodingStrategy`: Form param encoding strategy for request body of *x-www-form-urlencoded*. `automatic` for detecting encoding or not au
- `rest-client.addRequestBodyLineIndentationAroundBrackets`: Add line indentation around brackets(`{}`, `<>`, `[]`) in request body when pressing enter. (Default is
- `rest-client.decodeEscapedUnicodeCharacters`: Decode escaped unicode characters in response body. (Default is **false**)
- `rest-client.logLevel`: The verbosity of logging in the REST output panel. (Default is **error**)
- `rest-client.enableSendRequestCodeLens`: Enable/disable sending request CodeLens in request file. (Default is **true**)
- `rest-client.enableCustomVariableReferencesCodeLens`: Enable/disable custom variable references CodeLens in request file. (Default is **true**)

Rest Client extension respects the proxy settings made for Visual Studio Code (`http.proxy` and `http.proxyStrictSSL`). Only HTTP and HTTPS proxies are suppor

Per-request Settings

REST Client Extension also supports request-level settings for each independent request. The syntax is similar with the request name definition, `# @se`

Name	Syntax	Description
note	`# @note`	Use for request confirmation, especially for critical request

> All the above leading `#` can be replaced with `//`

License

MIT License

Change Log

See CHANGELOG here

Special Thanks

All the amazing contributors♡

Feedback

Please provide feedback through the GitHub Issue system, or fork the repository and submit PR.

vscode-favorites

vscode version

An extension that lets the developer mark resources (files or folders) as favorites, so they can be easily accessed.

Install

Launch VS Code Quick Open (`cmd/ctrl + p`), paste the following command, and press Enter.

```
ext install howardzuo.vscode-favorites
```

Usage

An **Add to Favorites** command in Explorer's context menu saves links to your favorite files or folders into your *XYZ*`.code-workspace` file if you are using

Your favorites are listed in a separate view and can be quickly accessed from there.

Configuration

```
{
    "favorites.resources": [], // resources path you prefer to mark
    "favorites.sortOrder": "ASC", // DESC, MANUAL
    "favorites.saveSeparated": false // whether to use an extra config file
}
```

> You normally don't need to modify this config manually. Use context menus instead.

Changelog

Changelog on Marketplace

LICENSE

GPL v3 License

js-beautify for VS Code

build passing Build status Licence VS Code Marketplace Rating Downloads Installs Donate

Beautify `javascript`, `JSON`, `CSS`, `Sass`, and `HTML` in Visual Studio Code.

VS Code uses js-beautify internally, but it lacks the ability to modify the style you wish to use. This extension enables running js-beautify in VS Co

For help on the settings in the `.jsbeautifyrc` see Settings.md

How we determine what settings to use

1. When not using a multi-root workspace:
    1. If there is a valid `.jsbeautifyrc` in the file's path tree, up to project root, these will be the only settings used.
    2. If an option is a file path or object of configuration specified in the user or workspace settings like this: `"beautify.config" : "string|Object.<st`
    3. If there is a valid `.jsbeautifyrc` in the file's path tree, above project root, these will be the only settings used.
    4. If there is a valid `.jsbeautifyrc` in your home directory, these will be the only settings used.
2. When using a multi-root workspace: Same as above, but the search ends at the nearest parent workspace root to the open file.

otherwise...

3. Settings are translated from your VS Code workspace/user settings.
4. Any open editor settings (indent spaces/tabs) for the specific file are merged in.
5. Editorconfig settings are searched for (See http://editorconfig.org/) and are merged in.

VS Code | .jsbeautifyrc settings map

.jsbeautifyrc setting	VS Code setting
eol	files.eol
tab_size	editor.tabSize
indent_with_tabs *(inverted)*	editor.insertSpaces
wrap_line_length	html.format.wrapLineLength
wrap_attributes	html.format.wrapAttributes
unformatted	html.format.unformatted
indent_inner_html	html.format.indentInnerHtml
preserve_newlines	html.format.preserveNewLines
max_preserve_newlines	html.format.maxPreserveNewLines
indent_handlebars	html.format.indentHandlebars
end_with_newline	html.format.endWithNewline (html)
end_with_newline	file.insertFinalNewLine (css, js)
extra_liners	html.format.extraLiners
space_after_anon_function	javascript.format .insertSpaceAfterFunctionKeywordForAnonymousFunctions
space_in_paren	javascript.format .insertSpaceAfterOpeningAndBeforeClosingNonemptyParenthesis

Note that the `html.format` settings will ONLY be used when the document is html. `javascript.format` settings are included always.

Also runs html and css beautify from the same package, as determined by the file extension. The schema indicates which beautifier each of the setting

The `.jsbeautifyrc` config parser accepts sub elements of `html`, `js` and `css` so that different settings can be used for each of the beautifiers (like sublime

Settings are inherited from the base of the file. Thus:

```
{
  "indent_size": 4,
  "indent_char": " ",
  "css": {
    "indent_size": 2
  }
}
```

Will result in the `indent_size` being set to 4 for JavaScript and HTML, but set to 2 for CSS. All will get the same `indent_char` setting.

If the file is unsaved, or the type is undetermined, you'll be prompted for which beautifier to use.

You can control which file types, extensions, or specific file names should be beautified with the `beautify.language` setting.

```
{
  "beautify.language": {
    "js": {
      "type": ["javascript", "json"],
      "filename": [".jshintrc", ".jsbeautifyrc"]
      // "ext": ["js", "json"]
      // ^^ to set extensions to be beautified using the javascript beautifier
    },
    "css": ["css", "scss"],
    "html": ["htm", "html"]
    // ^^ providing just an array sets the VS Code file type
  }
}
```

Beautify on save will be enabled when `"editor.formatOnSave"` is true.

Beautification on particular files using the built in **Format Document** (which includes formatting on save) can be skipped with the `beautify.ignore` option

Examples:

```
/* ignore all files named 'test.js' not in the root folder,
   all files directly in any 'spec' directory, and
   all files in any 'test' directory at any depth
*/
"beautify.ignore": ["*/test.js", "**/spec/*", "**/test/**/*"]
/* ignore all files ending in '_test.js' anywhere /
"beautify.ignore": "**/_test.js"
```

Note that the glob patterns are not used to test against the containing folder. You must match the filename as well.

Embedded version of js-beautify is v1.8.4

Keyboard Shortcut

Use the following to embed a beautify shortcut in keybindings.json. Replace with your preferred key bindings.

```
{
  "key": "cmd+b",
  "command": "HookyQR.beautify",
  "when": "editorFocus"
}
```

Contributing

For information on contributing see Contributing.md

Todo Tree

build passing

This extension quickly searches (using ripgrep your workspace for comment tags like TODO and FIXME, and displays them in a tree view in the explorer

Found TODOs can also be highlighted in open files.

Please see the wiki for configuration examples.

screenshot
screenshot

Notes:

- *The tree will only appear in the explorer pane when the extension finds some TODOs, unless* `todo-tree.tree.hideTreeWhenEmpty` *is set to false.*
- *User* `rg.conf` *files are ignored.*

Highlighting

Highlighting tags is configurable. Use `defaultHighlight` to set up highlights for all tags. If you need to configure individual tags differently, use `cust`

Both `defaultHighlight` and `customHighlight` allow for the following settings:

`foreground` - used to set the foreground colour of the highlight in the editor and the marker in the ruler.

`background` - used to set the background colour of the highlight in the editor.

Note: Foreground and background colours can be specified using HTML/CSS colour names (e.g. "Salmon"), RGB hex values (e.g. "#80FF00"), RGB CSS style v

`opacity` - percentage value used with the background colour. 100% will produce an opaque background which will obscure selection and other decorations.

`fontWeight`, `fontStyle`, `textDecoration` - can be used to style the highlight with standard CSS values.

`borderRadius` - used to set the border radius of the background of the highlight.

`icon` - used to set a different icon in the tree view. Must be a valid octicon (see https://octicons.github.com) or codicon (see https://microsoft.gith

`iconColour` - used to set the colour of the icon in the tree. If not specified, it will try to use the foreground colour or the background colour. Colou

`gutterIcon` - set to true to show the icon in the editor gutter.

`rulerColour` - used to set the colour of the marker in the overview ruler. If not specified, it will default to use the foreground colour. Colour can be

`rulerLane` - used to set the lane for the marker in the overview ruler. If not specified, it will default to the right hand lane. Use one of "left", "ce

`type` - used to control how much is highlighted in the editor. Valid values are:

- `tag` - highlights just the tag
- `text` - highlights the tag and any text after the tag
- `tag-and-comment` - highlights the comment characters (or the start of the match) and the tag
- `text-and-comment` - highlights the comment characters (or the start of the match), the tag and the text after the tag
- `line` - highlights the entire line containing the tag
- `whole-line` - highlights the entire line containing the tag to the full width of the editor

`hideFromTree` - used to hide tags from the tree, but still highlight in files

`hideFromStatusBar` - prevents the tag from being included in the status bar counts

Example:

```
    "todo-tree.highlights.defaultHighlight": {
        "icon": "alert",
        "type": "text",
        "foreground": "red",
        "background": "white",
        "opacity": 50,
        "iconColour": "blue"
    },
    "todo-tree.highlights.customHighlight": {
        "TODO": {
            "icon": "check",
            "type": "line"
        },
        "FIXME": {
            "foreground": "black",
            "iconColour": "yellow",
            "gutterIcon": true
        }
    }
```

Note: The highlight configuration is separate from the settings for the search. Adding settings in `customHighlight` does not automatically add the tags i

Installing

You can install the latest version of the extension via the Visual Studio Marketplace here.

Alternatively, open Visual Studio code, press `Ctrl+P` or `Cmd+P` and type:

```
    > ext install Gruntfuggly.todo-tree
```

Note: Don't forget to reload the window to activate the extension!

Source Code

The source code is available on GitHub here.

Controls

The tree view header can contain the following buttons:

collapse - Collapse all tree nodes
expand - Expand all tree nodes
flat - Show the tree view as a flat list, with the full filename for each TODO
tags - Show the view as a list of tags
tree - Show the tree view as a tree with expandable nodes for each folder (default)
tag - Group the TODOs in the tree by the tag
notag - Organise the TODOs by file (default)
filter - Only show items in the tree which match the entered filter text
clear-filter - Remove any active filter
refresh - Rebuild the tree
scan-open-files - Show tags from open files only
scan-workspace - Show tags from workspace
scan-current-file - Show the current file in the tree

Folder Filter Context Menu

Right clicking on a folder in the tree will bring up a context menu with the following options:

Hide This Folder - removes the folder from the tree

Only Show This Folder - remove all other folders and subfolders from the tree

Only Show This Folder And Subfolders - remove other folders from the tree, but keep subfolders

Reset Folder Filter - reset any folders previously filtered using the above

*Note: The current filters are shown in the debug log. Also, the filter can always be reset by right clicking the **Nothing Found** item in the tree. If yo*

Commands

Tags

To make it easier to configure the tags, there are two commands available:

Todo Tree: Add Tag - allows entry of a new tag for searching

Todo Tree: Remove Tag - shows a list of current tags which can be selected for removing

Note: The Remove Tag command can be used to show current tags - just press Escape or Enter with out selecting any to close it.

Export

The contents of the tree can be exported using **Todo Tree: Export Tree**. A read-only file will be created using the path specified with `todo-tree.general.`

Configuration

The extension can be customised as follows (default values in brackets):

todo-tree.general.debug (`false`)
Show a debug channel in the output view.

todo-tree.general.enableFileWatcher (`false`)
Set this to true to turn on automatic updates when files in the workspace are created, changed or deleted.

todo-tree.general.exportPath (`~/todo-tree-%Y%m%d-%H%M.txt`)
Path to use when exporting the tree. Environment variables will be expanded, e.g `${HOME}` and the path is passed through strftime (see https://github.c

todo-tree.general.rootFolder (`""`)
By default, any open workspaces will have a tree in the view. Use this to force another folder to be the root of the tree. You can include environmen
Note: Other open files (outside of the rootFolder) will be shown (as they are opened) with their full path in brackets.

todo-tree.general.tags (`["TODO","FIXME","BUG"]`)
This defines the tags which are recognised as TODOs. This list is automatically inserted into the regex.

todo-tree.general.tagGroups (`{}`)
This setting allows multiple tags to be treated as a single group. Example:

```
"todo-tree.general.tagGroups": {
    "FIXME": [
        "FIXME",
        "FIXIT",
        "FIX",
    ]
},
```

This treats any of `FIXME`, `FIXIT` or `FIX` as `FIXME`. When the tree is grouped by tag, all of these will appear under the `FIXME` node. This also means that cus

todo-tree.general.revealBehaviour (`start of todo`)
Change the cursor behaviour when selecting a todo from the explorer. Yo.u can choose from: `start of todo` (moves the cursor to the beginning of the todo

todo-tree.general.statusBar (`none`)
What to show in the status bar - nothing (`none`), total count (`total`), counts per tag (`tags`), counts for the top three tags (`top three`) or counts for the

todo-tree.general.statusBarClickBehaviour (`cycle`)
Set the behaviour of clicking the status bar to either cycle display formats, or reveal the tree.

todo-tree.filtering.includeGlobs (`[]`)
Globs for use in limiting search results by inclusion, e.g. `[\"**/unit-tests/*.js\"]` to only show .js files in unit-tests subfolders. Globs help. *Note:*

todo-tree.filtering.excludeGlobs (`[]`)
Globs for use in limiting search results by exclusion (applied after **includeGlobs**), e.g. `[\"**/*.txt\"]` to ignore all .txt files

todo-tree.filtering.includedWorkspaces (`[]`)
A list of workspace names to include as roots in the tree (wildcards can be used). An empty array includes all workspace folders.

todo-tree.filtering.excludedWorkspaces (`[]`)
A list of workspace names to exclude as roots in the tree (wildcards can be used).

todo-tree.filtering.passGlobsToRipgrep (`true`)
Set this to false to apply the globs *after* the search (legacy behaviour).

todo-tree.filtering.useBuiltInExcludes (`none`)
Set this to use VSCode's built in files or search excludes. Can be one of `none`, `file excludes` (uses Files:Exclude), `search excludes` (Uses Search:Exclude)

todo-tree.filtering.ignoreGitSubmodules (`false`)
If true, any subfolders containing a `.git` file will be ignored when searching.

todo-tree.filtering.includeHiddenFiles (`false`)
If true, files starting with a period (.) will be included.

todo-tree.highlights.enabled (`true`)
Set this to false to turn off highlighting.

todo-tree.highlights.highlightDelay (`500`)
The delay before highlighting (milliseconds).

todo-tree.highlights.defaultHighlight (`{}`)
Set default highlights. Example:

```
{
    "foreground": "white",
    "background": "red",
    "icon": "check",
    "type": "text"
}
```

todo-tree.highlights.customHighlight (`{}`)
Set highlights per tag (or tag group). Example:

```
{
    "TODO": {
        "foreground": "white",
        "type": "text"
    },
    "FIXME": {
        "icon": "beaker"
    }
}
```

todo-tree.highlights.schemes (`['file','untitled']`)
Editor schemes to show highlights in. To show highlights in settings files, for instance, add `vscode-userdata` or for output windows, add `output`.

