
Visual Studio Code - Tutorial and Tips & Tricks

Guide for the developer

Christophe Avonture

25 march 2021, 15:24

Visual Studio Code - Tutorial and Tips & Tricks

Figure 1: Banner

Tutorial and collection of tips and tricks for Visual Studio Code

- Discovering Visual Studio Code {#discovering}
 - Installation {#discovering-installation}
 - The first keyboard shortcuts to learn {#discovering-shortcuts}
 - Settings {#discovering-settings}
 - Working with files and folders {#working-with-files}
 - * Keyboard shortcuts {#working-with-files-shortcuts}
 - Code traversal - Navigation {#traversal}
 - * Keyboard shortcuts {#traversal-shortcuts}
 - * Select first {#traversal-select-first}
 - Snippets {#snippets}
 - * Global {#snippets-global}
 - Built-in {#snippets-built-in}
 - User-defined ones {#snippets-user-defined}
 - Make snippets configurable {#snippets-user-placeholders}
 - Using variables
 - Settings {#snippets-settings}
 - Extensions {#snippets-extensions}
 - * Project-based {#snippets-project-based}
 - * Generator {#snippets-generator}
 - * Video tutorial {#snippets-video}
 - Using the built-in terminal {#terminal}
 - * Extensions {#terminal-extensions}

- Search
 - * Save the search
- Working with code {#working-with-code}
 - Keyboard shortcuts {#working-with-code-shortcuts}
 - * Code folding {#working-with-code-folding}
 - * Comments {#working-with-code-comments}
 - Integrated GIT {#integrated-git}
 - * Using with the terminal {#integrated-git-terminal}
 - * Source Control - GIT {#integrated-git-source-control}
 - * Extensions {#integrated-git-extensions}
 - Code definition
 - PHP-Unit {#phpunit}
 - * Extensions {#phpunit-extensions}
 - Using xDebug {#xdebug}
 - * Installation
 - Check if xdebug is loaded
 - * Visual Studio Code
 - Settings
 - Use inside vscode
 - Breakpoint
 - Logpoint
 - Debug keyboard shortcuts
 - Also read
 - PHP Formatting {#php-formatting}
 - Multiple cursors {#multiple-cursors}
 - * Insert prefix on each line {#multiple-cursors-insert-prefix}
 - Navigate between problems
 - regions
 - Renaming
 - * a symbol
 - * a class
- Some settings {#settings}
 - Editor settings {#settings-editor}
 - Files settings {#settings-files}

- Search settings {#settings-search}
- Telemetry settings {#settings-telemetry}
- Window settings {#settings-window}
- Workbench settings {#settings-workbench}
- Extensions {#extensions}
 - Get the list of all installed extensions {extensions-list-extensions}
 - Bash - Shellsript {#extensions-sh}
 - * Shell-format {#extensions-shell-format}
 - Extend core features of Visual Studio Code {#extensions-core}
 - * Active File In StatusBar {#extensions-active-file-statusbar}
 - * Autofold {#extensions-autofold}
 - * Better comments {#extensions-better-comments}
 - * Bookmarks {#extensions-bookmarks}
 - * Bracket Pair Colorizer {#extensions-bracket-pair-colorizer}
 - * Change case {#extensions-change-case}
 - * Code runner {#extension-code-runner}
 - * Code Spell Checker {#extensions-code-spell-checker}
 - * EditorConfig {#extensions-editorconfig}
 - * Favorites {#extensions-favorites}
 - * git-project-manager {#extensions-git-project-manager}
 - * Gitlens {#extensions-gitlens}
 - * Vscode Google Translate {#extensions-google-translate}
 - Configuration Google Translate
 - * Coloring of source codes according to the language {#extensions-highlight}
 - Apache configuration file {#highlight-apache}
 - Log File Highlighter {#extensions_log-file-highlighter}
 - * Prettier {#extensions-prettier-vscode}
 - * Project Manager {#extensions-project-manager}
 - * Rerun the last command {#extensions-terminal-rerun}
 - * REST client {#extensions-rest-client}
 - * Select highlight in minimap {#extensions-select-highlight-minimap}
 - * Snippet-creator {#extensions-snippet-creator}
 - * Sort lines {#extensions-sort-lines}
 - * Start any shell {#extensions-start-any-shell}
 - * Surround {#extensions-surround}
 - * Syncing {#extensions-syncing}

- * TabNine {#extensions-tabnine}
 - * Todo Tree {#extensions-todo-tree}
 - * vscode-icons {#extensions-vscode-icons}
- Javascript {#extensions-javascript}
 - * ESLint {#extensions-eslint}
- Markdown {#extensions-markdown}
 - * Emoji {#extensions-emoji}
 - * Markdown All in One {#extensions-markdown-all-in-one}
 - * Markdownlint {#extensions-markdownlint}
 - Configuration Markdownlint {#extensions-markdownlint-configuration}
- PHP {#extensions-php}
 - * Better PHPUnit {#extensions-php-better-phpunit}
 - Configure
 - Run a test method
 - Run a test file
 - Run the entire suite
 - Run the previous test
 - * Code-runner {#extensions-php-better-phpunit}
 - Configuration Code-runner
 - * PHP intelephense {#extensions-php-intelephense}
 - Configuration PHP intelephense {#extensions-php-intelephense-configuration}
 - * Laravel Blade Snippets {#extensions-laravel-blade}
 - * PHP-CS-FIXER {#extensions-php-cs-fixer}
 - * PHP DocBlocker {#extensions-php-docblocker}
 - * PHP Extension Pack {#extensions-php-extension-pack}
 - * PHP getters and setters {#extensions-php-getters-setters}
 - * PHP IntelliSense {#extensions-php-intellisense}
 - * PHP Namespace Resolver {extension-php-namespace-resolver}
- Tips
 - editorconfig
 - Interface
 - * Use color to identify projects
 - Keyboard shortcuts for Windows
 - PHP-CS-FIXER {#php-cs-fixer}
 - * Installation {#php-cs-fixer-installation}

- * Remove unused imports {#php-cs-fixer-no-unused-imports}
- Search and replace - Regex
 - * Remove all empty lines
 - * Remove all lines except those matching a regex
- Custom tasks {#custom-tasks}
 - * Run a DOS batch {#custom-tasks-batch}
 - * Run PHP-Unit {#custom-tasks-phpunit}
- Troubleshooting {#troubleshooting}
 - Intelephense {#troubleshooting-intelephense}
 - Phan {#troubleshooting-phan}
 - PHP-CS-FIXER {#troubleshooting-php-cs-fixer}
 - * PHP General Error {#troubleshooting-php-cs-fixer-php-general-error}
- Create your own extension {#own-extension}
- Bookmarks
 - VS Code can do that
 - VS Code channel on youtube
- Annex
 - List of extensions I use
- License

Discovering Visual Studio Code

Installation

Download Visual Studio Code.

The Insiders version contains the very latest version but can be unstable.

The first keyboard shortcuts to learn

- CTRL-B toggle visibility of the side bar (the one with the list of files in the project).
- CTRL-O to open a file.
- CTRL-K-CTRL-O to open a folder (a project).
- CTRL-R to show the list of recent folders (recent project).

- **TIP** Before clicking on the desired folder , be sure to press the CTRL key to open the folder in a new window ([link](#))
- CTRL-P to quickly retrieve and open a file in an open project.
- CTRL-O to open a recent folder (a project).
- SHIFT-CTRL-F to open the search pane (when a project has been opened).
- CTRL-SHIFT-P to open the Command Palette to quick access all commands of the editor.
- CTRL-, to get access to the User Settings.
- CTRL-SHIFT-N to open a new instance of vscode.
- CTRL-SHIFT-T to reopen a closed file.
- CTRL-SHIFT-X to open the list of Extensions.
- CTRL-B show/hide the left side-bar (if CTRL-B isn't used like in markdown files to set in bold).
- CTRL-K-Z maximize the screen, aka Zen mode.

Settings

There are two sorts of settings: User or Workspace. Remember, you can quickly access settings thanks the CTRL-,

Working with files and folders

The most efficient way to do is to use the tree-view at the left and right-click where the file/folder should be created.

Select the parent folder, right-click and select New file or New folder.

Keyboard shortcuts

- SHIFT-CTRL-E when a file is opened will activate the tree-view, retrieve the file and highlight it.

Code traversal - Navigation

Keyboard shortcuts

- CTRL-P to quick open / go to an open file. Files are sorted in the list from the most recent till the last recent so it's easy to go back to the previously edited file.
- CTRL-P-CTRL-P to open the most recent file (the second file in the list). You can press CTRL-P again and again to go to the third, fourth, ... When you release keys, the selected file will be immediately open.

- CTRL-SHIFT-O to open (browse) a symbol¹ in the opened file. Navigating with the arrows will select the portion of code in the editor.
- CTRL-T to open (browse) a symbol² in the entire project. In a markdown file, CTRL-T will display the list of every headings f.i.
- CTRL-TAB to switch between open tabs (just like Windows and active applications).
- CTRL-K-CTRL-Q go back to the last edited line.

Select first

- Select a word, a function name, a variable, a constant then press CTRL-T to open a list where that selection can be found. For instance, by selecting a constant name and pressing CTRL-T you'll obtain the full list of files using that constant and the one who defines the constant.

Snippets

Global

Built-in Snippets are pieces of pre-programmed code (like a `try...catch...`) that allow you to generate code without typing it entirely.

Depending on the open file and its language (php, js, markdown, ...), *VSCode* will offer standard snippets and it will also be possible to program your own.

Press CTRL-SHIFT-P to open the Palette Command then type `Insert Snippet` to get the list of already existing snippets.

User-defined ones It is however, possible to write your own snippets: press CTRL-SHIFT-P then `Configure User Snippets`. The programming language is chosen, e.g. PHP, which will open an editor with, here, the open PHP file.

A file called `php.json` will be open and you can start to create a snippet; f.i.:

```
{
  "strict": {
    "prefix": "strict",
    "body": ["declare (strict_types = 1);", ""],
    "description": "Add strict_types"
```

¹A symbol is the name of a constant, of a function, part of a title (for a markdown document f.i.), ... You can get access to symbols of an opened file by using the CTRL-SHIFT-O keyboard shortcut.

²A symbol is the name of a constant, of a function, part of a title (for a markdown document f.i.), ... You can get access to symbols of an opened file by using the CTRL-SHIFT-O keyboard shortcut.


```
}
}
```

The code here above defines a snippet called `strict` and it's for a `.php` file. Open such file, start to type `stri` (you can type only a few letters) and press the CTRL-space keys. Select the snippet and press enter. You'll get the code defined in the body. It's magic.

Note: you can also desire to just press the TAB key. If so, check your `User Settings` and make sure the following key is well on on:

```
{
  "editor.tabCompletion": "on"
}
```

The example below will add two snippets, one called `img` and the other one `url`. These snippets are for markdown files so, when adding snippets to VSCode, in the `Configure User Snippets` command, select `markdown` as language first.

```
{
  "img": {
    "prefix": "img",
    "description": "Add an image tags",
    "body": ["![$${1:caption}](./images/${2:image}.png)"]
  },
  "url": {
    "prefix": "url",
    "description": "Add an url tags",
    "body": ["[${1:URL}](${1:URL})"]
  }
}
```

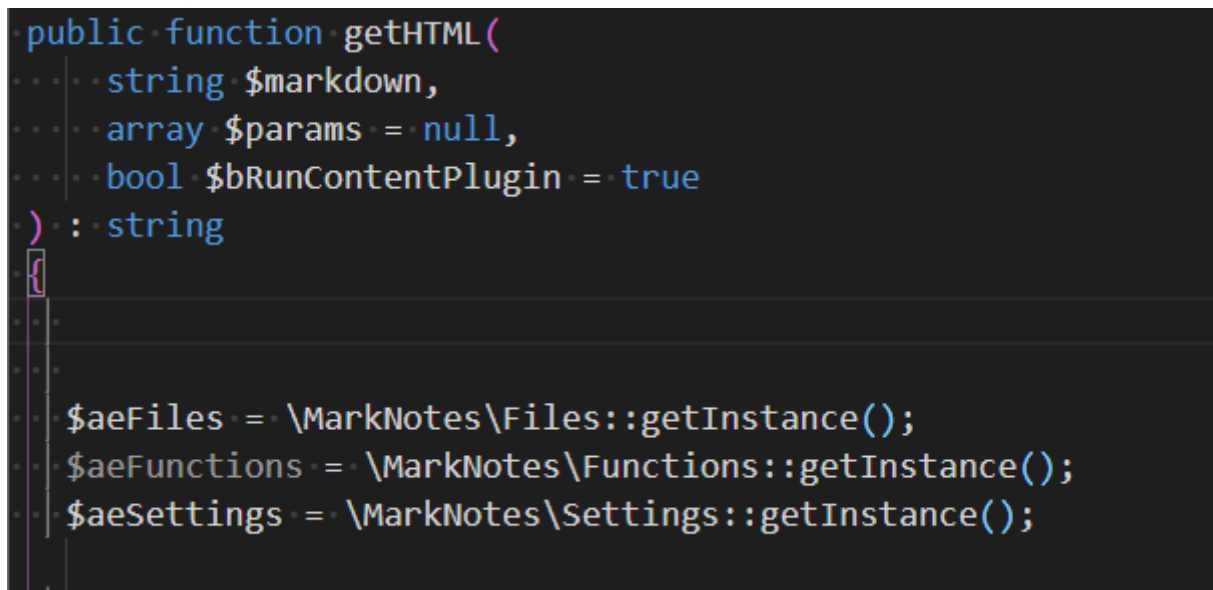
Make snippets configurable Take a look on the following snippet and the `$1` and `$2` placeholders.

```
{
  "var-dump": {
    "prefix": "vd",
    "body": [
      "echo '<pre>'.__METHOD__.'--'.__LINE__.'</pre>';",
      "echo '<pre>'.print_r(${1:\\$variableName}, true).'</pre>';",
      "die(${2:\\\"I'm dying right now\\\"});"
    ],
    "description": "Debug - Die and echo file/line info"
  }
}
```

```
}
}
```

Save this snippet in your editor and, in a php file, type `vd` followed by CTRL-space. You'll get now three new lines and the cursor will be placed where the `$1` placeholder was. Type a PHP variable and press TAB and see, the cursor will be now immediately put where `$2` was located. Thanks these placeholders it's easy to foresee, in a snippet, locations where you need to type dynamic content like variables, custom text, ...

Tip: use `${1:$variableName}` instead of just `$1` to show a place holder and/or a default value.



```
public function getHTML(
    string $markdown,
    array $params = null,
    bool $bRunContentPlugin = true
) : string
{
    $aeFiles = \MarkNotes\Files::getInstance();
    $aeFunctions = \MarkNotes\Functions::getInstance();
    $aeSettings = \MarkNotes\Settings::getInstance();
```

Figure 2: `var_dump`

Using variables When creating a snippet, we can f.i. use the current PHP filename. Imagine you've a file name `Customer.php` and there you wish to create a new class.

The following snippet will do the magic:

```
{
    "Class": {
        "prefix": "class",
        "body": ["class ${TM_FILENAME_BASE}", "{", "    return;", "}"],
        "description": "Create a new class"
    }
}
```

The `TM_FILENAME_BASE` variable will be replaced by VS Code to the filename (*without the extension*) so we'll get a new class called `Customer`.

See all variables here: https://code.visualstudio.com/docs/editor/userdefinedsnippets#_variables

Settings User snippets are stored in the `%APPDATA%\code\user\snippets` folder, one `.json` file by languages.

Extensions Also see the snippet-creator extension.

Project-based You can also define your snippets for your project and not globally. You can then share snippets with your colleagues f.i.

1. In the root folder of your project create a `.vscode` folder if not yet present;
2. Create a file with the `.code-snippets` extension like `my-project.code-snippets`;
3. You can configure your snippets there.

Below an example from this `VSCode-Tips` project:

```
{
  "Terminal-shortcut": {
    "scope": "markdown",
    "prefix": "terminal",
    "body": [
      "`Terminal`
      ↩ (<kbd>CTRL</kbd>-<kbd>SHIFT</kbd>-<kbd>ù</kbd>)"
    ],
    "description": "Show Terminal keyboard Shortcuts"
  }
}
```

Now, each time I'll type `terminal` I can decide to immediately add the keyboard shortcuts; nice.

Generator If you prefer to use a generator, <https://snippet-generator.app/> can help you.

Video tutorial <https://www.youtube.com/watch?v=Jlqk9UxgKEc>

Using the built-in terminal

Press `CTRL-SHIFT-ù` to open the built-in terminal.

Press CTRL-SHIFT-C to open a DOS Prompt session where the current folder will be the opened project.

You can choose for a DOS terminal, Powershell or bash one. Just select the `Select Default Shell` option and VS Code will prompt which shell application should be used.

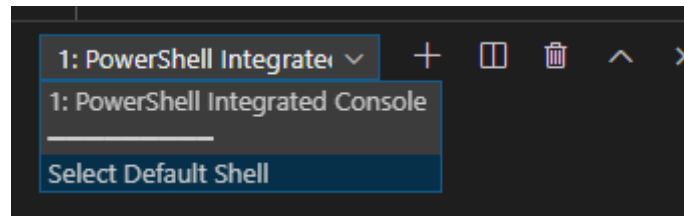


Figure 3: Terminal

Note: by clicking on the + button, we can create as many terminals we want.

Extensions Also see the Rerun last command extension.

Search

Save the search Just after having fired a search, it's possible to click on the `Open in editor` link so the result appears in the editor. The very nice thing then is to be able to see the context (i.e. xxx lines before/after the search match) and to be able to save the search as a file (the filename will be `.code-search`).

This can be useful when f.i. the search pattern is a complex regex or if working on the search result can take a long time so we can save the search as a file in the project and open it later on.

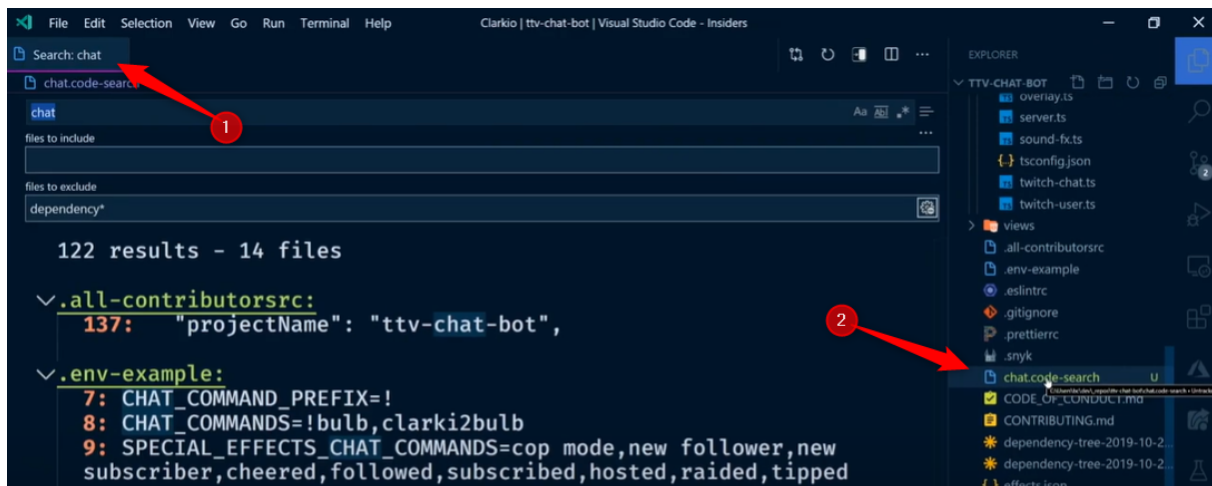


Figure 4: Save search

This video show this: <https://www.youtube.com/watch?v=zm7ZjENKUEc>

Working with code

Keyboard shortcuts

Code folding

- CTRL-K-CTRL-à to collapse all functions, headings, ... The same can be obtained by CTRL-SHIFT-P then Fold all.
- CTRL-K-CTRL-J to expand all functions, headings, ... The same can be obtained by CTRL-SHIFT-P then Unfold all.
- CTRL-K-CTRL-& - Collapse level 1.
- CTRL-K-CTRL-é - Collapse level 2.
- CTRL-K-CTRL-” - Collapse level 3.
- CTRL-K-CTRL-' - Collapse level 4.
- CTRL-K-CTRL-(- Collapse level 5.
- CTRL-K-CTRL-\$ - Collapse level 6.
- CTRL-K-CTRL-è - Collapse level 7.
- CTRL-K-CTRL-) - Collapse the bloc where the cursor is located.