todo-tree.regex.regex (`"((//|#|<!--|;|/\\*)\\s*($TAGS)|^\\s*- \\[ \\])"`)
This defines the regex used to locate TODOs. By default, it searches for tags in comments starting with `//`, `#`, `;`, `<!--` or `/`. *This should cover most*

todo-tree.regex.regexCaseSensitive (`true`)
Set to false to allow tags to be matched regardless of case.

todo-tree.ripgrep.ripgrep (`""`)
Normally, the extension will locate ripgrep itself as and when required. If you want to use an alternate version of ripgrep, set this to point to whe

todo-tree.ripgrep.ripgrepArgs (`"--max-columns=1000"`)
Use this to pass additional arguments to ripgrep. e.g. `"-i"` to make the search case insensitive. *Use with caution!*

todo-tree.ripgrep.ripgrepMaxBuffer (`200`)
By default, the ripgrep process will have a buffer of 200KB. However, this is sometimes not enough for all the tags you might want to see. This setti

todo-tree.tree.showInExplorer (`true`)
The tree is shown in the explorer view and also has it's own view in the activity bar. If you no longer want to see it in the explorer view, set this

todo-tree.tree.hideTreeWhenEmpty (`true`)
Normally, the tree is removed from the explorer view if nothing is found. Set this to false to keep the view present.

todo-tree.tree.filterCaseSensitive (`false`)
Use this if you need the filtering to be case sensitive. *Note: this does not the apply to the search.*

todo-tree.tree.trackFile (`true`)
Set to false if you want to prevent tracking the open file in the tree view.

todo-tree.tree.showBadges (`true`)
Set to false to disable SCM status and badges in the tree. Note: This also unfortunately turns off themed icons.

todo-tree.tree.expanded* (`false`)
Set to true if you want new views to be expanded by default.

todo-tree.tree.flat* (`false`)
Set to true if you want new views to be flat by default.

todo-tree.tree.grouped* (`false`)
Set to true if you want new views to be grouped by default.

todo-tree.tree.tagsOnly* (`false`)
Set to true if you want new views with tags only by default.

todo-tree.tree.sortTagsOnlyViewAlphabetically (`false`)
Sort items in the tags only view alphabetically instead of by file and line number.

todo-tree.tree.showCountsInTree (`false`)
Set to true to show counts of TODOs in the tree.

todo-tree.tree.labelFormat (`${tag} ${after}`)
Format of the TODO item labels. Available placeholders are `${line}`, `${column}`, `${tag}`, `${before}` (text from before the tag), `${after}` (text from after the

todo-tree.tree.scanMode (`workspace`)
By default the extension scans the whole workspace (`workspace`). Use this to limit the search to only open files (`open files`) or only the current file (

todo-tree.tree.showScanModeButton (`false`)
Show a button on the tree view header to switch the scanMode (see above).

todo-tree.tree.hideIconsWhenGroupedByTag (`false`)
Hide item icons when grouping by tag.

todo-tree.tree.disableCompactFolders (`false`)
The tree will normally respect the VSCode's `explorer.compactFolders` setting. Set this to true if you want to disable compact folders in the todo tree.

todo-tree.tree.tooltipFormat (`${filepath}, ${line}`)
Format of the tree item tooltips. Uses the same placeholders as `todo-tree.tree.labelFormat` (see above).

todo-tree.tree.buttons.reveal (`true`)
Show a button in the tree view title bar to reveal the current item (only when track file is not enabled).

todo-tree.tree.buttons.scanMode (`false`)
Show a button in the tree view title bar to change the Scan Mode setting.

todo-tree.tree.buttons.viewStyle (`true`)
Show a button in the tree view title bar to change the view style (tree, flat or tags only).

todo-tree.tree.buttons.groupByTag (`true`)
Show a button in the tree view title bar to enable grouping items by tag.

todo-tree.tree.buttons.filter (`true`)
Show a button in the tree view title bar allowing the tree to be filtered by entering some text.

todo-tree.tree.buttons.refresh (`true`)
Show a refresh button in the tree view title bar.

todo-tree.tree.buttons.expand (`true`)
Show a button in the tree view title bar to expand or collapse the whole tree.

todo-tree.tree.buttons.export (`false`)
Show a button in the tree view title bar to create a text file showing the tree content.

Only applies to new workspaces. Once the view has been changed in the workspace, the current state is stored.*

Multiline TODOs

If the regex contains `\n`, then multiline TODOs will be enabled. In this mode, the search results are processed slightly differently. If results are fo

```
"todo-tree.regex.regex": "(//)\\s*($TAGS).*(\\n\\s*//\\s{2,}.*)*"
```

This will now match multiline TODOs where the extra lines have at least two spaces between the comment characters and the TODO item. e.g.

```
// TODO multiline example
//  second line
//  third line
```

If you want to match multiline TODOs in C++ style multiline comment blocks, you'll need something like:

```
"todo-tree.regex.regex": "(/\\*)\\s*($TAGS).*(\\n\\s*(//|/\\*|\\*\\*)\\s{2,}.*)*"
```

which should match:

```
/* TODO multiline example
**  second line
**  third line
*/
```

Note: If you are modifying settings using the settings GUI, you don't need to escape each backslash.

Warning: Multiline TODOs will not work with markdown TODOs and may have other unexpected results. There may also be a reduction in performance.

Excluding files and folders

To restrict the set of folders which is searched, you can define `todo-tree.filtering.includeGlobs`. This is an array of globs which the search results are

To exclude folders/files from your search you can define `todo-tree.filtering.excludeGlobs`. If the search results match any of these globs, then the results

You can also include and exclude folders from the tree using the context menu. This folder filter is applied separately to the include/exclude globs.

Note: By default, ripgrep ignores files and folders from your `.gitignore` or `.ignore` files. If you want to include these files, set `todo-tree.ripgrep.ripgrep`

Open file

This extension enables the user to open a file under the current cursor position. Just right-click on a pathname within a open document and select the

If the string is has an tailing number separated by a colon (i.e. `:23`) it will open the file at the specified line number. `:23:45` means line 23 column

It is also possible to select one or more text segments in the document and open them.

Example

You have a document, containing some text, that exists in a folder, say `c:\Users\guest\Documents\myfile.txt`. In that file the path strings could look like

```
[...]
 c:\Users\guest\Documents\Temp\stuff.txt
[...]
 "..\..\administrator\readme.txt"
[...]
 "..\user\readme.txt:33"
[...]
```

With this extension you can right-click on such a path and choose `open file under cursor` and VSCode will open a new tab with that file.

Main Features

- support file string selection(s), or, no selection and auto *detect path outside quotes* or within quotes `'file-path'`.
- support *opening multiple files* at a time. (Either a multi-lined selection or multiple selections.)
- path lookup from multiple locations to find *nearest match*: absolute, current document's folder, workspace, workspace's `src`, defined search paths, (
- *line and column positioning* with `file:line:column`.
- possible to open like `[src/]class/SomeClass.php` from just `someClass`. (Use case insensitive file system to support the different in letter case.)
- allow `/path/to/sth` to be lookup as both absolute and relative path. Useful for code like `projectFolder + '/path/to/sth'`.
- support opening single or multiple files from *Linux `grep` output* with line number, which has the line pattern `file:line[:column]:content` (content is dis
- fallback to VS Code's "Quick Open" input box if file not found. (For handy custom search, or find containing files if the path is a folder in the
- include a simple "Open file like this file" command to call Quick Open with current file's relative path (without file extension), for lookup file

Path Lookup Detail

Relative paths are relative to the these folders (in listed order):

1. Currently opened document's folder, and

   (if it is within a workspace folder) the document's parent folders (up to the workspace folder).

2. All workspace folders, and their sub-folders listed in the option `seito-openfile.searchSubFoldersOfWorkspaceFolders`.

3. All search paths in the option `seito-openfile.searchPaths`.

Remarks: - Absolute paths `/...` (`/` or `\`), if not found, are searched like relative paths too. - If `~/...` paths not found from user's home, the step 2 of

Code Runner

Join the chat at https://gitter.im/formulahendry/vscode-code-runner Marketplace Version Installs Rating build passing

Run code snippet or code file for multiple languages: **C, C++, Java, JavaScript, PHP, Python, Perl, Perl 6, Ruby, Go, Lua, Groovy, PowerShell, BAT/CMD,**

Features

- Run code file of current active Text Editor
- Run code file through context menu of file explorer
- Run selected code snippet in Text Editor
- Run code per Shebang
- Run code per filename glob
- Run custom command
- Stop code running
- View output in Output Window
- Set default language to run
- Select language to run

- Support REPL by running code in Integrated Terminal

Usages

- To run code:
- use shortcut `Ctrl+Alt+N`
- or press `F1` and then select/type `Run Code`,
- or right click the Text Editor and then click `Run Code` in editor context menu
- or click `Run Code` button in editor title menu
- or click `Run Code` button in context menu of file explorer
- To stop the running code:
- use shortcut `Ctrl+Alt+M`
- or press `F1` and then select/type `Stop Code Run`
- or right click the Output Channel and then click `Stop Code Run` in context menu

Usage
Usage

- To select language to run, use shortcut `Ctrl+Alt+J`, or press `F1` and then select/type `Run By Language`, then type or select the language to run: e.g `php,`

Usage
Usage

- To run custom command, then use shortcut `Ctrl+Alt+K`, or press `F1` and then select/type `Run Custom Command`

Configuration

Make sure the executor PATH of each language is set in the environment variable. You could also add entry into `code-runner.executorMap` to set the executo

```
{
    "code-runner.executorMap": {
        "javascript": "node",
        "php": "C:\\php\\php.exe",
        "python": "python",
        "perl": "perl",
        "ruby": "C:\\Ruby23-x64\\bin\\ruby.exe",
        "go": "go run",
        "html": "\"C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe\"",
        "java": "cd $dir && javac $fileName && java $fileNameWithoutExt",
        "c": "cd $dir && gcc $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt"
    }
}
```

Supported customized parameters * *workspaceRoot* : *ThepathofthefolderopenedinVSCode* * dir: The directory of the code file being run * *dirWithoutTrailir*

Please take care of the back slash and the space in file path of the executor * Back slash: please use \\ * If there ares spaces in file path, please

You could set the executor per filename glob:

```
{
    "code-runner.executorMapByGlob": {
        "pom.xml": "cd $dir && mvn clean package",
        "*.test.js": "tap",
        "*.js": "node"
    }
}
```

Besides, you could set the default language to run:

```
{
    "code-runner.defaultLanguage": "javascript"
}
```

For the default language: It should be set with language id defined in VS Code. The languages you could set are `java`, `c`, `cpp`, `javascript`, `php`, `python`, `perl`,

Also, you could set the executor per file extension:

```
{
    "code-runner.executorMapByFileExtension": {
        ".vbs": "cscript //Nologo"
    }
}
```

To set the custom command to run:

```
{
    "code-runner.customCommand": "echo Hello"
}
```

To set the the working directory:

```
{
    "code-runner.cwd": "path/to/working/directory"
}
```

To set whether to clear previous output before each run (default is false):

```
{
    "code-runner.clearPreviousOutput": false
}
```

To set whether to save all files before running (default is false):

```
{
    "code-runner.saveAllFilesBeforeRun": false
}
```

To set whether to save the current file before running (default is false):

```
{
    "code-runner.saveFileBeforeRun": false
}
```

To set whether to show extra execution message like [Running] ... and [Done] ... (default is true):

```
{
    "code-runner.showExecutionMessage": true
}
```

[REPL support] To set whether to run code in Integrated Terminal (only support to run whole file in Integrated Terminal, neither untitled file nor coc

```
{
    "code-runner.runInTerminal": false
}
```

To set whether to preserve focus on code editor after code run is triggered (default is true, the code editor will keep focus; when it is false, Termi

```
{
    "code-runner.preserveFocus": true
}
```

`code-runner.ignoreSelection`: Whether to ignore selection to always run entire file. (Default is **false**)

`code-runner.showRunIconInEditorTitleMenu`: Whether to show 'Run Code' icon in editor title menu. (Default is **true**)

`code-runner.showRunCommandInEditorContextMenu`: Whether to show 'Run Code' command in editor context menu. (Default is **true**)

`code-runner.showRunCommandInExplorerContextMenu`: Whether to show 'Run Code' command in explorer context menu. (Default is **true**)

`code-runner.terminalRoot`: For Windows system, replaces the Windows style drive letter in the command with a Unix style root when using a custom shell as

`code-runner.temporaryFileName`: Temporary file name used in running selected code snippet. When it is set as empty, the file name will be random. (Default

`code-runner.respectShebang`: Whether to respect Shebang to run code. (Default is **true**)

About CWD Setting (current working directory)

1. By default, use the `code-runner.cwd` setting
2. If `code-runner.cwd` is not set and `code-runner.fileDirectoryAsCwd` is `true`, use the directory of the file to be executed
3. If `code-runner.cwd` is not set and `code-runner.fileDirectoryAsCwd` is `false`, use the path of root folder that is open in VS Code
4. If no folder is open, use the os temp folder

Note

- For Objective-C, it is only supported on macOS
- To run C# script, you need to install scriptcs
- To run TypeScript, you need to install ts-node
- To run Clojure, you need to install Leiningen and lein-exec

Telemetry data

By default, telemetry data collection is turned on to understand user behavior to improve this extension. To disable it, update the settings.json as

```
{
    "code-runner.enableAppInsights": false
}
```

Change Log

See Change Log here

Issues

Submit the issues if you find any bug or have any suggestion.

Contribution

Fork the repo and submit pull requests.

Git Project Manager

build passing

Git Project Manager (GPM) is a Microsoft VSCode extension that allows you to open a **new window targeting a git repository** directly from VSCode window.

Available commands

Currently there are 3 avaliable commands, all of them can be accessed via **Ctrl+Shift+P** *(Cmd+Alt+P on Mac)* typing **GPM**

GPM: Open Git Project *(Defaults to: Ctrl+Alt+P)*

Show a list of the available git repositories in all folders configured in **gitProjectManager.baseProjectsFolders**. The first time it searchs all folder

open Git Project
open Git Project

GPM: Refresh Projects

This commands refresh the cached repositories info for all configured folders.

GPM: Refresh specific project folder

This commands allows you to select a specific folder to refresh its repositories, without refreshing all folders.

GPM: Open Recent Git Project *(Defaults to Ctrl+Shift+Q)*

This command will bring a list of your most recent git projects, leting you swap even faster between them.