Comments Select a bloc of lines then,

- CTRL-K-CTRL-C to put that bloc in comments (add // in front of each line).
- CTRL-K-CTRL-U to remove the comments (remove // in front of each line).

Integrated GIT

Using with the terminal Once a project has been opened, you can get access to any git command by using the terminal. Open the Terminal (CTRL-SHIFT-ù).

For instance, if the project has never been sent to git, you can initialize the repository like this:

- Open the Terminal (CTRL-SHIFT-ù). You'll be placed in the root folder of the project;
- `git init` to initialize your local repository;
- `git add .` to add all files to the staging area of your repository. Visual Studio Code will change the color of any files and folders in the tree-view to reflect that files are now in the staging area and ready to be committed;
- `git commit -m 'My initial commit'` to commit everything to the local repository. Color in the tree-view will be back in white (no greener) meaning that files are now placed in the repo.;

If you now change a file by appending a new function, removing part of the code, changing a variable, ... VSCode will display a color (green, red, ...) in the left margin of the code editor reflecting the change.

- `git status` in the Terminal will show the list of uncommitted changes.

Note: You can get access to any GIT command in the Command Palette (CTRL-SHIFT-P).

Source Control - GIT

- CTRL-SHIFT-G will open the Source Control: GIT pane with the list of changes made to your source after the last `git add` command.

From there it's easy to see what files have been modified and, file by file, discard changes if you want.

Extensions Also see the Gitlens extension.

Code definition

By right-clicking on a variable, constant or method, we can get access to a few features like showing everywhere it's used.

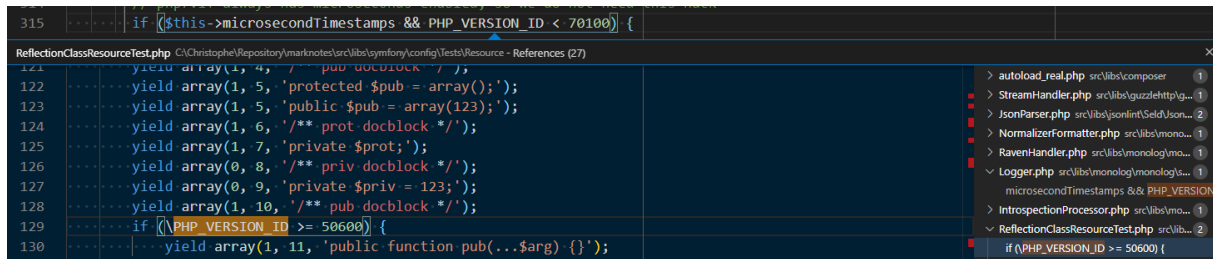


Figure 5: Go to References

- SHIFT-F12 to show all references

In the right area of the popup, we can navigate and click to see each portion of files.

Press F12 to jump in the definition of the method: click or select a method name like in `writeLog(...)` and press F12 once the cursor is located on `writeLog` and you'll jump in the file that implements the method, the method source code will be displayed too.

PHP-Unit

From the Terminal (CTRL-SHIFT-ù), you can directly run `phpunit` and the full suite of tests will be executed.

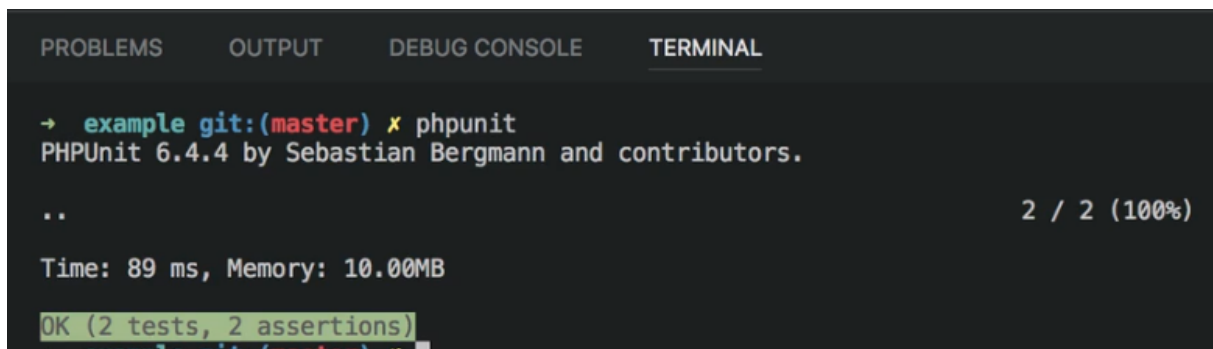


Figure 6: PHP-Unit

Extensions Also see the Better PHPUnit extension.

Using xDebug

URL: <https://xdebug.org>

Tutorial

Installation

- Open <http://localhost/?phpinfo=1> and make CTRL-A/CTRL-C
- Open <https://xdebug.org/wizard.php> and CTRLV there
- Click on the Analyze my phpinfo() output button

Follow installation steps:

- download the suggested .dll,
- save the file in the mentioned location,
- edit `php.ini` and add the reference to the .dll
- **extra step:** add the two lines below in the `php.ini` in the `[xdebug]` section

```
xdebug.remote_enable = 1
xdebug.remote_autostart = 1
```


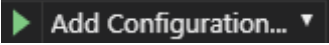
- save the file
- restart the webserver

The two variables below are important and should be initialized to 1 otherwise xdebug will not stop the code on breakpoints.

Check if xdebug is loaded

- Open <http://localhost/?phpinfo=1> once more
- Search for xdebug. A **xdebug** table should be there with a lot of keys/values.

Visual Studio Code Tutorial by Microsoft: <https://code.visualstudio.com/docs/editor/debugging>
configurations

- Install the PHP Debug extension
- Click on the bug button 
- Click on the  dropdown
- Select Add configuration and select PHP as language
- A list of options can be configured, the list is here: <https://marketplace.visualstudio.com/items?itemName=xdebug/debug#supported-launch.json-settings>

Settings Settings are saved in the `/.vscode/launch.json` file. A nice option is the `ignore` one who makes possible to ignore certain files from a debugger perspective. Here, ask not to go into vendors scripts (and stay in our own scripts):

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Listen for XDebug",
      "type": "php",
      "request": "launch",
      "port": 9000,
      "ignore": ["**/bin/**/*", "**/vendor/**/*"],
      "internalConsoleOptions": "openOnSessionStart"
    }
  ]
}
```

Notes:

- `ignore` is used to indicate to vscode to not stop on any error, exceptions, ... that can be retrieved in specific files or folders. If you're using dependencies, it's a good idea to not stop in files under the `vendor/` folder. Also, if you're using `.phar` archives for instance (let's say in folder `bin/`, it's also a good idea to ignore that folder / files),
- `internalConsoleOptions` will allow to immediately open the debug console in a new terminal, pretty easy.

The list of all settings can be retrieved here: https://code.visualstudio.com/docs/editor/debugging#_launchjson-attributes.

Use inside vscode Open your PHP file and add breakpoints. To do so, click on the line in your code where you want the browser to need to stop and click on the Debug menu then click on Toggle breakpoint. A red bullet will appear near the line number:

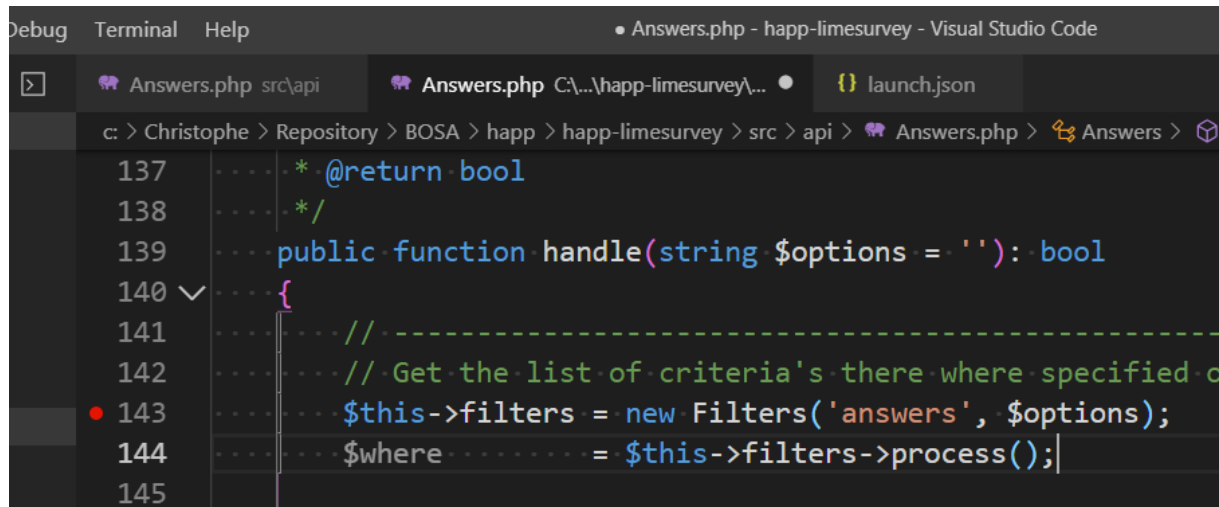


Figure 7: Red bullet

Then, once your breakpoints are in place, click on the Debug menu and click then on Start Debugging. You'll see an orange panel like the following one:

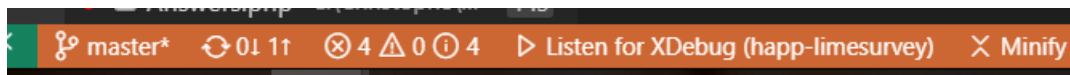


Figure 8: xDebug is running

You'll also have a new toolbar:



Figure 9: xDebug toolbar

Now, go back to your web browser and refresh your URL without any change; if your breakpoints are correctly initialized, the browser will be on *pause* and Visual Studio Code will activate the first met breakpoint.

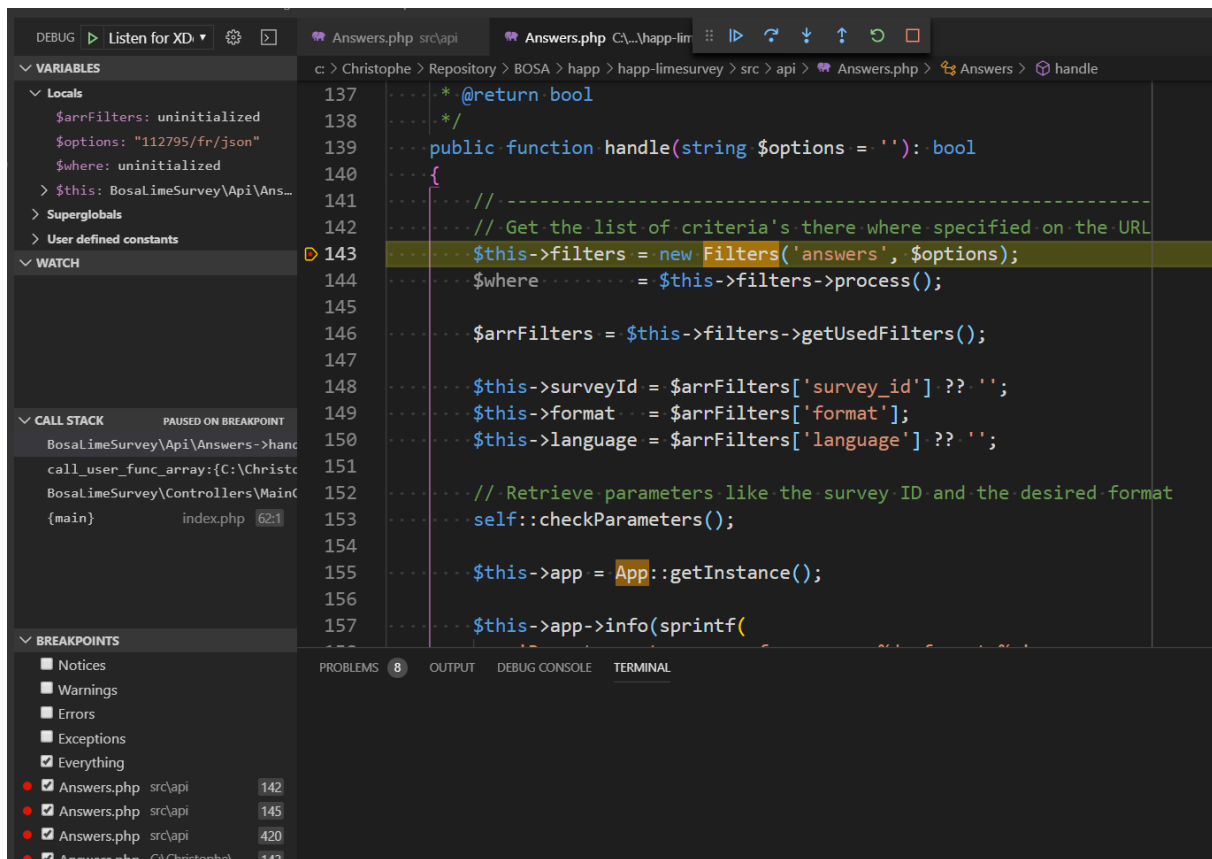


Figure 10: xDebug has stopped the execution

Use F10 to go to the next instruction or F11 to set into the next instruction (if the instruction is a function, go inside the function).

Breakpoint By adding a breakpoint, it's possible to edit it and set a condition.

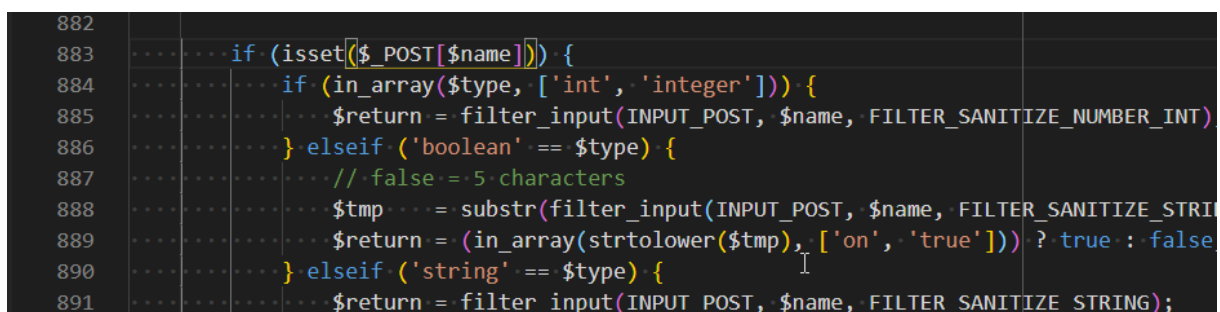


Figure 11: Conditional breakpoint

Logpoint Instead of writing here and there `echo '...';` or `console.log(...);` statements, VS Code can do it for us:

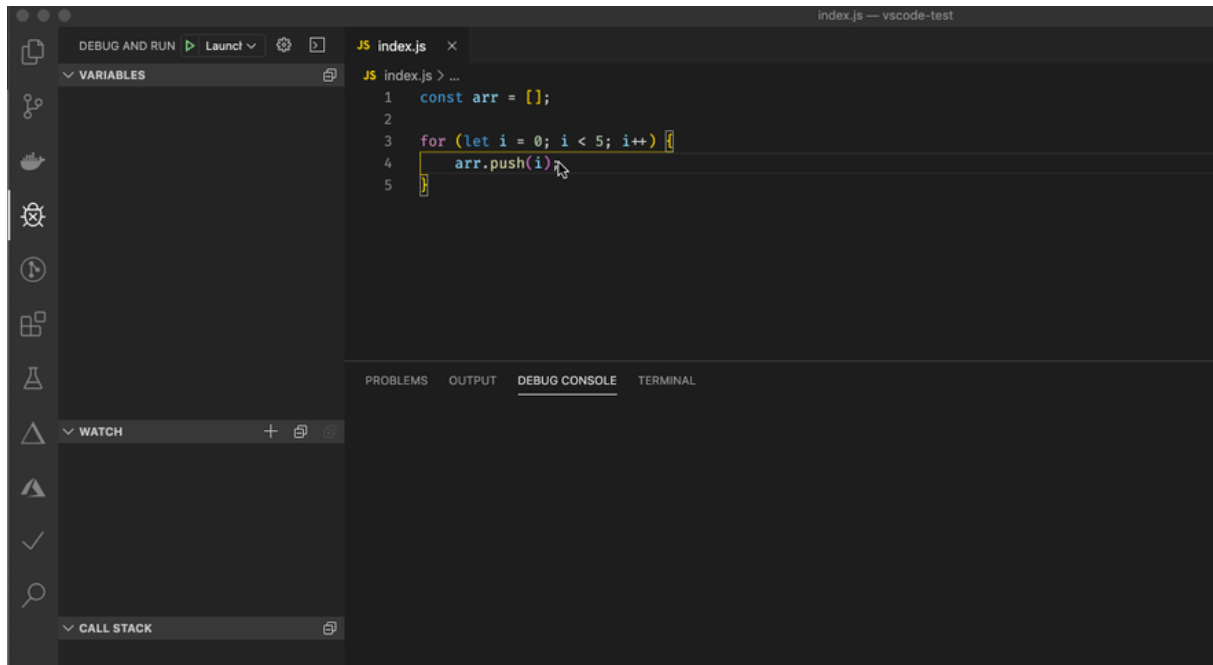


Figure 12: Add a logpoint

Debug keyboard shortcuts

- Show the Run pane (that pane contains all your variables so you can see, at runtime, their values): CTRL-SHIFT-D
- Toggle breakpoint: F9
- Step Info: F11
- Start / Continue: F5

Also read You can find more information on this repository, from Microsoft: <https://github.com/Microsoft/vscode-recipes/tree/master/PHP>

PHP Formatting

If you already have installed PHP intelephense, you can retrieve a `Format Document` command in the Command Palette (CTRL-SHIFT-P) but you can't really configure the rules (how the formatting should be done, tabs or spaces, removed unused classes, ...) and for this reason it's probably best to install PHP-CS-FIXER[#extensions-php-cs-fixer).

Once installed, php-cs-fixer will add command like `Fix this file` (ALT-SHIFT-F).

php-cs-fixer can be configured globally (in the User Settings (CTRL-,)) or for the Workspace (in this case, create or edit the `/.vscode/settings.json` file in your project).

```
{
  "php-cs-fixer.executablePath": "php-cs-fixer",
  "php-cs-fixer.executablePathWindows": "",
  "php-cs-fixer.onsave": false,
  "php-cs-fixer.rules": "@PSR2",
  "php-cs-fixer.config": ".php_cs;.php_cs.dist",
  "php-cs-fixer.allowRisky": false,
  "php-cs-fixer.pathMode": "override",
  "php-cs-fixer.exclude": [],
  "php-cs-fixer.autoFixByBracket": true,
  "php-cs-fixer.autoFixBySemicolon": false,
  "php-cs-fixer.formatHtml": false,
  "php-cs-fixer.documentFormattingProvider": true
}
```

By creating a `.php_cs` file in your root folder, php-cs-fixer will apply rules defined in that file.

Multiple cursors

VSCode supports multiple cursors: click here and there and there but just press the CTRL key after the first click.

You'll then have more than one cursor and starting type will do it in any places where a cursor was displayed.

For instance, you've a list of public functions, click before each `p` of public functions, press the delete key to remove the `public` word, it'll be done for all functions at a time and type now `private`.

You can also have an enumeration list, one word on each line. Click before each letter and then type `*` to add a bullet before each item.

Really convenient.

The CTRL-D shortcut will select the next occurrence: double-click on, f.i., the `public` word to select the first occurrence. Press CTRL-D to select the second, the third, and so on. Press CTRL-D again and again to select all occurrences. Then type `private` f.i. to overwrite selections and replace by the new word.

It's not really like a Search and Replace `all` since here we can decide how many occurrences we wish to replace. It's more interactive.

Insert prefix on each line SHIFT-ALT-I allow to enable multiple cursors, a nice use case is to select a bloc of lines and add a bullet so transform lines to a bullet list.

Here is how to do:

1. Select a bloc a line
2. Press SHIFT-ALT-I to enable multiple cursors
3. Press Home to put cursors at the beginning of each lines,
4. Press * followed by a space to transform the list of lines to a bullet list.

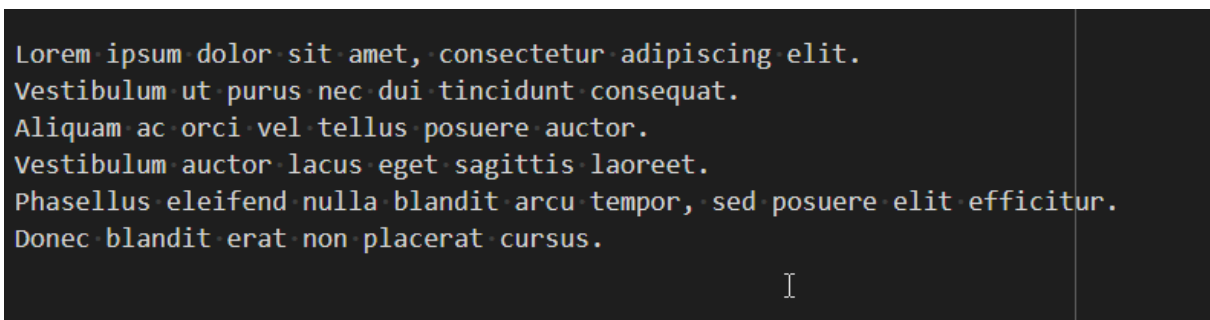


Figure 13: Make a bullet list

Navigate between problems

The list of problems detected by VS Code is displayed in a tab at the bottom of the screen, below the main editor.

You can see the list of problems and navigate between each of them by just pressing F8.

regions

Regions are a very smart features to make code editing more readable.

Consider the following function:

```
<?php
```

```
function addRows(array $data): array
{
    $result = [];

    //region 1. Add row only once for the same surveyId/sessionId
    foreach ($data as $participant) {
```

```
        // A few dozen lines of code...
        // A few dozen lines of code...
        // A few dozen lines of code...
    }

    //endregion

    //region 2. Add new participants
    try {
        // A few dozen lines of code...
        // A few dozen lines of code...
        // A few dozen lines of code...
    } catch (\Exception $e) {
        // Handle the exception
    }

    //endregion

    //region 3. Prepare resulting array
    // A few dozen lines of code...
    // A few dozen lines of code...
    // A few dozen lines of code...
    // A few dozen lines of code...
    //endregion

    return $result;
}
```

Your function is, logically, divided into three parts; the first to do checks, the second to do the job and the third to prepare the return data.

Using regions inside your function make the code's logic really easy to understand.

```
1  <?php
2
3  function addRows(array $data): array
4  {
5      $result = [];
6
7  >  ... #region 1. Add row only once for the same surveyId/sessionId ...
14
15 >  ... #region 2. Add new participants ...
24
25 >  ... #region 3. Prepare resulting array ...
31
32  ... return $result;
33  }
34
```

Figure 14: Working with regions

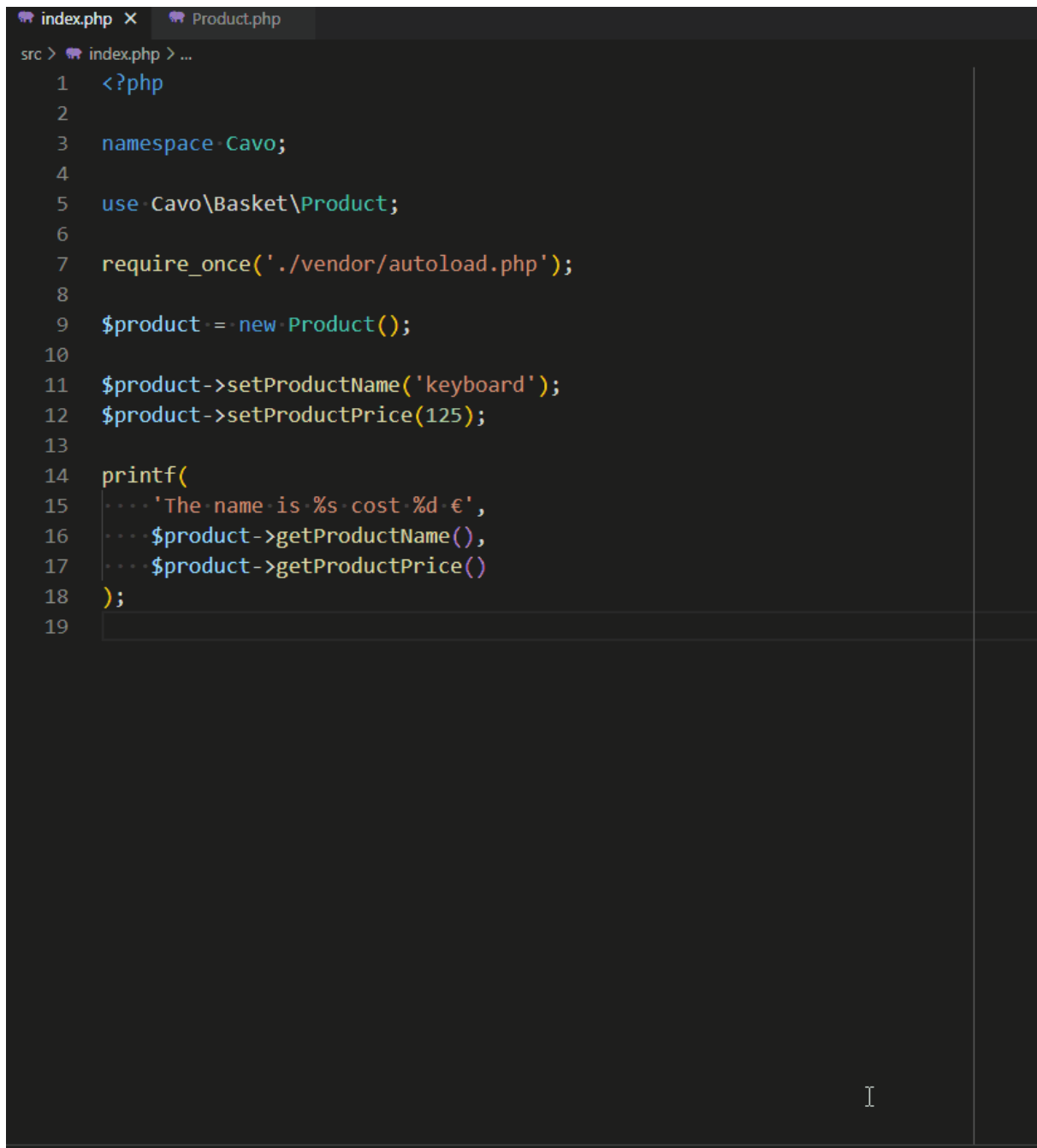
Renaming

<https://marketplace.visualstudio.com/items?itemName=st-pham.php-refactor-tool>

For PHP, for install the PHP Refactor Tool addon.

a symbol

1. Select a property,
2. Press F2 (or choose Rename Symbol in the Command palette (press CTRL-SHIFT-P))
3. Rename the property
4. Select Update Getter name and Setter name
5. Press Enter



```
src > index.php > ...
1  <?php
2
3  namespace Cavo;
4
5  use Cavo\Basket\Product;
6
7  require_once('./vendor/autoload.php');
8
9  $product = new Product();
10
11 $product->setProductName('keyboard');
12 $product->setProductPrice(125);
13
14 printf(
15     'The name is %s cost %d €',
16     $product->getProductName(),
17     $product->getProductPrice()
18 );
19
```

Figure 15: Renaming a symbol

Thanks to PHP Refactor Tool, all occurrences of the property will be renamed, the getter and the setter and everywhere these functions were used too. In all files of your project.

a class Anywhere in your code (in the class itself or where you use it), select the class name, press F2, rename it and press Enter.

The name will be changed everywhere, the name of the file will be changed too as you can see here below.

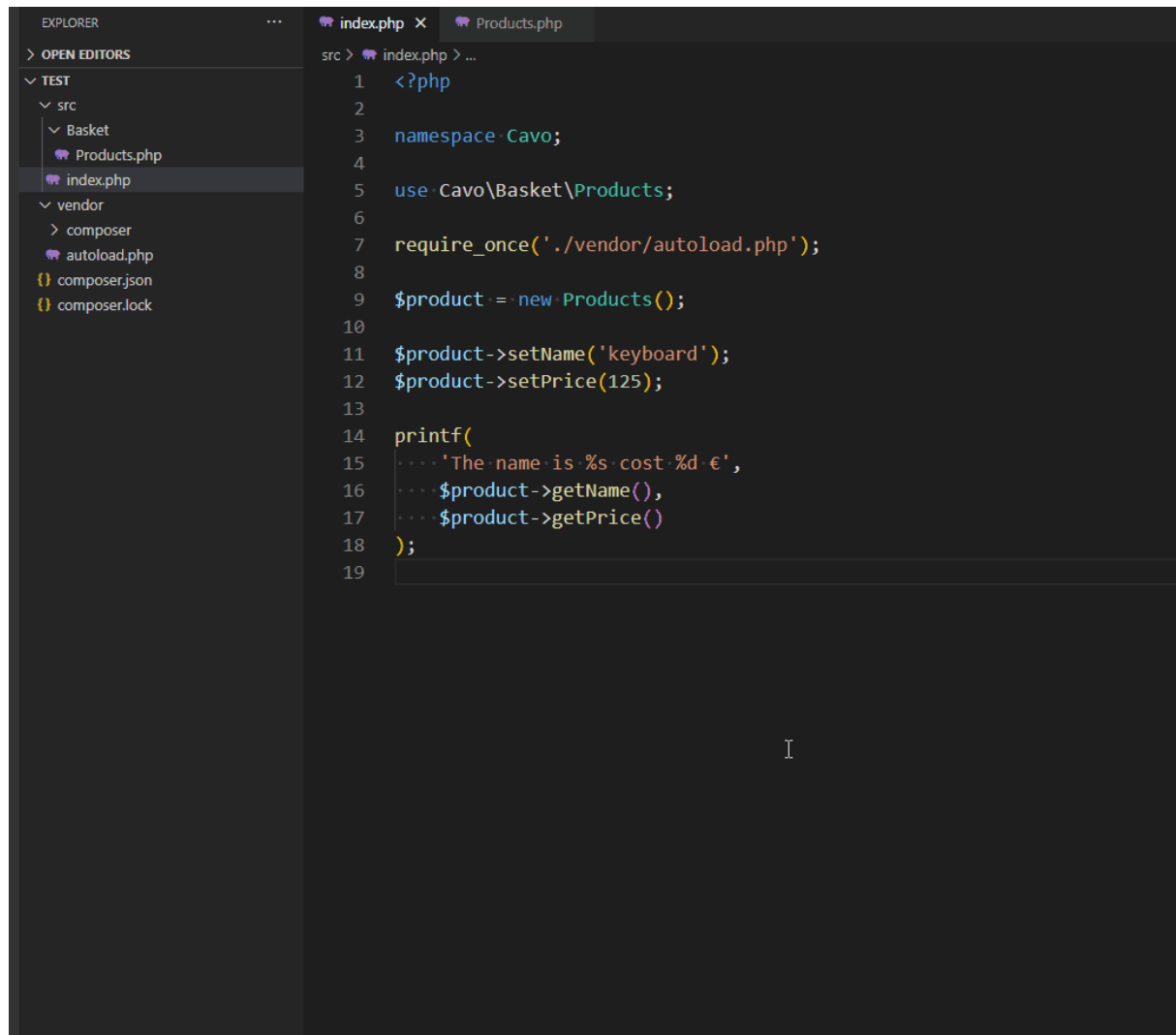


Figure 16: Renaming a class

Some settings

Editor settings

Adapts the size of the font used:

```
{  
  "editor.fontSize": 18  
}
```

Format the code during a copy/paste:

```
{  
  "editor.formatOnPaste": true  
}
```

Format the code when saving:

```
{  
  "editor.formatOnSave": true  
}
```

Indentation: tabs or spaces? true for spaces, false for tabs:

```
{  
  "editor.insertSpaces": false  
}
```

Stop displaying the list of open files (top left):

```
{  
  "explorer.openEditors.visible": 0  
}
```

Display a grayed out “dot” to make spaces visible:

```
{  
  "editor.renderWhitespace": "all"  
}
```

When using snippets, pressing the Tab key will activate the conversion from the shortcut to the content of the snippet:

```
{  
  "editor.tabCompletion": true  
}
```

Set the width of a tabulation:

```
{  
  "editor.tabSize": 3  
}
```

Files settings

Associating certain extensions with programming languages:

```
{
  "files.associations": {
    ".frm": "html",
    ".php-cs": "php"
  }
}
```

Specifies the default language for new :

```
{
  "files.defaultLanguage": "PHP"
}
```

Specifies the default format for CRLF (Windows) or LF (Unix) files:

```
{
  "files.eol": "\n"
}
```

If you don't want to see certain files / folders in your :

```
{
  "files.exclude": {
    "**/.DS_Store": true,
    "**/.git": true,
    "**/.hg": true,
    "**/.svn": true,
    "**/CVS": true,
    "**/libs/**": true
  }
}
```

Search settings

If you want the search feature to ignore certain files / folders :

```
{
  "search.exclude": {
    "**/bower_components": true,
    "**/node_modules": true,
    "**/libs/**": true
  }
}
```

```
}  
}
```

Telemetry settings

Disables telemetry to Microsoft:

```
{  
  "telemetry.enableCrashReporter": false,  
  "telemetry.enableTelemetry": false  
}
```

Window settings

When you open a file and then open another one without modifying the first one, the editor will open the second file in the same tab; the first file will then be closed.

If you want to open each time in a new tab, you have to adapt the parameter below:

```
{  
  "window.openFilesInNewWindow": "on"  
}
```

Changes the zoom level:

```
{  
  "window.zoomLevel": 5  
}
```

Workbench settings

Don't display the vertical menu bar on the left side which includes icons for files, search, Git, ...:

```
{  
  "workbench.activityBar.visible": false  
}
```

Stop displaying the status bar that appears at the bottom of the screen :

```
{  
  "workbench.statusBar.visible": false  
}
```

Extensions

Some extensions that will be useful for any PHP developer.

Get the list of all installed extensions {extensions-list-extensions}

You can use `vscode` on the command line with the `--list-extensions` option to retrieve the list of all installed extensions.

Open a DOS Prompt Session and run code `--list-extensions`. You'll get a list like below:

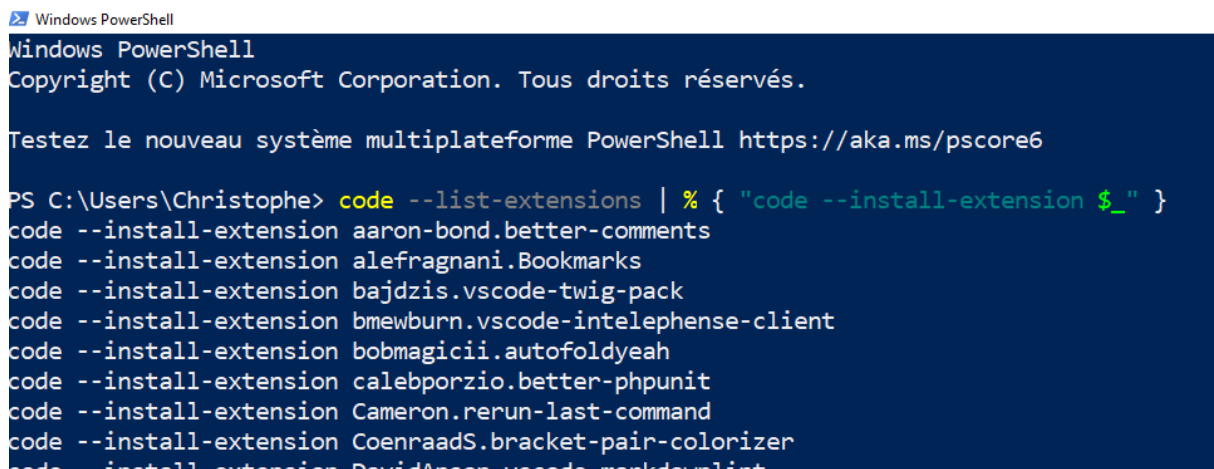
```
aaron-bond.better-comments
alefragnani.Bookmarks
bajdzis.vscode-twig-pack
bmewburn.vscode-intelephense-client
bobmagicii.autofoldyeah
calebporzio.better-phpunit
...
```

By running a Windows Powershell prompt, you can quickly generate a list of `vscode` instructions to install all these extensions. Start Windows Powershell and run the following command:

```
code --list-extensions | % { "code --install-extension $_" }
```

You'll get a list like below:

```
code --install-extension aaron-bond.better-comments
code --install-extension alefragnani.Bookmarks
code --install-extension bajdzis.vscode-twig-pack
code --install-extension bmewburn.vscode-intelephense-client
code --install-extension bobmagicii.autofoldyeah
code --install-extension calebporzio.better-phpunit
...
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