The size of the list if configured in `gitProjectManager.recentProjectsListSize`

Available settings

Before start using GPM you need to configure the base folders that the extension will search for git repositories. Edit settings.json from the **File -**

```
{
    "gitProjectManager.baseProjectsFolders": [
        "/home/user/nodeProjects",
        "/home/user/personal/pocs"
    ]
}
```

Another available configuration is **gitProjectManager.storeRepositoriesBetweenSessions** that allows git repositories information to be stored between se

```
{
    "gitProjectManager.storeRepositoriesBetweenSessions": true
}
```

If nothing happens when trying to open a found project it could be due to the Code command being used. To work around this issue set **gitProjectManager**

First: Define it as a simple string, with the path to code app

```
//Windows
{
    "gitProjectManager.codePath": "C:\\Program Files (x86)\\Microsoft VS Code\\bin\\code.cmd"
}
//Linux
{
"gitProjectManager.codePath": "/usr/local/bin/code"
}
```

Second: Use a object notation to define the path to code path on each platform

```
{
    "gitProjectManager.codePath" : {
        "windows": "C:\\Program Files (x86)\\Microsoft VS Code\\bin\\code.cmd",
        "linux": "/usr/local/bin/code"
    }
}
```

Third: An array of file paths, where at least one is a valid path

```
{
    "gitProjectManager.codePath" : [
        "C:\\Program Files (x86)\\Microsoft VS Code\\bin\\code.cmd",
        "/usr/local/bin/code"
    ]
}
```

To improve performance there are 2 new and important configurations that are: **ignoredFolders**: an array of folder names that will be ignored (*node_modu*

```
{
    "gitProjectManager.ignoredFolders": ["node_modules"]
}
```

maxDepthRecursion: indicates the maximum recursion depth that will be searched starting in the configured folder (default: `2`)

```
{
    "gitProjectManager.maxDepthRecursion": 4
}
```

In version 0.1.10 we also added the **"gitProjectManager.checkRemoteOrigin"** configuration that allows users to not check remote repository origin to imp

```
{
    "gitProjectManager.checkRemoteOrigin": false
}
```

Added in version 0.1.12, you can configure the behavior when opening a project if it'll be opened in the same window or in a new window. (*this option*

```
{
    "gitProjectManager.openInNewWindow": false
}
```

Participate

If you have any idea, feel free to create issues and pull requests.

Open in Code

Logo

Switch between Code and Code Insiders with ease.

Install

Follow the instructions in the Marketplace, or run the following in the command palette:

```
ext install fabiospampinato.vscode-open-in-code
```

Usage

It adds 2 commands to the command palette:

```
'Open in Code' // Open the current project and file in Code
'Open in Code Insiders' // Open the current project and file in Code Insiders
```

Contributing

If you found a problem, or have a feature request, please open an issue about it.

If you want to make a pull request you can debug the extension using Debug Launcher.

License

MIT © Fabio Spampinato

Reactjs

VS Code Reactjs snippets

Version Installs Ratings

This extension contains code snippets for Reactjs and is based on the awesome babel-sublime-snippets package.

Installation

In order to install an extension you need to launch the Command Palette (Ctrl + Shift + P or Cmd + Shift + P) and type Extensions. There you have eit

Supported languages (file extensions)

- JavaScript (.js)
- TypeScript (.ts)
- JavaScript React (.jsx)
- TypeScript React (.tsx)

Breaking change in version 2.0.0

Removed support for jsx language as it was giving errors in developer tools #39

Breaking change in version 1.0.0

Up until verion 1.0.0 all the JavaScript snippets where part of the extension. In order to avoid duplication the snippets are now included only to thi

Usage

When installing the extension React development could be really fun create react component

As VS Code from version 0.10.10 supports React components syntax inside js files the snippets are available for JavaScript language as well. In the fo
create react stateless component

Snippets

Below is a list of all available snippets and the triggers of each one. The ⇥ means the TAB key.

Trigger	Content
rcc→	class component skeleton
rrc→	class component skeleton with react-redux connect
rrdc→	class component skeleton with react-redux connect and dispatch
rccp→	class component skeleton with prop types after the class
rcjc→	class component skeleton without import and default export lines
rcfc→	class component skeleton that contains all the lifecycle methods
rwwd→	class component without import statements
rpc→	class pure component skeleton with prop types after the class
rsc→	stateless component skeleton
rscp→	stateless component with prop types skeleton
rscm→	memoize stateless component skeleton
rscpm→	memoize stateless component with prop types skeleton
rsf→	stateless named function skeleton
rsfp→	stateless named function with prop types skeleton
rsi→	stateless component with prop types and implicit return
fcc→	class component with flow types skeleton
fsf→	stateless named function skeleton with flow types skeleton
fsc→	stateless component with flow types skeleton
rpt→	empty propTypes declaration
rdp→	empty defaultProps declaration
con→	class default constructor with props
conc→	class default constructor with props and context
est→	empty state object
cwm→	componentWillMount method
cdm→	componentDidMount method
cwr→	componentWillReceiveProps method
scu→	shouldComponentUpdate method
cwup→	componentWillUpdate method
cdup→	componentDidUpdate method
cwun→	componentWillUnmount method
gsbu→	getSnapshotBeforeUpdate method
gdsfp→	static getDerivedStateFromProps method

cdc→	componentDidCatch method
ren→	render method
sst→	this.setState with object as parameter
ssf→	this.setState with function as parameter
props→	this.props
state→	this.state
bnd→	binds the this of method inside the constructor
disp→	MapDispatchToProps redux function

The following table lists all the snippets that can be used for prop types. Every snippet regarding prop types begins with `pt` so it's easy to group it

For example `pta` creates the `PropTypes.array` and `ptar` creates the `PropTypes.array.isRequired`

Trigger	Content
pta→	PropTypes.array,
ptar→	PropTypes.array.isRequired,
ptb→	PropTypes.bool,
ptbr→	PropTypes.bool.isRequired,
ptf→	PropTypes.func,
ptfr→	PropTypes.func.isRequired,
ptn→	PropTypes.number,
ptnr→	PropTypes.number.isRequired,
pto→	PropTypes.object,
ptor→	PropTypes.object.isRequired,
pts→	PropTypes.string,
ptsr→	PropTypes.string.isRequired,
ptsm→	PropTypes.symbol,
ptsmr→	PropTypes.symbol.isRequired,
ptan→	PropTypes.any,
ptanr→	PropTypes.any.isRequired,
ptnd→	PropTypes.node,
ptndr→	PropTypes.node.isRequired,
ptel→	PropTypes.element,
ptelr→	PropTypes.element.isRequired,
pti→	PropTypes.instanceOf(ClassName),
ptir→	PropTypes.instanceOf(ClassName).isRequired,
pte→	PropTypes.oneOf(['News', 'Photos']),
pter→	PropTypes.oneOf(['News', 'Photos']).isRequired,
ptet→	PropTypes.oneOfType([PropTypes.string, PropTypes.number]),
ptetr→	PropTypes.oneOfType([PropTypes.string, PropTypes.number]).isRequired,
ptao→	PropTypes.arrayOf(PropTypes.number),
ptaor→	PropTypes.arrayOf(PropTypes.number).isRequired,
ptoo→	PropTypes.objectOf(PropTypes.number),
ptoor→	PropTypes.objectOf(PropTypes.number).isRequired,
ptoos→	PropTypes.objectOf(PropTypes.shape()),
ptoosr→	PropTypes.objectOf(PropTypes.shape()).isRequired,
ptsh→	PropTypes.shape({color: PropTypes.string, fontSize: PropTypes.number}),
ptshr→	PropTypes.shape({color: PropTypes.string, fontSize: PropTypes.number}).isRequired,

jQuery Code Snippets

Over 130 jQuery Code Snippets for JavaScript code.

Just type the letters 'jq' to get a list of all available jQuery Code Snippets.

Image of Snippets
Image of Snippets

Snippets

column0

Trigger	Description			

func	An anonymous function.
jqAfter	Insert content, specified by the parameter, after each element in the set of matched elements.
jqAjax	Perform an asynchronous HTTP (Ajax) request.
jqAjaxAspNetWebService	Perform an asynchronous HTTP (Ajax) request to a ASP.NET web service.
jqAppend	Insert content, specified by the parameter, to the end of each element in the set of matched elements.
jqAppendTo	Insert every element in the set of matched elements to the end of the target.
jqAttrGet	Get the value of an attribute for the first element in the set of matched elements.
jqAttrRemove	Remove an attribute from each element in the set of matched elements.
jqAttrSet	Set one or more attributes for the set of matched elements.
jqAttrSetFn	Set one or more attributes for the set of matched elements.
jqAttrSetObj	Set one or more attributes for the set of matched elements.
jqBefore	Insert content, specified by the parameter, before each element in the set of matched elements.
jqBind	Attach a handler to an event for the elements.
jqBindWithData	Attach a handler to an event for the elements.
jqBlur	Bind an event handler to the "blur" JavaScript event, or trigger that event on an element.
jqChange	Bind an event handler to the "change" JavaScript event, or trigger that event on an element.
jqClassAdd	Adds the specified class(es) to each of the set of matched elements.
jqClassRemove	Remove a single class, multiple classes, or all classes from each element in the set of matched elements.
jqClassToggle	Add or remove one or more classes from each element in the set of matched elements, depending on either the class's p
jqClassToggleSwitch	Add or remove one or more classes from each element in the set of matched elements, depending on either the class's p
jqClick	Bind an event handler to the "click" JavaScript event, or trigger that event on an element.
jqClone	Create a deep copy of the set of matched elements.
jqCloneWithEvents	Create a deep copy of the set of matched elements.
jqCssGet	Get the computed style properties for the first element in the set of matched elements.
jqCssSet	Set one or more CSS properties for the set of matched elements.
jqCssSetObj	Set one or more CSS properties for the set of matched elements.
jqDataGet	Return the value at the named data store for the first element in the jQuery collection, as set by data(name, value)
jqDataRemove	Remove a previously-stored piece of data.
jqDataSet	Store arbitrary data associated with the matched elements.
jqDataSetObj	Store arbitrary data associated with the matched elements.
jqDie	Remove event handlers previously attached using .live() from the elements.
jqDieAll	Remove event handlers previously attached using .live() from the elements.
jqDieFn	Remove event handlers previously attached using .live() from the elements.
jqDocReady	Function to execute when the DOM is fully loaded.
jqDocReadyShort	Function to execute when the DOM is fully loaded.
jqEach	A generic iterator function, which can be used to seamlessly iterate over both objects and arrays. Arrays and array-l
jqEachElement	Iterate over a jQuery object, executing a function for each matched element.

jqEmpty	Remove all child nodes of the set of matched elements from the DOM.
jqFadeIn	Display the matched elements by fading them to opaque.
jqFadeInFull	Display the matched elements by fading them to opaque.
jqFadeOut	Hide the matched elements by fading them to transparent.
jqFadeOutFull	Hide the matched elements by fading them to transparent.
jqFadeTo	Adjust the opacity of the matched elements.
jqFadeToFull	Adjust the opacity of the matched elements.
jqFind	Get the descendants of each element in the current set of matched elements, filtered by a selector, jQuery object, or
jqFocus	Bind an event handler to the "focus" JavaScript event, or trigger that event on an element.
jqGet	Load data from the server using a HTTP GET request.
jqGetJson	Load JSON-encoded data from the server using a GET HTTP request.
jqGetScript	Load a JavaScript file from the server using a GET HTTP request, then execute it.
jqHasClass	Determine whether any of the matched elements are assigned the given class.
jqHeightGet	Get the current computed height for the first element in the set of matched elements.
jqHeightSet	Set the CSS height of every matched element.
jqHide	Hide the matched elements.
jqHideFull	Hide the matched elements.
jqHover	Bind two handlers to the matched elements, to be executed when the mouse pointer enters and leaves the elements.
jqHtmlGet	Get the HTML contents of the first element in the set of matched elements.
jqHtmlSet	Set the HTML contents of each element in the set of matched elements.
jqInnerHeight	Get the current computed height for the first element in the set of matched elements, including padding but not borde
jqInnerWidth	Get the current computed inner width for the first element in the set of matched elements, including padding but not
jqInsertAfter	Insert every element in the set of matched elements after the target.
jqInsertBefore	Insert every element in the set of matched elements before the target.
jqKeyDown	Bind an event handler to the "keydown" JavaScript event, or trigger that event on an element.
jqKeyPress	Bind an event handler to the "keypress" JavaScript event, or trigger that event on an element.
jqKeyUp	Bind an event handler to the "keyup" JavaScript event, or trigger that event on an element.
jqLoadGet	Load data from the server and place the returned HTML into the matched element.
jqLoadPost	Load data from the server and place the returned HTML into the matched element.
jqMap	Translate all items in an array or object to new array of items.
jqMouseDown	Bind an event handler to the "mousedown" JavaScript event, or trigger that event on an element.
jqMouseEnter	Bind an event handler to be fired when the mouse enters an element, or trigger that handler on an element.
jqMouseLeave	Bind an event handler to be fired when the mouse leaves an element, or trigger that handler on an element.
jqMouseMove	Bind an event handler to the "mousemove" JavaScript event, or trigger that event on an element.
jqMouseOut	Bind an event handler to the "mouseout" JavaScript event, or trigger that event on an element.
jqMouseOver	Bind an event handler to the "mouseover" JavaScript event, or trigger that event on an element.
jqMouseUp	Bind an event handler to the "mouseup" JavaScript event, or trigger that event on an element.
jqNamespace	A namespace template. ref: http://enterprisejquery.com/2010/10/how-good-c-habits-can-encourage-bad-javascript-habits-
jqOffsetGet	Get the current coordinates of the first element, or set the coordinates of every element, in the set of matched elem
jqOffsetParent	Get the closest ancestor element that is positioned.
jqOn	Attach an event handler function for one or more events to the selected elements.
jqOne	Attach a handler to an event for the elements. The handler is executed at most once per element per event type.
jqOneWithData	Attach a handler to an event for the elements. The handler is executed at most once per element per event type.
jqOuterHeight	Get the current computed height for the first element in the set of matched elements, including padding, border, and
jqOuterWidth	Get the current computed width for the first element in the set of matched elements, including padding and border.
jqPlugin	Plugin template.
jqPosition	Get the current coordinates of the first element in the set of matched elements, relative to the offset parent.
jqPost	Load data from the server using a HTTP POST request.
jqPrepend	Insert content, specified by the parameter, to the beginning of each element in the set of matched elements.
jqPrependTo	Insert every element in the set of matched elements to the beginning of the target.
jqRemove	Remove the set of matched elements from the DOM.
jqRemoveExp	Remove the set of matched elements from the DOM.
jqReplaceAll	Replace each target element with the set of matched elements.