PS C:\Users\Christophe> code --list-extensions | % { "code --install-extension $_" }
code --install-extension aaron-bond.better-comments
code --install-extension alefragnani.Bookmarks
code --install-extension bajdzis.vscode-twig-pack
code --install-extension bmewburn.vscode-intelephense-client
code --install-extension bobmagicii.autofoldyeah
code --install-extension calebporzio.better-phpunit
code --install-extension Cameron.rerun-last-command
code --install-extension CoenraadS.bracket-pair-colorizer
code --install-extension DavidAnson.vscode-markdownlint
```

Figure 17: List of all installed extensions

Just copy/paste the full output and, f.i. send it by mail to someone. He'll just need to paste the list in a DOS Prompt Session to install them.

Bash - Shellscript

Shell-format

<https://marketplace.visualstudio.com/items?itemName=foxundermoon.shell-format>

Linter for .sh scripts.

Extend core features of Visual Studio Code

Active File In StatusBar

<https://marketplace.visualstudio.com/items?itemName=RoscoP.ActiveFileInStatusBar>

Displays the full name of the file being edited at the bottom of the screen and allows, for example, to copy/paste its name by clicking on it.

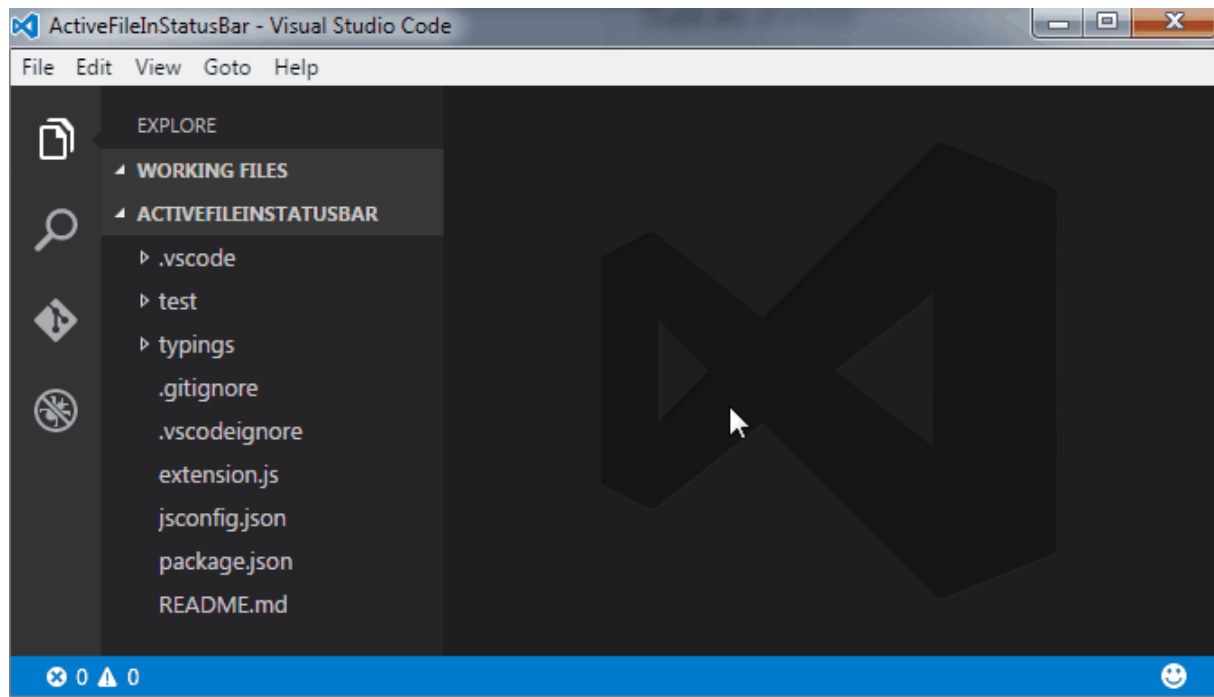


Figure 18: Active file in status bar

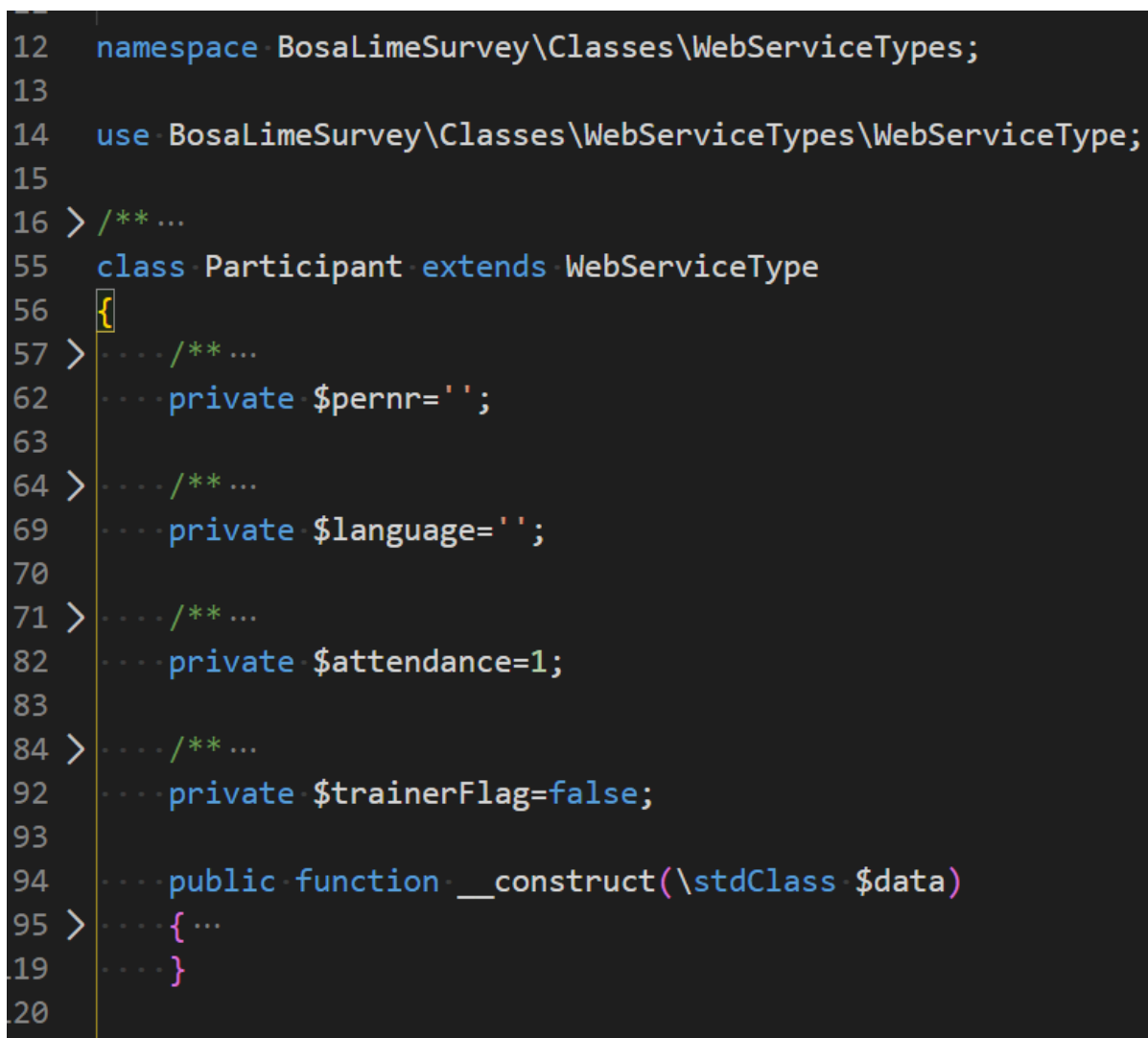
Autofold

<https://marketplace.visualstudio.com/items?itemName=bobmagicii.autofoldyeah>

By opening a file, the autofold extension can automatically collapse docblocks, functions, ...

The level (collapse all, from the second, third, ... level) can be configured for each file extension separately.

Here is a sample: by opening that source file, comments and function code is collapsing automatically, we've directly a better view on what that file is doing.



```
12 namespace BosaLimeSurvey\Classes\WebServiceTypes;
13
14 use BosaLimeSurvey\Classes\WebServiceTypes\WebServiceType;
15
16 > /** ...
55 class Participant extends WebServiceType
56 {
57 > ... /** ...
62 ... private $pernr='';
63
64 > ... /** ...
69 ... private $language='';
70
71 > ... /** ...
82 ... private $attendance=1;
83
84 > ... /** ...
92 ... private $trainerFlag=false;
93
94 ... public function __construct(\stdClass $data)
95 > ... { ...
19 ... }
20
```

Figure 19: Autofold

Better comments

<https://marketplace.visualstudio.com/items?itemName=aaron-bond.better-comments>

The Better Comments extension will help you create more human-friendly comments in your code.

```
2  export class MyClass {
3
4      /**
5       * MyMethod
6       * * Important information is highlighted
7       * ! Deprecated method, do not use
8       * ? Should this method be exposed in the public API?
9       * TODO: refactor this method so that it conforms to the API
10      * @param myParam The parameter for this method
11      */
12     public MyMethod(myParam: any): void {
13         let myVar: number = 123;
14
15         /** This is highlighted
16         if (myVar > 0) {
17             throw new TypeError(); //! This is an alert
18         }
19
20         /**? This is a query
21         let x = 1;
22
23         /// this.lineOfCode() == commentedOut;
24
25         //TODO: Create some test cases
26     }
27 }
```

Figure 20: Better comments

Bookmarks

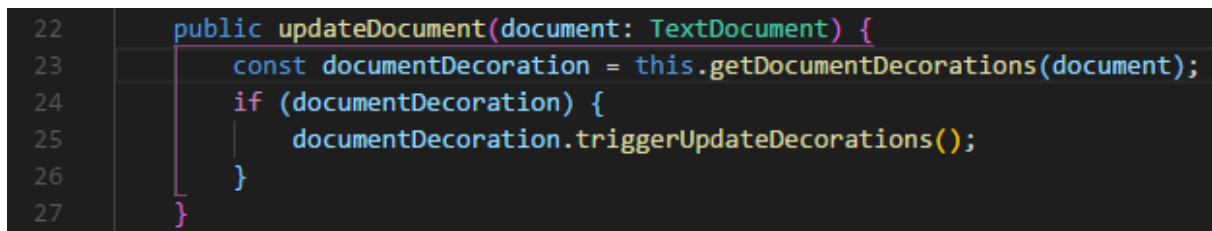
<https://marketplace.visualstudio.com/items?itemName=alefragnani.Bookmarks>

Allows you to put files in a bookmark folder, displayed in the icon bar on the left. This allows you to jump very quickly to a file, to a specific line (which would have been bookmarked).

Bracket Pair Colorizer

<https://marketplace.visualstudio.com/items?itemName=CoenraadS.bracket-pair-colorizer>

Uses different colors in the editor to properly identify brackets and opening / closing braces.



```
22 public updateDocument(document: TextDocument) {  
23     const documentDecoration = this.getDocumentDecorations(document);  
24     if (documentDecoration) {  
25         documentDecoration.triggerUpdateDecorations();  
26     }  
27 }
```

Figure 21: Bracket Pair Colorizer

Change case

<https://marketplace.visualstudio.com/items?itemName=wmaurer.change-case>

Easily change the case of variable names; like switching to pascal case, camelCase, CONSTANT, ...



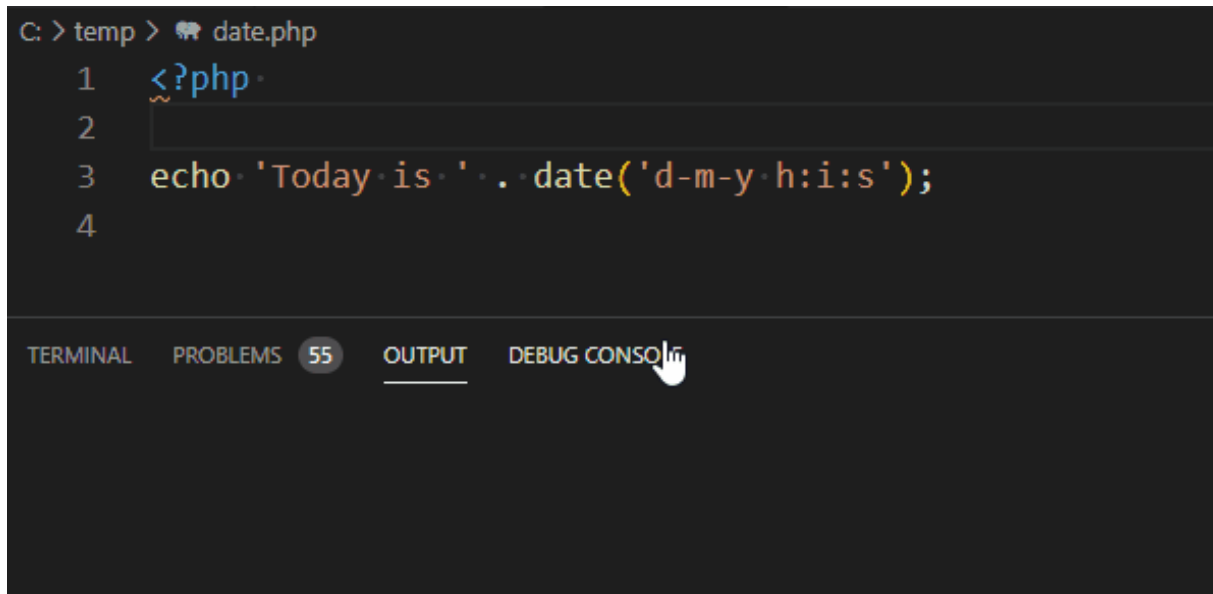
```
change-case.ts  
1 const fooBar = 0;  
2
```

Figure 22: Change case

Code runner

<https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner>

With Code runner, open a supported code source (like PHP or Python) and press CTRL-SHIFT-R to run that code and get the result in the console. You even don't need to save the file.



```
C: > temp > date.php
1 <?php
2
3 echo 'Today is ' . date('d-m-y h:i:s');
4
```

TERMINAL PROBLEMS 55 OUTPUT DEBUG CONSOLE

Figure 23: Code run

Code Spell Checker

<https://marketplace.visualstudio.com/items?itemName=streetsidesoftware.code-spell-checker>

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.

```
/**
 * Split camelCase words into an array of strings.
 */
export function splitCamelCaseWord(word: string): string[] {
  const separator = '_<^*_>_';
  const pass1 = XRegExp.replace(word, regExSplitWords, '$1' + separator + '$2');
  const pass2 = XRegExp.replace(pass1, regExSplitWords2, '$1' + separator + '$2');
  return XRegExp.split(pass2, separator);
}

const firstVallue = 25;

/**
 * Extract out whole words from a string of text.
 */
export function extractWordsFromText1(text: string): WordOffset[] {
  const words: WordOffset[] = [];

  const reg = XRegExp(regExWords);
  let match: RegExpExecArray;

  while ( match = reg.exec(text) ) {
    words.push({
      word: match[0],
      offset: match.index
    });
  }

  return words;
}
```

Figure 24: Code Spell Checker

EditorConfig

<https://github.com/editorconfig/editorconfig-vscode>

EditorConfig extension for Visual Studio Code <https://editorconfig.org/>

Favorites

<https://marketplace.visualstudio.com/items?itemName=kdcro101.favorites>

Allows you to group shortcuts to files that, for example, you often have to open.

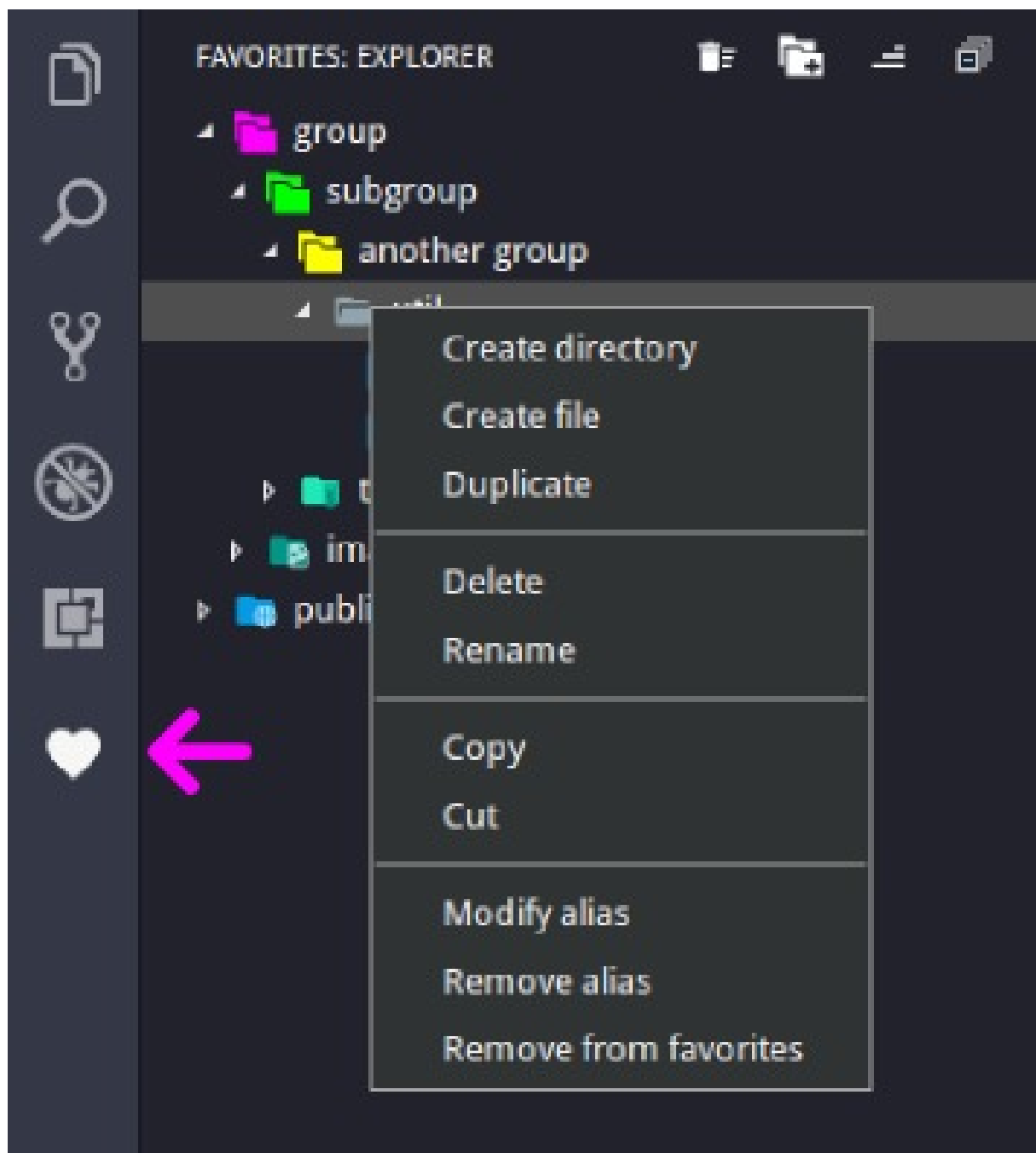


Figure 25: Favorites

git-project-manager

<https://github.com/felipecaputo/git-project-manager>

A Git Project Manager extension for vsCode.

Using GPM, it's now really easy to open a project: just press CTRL-P to open the Command Palette and select GPM: Open Git Project. Then you'll get the list of all folders on your disk with Git repositories.

First, you'll need to configure the `gitProjectManager.baseProjectsFolders` setting with the list of directories with your project; let's say `C:\Christophe\Repository`. You can mention several folders if you desire to.

You can also specify a root folder and playing with the `gitProjectManager.maxDepthRecursion` setting, you can define the recursion (*the higher the number, the slower the system will be to retrieve the list of projects*).