jqReplaceWith	Replace each element in the set of matched elements with the provided new content and return the set of elements that
jqResize	Bind an event handler to the "resize" JavaScript event, or trigger that event on an element.
jqScroll	Bind an event handler to the "scroll" JavaScript event, or trigger that event on an element.
jqScrollLeftGet	Get the current horizontal position of the scroll bar for the first element in the set of matched elements.
jqScrollLeftSet	Set the current horizontal position of the scroll bar for each of the set of matched elements.
jqScrollTopGet	Get the current vertical position of the scroll bar for the first element in the set of matched elements or set the
jqScrollTopSet	Set the current vertical position of the scroll bar for each of the set of matched elements.
jqSelect	Bind an event handler to the "select" JavaScript event, or trigger that event on an element.
jqSelectTrigger	Bind an event handler to the "select" JavaScript event, or trigger that event on an element.
jqShow	Display the matched elements.
jqShowFull	Display the matched elements.
jqSlideDown	Display the matched elements with a sliding motion.
jqSlideDownFull	Display the matched elements with a sliding motion.
jqSlideToggle	Display or hide the matched elements with a sliding motion.
jqSlideToggleFull	Display or hide the matched elements with a sliding motion.
jqSlideUp	Display the matched elements with a sliding motion.
jqSlideUpFull	Display the matched elements with a sliding motion.
jqSubmit	Bind an event handler to the "submit" JavaScript event, or trigger that event on an element.
jqSubmitTrigger	Bind an event handler to the "submit" JavaScript event, or trigger that event on an element.
jqTextGet	Get the combined text contents of each element in the set of matched elements, including their descendants.
jqTextSet	Set the content of each element in the set of matched elements to the specified text.
jqToggle	Display or hide the matched elements.
jqToggleFull	Display or hide the matched elements.
jqToggleSwitch	Display or hide the matched elements.
jqTrigger	Execute all handlers and behaviors attached to the matched elements for the given event type.
jqTriggerHandler	Execute all handlers attached to an element for an event.
jqTriggerHandlerWithData	Execute all handlers attached to an element for an event.
jqTriggerWithData	Execute all handlers and behaviors attached to the matched elements for the given event type.
jqUnbind	Remove a previously-attached event handler from the elements.
jqUnbindAll	Remove a previously-attached event handler from the elements.
jqUnload	Bind an event handler to the "unload" JavaScript event.
jqValGet	Get the current value of the first element in the set of matched elements.
jqValSet	Set the value of each element in the set of matched elements.
jqWidthGet	Get the current computed width for the first element in the set of matched elements.
jqWidthSet	Set the CSS width of each element in the set of matched elements.
jqWrap	Wrap an HTML structure around each element in the set of matched elements.
jqWrapAll	Wrap an HTML structure around all elements in the set of matched elements.
jqWrapInner	Wrap an HTML structure around the content of each element in the set of matched elements.

Source

Github

All snippets have been taken from the Visual Studio 2015 jQuery Code Snippets Extension. Credit given where due.

License

MIT

Markdown table prettifier extension for Visual Studio Code

build passing

Makes tables more readable for humans. Compatible with the Markdown writer plugin's table formatter feature in Atom.

Features

- Remove redundant ending table border if the beginning has no border, so the table *will not end* with "|".
- Create missing ending table border if the beginning already has a border, so the table *will end* with "|".
- Save space by not right-padding the last column if the table has no border.
- Support empty columns inside tables.
- Support column alignment options with ":".
- CLI and docker support to prettify files.

feature X
feature X

CLI formatting

Formatting files or checking if they're already formatted is possible from the command line. This requires `node` and `npm`.

The extension has to be downloaded and compiled: - Locate the installed extension path or download the extension from Github. - Go to the extension di

The typical location of the installed extension (your actual version might differ): - Windows %USERPROFILE%.vscode.markdown-table-prettify-3.0.0 - mac

Available features from the command line: - To prettify a file: `npm run --silent prettify-md < input.md`. - To prettify a file and save the output: `npm run --`

> Note: the `--silent` switch sets the npm log level to silent, which is useful to hide the executed file name and concentrate on the actual output.

Formatting with docker

Available features from docker: - To prettify a file: `docker container run -i darkriszty/prettify-md < input.md`. - To prettify a file and save the output: `dock`

Extension Settings

The extension is available for markdown language mode. It can either prettify a selected table (`Format Selection`) or the entire document (`Format Document`)

Configurable settings: - The maximum texth length of a selection/entire document to consider for formatting. Defaults to 1M chars. There is no limit w

Known Issues

- **Tables with mixed character widths (eg: CJK) are not always properly formatted (issue #4).**

<===============(vscode-goto-documentation)===============>

vscode-goto-documentation

A Visual Studio Code extension to jump to documentation for the current keyword, ported from sublime-text-2-goto-documentation

Supports

- PHP
- JS / CoffeeScript
- HTML
- CSS/SASS/LESS
- Python
- Clojure
- Go
- Ruby
- C / C++
- Perl
- C#
- Lua
- Erlang
- Haskell
- ...you can add any other language via settings

Installation

Search for `goto documentation`

How to use

Move the cursor inside the word you want the docs for and: * Press `Super+Shift+H` or
 * mouse right click the word and select **gotoDocument**

Edit the urls

GotoDocumentation allows you to edit the url that opens by editing the settings. ### The available settings are:

```
"goto-documentation.customDocs": {
    // the key value pair represent scope -> doc url
    // supported placeholders:
    //  - ${query} the selected text/word
   "css": "http://devdocs.io/#q=${query}",
}
```

VSCode DevTools for Chrome

A VSCode extension to host the chrome devtools inside of a webview.

If you are looking for a more streamlined and officially supported devtools extension, you should try VS Code - Elements for Microsoft Edge (Chromium)

Marketplace badge

Attaching to a running chrome instance:

Demo1
Demo1

Launching a 'debugger for chrome' project and using screencast:

Demo2
Demo2

Using the extension

Launching as a Debugger

You can launch the Chrome DevTools hosted in VS Code like you would a debugger, by using a launch.json config file. However, the Chrome DevTools aren'

To do this in your `launch.json` add a new debug config with two parameters. - `type` - The name of the debugger which must be `devtools-for-chrome`. Required. -

```
{
    "version": "0.1.0",
    "configurations": [
        {
            "type": "devtools-for-chrome",
            "request": "launch",
            "name": "Launch Chrome DevTools",
            "file": "${workspaceFolder}/index.html"
        },
        {
            "type": "devtools-for-chrome",
            "request": "attach",
            "name": "Attach Chrome DevTools",
            "url": "http://localhost:8000/"
        }
    ]
}
```

Launching Chrome manually

- Start chrome with no extensions and remote-debugging enabled on port 9222:
  - `chrome.exe --disable-extensions --remote-debugging-port=9222`
- Open the devtools inside VS Code:
  - Run the command - `DevTools for Chrome: Attach to a target`
  - Select a target from the drop down

Launching Chrome via the extension

- Start chrome:
  - Run the command - `DevTools for Chrome: Launch Chrome and then attach to a target`
  - Navigate to whatever page you want

- Open the devtools inside VS Code:
  - Select a target from the drop down

Known Issues

- Prototyping stage
- Having the DevTools in a non-foreground tab can cause issues while debugging
  - This is due to VS Code suspending script execution of non-foreground webviews
  - The workaround is to put the DevTools in a split view tab so that they are always visible while open
- Chrome browser extensions can sometimes cause the webview to terminate

Developing the extension itself

- Start chrome with remote-debugging enabled on port 9222
  - `chrome.exe --disable-extensions --remote-debugging-port=9222`
- Run the extension
  - `npm install`
  - `npm run watch` or `npm run build`
  - Open the folder in VSCode
  - `F5` to start debugging
- Open the devtools
  - Run the command - `DevTools for Chrome: Attach to a target`
  - Select a target from the drop down

JS Refactor

JS Refactor is the Javascript automated refactoring tool for Visual Studio Code, built to smooth and streamline your development experience. It provic

Donate to the JSR Cause

If you use JS Refactor and find it useful, please consider visiting my Patreon page and donating. Anything helps to keep updates and maintenance happe

https://www.patreon.com/sdlcpunk

Supported Language Files

- JavaScript/ECMAScript (.js)
- Vue single file components (.vue)
- HTML (.htm, .html)

Experimental Support

- JavaScript React (.jsx)
- TypeScript support (.ts)
- TypeScript React (.tsx)

Installation

Extensions Panel:

Click on the extensions icon on the left-hand side of your editor. In the search bar type "JS Refactor." Find the extension in the list and click the

Command Pallette

Open VS Code, press F1 and enter `ext install` to open the extensions panel, follow the instructions above

Find Me Online

- ChrisStead.net
- Visit me on Twitter
- Fork and Contribute on Github

Automated Refactorings

Basic usage: Make a selection, right click and select the refactoring you want to perform from the context menu.

Command Pallette: You can press F1 then simply type the name of the refactoring and press enter if you know the name of the refactoring you need.

Shortcuts: Finally, there are hotkey combinations for some of the most common refactorings you might want. Hotkeys are listed in the keybindings secti

JS Refactor supports the following refactorings (explanations below):

Common Refactorings: - Extract Method - Extract Variable - Inline Variable - Rename Variable (Alias of VS Code internal command)

Other Utilities: - Convert To Arrow Function - Convert To Function Declaration - Convert To Function Expression - Convert To Template Literal - Export

Keybindings

- Extract method
  - Windows/Linux: ctrl+shift+j m
  - Mac: cmd+shift+j m
- Extract variable
  - Windows/Linux: ctrl+shift+j v
  - Mac: cmd+shift+j v
- Inline variable
  - Windows/Linux: ctrl+shift+j i
  - Mac: cmd+shift+j i

- Rename variable (VS Code internal) - F2
- Convert to Arrow Function
  - Windows/Linux: ctrl+shift+j a
  - Mac: cmd+shift+j a
- Convert to Function Expression
  - Windows/Linux: ctrl+shift+j f
  - Mac: cmd+shift+j f
- Convert to Template Literal
  - Windows/Linux: ctrl+shift+j l
  - Mac: cmd+shift+j l
- Export function
  - Windows/Linux: ctrl+shift+j x
  - Mac: cmd+shift+j x
- Mark function as async
  - Windows/Linux: ctrl+shift+j s
  - Mac: cmd+shift+j s
- Shift parameters
  - Windows/Linux: ctrl+shift+j p
  - Mac: cmd+shift+j p
- Wrap selection
  - Windows/Linux: ctrl+shift+j w
  - Mac: cmd+shift+j w

Usage

Select the code you wish to refactor and then press the F1 key to open the command pallette. Begin typing the name of the refactoring and select the

Explanations

Core Refactorings

Extract Method Creates new function with original selection as the body.

Extract Variable Creates new assigned variable declaration and replaces original selection.

Inline Variable Replaces all references to variable with variable initialization expression, deletes variable declaration.

Other Utilities

Convert To Arrow Function Converts a function expression to an arrow function.

Convert To Function Declaration Converts a function expression, assigned to a variable, to a function declaration.

Convert To Function Expression Converts an arrow function to a function expression.

Convert To Template Literal Converts a string concatenation expression to a template literal.

Export Function creates new export declaration for selected function or function name

Introduce Function creates new function from existing function call or variable assignment

Lift and Name Function Expression Lifts function expression from current context, replacing it with provided name and adds name to expression

Shift Parameters Shifts function parameters to the left or right by the selected number of places

Wrap In Condition Wraps selected code in an if statement, adding indentation as necessary

Wrap in function takes your selected code, at line-level precision, and wraps all of the lines in a named function.

Wrap in IIFE wraps selected code in an immediately invoked function expression (IIFE).

Snippets

JS Refactor supports several common code snippets:

- Anonymous Function (anon)
- Arrow Function (arrow)
- Async Function (async)
- Class
    - Definition (class)
    - Constructor (ctor)
    - Method (method)
- Condition Block (cond)
- Console Log (log)
- Export statement -- single variable (export)
- Export statement -- object literal (exportObj)
- Function (fn)
- Generator (generator)
- Lambda function (lfn)
- Immediately Invoked Function Expression (iife)
- Member Function (mfn)
- Prototypal Object Definition (proto)
- Require statement (require)
- Try/Catch Block (tryCatch)
- Use Strict (strict)

Usage

Type the abbreviation, such as fn, in your code and hit enter. When the snippet is executed, the appropriate code will be inserted into your document.

Explanations

anon Inserts a tab-stopped anonymous function snippet into your code

export Adds a module.exports single var assignment in your module

exportObj Adds a module.exports assignment with an object literal

fn Inserts a tab-stopped named function snippet into your code

lfn Inserts a tab-stopped lambda function snippet into your code

iife Inserts a new, tab-stopped IIFE into your code

mfn Inserts a new color-delimited member function to your prototype -- Protip be inside a prototype object when using this.

require Inserts a new require statement in your module

strict Inserts 'use strict' into your code

Npm Intellisense

Visual Studio Code plugin that autocompletes npm modules in import statements.

auto complete
auto complete

Sponsors

Eliminate context switching and costly distractions. Create and merge PRs and perform code reviews from inside your IDE while using jump-to-definiti
Learn more

Installation

In the command palette (cmd-shift-p) select Install Extension and choose npm Intellisense.

install
install

Contributing

Something missing? Found a bug? - Create a pull request or an issue. Github

Features

Import command

import command
import command

```
{
    "npm-intellisense.importES6": true,
    "npm-intellisense.importQuotes": "'",
    "npm-intellisense.importLinebreak": ";\r\n",
    "npm-intellisense.importDeclarationType": "const",
}
```

Import command (ES5)

import command
import command

```
{
    "npm-intellisense.importES6": false,
    "npm-intellisense.importQuotes": "'",
    "npm-intellisense.importLinebreak": ";\r\n",
    "npm-intellisense.importDeclarationType": "const",
}
```

Scan devDependencies

Npm intellisense scans only dependencies by default. Set scanDevDependencies to true to enable it for devDependencies too.

```
{
    "npm-intellisense.scanDevDependencies": true,
}
```

Show build in (local) libs

Shows build in node modules like 'path' of 'fs'

```
{
    "npm-intellisense.showBuildInLibs": true,
}
```

Lookup package.json recursive

Look for package.json inside nearest directory instead of workspace root. It's enabled by default.

```
{
    "npm-intellisense.recursivePackageJsonLookup": true,
}
```

Experimental: Package Subfolder Intellisense

Open subfolders of a module. This feature is work in progress and experimental.

```
{
    "npm-intellisense.packageSubfoldersIntellisense": false,
}
```

License

This software is released under MIT License

vscode-standardjs

VSCode extension to integrate JavaScript Standard Style into VSCode.

We support JavaScript Semi-Standard Style too, if you prefer keeping the semicolon.

JavaScript Standard Style with custom tweaks is also supported if you want to fine-tune your ESLint config while keeping the power of Standard.

This plugin also works with [TypeScript Standard Style][https://github.com/toddbluhm/ts-standard] which has additonal rules for TypeScript based proje

How to use

1. **Install the 'JavaScript Standard Style' extension**

   If you don't know how to install extensions in VSCode, take a look at the documentation.

   You will need to reload VSCode before new extensions can be used.

2. **Install `standard`, `semistandard`, `standardx` or `ts-standard`**

   This can be done globally or locally. We recommend that you install them locally (i.e. saved in your project's `devDependencies`), to ensure that othe

3. **Disable the built-in VSCode validator**

   To do this, set `"javascript.validate.enable": false` in your VSCode `settings.json`.