```
{
  "gitProjectManager.baseProjectsFolders": [
    "C:\\Christophe\\Repository"
  ],
  "gitProjectManager.storeRepositoriesBetweenSessions": true,
  "gitProjectManager.ignoredFolders": ["node_modules", "vendor"],
  "gitProjectManager.maxDepthRecursion": 2,
  "gitProjectManager.checkRemoteOrigin": false,
  "gitProjectManager.openInNewWindow": false
}
```

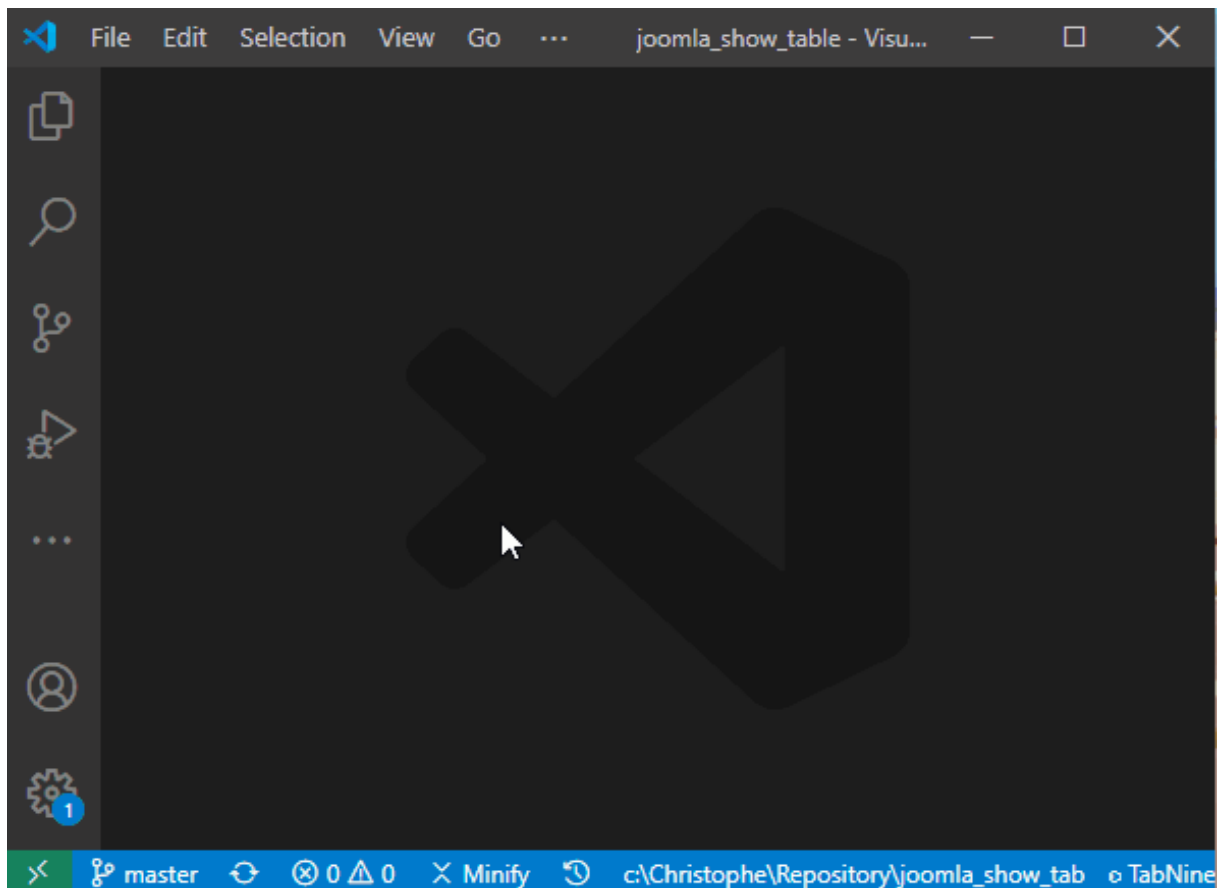
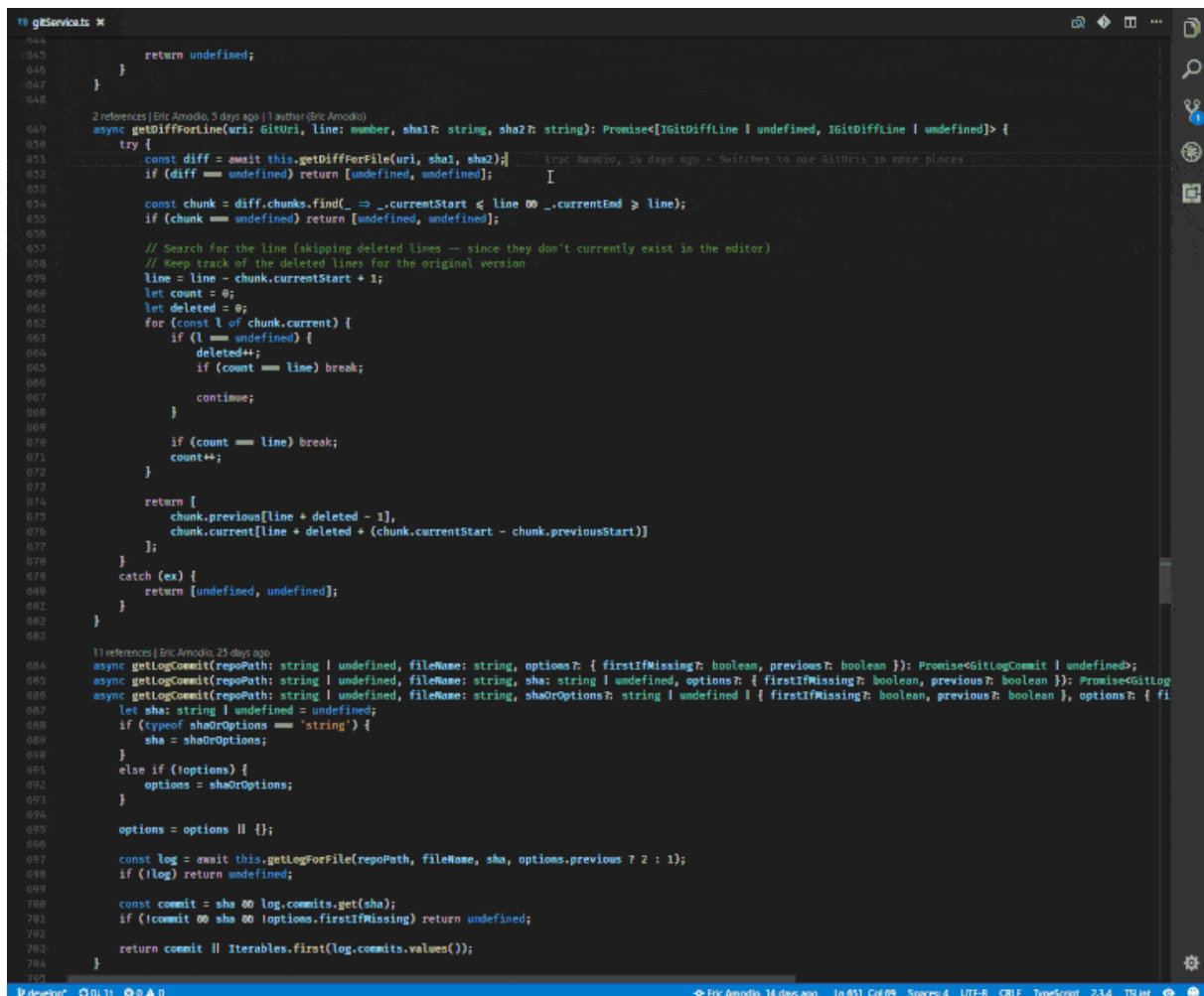


Figure 26: git-project-manager

Gitlens

<https://github.com/eamodio/vscode-gitlens>

Supercharge the Git capabilities built into Visual Studio Code — Visualize code authorship at a glance via Git blame annotations and code lens, seamlessly navigate and explore Git repositories, gain valuable insights via powerful comparison commands, and so much more



```
644 return undefined;
645 }
646 }
647 }
648 }
649
650 2 references | Eric Amodio, 5 days ago | 1 author (Eric Amodio)
651 async getDiffForLine(uri: GitUri, line: number, sha1: string, sha2?: string): Promise<[IGitDiffLine | undefined, IGitDiffLine | undefined]> {
652   try {
653     const diff = await this.getDiffForFile(uri, sha1, sha2);
654     if (diff === undefined) return [undefined, undefined];
655     const chunk = diff.chunks.find(_ => _._currentStart <= line && _._currentEnd >= line);
656     if (chunk === undefined) return [undefined, undefined];
657     // Search for the line (skipping deleted lines -- since they don't currently exist in the editor)
658     // Keep track of the deleted lines for the original version
659     line = line - chunk._currentStart + 1;
660     let count = 0;
661     let deleted = 0;
662     for (const l of chunk._current) {
663       if (l === undefined) {
664         deleted++;
665         if (count === line) break;
666       }
667       continue;
668     }
669     if (count === line) break;
670     count++;
671   }
672   return [
673     chunk._previous[line + deleted - 1],
674     chunk._current[line + deleted + (chunk._currentStart - chunk._previousStart)]
675   ];
676 }
677 catch (ex) {
678   return [undefined, undefined];
679 }
680 }
681 }
682 }
683 }
684
685 11 references | Eric Amodio, 25 days ago
686 async getLogCommit(repoPath: string | undefined, fileName: string, options?: { firstIfMissing?: boolean, previous?: boolean }): Promise<GitLogCommit | undefined>;
687 async getLogCommit(repoPath: string | undefined, fileName: string, sha: string | undefined, options?: { firstIfMissing?: boolean, previous?: boolean }): Promise<GitLogCommit | undefined>;
688 async getLogCommit(repoPath: string | undefined, fileName: string, shaOptions?: string | undefined | { firstIfMissing?: boolean, previous?: boolean }, options?: { firstIfMissing?: boolean, previous?: boolean }): Promise<GitLogCommit | undefined>;
689 if (typeof shaOptions === 'string') {
690   sha = shaOptions;
691 }
692 else if (options) {
693   options = shaOptions;
694 }
695 options = options || {};
696 const log = await this.getLogForFile(repoPath, fileName, sha, options.previous ? 2 : 1);
697 if (!log) return undefined;
698 const commit = sha && log.commits.get(sha);
699 if (!commit && sha && options.firstIfMissing) return undefined;
700 return commit || Iterables.first(log.commits.values());
701 }
702 }
```

Figure 27: GitLens

Vscode Google Translate

<https://marketplace.visualstudio.com/items?itemName=funkyremi.vscode-google-translate>

Quickly translate text right in your code.

```

Snippets are pieces of pre-programmed code (like a try...catch...) that allow
you to generate code without typing it entirely.

Depending on the open file and its language (php, js, markdown, ...), VSCode
will offer standard snippets and it will also be possible to program your
own.

Press <kbd>CTRL</kbd>-<kbd>SHIFT</kbd>-<kbd>P</kbd> to open the Palette Command
then type Insert Snippet to get the list of already existing snippets.

### User defined ones {#snippets-user-defined}

```

Figure 28: Translate

Configuration Google Translate By setting your preferred language, you'll not be prompted each time about the target language.

```

{
  "vscodeGoogleTranslate.preferredLanguage": "fr"
}

```

Also, it's best to assign keyboard shortcut like ALT-SHIFT-T to translate the text.

```

{
  "key": "shift+alt+t",
  "command": "extension.translateTextPreferred"
}

```

Coloring of source codes according to the language

Apache configuration file

<https://marketplace.visualstudio.com/items?itemName=mrmlnc.vscode-apache>

Support (coloring) Apache files such as `.htaccess`, `.htpasswd`, `.conf` and `.htgroups`.

```
1  <IfModule mod_rewrite.c>
2
3      <IfModule mod_negotiation.c>
4          Options -MultiViews
5      </IfModule>
6
7      RewriteEngine On
8
9      ##
10     ## You may need to uncomment the following line for some hosting environments,
11     ## if you have installed to a subdirectory, enter the name here also.
12     ##
13     # RewriteBase /
14
15     ##
16     ## Black list protected files
17     ##
18     RewriteRule ^themes/.*/(layouts|pages|partials)/.*.htm index.php [L,NC]
19     RewriteRule ^bootstrap/..* index.php [L,NC]
20     RewriteRule ^config/..* index.php [L,NC]
21     RewriteRule ^vendor/..* index.php [L,NC]
22     RewriteRule ^storage/cms/..* index.php [L,NC]
23     RewriteRule ^storage/logs/..* index.php [L,NC]
24     RewriteRule ^storage/framework/..* index.php [L,NC]
25     RewriteRule ^storage/temp/protected/..* index.php [L,NC]
26     RewriteRule ^storage/app/uploads/protected/..* index.php [L,NC]
27
28     ##
29     ## White listed folders and files
30     ##
31     RewriteCond %{REQUEST_FILENAME} -f
32     RewriteCond %{REQUEST_URI} !\.js$
33     RewriteCond %{REQUEST_URI} !\.map$
34     RewriteCond %{REQUEST_URI} !\.ico$
35     RewriteCond %{REQUEST_URI} !\.jpg$
36     RewriteCond %{REQUEST_URI} !\.jpeg$
37     RewriteCond %{REQUEST_URI} !\.bmp$
```

Figure 29: Apache coloring

Log File Highlighter

<https://marketplace.visualstudio.com/items?itemName=emilast.LogFileHighlighter>

Log file highlight.