Plugin options

We give you some options to customize vscode-standardjs in your VSCode `settings.json`.

Option	Description	Default
`standard.enable`	enable or disable JavaScript Standard Style	
`standard.run`	run linter `onSave` or `onType`	
`standard.autoFixOnSave`	enable or disable auto fix on save. It is only available when VSCode's `files.autoSave` is either `off`, `onFocusChange` or `onWin`	
`standard.nodePath`	use this setting if an installed `standard` package can't be detected.	
`standard.validate`	an array of language identifiers specify the files to be validated	
`standard.workingDirectories`	an array for working directories to be used.	
`standard.engine`	You can use `semistandard`, `standardx` or `ts-standard` instead of `standard`. **Just make sure you've installed the `semistandard`, the**	
`standard.usePackageJson`	if set to `true`, JavaScript Standard Style will use project's `package.json` settings, otherwise globally installed `standard`	

Configuring Standard

You can still configure `standard` itself with the `standard.options` setting, for example:

```
"standard.options": {
    "globals": ["$", "jQuery", "fetch"],
    "ignore": [
        "node_modules/**"
    ],
    "plugins": ["html"],
    "parser": "babel-eslint",
    "envs": ["jest"]
}
```

It's recommended to change these options in your `package.json` file on a per-project basis, rather than setting them globally in `settings.json`. For exampl

```
"standard": {
    "plugins": ["html"],
    "parser": "babel-eslint"
}
```

If you've got multiple projects within a workspace (e.g. you're inside a monorepo), VSCode prevents extensions from accessing multiple `package.json` fil

If you want this functionality, you should add each project folder to your workspace (`File -> Add folder to workspace...`). If you can't see this option, dc

Commands

When you open the Command Palette in VSCode (⇧⌘P or Ctrl+Shift+P), this plugin has the following options:

- `Fix all auto-fixable problems` - applies JavaScript Standard Style auto-fix resolutions to all fixable problems.
- `Disable JavaScript Standard Style for this Workspace` - disable JavaScript Standard Style extension for this workspace.
- `Enable JavaScript Standard Style for this Workspace` - enable JavaScript Standard Style extension for this workspace.
- `Show output channel` - view the linter output for JavaScript Standard Style.

FAQ

1. How do I lint `script` tags in `vue` or `html` files?

   You can lint them with `eslint-plugin-html`. Make sure it's installed, then enable linting for those file types in your `settings.json`:

   ```
   "standard.validate": [
       "javascript",
       "javascriptreact",
       "html"
   ],
   "standard.options": {
       "plugins": ["html"]
   },
   "files.associations": {
       "*.vue": "html"
   },
   ```

   If you want to enable `autoFix` for the new languages, you should enable it yourself:

   ```
   "standard.validate": [
       "javascript",
       "javascriptreact",
       {
           "language": "html",
           "autoFix": true
       }
   ],
   "standard.options": {
       "plugins": ["html"]
   }
   ```

How to develop

1. Fork this repo, and clone your fork locally.
2. Run `npm install` right under project root.
3. Open project in VSCode. The plugin should be disabled whilst developing.
4. Run the `watch` build task (⇧⌘B or Ctrl+Shift+B) to compile the client and server.
5. To run/debug the extension, use the `Launch Extension` launch configuration (from the VSCode debug panel).
6. To debug the server, use the `Attach to Server` launch configuration.

How to package

1. Run `npm install`,
2. Run `npm run compile`,
3. Run `npm run package` to build a .vsix file, then you can install it with `code --install-extension vscode-standardjs.vsix`.

TODO

1. [ ] add tests

File Header Comment - Visual Studio Marketplace

> Extension for Visual Studio Code - Insert File Header Comment such as date, time

This extension allow you to insert timestamp, copyright or any information to your file like comment below

```
/*
 * Created on Tue Feb 18 2020
 *
 * Copyright (c) 2020 - Your Company
 */
```

Features

- insert defined parameter like `date`, `time`, `datetime`, `day`, `month`, `year`, `hour`, `minute`, `second`, `company`, `filename`
- insert your own parameter and template
- define multiple templates

Install

```
ext install fileheadercomment
```

Extension Settings

By default you don't have to set anything. It will detect most programming language for appropriate comment syntax.

Execute it from `Command Pallete` (menu View - Command Pallete...) then type command below:

1. `FileHeaderComment: Insert Default Template at Cursor`
2. `FileHeaderComment: Select from Available Templates`

The second command will show your available templates defined in Settings

If you want to set your own parameter and template (set from menu Preferences - User Settings), you can read explanation below

This is default configuration

```
      "fileHeaderComment.parameter":{
          "*":{
              "commentbegin": "/*",
              "commentprefix": " *",
              "commentend": " */",
              "company": "Your Company"
          }
      },
      "fileHeaderComment.template":{
          "*":[
              "${commentbegin}",
              "${commentprefix} Created on ${date}",
              "${commentprefix}",
              "${commentprefix} Copyright (c) ${year} ${company}",
              "${commentend}"
          ]
      }
```

Define all custom variables/paramenters in asterisk * like

```
  "fileHeaderComment.parameter":{
      "*":{
          "company": "Your Company"
          "myvar1": "My Variable 1",
          "myvar2": "My Variable 2"
      }
  }
```

Use your variable in template like (asterisk * will be default template)

```
  "fileHeaderComment.template":{
      "*":[
          "${commentbegin}",
          "${commentprefix} Created on ${date}",
          "${commentprefix}",
          "${commentprefix} Copyright (c) ${year} ${company}",
          "${commentprefix} my variables are ${myvar1} and ${myvar2}",
          "${commentend}"
      ]
  }
```

You can define multiple templates, for instance template for MIT License

```
  "fileHeaderComment.parameter":{
      "*":{
          "author": "Your Name",
          "license_mit":[
              "The MIT License (MIT)",
              " Copyright (c) ${year} ${author}",
              "",
              " Permission is hereby granted, free of charge, to any person obtaining a copy of this software",
              " and associated documentation files (the \"Software\"), to deal in the Software without restriction,",
              " including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense,",
              " and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so,",
              " subject to the following conditions:",
              "",
              " The above copyright notice and this permission notice shall be included in all copies or substantial",
              " portions of the Software.",
              "",
              " THE SOFTWARE IS PROVIDED \"AS IS\", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED",
              " TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL",
              " THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,",
              " TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE."
          ]
      }
  },
  "fileHeaderComment.template":{
      "mit":[
          "${commentbegin}",
          "${commentprefix} Created on ${date}",
          "${commentprefix}",
          "${commentprefix} ${license_mit}",
          "${commentend}"
      ]
  }
```

You can use your `mit` template above by calling it through `Command Pallete` and choose `FileHeaderComment: Select from Available Templates`.

You can use parameters below in your template

- `date` : print current date

- `time` : print current time
- `time24h` : print current time in 24 hour format
- `datetime`: print current date + time
- `datetime24h` : print current date + time in 24 hour format
- `company` : print "Your Company"
- `day`: print day of the month
- `month`: print current month
- `year`: print current year
- `hour`: print current hour (24h)
- `minute`: print current minute
- `second`: print current second
- `filename`: print filename

Release Notes

0.0.5

- fixing python comment style (thanks to @ronak1009)

0.0.4

- support yaml, shellscript language (thanks to @waddyvic)
- add day, month, hour, minute, second, filename parameter (thanks to @rcabg, @ternvein)

0.0.3

- fixing "unknown configuration setting" message in Settings (thanks to @isuda)

0.0.2

- multiple templates
- bugfixes

0.0.1

- Initial release

Source

Bookmarks Logo

What's new in Bookmarks 11.4

- Adds an all-new **Side Bar**
- Adds **Label suggestion** based on selection
- Adds **Multi cursor** support
- Adds **Column Position** and **Label** support
- Adds **Localization** support
- Adds `Collapse All` command in the **Side Bar**
- Adds **Hover Buttons** for files and Bookmarks in the **Side Bar**
- Adds **workbench.colorCustomizations** support

Support

Bookmarks is an extension created for **Visual Studio Code**. If you find it useful, please consider supporting it.

Sponsors

Discussing code is now as easy as highlighting a block and typing a comment right from your IDE. Take the pain out of code reviews and improve code
Try it free

Bookmarks

It helps you to navigate in your code, moving between important positions easily and quickly. *No more need to search for code.* It also supports a set

Here are some of the features that **Bookmarks** provides:

- **Mark/unmark positions** in your code
- Mark positions in your code and **give it name**
- **Jump** forward and backward between bookmarks
- Icons in **gutter** and **overview ruler**
- See a list of all Bookmarks in one **file**
- See a list of all Bookmarks in your **project**
- A dedicated **Side Bar**
- **Select lines** with bookmarks
- **Select regions** between bookmarks

Features

Available commands

- `Bookmarks: Toggle` Mark/unmark positions with bookmarks
- `Bookmarks: Toggle Labeled` Mark labeled bookmarks
- `Bookmarks: Jump to Next` Move the cursor forward, to the bookmark below
- `Bookmarks: Jump to Previous` Move the cursor backward, to the bookmark above
- `Bookmarks: List` List all bookmarks in the current file
- `Bookmarks: List from All Files` List all bookmarks from all files
- `Bookmarks: Clear` remove all bookmarks in the current file
- `Bookmarks: Clear from All Files` remove all bookmarks from all files
- `Bookmarks (Selection): Select Lines` Select all lines that contains bookmarks
- `Bookmarks (Selection): Expand Selection to Next` Expand the selected text to the next bookmark
- `Bookmarks (Selection): Expand Selection to Previous` Expand the selected text to the previous bookmark
- `Bookmarks (Selection): Shrink Selection` Shrink the select text to the Previous/Next bookmark

Manage your bookmarks

Toggle / Toggle Labeled

You can easily Mark/Unmark bookmarks on any position. You can even define **Labels** for each bookmark.

Toggle
Toggle

Navigation

Jump to Next / Previous

Quicky move between bookmarks backward and forward, even if located outside the active file.

List / List from All Files

List all bookmarks from the current file/project and easily navigate to any of them. It shows a line preview and temporarily scroll to its position.

List
List

- Bookmarks from the active file only shows the line number and its contents
- Bookmarks from other files in the project also shows the relative path and filename
- Bookmarks from files outside the project are denoted with Folder

Selection

You can use **Bookmarks** to easily select lines or text blocks. Simply toggle bookmarks in any position of interest and use some of the *Selection* commanc

Select Lines

Select all bookmarked lines. Specially useful while working with log files.

Select Lines
Select Lines

Expand Selection to the Next/Previous Bookmark or Shrink the Selection

Manipulate the selection of lines *between* bookmarks, up and down.

Available Settings

- Allow navigation through all files that contains bookmarks (`false` by default)

```
"bookmarks.navigateThroughAllFiles": true
```

- Allow navigation to wrap around at the first and last bookmarks in scope (current file or all files) (`true` by default)

```
"bookmarks.wrapNavigation": true
```

- Bookmarks are always saved between sessions, and you can decide if it should be saved *in the Project*, so you can add it to your Git/SVN repo and h

```
"bookmarks.saveBookmarksInProject": true
```

- Path to another image to be shown as Bookmark (16x16 px)

```
"bookmarks.gutterIconPath": "c:\\temp\\othericon.png"
```

- Choose the background color to use on a bookmarked line

```
"bookmarks.backgroundLineColor"
```

> Deprecated in 10.7: Use `workbench.colorCustomizations` instead. More info in Available Colors

- Allow bookmarks commands, (Toggle, Jump to Next/Previous), to be displayed on the editor contex menu (`true` by default)

```
"bookmarks.showCommandsInContextMenu": true
```

- Use a **workaround** for formatters, like Prettier, which does not notify on document changes and messes Bookmark's *Sticky* behavior *(`false` by default)*

```
"bookmarks.useWorkaroundForFormatters": true
```

> This workaround should be temporary, until a proper research and suggested APIs are available

- Choose if the Side Bar should start expanded (`false` by default)

```
"bookmarks.sideBar.expanded": true
```

- Choose how multi cursor handles already bookmarked lines (`allLinesAtOnce` by default)

- `allLinesAtOnce`: Creates bookmarks in all selected lines at once, if at least one of the lines don't have a bookmark

- `eachLineIndependently`: Literally toggles a bookmark in each line, instead of making all lines equals

```
"bookmarks.multicursor.toggleMode": "eachLineIndependently"
```

- Choose how labels are suggested when creating bookmarks (`dontUse` by default)

- `dontUse`: Don't use the selection (original behavior)
- `useWhenSelected`: Use the selected text *(if available)* directly, no confirmation required
- `suggestWhenSelected`: Suggests the selected text *(if available)*. You still need to confirm.

- `suggestWhenSelectedOrLineWhenNoSelected`: Suggests the selected text *(if available)* or the entire line (when has no selection). You still need to confirm

```
"bookmarks.label.suggestion": "useWhenSelected"
```

Available Colors

- Choose the background color to use on a bookmarked line

```
"workbench.colorCustomizations": {
  "bookmarks.lineBackground": "#157EFB22"
}
```

- Choose the border color to use on a bookmarked line

```
"workbench.colorCustomizations": {
  "bookmarks.lineBorder": "#FF0000"
}
```

- Choose marker color to use in the overview ruler

```
"workbench.colorCustomizations": {
  "bookmarks.overviewRuler": "#157EFB88"
```

```
    }
```

Side Bar

The **Bookmarks** extension has its own **Side Bar**, giving you more free space in your Explorer view. You will have a few extra commands available:

- Jump to Bookmark
- Edit Label
- Remove Bookmark
- Clear Bookmark's file

Treeview
Treeview

Project and Session Based

The bookmarks are saved *per session* for the project that you are using. You don't have to worry about closing files in *Working Files*. When you reopen

It also works even if you only *preview* a file (simple click in TreeView). You can put bookmarks in any file and when you preview it again, the bookmar

License

MIT © Alessandro Fragnani

yzane/vscode-markdown-pdf

> Markdown converter for Visual Studio Code. Contribute to yzane/vscode-markdown-pdf development by creating an account on GitHub.

here be dragons

Include markdown fragment files: `:[alternate-text](relative-path-to-file.md)`.

Chromium download starts automatically when Markdown PDF is installed and Markdown file is first opened with Visual Studio Code.

However, it is time-consuming depending on the environment because of its large size (~ 170Mb Mac, ~ 282Mb Linux, ~ 280Mb Win).

During downloading, the message `Installing Chromium` is displayed in the status bar.

If you are behind a proxy, set the `http.proxy` option to settings.json and restart Visual Studio Code.