```
10 2015-12-09 09:24:09,424 WARN EPIServer.Scheduler.WaitableTimer: 10s warning - due time is den 8 december 2015 15:24:58
11 2015-12-09 09:24:19,842 WARN EPIServer.Scheduler.WaitableTimer: 10s warning - due time is den 8 december 2015 16:12:59
12 2015-12-09 09:24:19,852 INFO Web.SiteMap.Framework.Jobs.SiteMapGeneratorJob: Starting SiteMapGeneratorJob on 'CLT-43587CM'
13 2015-12-09 09:27:25,114 WARN EPIServer.Core.ContentProvider: ContentLink=4656 464757, ReadAndCacheObject delegate...LoadContent returned null
14 2015-12-09 09:28:35,016 INFO Web.ServiceReferences.Booking.BookingFacade<GetConfirmedBooking>d_8: Attempted to get confirmed booking returned NotFound, bookingId=5000001
15 2015-12-09 09:28:35,025 ERROR Web.Controllers.Booking.BookingReceiptPageController<Email>d_17: [GET] Email failed. bookingId = 5000001, lastName = %c3%96z1l
16 2015-12-09 09:29:02,258 INFO Web.Helpers.HotelDataHelper<CreateViewModelFromHotel>d_8: CreateViewModelFromHotel retrieved hotel image from DAM in 200 ms
17 2015-12-09 09:29:02,263 INFO Web.Services.Episerver.HotelInformationPageProvider: GetHotelPageByDataPageId HotelPage pagetypeid = 91
18 2015-12-09 09:29:14,695 INFO Web.Helpers.HotelDataHelper<CreateViewModelFromHotel>d_8: CreateViewModelFromHotel retrieved hotel image from DAM in 169 ms
19 2015-12-09 09:29:14,696 INFO Web.Services.Episerver.HotelInformationPageProvider: GetHotelPageByDataPageId HotelPage pagetypeid = 91
20 2015-12-09 09:29:25,369 WARN EPIServer.Core.ContentProvider: ContentLink=67934, ReadAndCacheObject delegate...LoadContent returned null
21 2015-12-09 09:30:01,884 ERROR Common.ErrorHandling.ViewModelExtensions: SafeGet caught an exception in the expression <Execute>b_2_2.
22 System.NullReferenceException: Object reference not set to an instance of an object.
23 -- at ASP_Views_Pages_BookingRoomsAndRatesPage_Index_cshtml.<c.<Execute>b_2_2(BookingRoomsAndRatesPageViewModel m) in C:\Code\Web\obj\CodeGen\Views\Views\Pages\BookingR
24 -- at Common.ErrorHandling.ViewModelExtensions.SafeGet[TModel,TResult](TModel model, Func`2 func, TResult fallback) in C:\Code\Common\ErrorHandling\ViewModelExtensions.cs:
25 2015-12-09 09:30:01,921 INFO Web.Helpers.HotelDataHelper<CreateViewModelFromHotel>d_8: CreateViewModelFromHotel retrieved hotel image from DAM in 30 ms
26 2015-12-09 09:30:01,921 INFO Web.Services.Episerver.HotelInformationPageProvider: GetHotelPageByDataPageId HotelPage pagetypeid = 91
27 2015-12-09 09:30:06,211 INFO Web.Helpers.HotelDataHelper<CreateViewModelFromHotel>d_8: CreateViewModelFromHotel retrieved hotel image from DAM in 30 ms
28 2015-12-09 09:30:06,211 INFO Web.Services.Episerver.HotelInformationPageProvider: GetHotelPageByDataPageId HotelPage pagetypeid = 91
29 2015-12-09 09:30:06,690 INFO Web.Helpers.HotelDataHelper<CreateViewModelFromHotel>d_8: CreateViewModelFromHotel retrieved hotel image from DAM in 33 ms
30 2015-12-09 09:30:06,691 INFO Web.Services.Episerver.HotelInformationPageProvider: GetHotelPageByDataPageId HotelPage pagetypeid = 91
31 2015-12-09 09:30:07,370 INFO Web.Helpers.HotelDataHelper<CreateViewModelFromHotel>d_8: CreateViewModelFromHotel retrieved hotel image from DAM in 32 ms
```

Figure 30: logfile_highlight.png

Prettier

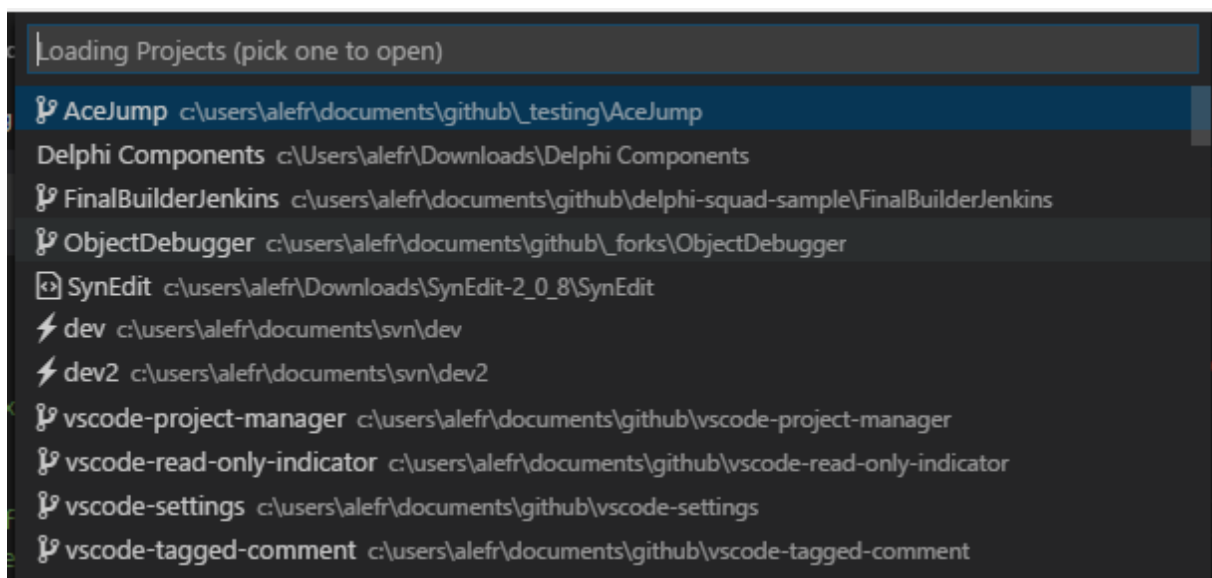
<https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode>

Prettier is an opinionated code formatter. It enforces a consistent style by parsing your code and re-printing it with its own rules that take the maximum line length into account, wrapping code when necessary.

Project Manager

<https://marketplace.visualstudio.com/items?itemName=alefragnani.project-manager>

Easily switch between projects.

**Figure 31:** Project Manager

Rerun the last command <https://marketplace.visualstudio.com/items?itemName=Cameron.rerun-last-command>

Quickly repeat the last command in your terminal without leaving the text editor.

Just press CTRL-F7 to rerun it.

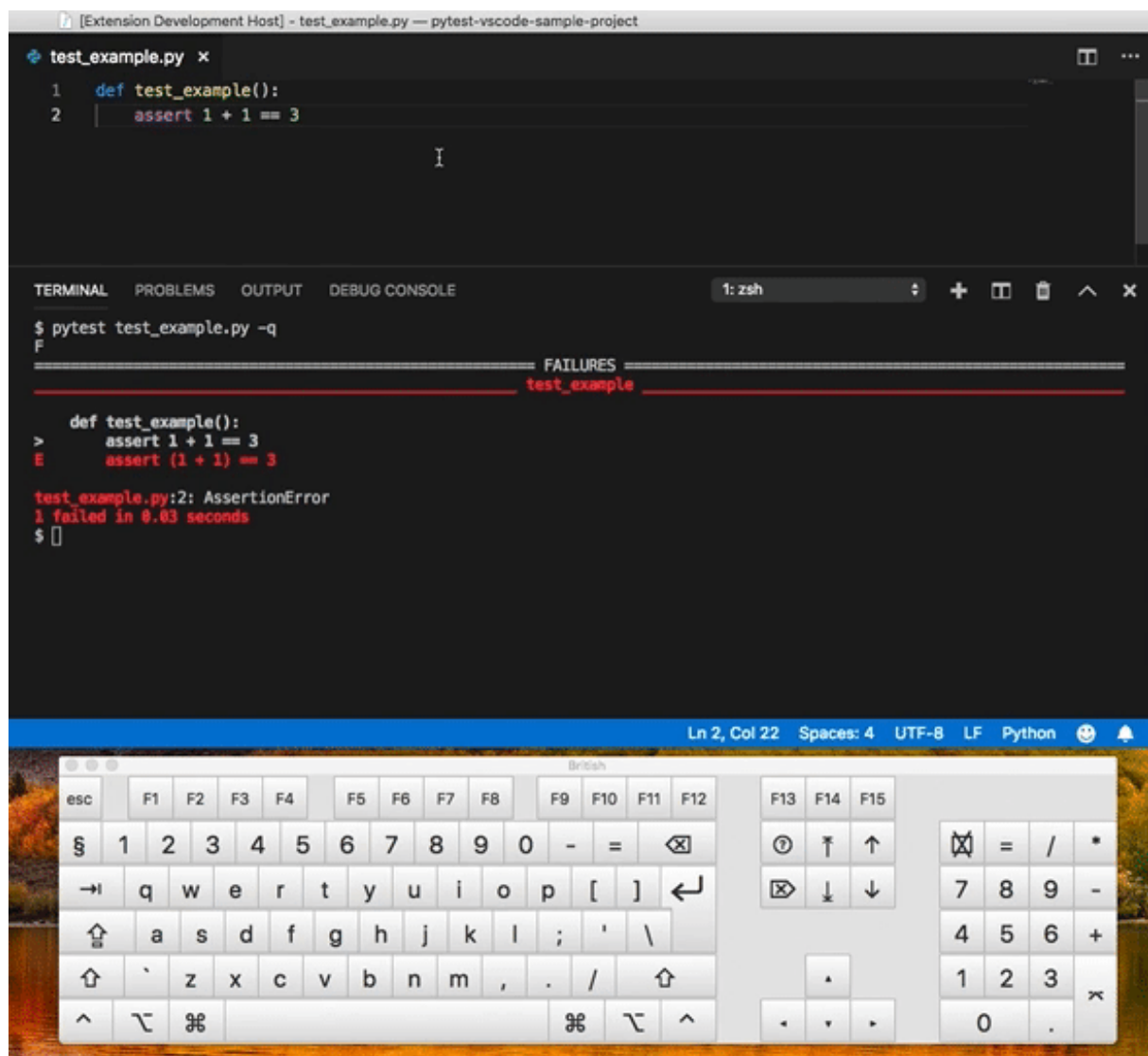


Figure 32: Rerun last command

REST client

<https://marketplace.visualstudio.com/items?itemName=humao.rest-client>

Make easy to debug JSON API without leaving the IDE.

Select highlight in minimap

<https://marketplace.visualstudio.com/items?itemName=mde.select-highlight-minimap>

Highlights the selected code (e.g. a function name) in the minimap so that you can quickly identify where in the currently edited file the same function is called.

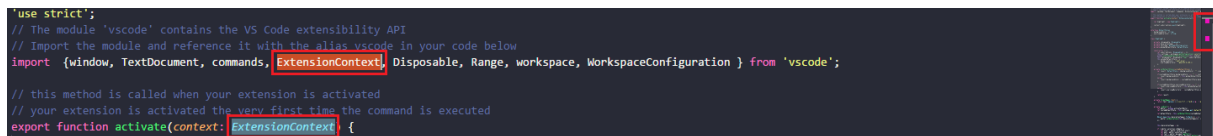


Figure 33: Select highlight in minimap

Snippet-creator The snippet-creator extension simplifies the creation of a snippet from existing code. Simply select the code already present in a file and then do a CTRL-SHIFT-P, type Create Snippet to call the wizard.

Note: the extension is actually deprecated (may 2020).

Sort lines

<https://marketplace.visualstudio.com/items?itemName=Tyriar.sort-lines>

Sorts lines of text.

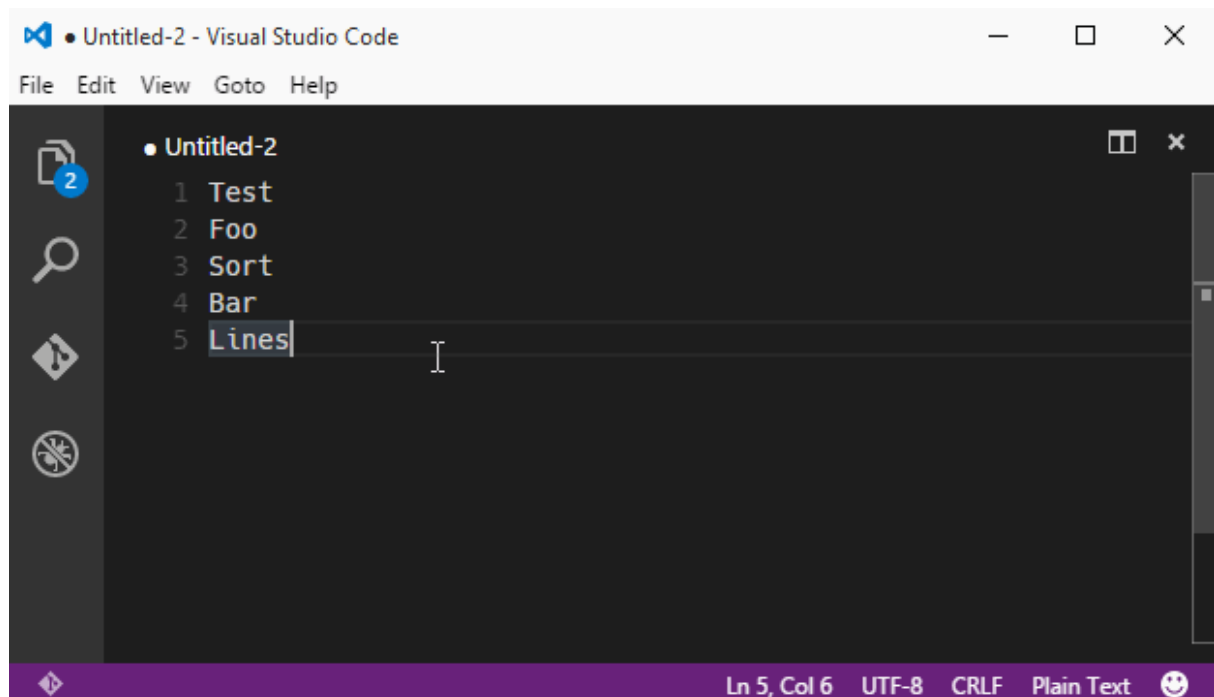


Figure 34: Sort lines

Start any shell

<https://github.com/remcoros/vscode-startanyshell>

Press CTRL-SHIFT-C to start a shell like DOS Prompt, Windows Powershell or Git Bash. The startup folder will be the root of your project.

So easy to run DOS batch, powershell or Bash scripts or any other commands (git, composer, ...).

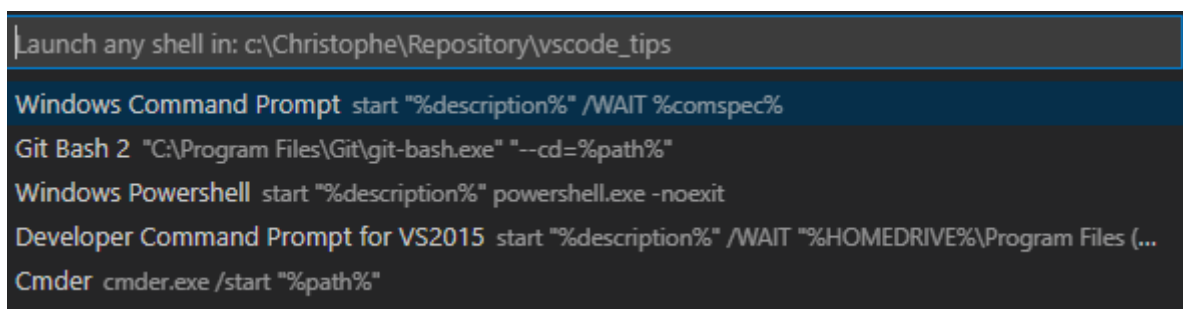


Figure 35: Start any shell

Surround

<https://marketplace.visualstudio.com/items?itemName=yatki.vscode-surround>

We select a block of lines of code and then, thanks to Surround, we can include it in an `if/else`, `try/catch`, ... The tool indents the code itself.

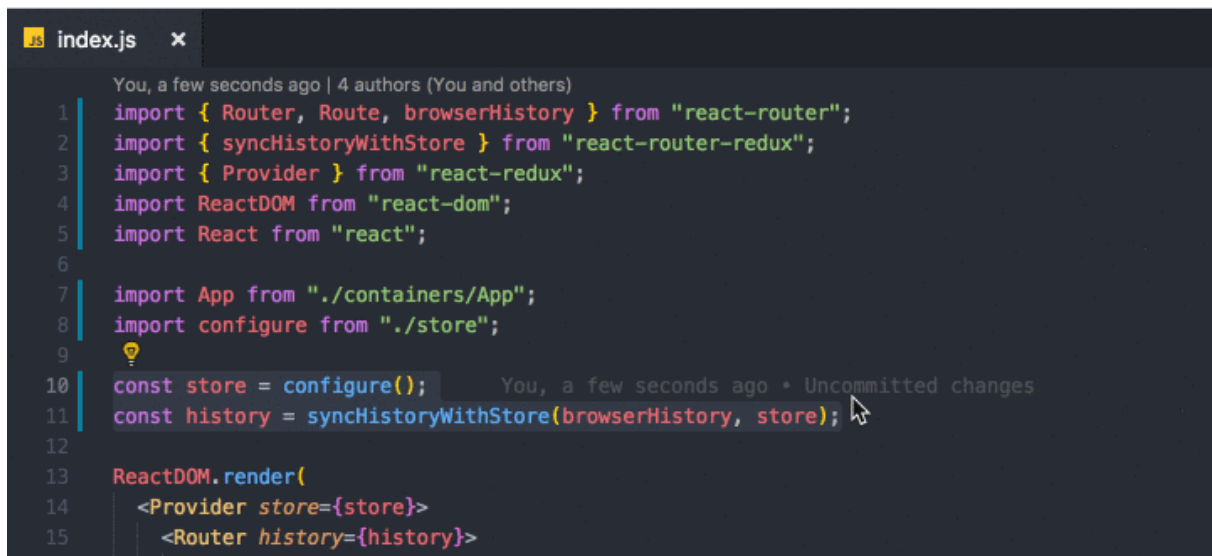


Figure 36: VS Code - Surround

Syncing Synchronization is now (August 2020) part of Visual Studio Code: <https://code.visualstudio.com/docs/editor/settings-sync>

TabNine AMAZING code completion extension.

TabNine is using AI to learn how you are using vscode (no matter the used language) and will predict the next words.

This addon is really a MUST HAVE: <https://marketplace.visualstudio.com/items?itemName=TabNine.tabnine-vscode>

Todo Tree

<https://marketplace.visualstudio.com/items?itemName=Gruntfuggly.todo-tree>

Displays an icon on the left side of the screen, in the form of a tree, which allows you to find, in one place, the list of TODOs that you have to make, i.e. comments beginning with `// TODO` that have been encoded in the source files.

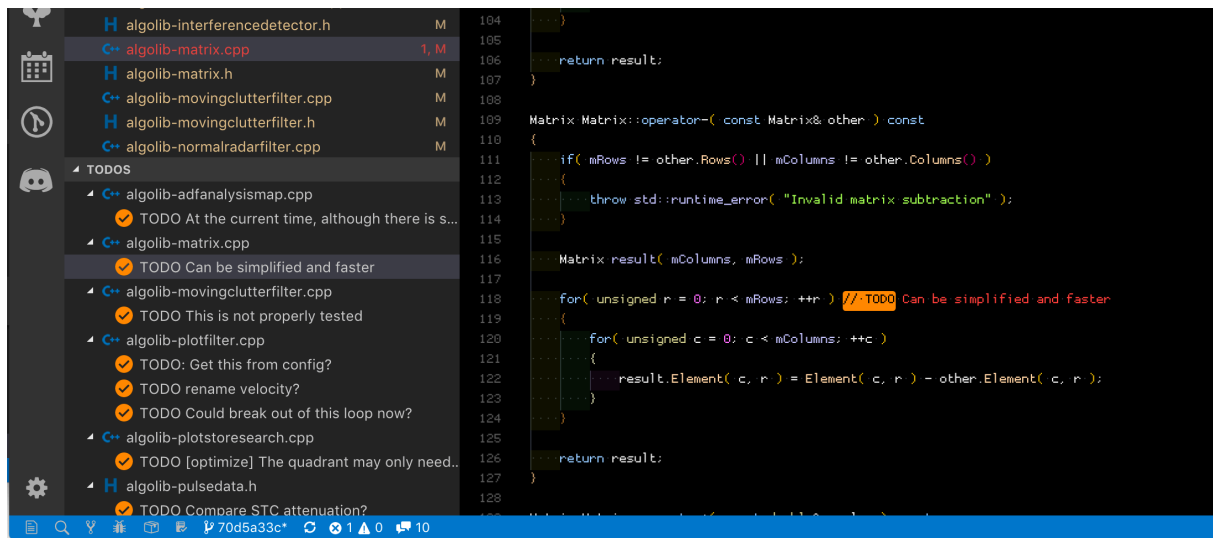


Figure 37: Todo Tree

vscode-icons

<https://marketplace.visualstudio.com/items?itemName=robertohuertasm.vscode-icons>

Adapt the tree-view with the list of files to use an icon associated to the type of file (css, html, php, ...)

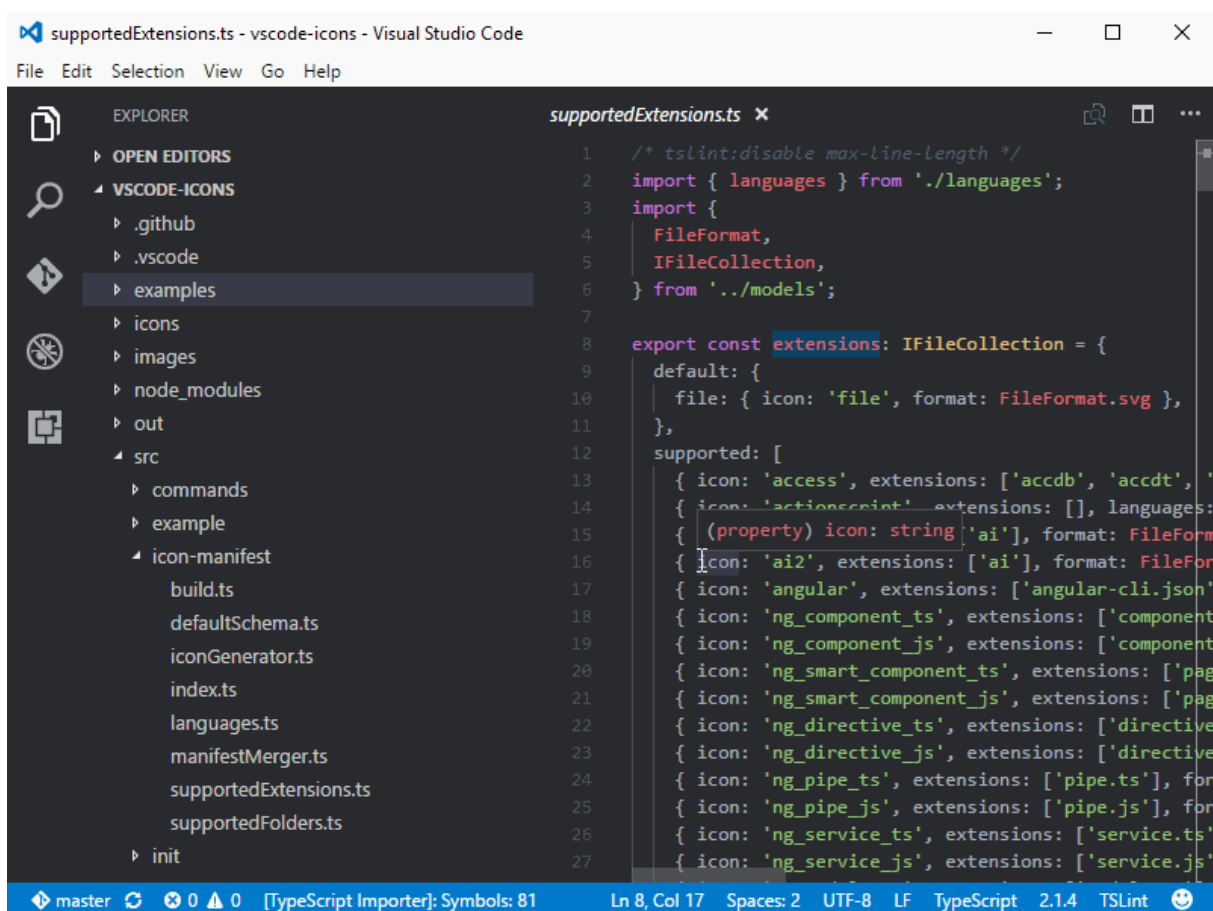


Figure 38: VS Code Icons

Javascript

ESLint

<https://marketplace.visualstudio.com/items?itemName=dbaumeier.vscode-eslint>

Javascript linter

Markdown

Emoji

<https://marketplace.visualstudio.com/items?itemName=perkovec.emoji>

Insert an emoji without leaving the editor. The

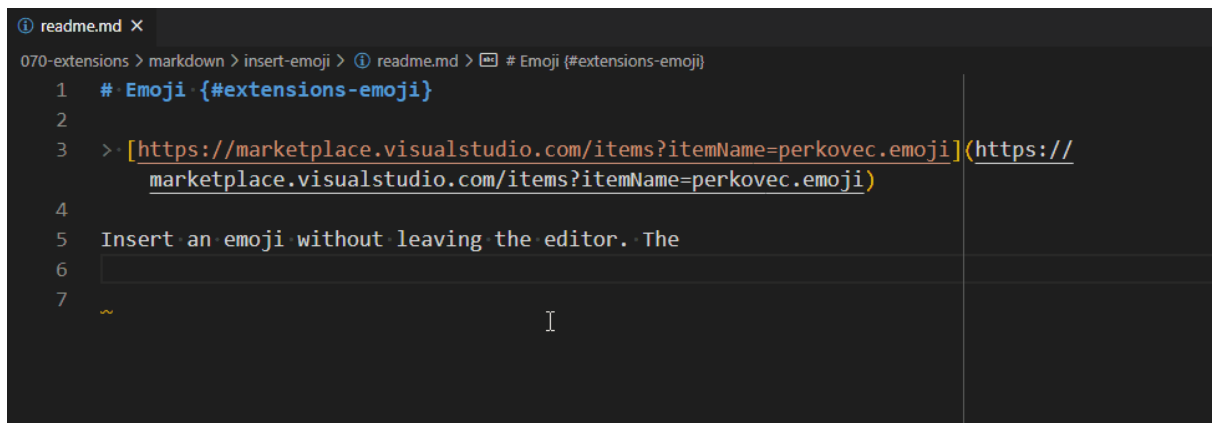


Figure 39: Insert emoji

Markdown All in One

<https://marketplace.visualstudio.com/items?itemName=yzhang.markdown-all-in-one>

Implements keyboard shortcuts for the Markdown language allows the generation of tables of contents, preview, ...

Markdownlint

- <https://github.com/DavidAnson/vscode-markdownlint>

Configuration Markdownlint You can configure markdownlint to ignore some files or some rules.

The list of rules can be retrieved from <https://github.com/DavidAnson/markdownlint/blob/master/doc/Rules.md>.

To ignore f.i. rules MD033 and MD041, just add the following configuration in your `settings.json` file:

```
{
  "markdownlint.config": {
    "MD033": false,
    "MD041": false
  }
}
```

To ignore files use the `ignore` key:

```
{  
  "markdownlint.config": {  
    "ignore": ["!**/History.md"]  
  }  
}
```

Set your custom rules:

```
{  
  "markdownlint.customRules":  
    ↪ ["C:\\\\Christophe\\\\.config\\\\.markdownlint.json"]  
}
```

PHP

Better PHPUnit

<https://github.com/calebporzio/better-phpunit>

Better PHPUnit is the most popular, cleanest, and fastest PHPUnit runner for VS Code.



Figure 40: Better PHPUnit

Configure Make sure to edit the Workspace settings and add these two keys:

```
{ "better-phpunit.phpunitBinary": "c:\\your_project\\vendor\\bin\\phpunit.bat", "better-phpunit.xmlConfigFilepath":  
  "c:\\your_project\\phpunit.xml" }
```

Should be the absolute path to the `phpunit.bat` file and your `phpunit.xml` configuration file.

Run a test method

- Place your cursor in/on the method you want to run
- CTRL-SHIFT-P to open the Command Palette
- Select: Better PHPUnit: run (CTRL-K-CTRL-R)

Run a test file

- CTRL-SHIFT-P to open the Command Palette
- Select: Better PHPUnit: run-file (CTRL-K-CTRL-F)

Run the entire suite

- CTRL-SHIFT-P to open the Command Palette
- Select: Better PHPUnit: run suite

Run the previous test

- CTRL-SHIFT-P to open the Command Palette
- Select: Better PHPUnit: run previous (CTRL-K-CTRL-P)

Code-runner

<https://github.com/formulahendry/vscode-code-runner>

With code-runner, open a PHP file and just run it from within vscode. Useful to run samples scripts f.i.

Configuration Code-runner Be sure to enable the `fileDirectoryAsCwd` setting. This will force code-runner to run the script by first setting the current directory to the one of the PHP file. Needed to make sure your require (like your autoloader) statements can retrieve files.

```
{ "code-runner.clearPreviousOutput": true, "code-runner.defaultLanguage": "php", "code-  
runner.enableAppInsights": false, "code-runner.fileDirectoryAsCwd": true }
```

PHP intelephense

<https://github.com/bmewburn/vscode-intelephense>

Configuration PHP intelephense After installation, we need to disable the built-in PHP feature of VSCode.

1. Go to the extensions tab (CTRL-SHIFT-X)
2. Search for `@builtin php`
3. Select PHP Language Features
4. Disable it.

Laravel Blade Snippets

<https://marketplace.visualstudio.com/items?itemName=onecentlin.laravel-blade>

Helper for working with Laravel Blade templates.

PHP-CS-FIXER

<https://marketplace.visualstudio.com/items?itemName=junstyle.php-cs-fixer>

Automatic correction of the quality of PHP code formatting thanks to quality standards (e.g. PSR2), which can be overloaded through a configuration file.

Once installed, go to the Command Palette (CTRL-SHIFT-P) and select `php-cs-fixer: fix this file` to correctly format the file.

Note: you can add PHP-CS-FIXER as a global dependency (thus for all your projects) by running `composer require friendsofphp/php-cs-fixer global` on the command line.

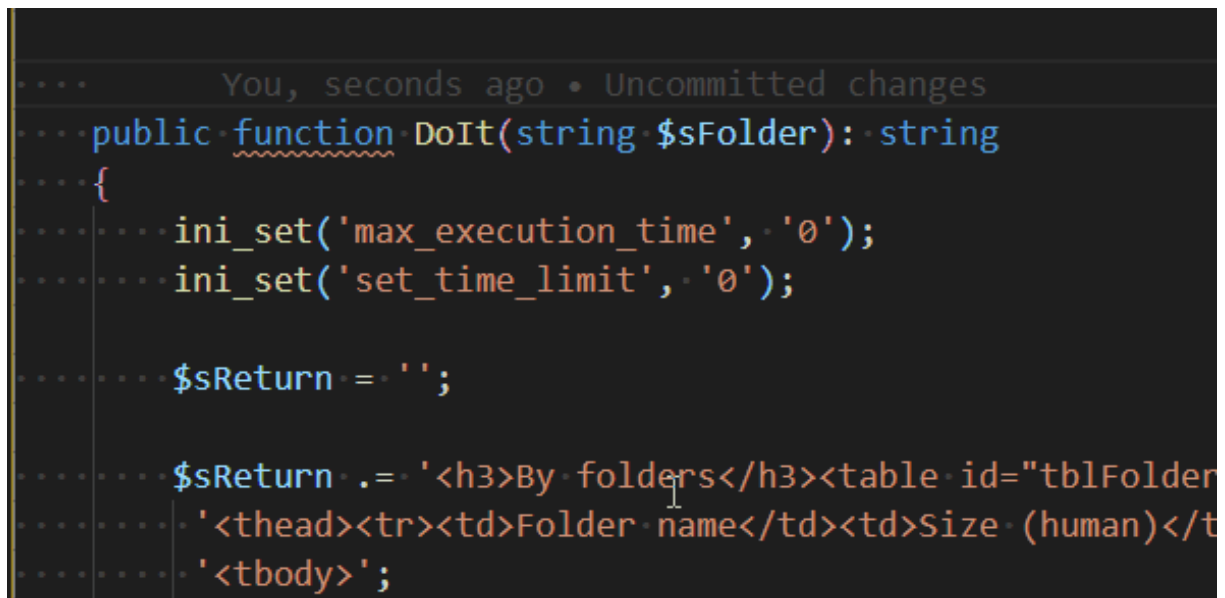
php-cs-fixer is using a `.php-cs` file for his configuration; there are a lot of items that can be configured. See <https://github.com/FriendsOfPHP/PHP-CS-Fixer> for more information's.

PHP DocBlocker

<https://marketplace.visualstudio.com/items?itemName=neilbrayfield.php-docblocker>

Allows generating documentation blocks of classes, methods, ...

Essential extension for any PHP programmer because it allows generating docblocks to document methods, properties, constants, ... in a PHP source code.

A screenshot of the Visual Studio Code editor interface. At the top, a status bar indicates 'You, seconds ago • Uncommitted changes'. The main editor area shows a PHP code snippet with a function definition. The function signature is 'public function DoIt(string \$sFolder): string'. The function body contains two lines of code: 'ini_set('max_execution_time', '0');' and 'ini_set('set_time_limit', '0');'. Below these, there is a line '\$sReturn = '';' and then a line '\$sReturn .= '<h3>By folders</h3><table id="tblFolder' followed by a line '<thead><tr><td>Folder name</td><td>Size (human)</td></tr></thead><tbody>';'. The code is color-coded: 'public' is blue, 'function' is blue, 'DoIt' is blue, 'string' is blue, '\$sFolder' is blue, ': string' is blue, '{' is blue, 'ini_set' is orange, 'max_execution_time' is orange, '0' is orange, 'set_time_limit' is orange, '0' is orange, '\$sReturn' is blue, '=<h3>By folders</h3><table id="tblFolder' is orange, '<thead><tr><td>Folder name</td><td>Size (human)</td></tr></thead><tbody>';' is orange, and ';' is blue. The code is wrapped in a DocBlock comment structure with '....' markers.

```
.... You, seconds ago • Uncommitted changes
.... public function DoIt(string $sFolder): string
.... {
....     ini_set('max_execution_time', '0');
....     ini_set('set_time_limit', '0');

....     $sReturn = '';

....     $sReturn .= '<h3>By folders</h3><table id="tblFolder
....     '<thead><tr><td>Folder name</td><td>Size (human)</td></tr></thead><tbody>';
.... }
```

Figure 41: PHP DocBlock

PHP Extension Pack

<https://marketplace.visualstudio.com/items?fitemname=felixfbecker.php-pack>

Includes the most important extensions to get you started with PHP development in Visual Studio Code.

PHP getters and setters

<https://github.com/phpproberto/vscode-php-getters-setters>

Create a new property in your class; right-click on it and choose Insert PHP Getter & Setter to generate PHP code for both actions.

Really easy...

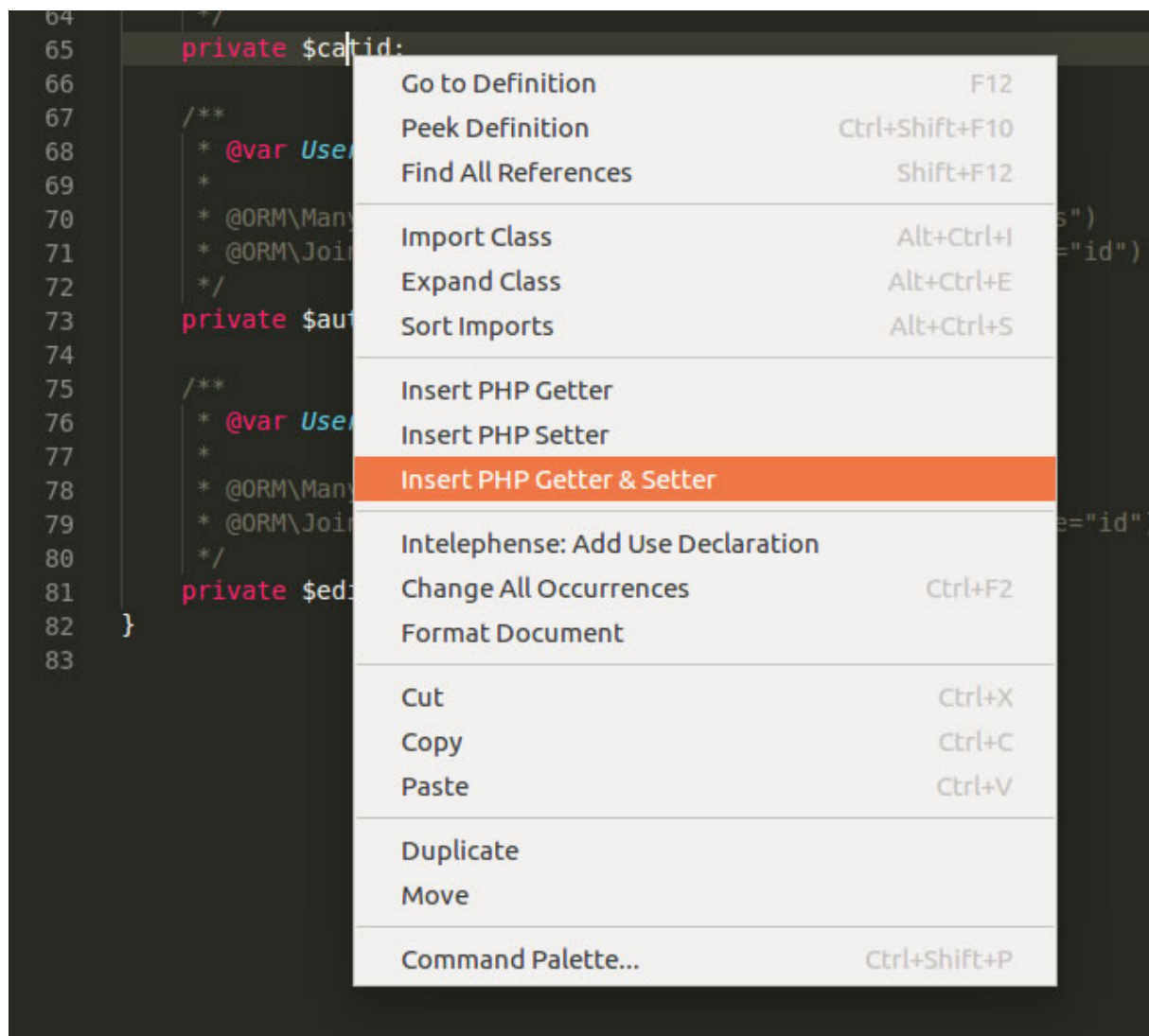


Figure 42: Insert PHP Getter & Setter

The extension can be configured (see <https://github.com/phproberto/vscode-php-getters-setters#extension-settings>) and, if you wish, you can redefine the code of the Setter for instance (as an example, you can decide not to return the value).

PHP IntelliSense

<https://marketplace.visualstudio.com/items?fitemname=felixfbecker.php-intellisense>

Advanced PHP IntelliSense for Visual Studio Code.

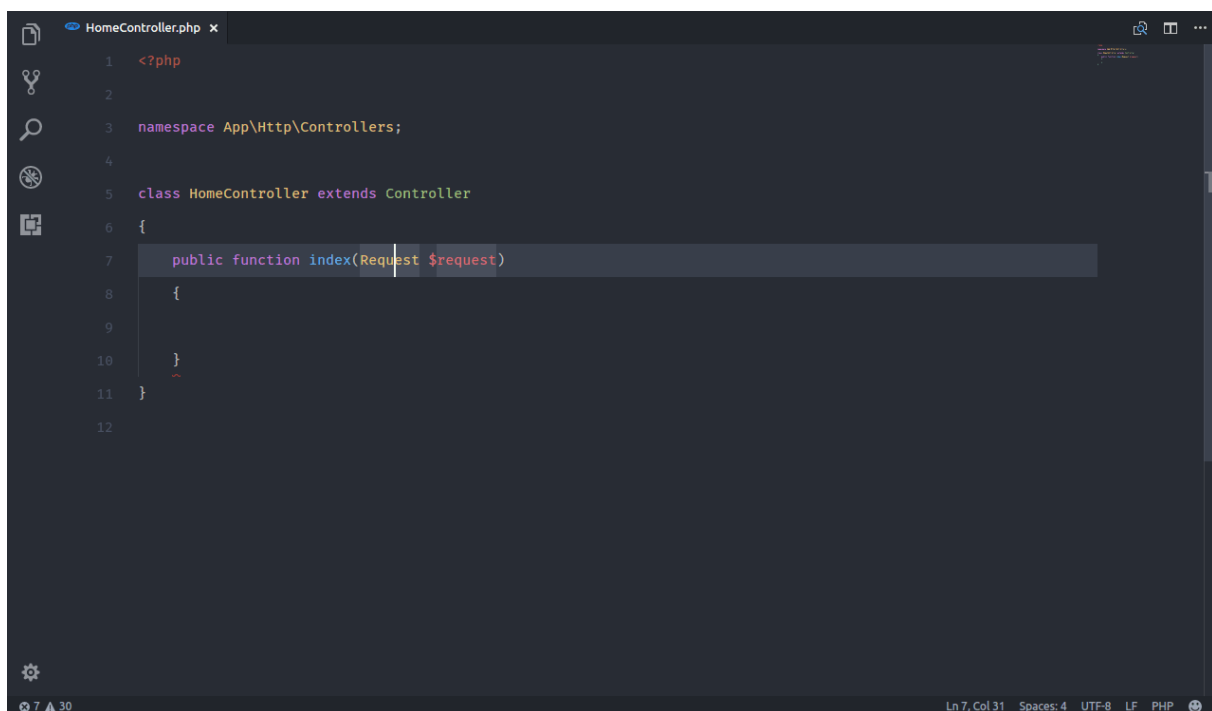

```
39
40 /**
41  * @param int $i Global function param one
42  * @param bool $b Default false param
43  * @param Test|null ...$things Test things
44  */
45 function foo(int $i, bool $b = false, Test ...$things = null)
46 {
47 }
48
49
50
```

Figure 43: PHP IntelliSense

PHP Namespace Resolver {extension-php-namespace-resolver}

<https://marketplace.visualstudio.com/items?itemName=MehediDracula.php-namespace-resolver>

In a PHP file, right-click on a class name and PHP Namespace Resolver will search where that class is defined (based on your composer .json file). If there is only one occurrence, the class will be immediately added in a use command. Otherwise, you'll be prompted to select the desired one.



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 class HomeController extends Controller
6 {
7     public function index(Request $request)
8     {
9
10    }
11 }
12
```

Figure 44: PHP Namespace Resolver

Tips

editorconfig

<https://editorconfig.org>

Just place a file like the one below in your project folder or **any parent folder**. The file should be named `.editorconfig`.

There is **nothing** to do in VSCode except installing the extension. There is no settings... The extension will automatically check for the presence of the `.editorconfig` file in the project folder or any parent folder. If a file is found, if the `root = true` line is in it, the extension knows that this is the final file.

So we can have a file in our project folder without the root statement for overriding some features and a global file in the parent folder.

```
# http://editorconfig.org/

# top-most EditorConfig file
root = true

[*]
charset = utf-8
indent_style = spaces
indent_size = 4
end_of_line = lf
trim_trailing_whitespace = true
insert_final_newline = true

[*.md]
trim_trailing_whitespace = false

[*.Markdown]
trim_trailing_whitespace = false
```

Interface

Use color to identify projects Colors have to be set in the `.vscode/settings.json` file in the project root folder; example:

```
{
  "workbench.colorCustomizations": {
    "titleBar.activeBackground": "#f3a2bc",
    "titleBar.inactiveBackground": "#FF2CCC",
    "titleBar.activeForeground": "#0e0b0b",
    "titleBar.inactiveForeground": "#FFF000"
  }
}
```

To open that setting file, just go to your Workspace settings page.