If the download is not successful or you want to avoid downloading every time you upgrade Markdown PDF, please specify the installed Chrome or 'Chromi

Usage

Command Palette

1. Open the Markdown file
2. Press `F1` or `Ctrl+Shift+P`

3. Type `export` and select below
    - `markdown-pdf: Export (settings.json)`
    - `markdown-pdf: Export (pdf)`
    - `markdown-pdf: Export (html)`
    - `markdown-pdf: Export (png)`
    - `markdown-pdf: Export (jpeg)`
    - `markdown-pdf: Export (all: pdf, html, png, jpeg)`

usage1

Menu

1. Open the Markdown file
2. Right click and select below
    - `markdown-pdf: Export (settings.json)`
    - `markdown-pdf: Export (pdf)`
    - `markdown-pdf: Export (html)`
    - `markdown-pdf: Export (png)`
    - `markdown-pdf: Export (jpeg)`
    - `markdown-pdf: Export (all: pdf, html, png, jpeg)`

usage2

Auto convert

1. Add `"markdown-pdf.convertOnSave": true` option to **settings.json**
2. Restart Visual Studio Code
3. Open the Markdown file
4. Auto convert on save

Extension Settings

Visual Studio Code User and Workspace Settings

1. Select **File > Preferences > UserSettings or Workspace Settings**
2. Find markdown-pdf settings in the **Default Settings**
3. Copy `markdown-pdf.*` settings
4. Paste to the **settings.json**, and change the value

demo

Options

List

Category	Option name	Configuration scope
Save options	markdown-pdf.type	
	markdown-pdf.convertOnSave	
	markdown-pdf.convertOnSaveExclude	
	markdown-pdf.outputDirectory	
	markdown-pdf.outputDirectoryRelativePathFile	
Styles options	markdown-pdf.styles	
	markdown-pdf.stylesRelativePathFile	
	markdown-pdf.includeDefaultStyles	
Syntax highlight options	markdown-pdf.highlight	
	markdown-pdf.highlightStyle	
Markdown options	markdown-pdf.breaks	
Emoji options	markdown-pdf.emoji	
Configuration options	markdown-pdf.executablePath	
Common Options	markdown-pdf.scale	
PDF options	markdown-pdf.displayHeaderFooter	resource
	markdown-pdf.headerTemplate	resource
	markdown-pdf.footerTemplate	resource
	markdown-pdf.printBackground	resource
	markdown-pdf.orientation	resource
	markdown-pdf.pageRanges	resource
	markdown-pdf.format	resource
	markdown-pdf.width	resource
	markdown-pdf.height	resource
	markdown-pdf.margin.top	resource
	markdown-pdf.margin.bottom	resource
	markdown-pdf.margin.right	resource
	markdown-pdf.margin.left	resource
PNG JPEG options	markdown-pdf.quality	
	markdown-pdf.clip.x	
	markdown-pdf.clip.y	
	markdown-pdf.clip.width	
	markdown-pdf.clip.height	
	markdown-pdf.omitBackground	
PlantUML options	markdown-pdf.plantumlOpenMarker	
	markdown-pdf.plantumlCloseMarker	
	markdown-pdf.plantumlServer	
markdown-it-include options	markdown-pdf.markdown-it-include.enable	
mermaid options	markdown-pdf.mermaidServer	

Save options

markdown-pdf.type

- Output format: pdf, html, png, jpeg
- Multiple output formats support
- Default: pdf

```
"markdown-pdf.type": [ "pdf", "html", "png", "jpeg" ],
```

markdown-pdf.convertOnSave

- Enable Auto convert on save
- boolean. Default: false
- To apply the settings, you need to restart Visual Studio Code

markdown-pdf.convertOnSaveExclude

- Excluded file name of convertOnSave option

```
"markdown-pdf.convertOnSaveExclude": [ "^work", "work.md$", "work|test", "[0-9][0-9][0-9][0-9]-work", "work\\test" // All '\' need to be written as '\
```

markdown-pdf.outputDirectory

- Output Directory
- All `\` need to be written as `\\` (Windows)

```
"markdown-pdf.outputDirectory": "C:\\work\\output",
```

- Relative path
  - If you open the `Markdown file`, it will be interpreted as a relative path from the file
  - If you open a `folder`, it will be interpreted as a relative path from the root folder
  - If you open the `workspace`, it will be interpreted as a relative path from the each root folder
    - See Multi-root Workspaces

```
"markdown-pdf.outputDirectory": "output",
```

- Relative path (home directory)
  - If path starts with `~`, it will be interpreted as a relative path from the home directory

```
"markdown-pdf.outputDirectory": "~/output",
```

- If you set a directory with a `relative path`, it will be created if the directory does not exist
- If you set a directory with an `absolute path`, an error occurs if the directory does not exist

markdown-pdf.outputDirectoryRelativePathFile

- If `markdown-pdf.outputDirectoryRelativePathFile` option is set to `true`, the relative path set with markdown-pdf.outputDirectory is interpreted as relative
- It can be used to avoid relative paths from folders and workspaces
- boolean. Default: false

Styles options

markdown-pdf.styles

- A list of local paths to the stylesheets to use from the markdown-pdf
- If the file does not exist, it will be skipped
- All `\` need to be written as `\\` (Windows)

```
"markdown-pdf.styles": [ "C:\\Users\\\\Documents\\markdown-pdf.css", "/home//settings/markdown-pdf.css", ],
```

- Relative path
  - If you open the `Markdown file`, it will be interpreted as a relative path from the file
  - If you open a `folder`, it will be interpreted as a relative path from the root folder
  - If you open the `workspace`, it will be interpreted as a relative path from the each root folder
    - See Multi-root Workspaces

```
"markdown-pdf.styles": [ "markdown-pdf.css", ],
```

- Relative path (home directory)
  - If path starts with `~`, it will be interpreted as a relative path from the home directory

```
"markdown-pdf.styles": [ "~/.config/Code/User/markdown-pdf.css" ],
```

- Online CSS (https://xxx/xxx.css) is applied correctly for JPG and PNG, but problems occur with PDF #67

```
"markdown-pdf.styles": [ "https://xxx/markdown-pdf.css" ],
```

markdown-pdf.stylesRelativePathFile

- If `markdown-pdf.stylesRelativePathFile` option is set to `true`, the relative path set with markdown-pdf.styles is interpreted as relative from the file
- It can be used to avoid relative paths from folders and workspaces
- boolean. Default: false

markdown-pdf.includeDefaultStyles

- Enable the inclusion of default Markdown styles (VSCode, markdown-pdf)
- boolean. Default: true

Syntax highlight options

markdown-pdf.highlight

- Enable Syntax highlighting
- boolean. Default: true

markdown-pdf.highlightStyle

- Set the style file name. for example: github.css, monokai.css ...
- file name list
- demo site : https://highlightjs.org/static/demo/

```
"markdown-pdf.highlightStyle": "github.css",
```

Markdown options

markdown-pdf.breaks

- Enable line breaks
- boolean. Default: false

Emoji options

markdown-pdf.emoji

- Enable emoji. EMOJI CHEAT SHEET
- boolean. Default: true

Configuration options

markdown-pdf.executablePath

- Path to a Chromium or Chrome executable to run instead of the bundled Chromium
- All \ need to be written as \\ (Windows)
- To apply the settings, you need to restart Visual Studio Code

```
"markdown-pdf.executablePath": "C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe"
```

Common Options

markdown-pdf.scale

- Scale of the page rendering
- number. default: 1

PDF options

- pdf only. puppeteer page.pdf options

markdown-pdf.displayHeaderFooter

- Enable display header and footer
- boolean. Default: true

markdown-pdf.headerTemplate

markdown-pdf.footerTemplate

- HTML template for the print header and footer
- `<span class='date'></span>` : formatted print date
- `<span class='title'></span>` : markdown file name
- `<span class='url'></span>` : markdown full path name
- `<span class='pageNumber'></span>` : current page number
- `<span class='totalPages'></span>` : total pages in the document

```
"markdown-pdf.headerTemplate": "



",

"markdown-pdf.footerTemplate": "
 /
",
```

markdown-pdf.printBackground

- Print background graphics
- boolean. Default: true

markdown-pdf.orientation

- Paper orientation
- portrait or landscape
- Default: portrait

markdown-pdf.pageRanges

- Paper ranges to print, e.g., '1-5, 8, 11-13'
- Default: all pages

```
"markdown-pdf.pageRanges": "1,4-",
```

markdown-pdf.format

- Paper format
- Letter, Legal, Tabloid, Ledger, A0, A1, A2, A3, A4, A5, A6
- Default: A4

"markdown-pdf.format": "A4",

markdown-pdf.width

markdown-pdf.height

- Paper width / height, accepts values labeled with units(mm, cm, in, px)
- If it is set, it overrides the markdown-pdf.format option

"markdown-pdf.width": "10cm", "markdown-pdf.height": "20cm",

markdown-pdf.margin.top

markdown-pdf.margin.bottom

markdown-pdf.margin.right

markdown-pdf.margin.left

- Paper margins.units(mm, cm, in, px)

"markdown-pdf.margin.top": "1.5cm", "markdown-pdf.margin.bottom": "1cm", "markdown-pdf.margin.right": "1cm", "markdown-pdf.margin.left": "1cm",

PNG JPEG options

- png and jpeg only. puppeteer page.screenshot options

markdown-pdf.quality

- jpeg only. The quality of the image, between 0-100. Not applicable to png images

"markdown-pdf.quality": 100,

markdown-pdf.clip.x

markdown-pdf.clip.y

markdown-pdf.clip.width

markdown-pdf.clip.height

- An object which specifies clipping region of the page
- number

// x-coordinate of top-left corner of clip area "markdown-pdf.clip.x": 0,

// y-coordinate of top-left corner of clip area "markdown-pdf.clip.y": 0,

// width of clipping area "markdown-pdf.clip.width": 1000,

// height of clipping area "markdown-pdf.clip.height": 1000,

markdown-pdf.omitBackground

- Hides default white background and allows capturing screenshots with transparency
- boolean. Default: false

PlantUML options

markdown-pdf.plantumlOpenMarker

- Oppening delimiter used for the plantuml parser.
- Default: @startuml

markdown-pdf.plantumlCloseMarker

- Closing delimiter used for the plantuml parser.
- Default: @enduml

markdown-pdf.plantumlServer

- Plantuml server. e.g. http://localhost:8080
- Default: http://www.plantuml.com/plantuml
- For example, to run Plantuml Server locally #139 :

  ```
  docker run -d -p 8080:8080 plantuml/plantuml-server:jetty
  ```

  plantuml/plantuml-server - Docker Hub

markdown-it-include options

markdown-pdf.markdown-it-include.enable

- Enable markdown-it-include.
- boolean. Default: true

mermaid options

markdown-pdf.mermaidServer

- mermaid server
- Default: https://unpkg.com/mermaid/dist/mermaid.min.js

FAQ

How can I change emoji size ?

1. Add the following to your stylesheet which was specified in the markdown-pdf.styles

Auto guess encoding of files

Using files.autoGuessEncoding option of the Visual Studio Code is useful because it automatically guesses the character code. See files.autoGuessEncoding

"files.autoGuessEncoding": true,

Output directory

If you always want to output to the relative path directory from the Markdown file.

For example, to output to the "output" directory in the same directory as the Markdown file, set it as follows.

"markdown-pdf.outputDirectory" : "output", "markdown-pdf.outputDirectoryRelativePathFile": true,

Page Break

Please use the following to insert a page break.

Source

<===============()===============>

194.WallabyJs.quokka-vscode

Quokka

- Install
- Docs
- Support

- Contact

PRO

Rapid JavaScript Prototyping in your Editor

Alt Text
Alt Text

Quokka.js is a developer productivity tool for rapid JavaScript / TypeScript prototyping. Runtime va

Live Execution and Results

Code runs immediately as you type, on unsaved changes; no need to do anything manually or switch context. Error messages are displayed right next to t

Alt Text
Alt Text

Live Coverage

Indicators in the gutter of your code editor are constantly updated in realtime to display code coverage so you can quickly see which lines of code ar

Alt Text
Alt Text

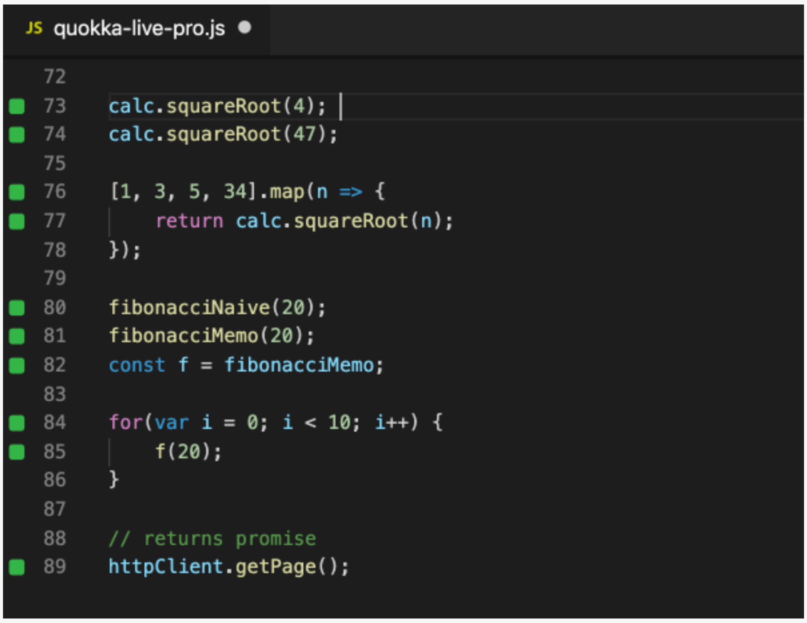
Value Explorer

Value Explorer allows non-primitive runtime values to be viewed and explored in an easy-to-navigate real-time treeview. This feature is great for expl

Alt Text
Alt Text

Live Comments and Values

Show and copy expression values by selecting them in your editor or with editor commands, accessible using keyboard shortcuts. A special comment forma



```
JS quokka-live-pro.js ●

calc.squareRoot(4);
calc.squareRoot(47);

[1, 3, 5, 34].map(n => {
    return calc.squareRoot(n);
});

fibonacciNaive(20);
fibonacciMemo(20);
const f = fibonacciMemo;

for(var i = 0; i < 10; i++) {
    f(20);
}

// returns promise
httpClient.getPage();
```

Alt Text

Project Files Import

Pro

Import any files from your project into your Quokka file. Quokka will watch project files for changes and automatically update when dependent files ch

Alt Text
Alt Text

Quick Package Install

Pro

Quickly install any node package, without switching away from your editor, and without having to type the package name. When you install, choose wheth

Alt Text
Alt Text

Quick Start

To get started with Quokka.js in **VS Code**, install the extension first by clicking on the Extensions icon in the Activity Bar on the side of VS Code ar

## Live Feedback

```
You may create a new Quokka file, or start Quokka on an existing file.
The results of the execution are displayed right in the editor. To see
the full execution output, you may view the Quokka Console by invoking
the `Show Output`  command or
clicking the bottom-right status bar indicator.
```

You may create a new Quokka file, or start Quokka on an existing file. The results of the execution are displayed right in the editor. To see the full execution output, you may

```
You may create a new Quokka file, or start Quokka on an existing file by
using the `Start Quokka`  context
menu action in any opened file editor (you may also assign some shortcut
to the action). The results of the execution are displayed right in the
editor. To see the full execution output, you may view the Quokka
Console by clicking the bottom-right status bar indicator.
```

```
If you create a new Quokka scratch file and want to save it, then you
may press `F5`  to do so. Later
on, you may open the file and run Quokka using the
`Start Quokka`  context menu
action in the opened file editor.
```

```
It is recommended that you memorize a couple of Quokka keyboard
shortcuts (you may see them when using the editor's command palette).
This will make working with Quokka much faster.
```

```
To open a new Quokka file use `Cmd/Ctrl + K, J`{.highlighter-rouge
.language-plaintext} for JavaScript, or
`Cmd/Ctrl + K, T`  for
TypeScript. To start/restart Quokka on an existing file, use
`Cmd/Ctrl + K, Q` .
```

Live Logging/Compare

```
You may **use `console.log`  or
identifier expressions (i.e. just typing a variable name) to log any
values**.\
You may also **use sequence expressions to compare objects**:
```

Note that when using identifier expressions for logging (for example, typing `a` to see the value of the `a` variable), you may hit some limits in terms o

```
In this case, you may use
`console.log(a)`  to display
objects without the limitations.
```

Please also note that Boolean, Number and Function data types are **not supported** when use sequence expressions to compare objects (e.g. `obj1, obj2`).

Live Code Coverage

Once Quokka.js is running, you can see the code coverage in the gutter of your editor. The coverage is live, so you can start changing your code and t

As you may see, there are various colored squares displayed for each line of your source code.

- Gray squares mean that the source line has not been executed.
- Green squares mean that the source line has been executed.
- Yellow squares mean that the source line has only been partially executed. Quokka supports branch code coverage level, so if a line contains a log
- Red squares mean that the source line is the source of an error, or is in the stack of an error.

Value Explorer

Value Explorer allows you to inspect everything that is logged in Quokka with `console.log` , identifier expressions, live comments, and the `Show Value` com

Expression paths and values can be copied using the tree node's context menu.

Note that while Value Explorer is available for both Community and Pro editions, in Community edition only 2 levels of the explorer's tree can be expa

Auto-Expand Value Explorer Objects

Identifier Expressions and **Live Comments** can be provided with an additional hint to automatically expand objects when they are logged to Value Explore

Use this feature with small- to medium-sized objects when you want to expand all properties in Value Explorer. Having the properties expanded also hel

Note that automatically expanded objects have the following limitations:

- Cyclic Depedencies are not automatically expanded
- Functions are not automatically expanded
- Strings beyond 8192 characters are not automatically expanded
- Only the first 100 properties on a single object will be expanded
- Only the first 100 elements of an array will be expanded
- Only the first 10 levels of nested properties will be expanded
- Only the first 5000 properties across all objects will be expanded

VS Code Live Share Integration

When Microsoft's VS Code plugin for Live Share is installed alongside Quokka, Quokka provides software developers with the ability to **interactively co**

How does it work?

When the host and client both have Quokka plugin installed during a Live Share collaboration, when code changes are made and Quokka is running, Quokka

Members of the Live Share collaboration see the same Quokka display values when they are visible to the host. If the Live Share session is configured

Interactive Examples

The Wallaby.js team curates a set of interactive examples that may be launched from within VS Code using the `Create File (Ctrl + K, L)` command. These exar

Live Comments

PRO

While `console.log` may do a great job for displaying values, sometimes you may need to see the value right in the middle of an expression. For example,

The most powerful way to log **any** expression is to **use a special comment right after the expression you want to log**.

Inserting the special comment `/*?*/` after an expression (or just `//?` after a statement) will log just the value of that expression.

For example,

```
a.b()/*?*/.c().d()
```

Copy

will output the result of `a.b()` expression, and

```
a.b().c().d() /*?*/
// or just
a.b().c().d() //?
```

Copy

will output the result of the full `a.b().c().d()` expression.

You may also write **any JavaScript code right in the comment** to shape the output. The code has the access to the `$` variable which is the expression tha

Note that there's no constraints in terms of what the comment code can do. For example, the watch comment below is incrementing `d.e` value, and returni

Also, unlike console logging, the special comment logging has some built-in smarts. For example, when you place the comment after an expression that i

Live comment snippet

To save some time on typing the comment when you need it, you may create a code snippet with a custom keybinding.

To save some time on typing the comment when you need it, you may create a live template.

To save some time on typing the comment when you need it, you may create a code snippet with a custom keybinding.

To save some time on typing the comment when you need it, you may create code snippets.

Live Value Display

PRO

While the Live Comments feature provides an excellent way to log expression values and will keep displaying values when you change your code, sometime

Show Value

If you want to quickly display some expression value, but without modifying your code, you may do it by **simply selecting an expression in an editor**.

If you want to quickly display some expression value, but without modifying your code, you may do it by **simply selecting an expression in an editor**.

Copy Value

If you want to **copy an expression value to your clipboard without modifying your code**, Live Value Display allows you to do it with a special command (

If you want to **copy an expression value to your clipboard without modifying your code**, Live Value Display allows you to do it with a special intentior

Project Files Import

PRO

Quokka 'Community' edition allows you to import any locally installed node modules to a Quokka file. In addition to that, Quokka 'Pro' edition also al

- with Babel or TypeScript compilation if required;
- Quokka will also watch project files for changes and automatically update when dependent files change.

Quokka also resolves imported modules with non-relative names when using `tsconfig.json` and `jsconfig.json` files that specify `baseUrl` and/or `paths` settings.

Quick Package Install

PRO

This feature allows you to quickly install any node package, via `npm` or `yarn` , **without switching away from your editor, and without even having to type**

You may install missing packages via intention actions (`Alt+Enter`), or via the links in the Quokka output.

You may install missing packages by using the hover message, or with the `Cmd + K, I` keyboard shortcut (whenever there's a missing package), via command

By default Quokka uses the `npm install {packageName}` command. You may change it to `yarn add {packageName}` by setting the `installPackageCommand` value in your Quc

```
{
    "installPackageCommand": "yarn add {packageName}"
}
```

Copy

Live Performance Testing

PRO

The feature allows to **quickly see how various parts of your code perform.** It can be very helpful for identifying possible bottlenecks in your app and

Inserting the special comment `/*?.*/` **after any expression will report how much time it took to execute the expression.**

Adding the comment to an expression that gets executed multiple times, for example inside a loop, will make the tool to display total execution time,

You may also use the execution time reporting comment and add to it any expression that you may use in live comments to display both an expression exe

For example,

```
a() //?. $
```

Copy

displays `a()` execution time and result.

Run on Save/Run Once

PRO

Choose how and when Quokka should run on your files. In addition to the community edition's **Automatic** mode, you can start Quokka and only **Run on Save**

The **Run on Save** and **Run Once** actions are available along-side other Quokka commands, in your IDE's list of commands. If you use these commands often,

Runtime

Quokka.js is using your system's node.js to run your code. You also may configure Quokka to use any specific node.js version, if you are using `nvm` for

By default Quokka uses `esm` module for JavaScript files, allowing ES imports and top level await to be used with zero configuration.

Browser-like Runtime

To simulate a **browser environment**, Quokka supports jsdom. The `jsdom` package must be installed and available from your project as well adding as a few

Configuration

Quokka.js does not need any configuration by default. It will run your code using your system's node.js. It may also run your TypeScript code without

If you would like to use Babel/React JSX or override your `tsconfig.json` settings, then you may configure Quokka.js.

If you are using VS Code, you can override our editor display settings with VS Code User Setting overrides. You can view the overridable settings in t

You may override the coverage indicator colors in VS Code's user settings (`settings.json` file). The snippet below shows the config with Quokka's defaul

```
{
    ...

   "quokka.colors": {
       "covered": "#62b455",
       "errorPath": "#ffa0a0",
       "errorSource": "#fe536a",
       "notCovered": "#cccccc",
       "partiallyCovered": "#d2a032"
   }



}
```

Copy

If you are using VS Code, you may change your Quokka console output to `compact` mode by updating your VS Code setting `quokka.compactMessageOutput` to `true` .

Plugins

Quokka.js provides a plugin model that may be configured by using the `plugins` setting. Quokka's plugin model allows you to specify a file or a node moc

Examples

Because Quokka.js runs your code using `node.js`, sometimes a little more configuration may be required to match your project's runtime configuration. If

- Create-React-App
- Babel + import
- TypeScript + import
- React + Babel + jsdom plugin
- Quokka Alias

Questions and issues

If you find an issue, please report it in our support repository.

Configuration**

Please enable JavaScript to view the comments powered by Disqus.

1. Getting started
2. Live Feedback
3. Live Logging/Compare
4. Live Code Coverage
5. Value Explorer
6. VS Code Live Share Integration
7. Interactive Examples
8. Live Comments

9. Live Value Display
10. Project Files Import
11. Quick Package Install
12. Live Performance Testing
13. Run on Save/Run Once
14. Runtime
15. Browser-like Runtime
16. Configuration
17. Plugins
18. Examples
19. Questions and issues

Adds :emoji: syntax support to VS Code's built-in Markdown preview

Features

- Adds support for :emoji: syntax to VS Code's built-in markdown preview

Microsoft Edge Tools for VS Code - Visual Studio Marketplace

> Extension for Visual Studio Code - Use the Microsoft Edge Tools from within VS Code to see your site's runtime HTML structure, alter its layout, fi

Show the browser's Elements and Network tool inside the Visual Studio Code editor and use it to fix CSS issues with your site and inspect network acti

A Visual Studio Code extension that allows you to use the browser's Elements and Network tool from within the editor. The DevTools will connect to an

Note: This extension *only* supports Microsoft Edge (version greater than 80.0.361.48)

Microsoft Edge Tools - Demo
Microsoft Edge Tools - Demo

Supported Features

- Debug configurations for launching Microsoft Edge browser in remote-debugging mode and auto attaching the tools.
- Side Bar view for listing all the debuggable targets, including tabs, extensions, service workers, etc.
- Fully featured Elements and Network tool with views for HTML, CSS, accessibility and more.
- Screen-casting feature to allow you to see your page without leaving Visual Studio Code.
- Go directly to the line/column for source files in your workspace when clicking on a link or CSS rule inside the Elements tool.
- Auto attach the Microsoft Edge Tools when you start debugging with the Debugger for Microsoft Edge extension.
- Debug using a windowed or headless version of the Microsoft Edge Browser

Using the Extension

Getting Started

For use inside Visual Studio Code:

1. Install any channel (Canary/Dev/etc.) of Microsoft Edge.
2. Install the extension Microsoft Edge Tools.
3. Open the folder containing the project you want to work on.
4. (Optional) Install the Debugger for Microsoft Edge extension.

Changing Extension Settings

You can customize the extension to your needs. You can reach the settings by clicking the gear icon of the extension listing or via the settings menu.

The gear icon in the extension listing
The gear icon in the extension listing

Turning on Network Inspection

You can enable the Network Pane to inspect any network request of the attached browser. To do this, change the setting and restart the extension.

Example how to turn on the network pane
Example how to turn on the network pane

You can see an example of the change in the following screencast. After restart, you get an extra tab with network functionality.

Example how to turn on the network pane
Example how to turn on the network pane

Turning on Headless Mode

By default, the extension will launch the browser in its own window. This means you get an extra browser icon in your task bar and you need to turn or

Note: In the past we had issues on Macintosh computers where the Microsoft Edge instance reported itself as "inactive" when the window wasn't visible.

Example how to turn on the network pane
Example how to turn on the network pane

You can see an example of the change in the following screencast:

Example how to turn on the headless mode
Example how to turn on the headless mode

Using the tools

The extension operates in two modes - it can launch an instance of Microsoft Edge navigated to your app, or it can attach to a running instance of Mic

You can now use the high-fidelity tools to tweak your CSS and inspect network calls and go directly back to your code without leaving the editor.

Microsoft Edge Tools - Demo
Microsoft Edge Tools - Demo

Opening source files from the Elements tool

One of the features of the Elements tool is that it can show you what file applied the styles and event handlers for a given node.

Microsoft Edge Tools - Links
Microsoft Edge Tools - Links

The source files for these applied styles and attached event handlers appear in the form of links to a url specified by the browser. Clicking on one

An example webpack configuration for sass and typescript is given below:

```
module.exports = {
  devtool: "source-map",
  module: {
    rules: [
      {
        test: /\.ts$/,
        loader: "ts-loader"
      },
      {
        test: /\.(s*)css$/,
        use: [
          { loader: MiniCssExtractPlugin.loader },
          { loader: "css-loader", options: { sourceMap: true } },
          { loader: "sass-loader", options: { sourceMap: true } }
        ]
      },
    ]
  }
}
```

With source maps enabled, you may also need to configure the extension settings/launch.json config to add customized paths between your runtime urls

Debug Configuration

You can launch the Microsoft Edge Tools extension like you would a debugger, by using a `launch.json` config file.

Microsoft Edge Tools works great when paired with Debugger for Microsoft Edge, you can use the first one to design your frontend and the latter to de

To add a new debug configuration, in your `launch.json` add a new debug config with the following parameters:

- `type` - The name of the debugger which must be `vscode-edge-devtools.debug`. **Required.**
- `request` - `launch` to open a new browser tab or `attach` to connect to an existing tab. **Required.**
- `name` - A friendly name to show in the Visual Studio Code UI. **Required.**
- `url` - The url for the new tab or of the existing tab. **Optional.**
- `file` - The local file path for the new tab or of the existing tab. **Optional.**
- `webRoot` - The directory that files are served from. Used to resolve urls like `http://localhost:8000/app.js` to a file on disk like `/out/app.js`. **Optional.**

```
{ "version": "0.1.0", "configurations": [ { "type": "vscode-edge-devtools.debug", "request": "launch", "name": "Launch Microsoft Edge and open the
```

Other optional launch config fields

- `browserPath`: The full path to the browser executable that will be launched. If not specified the most stable channel of Microsoft Edge will be laun
- `hostname`: By default the extension searches for debuggable instances using `localhost`. If you are hosting your web app on a remote machine you can sp
- `port`: By default the extension will set the remote-debugging-port to `9222`. Use this option to specify a different port on which to connect.

- `userDataDir`: Normally, if Microsoft Edge is already running when you start debugging with a launch config, then the new instance won't start in remo
- `useHttps`: By default the extension will search for attachable instances using the `http` protocol. Set this to true if you are hosting your web app ov
- `sourceMaps`: By default, the extension will use sourcemaps and your original sources whenever possible. You can disable this by setting `sourceMaps` to
- `pathMapping`: This property takes a mapping of URL paths to local paths, to give you more flexibility in how URLs are resolved to local files. `"webRoo`
- `sourceMapPathOverrides`: A mapping of source paths from the sourcemap, to the locations of these sources on disk. See Sourcemaps for more information
- `urlFilter`: A string that can contain wildcards that will be used for finding a browser target, for example, "localhost:*/app" will match either "ht
- `timeout`: The number of milliseconds that the Microsoft Edge Tools will keep trying to attach to the browser before timing out. Defaults to 10000ms.

Sourcemaps

The elements tool uses sourcemaps to correctly open original source files when you click links in the UI, but sometimes the sourcemaps aren't generate

The left hand side of the mapping is a pattern that can contain a wildcard, and will be tested against the `sourceRoot` + `sources` entry in the source map.

A few mappings are applied by default, corresponding to some common default configs for Webpack and Meteor: Note: These are the mappings that are incl

```
"sourceMapPathOverrides": {
    "webpack:///./~/*": "${webRoot}/node_modules/*",
    "webpack:///./*": "${webRoot}/*",
    "webpack:///*": "*",
    "webpack:///src/*": "${webRoot}/*",
    "meteor://💻app/*": "${webRoot}/*"
}
```

If you set `sourceMapPathOverrides` in your launch config, that will override these defaults. `${workspaceFolder}` and `${webRoot}` can be used there.

See the following examples for each entry in the default mappings (`webRoot = /Users/me/project`):

```
"webpack:///./~/*": "${webRoot}/node_modules/*"
Example:
"webpack:///./~/querystring/index.js"
-> "/Users/me/project/node_modules/querystring/index.js"
"webpack:///./":   "${webRoot}/"
Example:
"webpack:///./src/app.js" -> "/Users/me/project/src/app.js"


"webpack:///": ""
Example:
"webpack:///project/app.ts" -> "/project/app.ts"


"webpack:///src/": "${webRoot}/"
Example:
"webpack:///src/app.js" -> "/Users/me/project/app.js"


"meteor://💻app/": "${webRoot}/"
Example:
"meteor://💻app/main.ts"-&gt;"/Users/me/project/main.ts"
```

Ionic/gulp-sourcemaps note

Ionic and gulp-sourcemaps output a sourceRoot of `"/source/"` by default. If you can't fix this via your build config, try this setting:

```
"sourceMapPathOverrides": {
    "/source/*": "${workspaceFolder}/*"
}
```

Launching the browser via the side bar view

- Start Microsoft Edge via the side bar
  - Click the `Microsoft Edge Tools` view in the side bar.
  - Click the `Open a new tab` icon to launch the browser (if it isn't open yet) and open a new tab.

- Attach the Microsoft Edge Tools via the side bar view
  - Click the `Attach` icon next to the tab to open the Microsoft Edge Tools.

Launching the browser manually

- Start Microsoft Edge with remote-debugging enabled on port 9222:
  - `msedge.exe --remote-debugging-port=9222`
  - Navigate the browser to the desired URL.
- Attach the Microsoft Edge Tools via a command:
  - Run the command `Microsoft Edge Tools: Attach to a target`
  - Select a target from the drop down.

Attaching automatically when launching the browser for debugging

- Install the Debugger for Microsoft Edge extension
- Setup your `launch.json` configuration to launch and debug Microsoft Edge.
  - See Debugger for Microsoft Edge Readme.md.
- Start Microsoft Edge for debugging.
  - Once debugging has started, the Microsoft Edge Tools will auto attach to the browser (it will keep retrying until the Debugger for Microsoft E
  - This auto attach functionality can be disabled via the `vscode-edge-devtools.autoAttachViaDebuggerForEdge` Visual Studio Code setting.

This project welcomes contributions and suggestions. Most contributions require you to agree to a Contributor License Agreement (CLA) declaring that y

See CONTRIBUTING.md for more information.

Data/Telemetry

This project collects usage data and sends it to Microsoft to help improve our products and services. Read Microsoft's privacy statement to learn more

Source

<===============(Turbo Console Log)===============>

Turbo Console Log

> Extension for Visual Studio Code - Automating the process of writing meaningful log messages.

Main Functionality

This extension make debugging much easier by automating the operation of writing meaningful log message.

Features

I. Insert meaningful log message automatically

Two steps:

- Selecting the variable which is the subject of the debugging
- Pressing ctrl + alt + L

The log message will be inserted in the next line relative to the selected variable like this:

```
console.log("SelectedVariableEnclosingClassName -> SelectedVariableEnclosingFunctionName -> SelectedVariable", SelectedVariable)
```

alt text
alt text

Multiple cursor support:

alt text
alt text

Properties:

- turboConsoleLog.wrapLogMessage (boolean): Whether to wrap the log message or not.
- turboConsoleLog.logMessagePrefix (string): The prefix of the log message.
- turboConsoleLog.addSemicolonInTheEnd (boolean): Whether to put a semicolon in the end of the log message or not.
- turboConsoleLog.insertEnclosingClass (boolean): Whether to insert or not the enclosing class of the selected variable in the log message.
- turboConsoleLog.insertEnclosingFunction (boolean): Whether to insert or not the enclosing function of the selected variable in the log message.
- turboConsoleLog.quote (enum): Double quotes (""), single quotes ('') or backtick(``).

A wrapped log message :

alt text
alt text

II. Comment all log messages, inserted by the extension, from the current document

All it takes to comment all log messages, inserted by the extension, from the current document is to press alt + shift + c

alt text
alt text

III. Uncomment all log messages, inserted by the extension, from the current document

All it takes to uncomment all log messages, inserted by the extension, from the current document is to press alt + shift + u

alt text
alt text

IV. Delete all log messages, inserted by the extension, from the current document

All it takes to delete all log messages, inserted by the extension, from the current document is to press alt + shift + d

alt text
alt text

Regex Previewer - Visual Studio Marketplace

> Extension for Visual Studio Code - Regex matches previewer for JavaScript, TypeScript, PHP and Haxe in Visual Studio Code.

Features

Shows the current regular expression's matches in a side-by-side document. This can be turned on/off with `Ctrl+Alt+M` (`⌥⌘M`).

Global and multiline options can be added for evaluation with a side-by-side document through a status bar entry. This can be useful when the side-by-

Regex Previewer in Action
Regex Previewer in Action

Source

<===============(JavaScript Snippet Pack)===============>

JavaScript Snippet Pack - Visual Studio Marketplace

> Extension for Visual Studio Code - A snippet pack to make you more productive working with JavaScript

JavaScript Snippet Pack for Visual Studio Code

Download this extension from the Visual Studio Code Marketplace

Usage

A snippet pack to make you more productive working with JavaScript. Based on Visual Studio extension by Mads Kristensen, which is based on Atom snippe

This extension ships a bunch of useful code snippets for the JavaScript and TypeScript editors.

Here's the full list of all the snippets:

Console

[cd] console.dir

```
console.dir(${1});
```

[ce] console.error

```
console.error(${1});
```

[ci] console.info

```
console.info(${1});
```

[cl] console.log

```
console.log(${1});
```

[cw] console.warn

```
console.warn(${1});
```

[de] debugger

```
debugger;
```

DOM

[ae] addEventListener

```
${1:document}.addEventListener('${2:load}', function(e) {
    ${3:// body}
});
```

[ac] appendChild

```
${1:document}.appendChild(${2:elem});
```

[rc] removeChild

```
${1:document}.removeChild(${2:elem});
```

[cel] createElement

```
${1:document}.createElement(${2:elem});
```

[cdf] createDocumentFragment

```
${1:document}.createDocumentFragment();
```

[ca] classList.add

```
${1:document}.classList.add('${2:class}');
```

[ct] classList.toggle

```
${1:document}.classList.toggle('${2:class}');
```

[cr] classList.remove

```
${1:document}.classList.remove('${2:class}');
```

[gi] getElementById

```
${1:document}.getElementById('${2:id}');
```

[gc] getElementsByClassName

```
${1:document}.getElementsByClassName('${2:class}');
```

[gt] getElementsByTagName

```
${1:document}.getElementsByTagName('${2:tag}');
```

[ga] getAttribute

```
${1:document}.getAttribute('${2:attr}');
```

[sa] setAttribute

```
${1:document}.setAttribute('${2:attr}', ${3:value});
```

[ra] removeAttribute

```
${1:document}.removeAttribute('${2:attr}');
```

[ih] innerHTML

```
${1:document}.innerHTML = '${2:elem}';
```

[tc] textContent

```
${1:document}.textContent = '${2:content}';
```

[qs] querySelector

```
${1:document}.querySelector('${2:selector}');
```

[qsa] querySelectorAll

```
${1:document}.querySelectorAll('${2:selector}');
```

Loop

[fe] forEach

```
${1:array}.forEach(function(item) {
    ${2:// body}
});
```

Function

[fn] function

```
function ${1:methodName} (${2:arguments}) {
    ${3:// body}
}
```

[afn] anonymous function

```
function(${1:arguments}) {
    ${2:// body}
}
```

[pr] prototype

```
${1:object}.prototype.${2:method} = function(${3:arguments}) {
    ${4:// body}
}
```

[iife] immediately-invoked function expression

```
(function(${1:window}, ${2:document}) {
    ${3:// body}
})(${1:window}, ${2:document});
```

[call] function call

```
${1:method}.call(${2:context}, ${3:arguments})
```

[apply] function apply

```
${1:method}.apply(${2:context}, [${3:arguments}])
```

[ofn] function as a property of an object

```
${1:functionName}: function(${2:arguments}) {
    ${3:// body}
}
```

JSON

[jp] JSON.parse

```
JSON.parse(${1:obj});
```

[js] JSON.stringify

```
JSON.stringify(${1:obj});
```

Timer

[si] setInterval

```
setInterval(function() {
    ${0:// body}
}, ${1:1000});
```

[st] setTimeout

```
setTimeout(function() {
    ${0:// body}
}, ${1:1000});
```

Misc

[us] use strict

```
'use strict';
```

[al] alert

```
alert('${1:msg}');
```

[co] confirm

```
confirm('${1:msg}');
```

[pm] prompt

prompt('${1:msg}');

<===============(Node.js Exec)===============>

Node.js Exec - Visual Studio Marketplace

> Extension for Visual Studio Code - Execute the current file or your selected code with node.js.

Execute the current file or your selected code with node.js.

Usage

- To execute the current file or the selection press `F8` or use the command `Execute Node.js`
- To cancel a running process press `F9`

Configuration

Clear output before execution

```
{
  "miramac.node.clearOutput": true
}
```

Show start and end info

```
{
  "miramac.node.showInfo": true
}
```

Show stdout and stderr

```
{
  "miramac.node.showStdout": true,
  "miramac.node.showStderr": true
}
```

If `miramac.node.legacyMode` is `true` (default) the extention will not use new features and options. Because of some strange problems I can't reproduce, the

```
{
  "miramac.node.legacyMode": false
}
```

The folloing options need to set the legacyMode off

Set environment variables for execution:

```
{
  "miramac.node.env": {
      "NODE_ENV": "production"
  }
}
```

Add arguments for execution:

```
{
  "miramac.node.args": ["--port", "1337"]
}
```

Add options for execution:

```
{
  "miramac.node.options": ["--require", "babel-register"]
}
```

Change the node binary for execution

```
{
  "miramac.node.nodeBin": "/path/to/some/bin/node-7.0"
}
```

Some code that is executed with each run

```
{
  "miramac.node.includeCode": "const DEBUG = true; const fs = require('fs'); "
}
```

How it works

The selected code or if nothing is selected, the active file, is written in a temporarily file (something like `node_<random-sring>.tmp`). You don't have t

```
require('child_process').spawn('node', options,[tmpFile, args])
```

Search Editor: Apply Changes - Visual Studio Marketplace

> Extension for Visual Studio Code - Apply local Search Editor edits to your workspace

In the marketplace here.

Apply changes in a Search Editor to files in a workspace.

Example usage
Example usage

Steps:

- Run a search
- Edit results
- Run command "Apply Search Editor changes to worksapce"

> Warning: Ensure the workspace is in sync with the Search Editor before starting to make changes, otherwise data loss may occur. This can be done by

Search editor changes will overwrite their target files at the lines specified in the editor - if the lines in the target document have been modified

This is a very early expermient of what writing local search editor changes out to the workspace might look like, please file bugs and feature request

Known Limitations

- No way to apply edits that insert or delete lines.

Keybindings

You may find the following keybinding helpful for making the default "save" behaviour write out to workspace rather than save a copy of the results:

```
{
  "key": "cmd+s",
  "command": "searchEditorApplyChanges.apply",
  "when": "inS,earchEditor"
}
```

Source

<===============(Node TDD)===============>

Node TDD - Visual Studio Marketplace

> Extension for Visual Studio Code - Ease test-driven development in Node and JavaScript

> A Visual Studio Code extension to ease test-driven development in Node and JavaScript.

Features

node-tdd demo
node-tdd demo

- Activates when a workspace containing `package.json` is opened.
- Triggers an automatic test build whenever source files are updated.
- Shows a colour-coded build summary.
- Shows the average test coverage (experimental).
- Optionally, if your test runner generates TAP outputs, use the `nodeTdd.reporter` setting to provide a more meaningful test summary:

tap
tap

- Finally, use `nodeTdd.minimal` to reclaim some status bar real estate:

minimal mode
minimal mode

Settings

This extension contributes the following settings:

Setting	Type	Default	Description
`nodeTdd.activateOnStartup`	boolean	`true`	Activate TDD mode when workspace is opened
`nodeTdd.testScript`	string	`test`	The npm script to run tests
`nodeTdd.glob`	string	`{src,test}/**/*.{js,ts,jsx,tsx}`	The glob pattern for files to watch, relative to the workspace root
`nodeTdd.verbose`	boolean / object	`false` / `{onlyOnFailure: true}`	Show build status dialogs
`nodeTdd.minimal`	boolean	`false`	Minimise status bar clutter
`nodeTdd.buildOnActivation`	boolean	`false`	Run tests when TDD mode is activated
`nodeTdd.buildOnCreate`	boolean	`false`	Run tests when matching files are created
`nodeTdd.buildOnDelete`	boolean	`false`	Run tests when matching files are deleted
`nodeTdd.showCoverage`	boolean	`false`	Show the average test coverage if reported (experimental)
`nodeTdd.coverageThreshold`	number / null	`null`	The coverage threshold percentage, used to colour-code the coverage
`nodeTdd.reporter`	string / null	`null`	The test reporter used (currently only "tap" is supported)

Commands

The following commands are available from the commands menu as well as status bar buttons:

Command	Action
activate	Activate node-tdd in a workspace
deactivate	Deactivate node-tdd in a workspace
toggleOutput	Toggle detailed test results output
stopBuild	Stop a running build

Limitations and known issues

- The extension doesn't get activated if package.json was not initially present when the workspace was opened; a window restart will be required to de
- It doesn't work with watch mode/incremental test builds. The build process used for running tests must exit on each execution, otherwise it will n
- showCoverage is an experimental setting that currently only works with the text-summary reports from Lab, Istanbul and nyc. Disable it if it doesn't
- Despite recent fixes, it might still be flaky on Windows. Please report any issues.

- **Ironically for a TDD extension, it has very few tests of its own because I don't yet know how to**

  # <===============()===============>