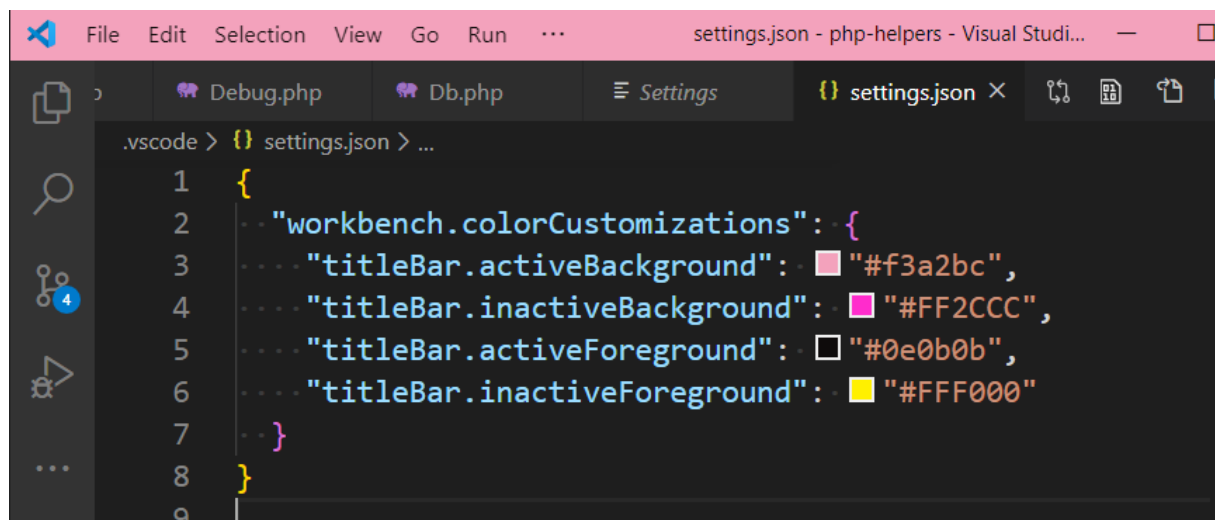


Figure 45: Interface colors

Tips:

- Install Peacock to make changes easily. Peacock will use some default schemas or allow to choose a random one.
- you can use a tool like <https://contrastchecker.com/> to find a good contrast schema.
- You can also deeper in the configuration of the editor; please take a look to <https://code.visualstudio.com/api/references/theme-color>

Happy coloring ☒

Keyboard shortcuts for Windows

- Cheat sheet for Windows users: <https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf>
- Online and interactive tutorial

PHP-CS-FIXER

The use of php-cs-fixer can help to maintain a proper code. More info below.

Installation Install PHP-CS-FIXER <https://github.com/FriendsOfPHP/PHP-CS-Fixer>.

Create your `php-cs-fixer.php` file and put it there your settings. More info below.

Open VSCode, go to the settings page (press CTRL-,), search for `php-cs-fixer.config` and initialize the settings to the path of your `php-cs-fixer.php` (f.i. `C:\\config\\php-cs-fixer.php`).

Remove unused imports The `no_unused_imports` setting, when set to `true`, allow php-cs-fixer to detect unused imports like below and to remove it automatically.

```
<?php
```

```
require 'vendor/autoload.php';
```

```
use Avonture\Pandoc;
```

```
echo __DIR__;
```

This is done when you'll save your file.

```
<?php
```

```
return PhpCsFixer\Config::create()
    ->setRules([
        'no_unused_imports'=> true
    ]);
```

Search and replace - Regex

Remove all empty lines

<https://dev.to/gyurisc/easily-remove-empty-lines-using-regular-expression-in-visual-studio-code-1230>

- Search `^(?:[\t]*(?:\r?\n|\r))+`
- Replace by *let this field empty*

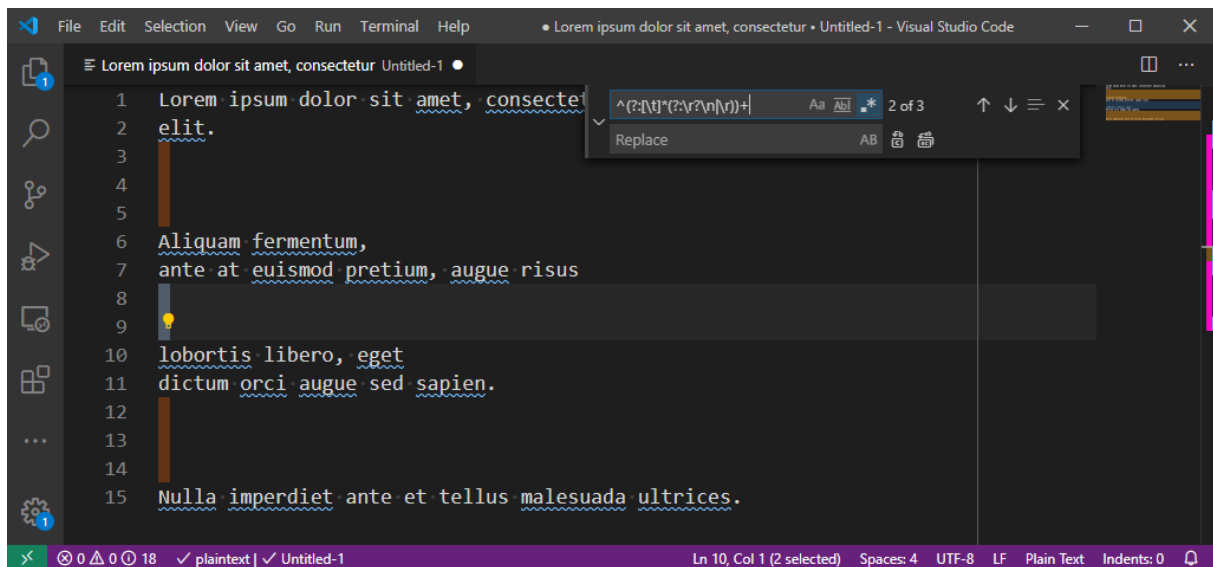


Figure 46: Remove all empty lines

Remove all lines except those matching a regex Imagine you've a big JSON file with f.i. a list of users like this:

```
[
  {
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz"
  },
  {
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv"
  },
  {
    "name": "Clementine Bauch",
    "username": "Samantha",
    "email": "Nathan@yesenia.net"
  },
  {
    "name": "Patricia Lebsack",
    "username": "Karianne",
    "email": "Julianne.OConner@kory.org"
  },
  {
```

```
    "name": "Chelsey Dietrich",  
    "username": "Kamren",  
    "email": "Lucio_Hettinger@annie.ca"  
  }  
]
```

From that list you wish to keep only lines with emails.

The idea is to use a regex expression so we can match email lines and use a negative search so not email lines are selected:

To match email lines (with the carriage return at the end), we'll use `. *email.*\n`.

For the use case, we'll search for `. *email.*\n` and replace by nothing: emails lines will be removed in the entire file.



```
{ Untitled-1  
1  [  
2    {  
3      "name": "Leanne Graham",  
4      "username": "Bret",  
5      "email": "Sincere@april.biz"  
6    },  
7    {  
8      "name": "Ervin Howell",  
9      "username": "Antonette",  
10     "email": "Shanna@melissa.tv"  
11   },  
12   {  
13     "name": "Clementine Bauch",  
14     "username": "Samantha",  
15     "email": "Nathan@yesenia.net"  
16   },  
17   {  
18     "name": "Patricia Lebsack",  
19     "username": "Karianne",  
20     "email": "Julianne.OConner@kory.org"  
21   }  
22 ]
```

Figure 47: Search and replace

Using the negative regex `^(?!.*email).*$\n` will match all lines except the ones containing the email word and, here too, if we search and replace by nothing, we'll remove all lines except emails:



```
1  [
2    {
3      "name": "Leanne Graham",
4      "username": "Bret",
5      "email": "Sincere@april.biz"
6    },
7    {
8      "name": "Ervin Howell",
9      "username": "Antonette",
10     "email": "Shanna@melissa.tv"
11   },
12   {
13     "name": "Clementine Bauch",
14     "username": "Samantha",
15     "email": "Nathan@yesenia.net"
16   },
17   {
18     "name": "Patricia Lebsack",
19     "username": "Karianne",
20     "email": "Julianne.OConner@kory.org"
21   }
22 ]
```

Figure 48: Search and replace negative regex

Custom tasks

<https://code.visualstudio.com/docs/editor/tasks>

<https://code.visualstudio.com/docs/editor/variables-reference>

By creating a `./.vscode/tasks.json`, it's possible to add custom tasks to Visual Studio Code.

A custom task can be running PHP Unit tests or any other shell commands.

Run a DOS batch The following example define a task called `concat-md`, it's a shell command (i.e. a DOS task).

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "concat-md",
      "type": "shell",
```

```

        "command": "${workspaceFolder}/concat-md.cmd",
        "group": "build",
        "presentation": {
            "reveal": "always"
        },
        "problemMatcher": []
    }
]
}

```

Note: `${workspaceFolder}` has been used instead of just `./concat-md.cmd` to make sure vs-code will use the file from the repo's root folder.

To run that task, we just need to call the Command Palette (CTRL-SHIFT-P) then select **Tasks: Run task**. You'll then be prompted to select the task and `concat-md` will appear.

All your custom tasks will be accessible in the **Tasks: Run task** list.

Tip: it's possible to specify more than one command using the `&&` syntax; f.i. `"command": "cd tests/ && run_all.cmd"`,

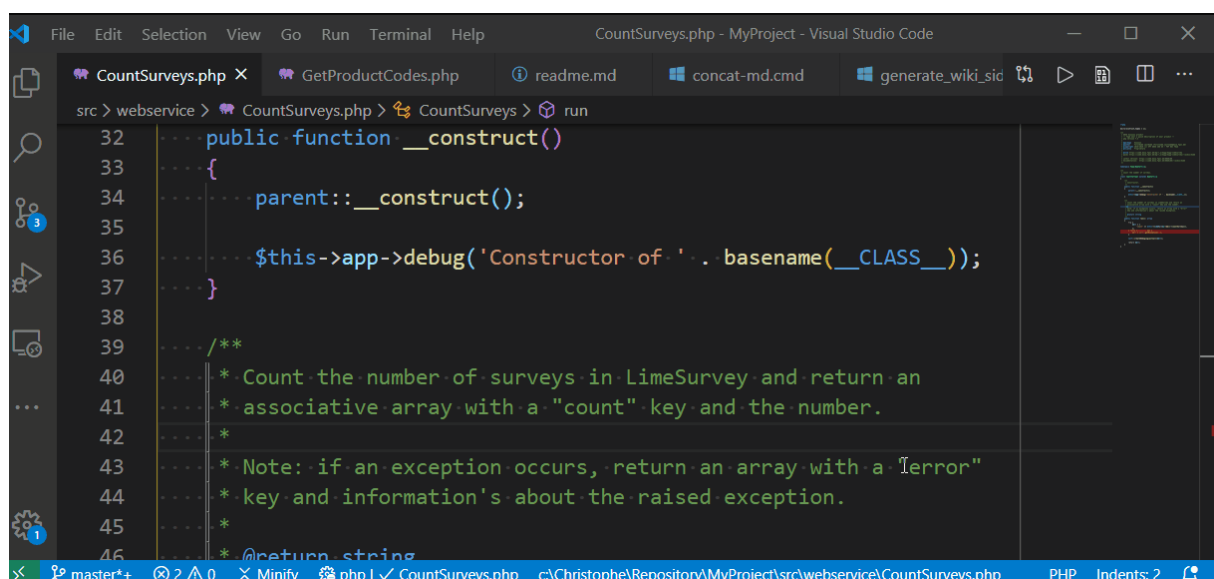


Figure 49: Running the pre-commit custom task

Run PHP-Unit Another example can be to run PHP Unit with some command-line options.

The example below defines a `phpunit: run tests\Api` task so we can execute all tests under the `Api` folder. We can of course use any command line options supported by PHP Unit.


```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "phpunit: run tests\\Api",
      "type": "process",
      "command": "${workspaceFolder}\\vendor\\bin\\phpunit.bat",
      "args": [
        "${workspaceFolder}\\tests\\Api"
      ],
      "presentation": {
        "echo": true,
        "panel": "new"
      },
      "problemMatcher": []
    }
  ]
}
```

Note: the type defined in the JSON file is process, this means that the defined command will be executed within vscode, in the Terminal. The process should then be an executable and any parameters have to be defined in the args array;

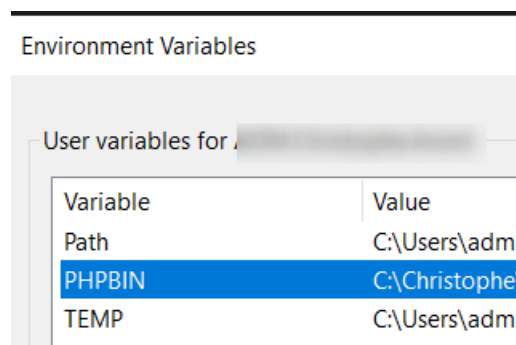
Troubleshooting

Intelephense

In order to avoid errors with Intelephense like Undefined function xxx (f.i. trim) or Undefined type xxx (f.i. stdClass) for core PHP features, make sure to specify in your settings.json file the correct PHP version you're using.

"intelephense.environment.phpVersion": "7.4.4"

That version should match the one you've defined in the PHPBIN environment variable:

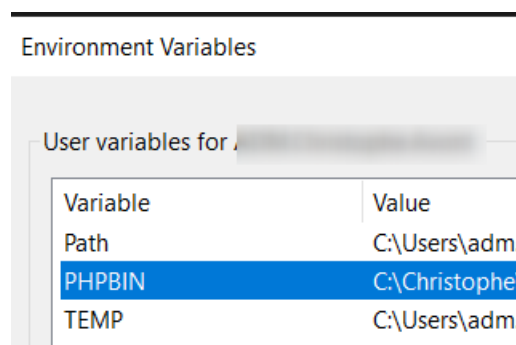
**Figure 50:** PHPBIN

Phan

Make sure to specify in your settings .json file the PHP version you're using:

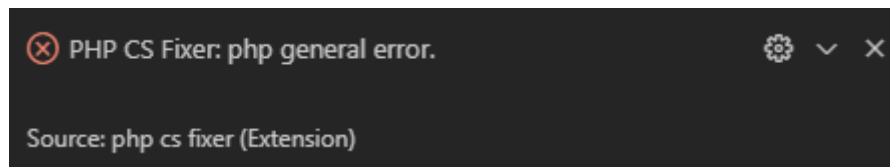
"phan.phpExecutablePath": "C:\\wamp64\\bin\\php\\php7.4.4\\php.exe"

That version should match the one you've defined in the PHPBIN environment variable:

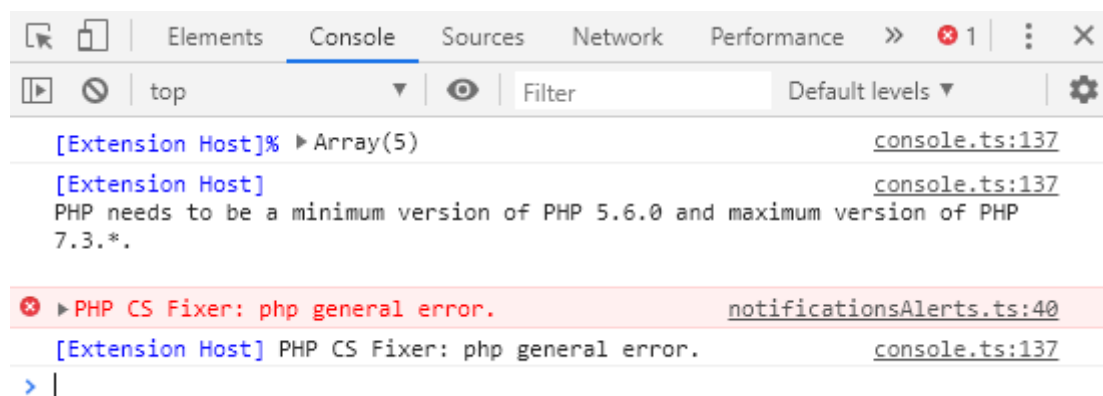
**Figure 51:** PHPBIN

PHP-CS-FIXER

PHP General Error By calling a PHP-CS-FIXER function like running `Fix this file` (ALT-SHIFT-F), you can get the following error:

**Figure 52:** PHP General Error

To get more information, make sure to display the Developer Tools (CTRL-SHIFT-I) in the Help menu. You'll get there extra information's.

**Figure 53:** Console

As we can see here above, our PHP-CS-FIXER needs to be updated: we're using a more recent version of PHP and the maximum supported by the current PHP-CS-FIXER installed version is an old one.

Create your own extension

<https://www.youtube.com/watch?v=srwsnNhiqv8>

<https://code.visualstudio.com/api/get-started/your-first-extension>

It's quite easy to create his own extension by the use of Yo Code - Extension Generator.

The YouTube video is showing how to create a snippet extension. Really easy!

Bookmarks

List of nice website for learning VS Code.

VS Code can do that

All the best things about Visual Studio Code that nobody ever bothered to tell you.

<https://vscodecandothat.com/>

VS Code channel on youtube

https://www.youtube.com/channel/UCs5Y5_7XK8HLDX0SLNwkd3w

Annex

List of extensions I use

You can create the same list by running code `--list-extensions | % { "code --install-extension $_" }` in a Windows Powershell prompt.

```
code --install-extension aaron-bond.better-comments
code --install-extension alefragnani.Bookmarks
code --install-extension bajdzis.vscode-twig-pack
code --install-extension bmewburn.vscode-intelephense-client
code --install-extension bobmagicii.autofoldyeah
code --install-extension calebporzio.better-phpunit
code --install-extension Cameron.rerun-last-command
code --install-extension CoenraadS.bracket-pair-colorizer
code --install-extension DavidAnson.vscode-markdownlint
code --install-extension dbaeumer.vscode-eslint
code --install-extension DotJoshJohnson.xml
code --install-extension druideinformatique.antidote
code --install-extension eamodio.gitlens
code --install-extension EditorConfig.EditorConfig
code --install-extension emilast.LogFileHighlighter
code --install-extension esbenp.prettier-vscode
code --install-extension felipecaputo.git-project-manager
code --install-extension felixfbecker.php-debug
code --install-extension felixfbecker.php-intellisense
code --install-extension felixfbecker.php-pack
code --install-extension formulahendry.auto-close-tag
```

```
code --install-extension formulahendry.code-runner
code --install-extension GrapeCity.gc-excelviewer
code --install-extension Gruntfuggly.todo-tree
code --install-extension hollowtree.vue-snippets
code --install-extension ikappas.phpcs
code --install-extension ionutvmi.reg
code --install-extension johnpapa.vscode-peacock
code --install-extension junstyle.php-cs-fixer
code --install-extension Kasik96.latte
code --install-extension marsl.vscode-php-refactoring
code --install-extension mblode.twig-language
code --install-extension mde.select-highlight-minimap
code --install-extension mechatroner.rainbow-csv
code --install-extension MehediDracula.php-namespace-resolver
code --install-extension mikestead.dotenv
code --install-extension mrmlnc.vscode-apache
code --install-extension mrmlnc.vscode-duplicate
code --install-extension ms-vscode-remote.remote-wsl
code --install-extension ms-vscode.powershell
code --install-extension neilbrayfield.php-docblocker
code --install-extension NiclasvanEyk.am-i-behind
code --install-extension nonoroazoro.syncing
code --install-extension octref.vetur
code --install-extension olback.es6-css-minify
code --install-extension Otiel.vscode-xyplorer
code --install-extension Perkovec.emoji
code --install-extension persoderlind.vscode-phpcbf
code --install-extension phproberto.vscode-php-getters-setters
code --install-extension RoscoP.ActiveFileInStatusBar
code --install-extension sdras.vue-vscode-snippets
code --install-extension SirTori.indenticator
code --install-extension slevesque.vscode-autohotkey
code --install-extension slevesque.vscode-hexdump
code --install-extension spences10.vba
code --install-extension st-pham.php-refactor-tool
code --install-extension streetsidesoftware.code-spell-checker
code --install-extension TabNine.tabnine-vscode
code --install-extension tomoyukim.vscode-mermaid-editor
```

```
code --install-extension Tyriar.sort-lines
code --install-extension TysonAndre.php-phan
code --install-extension vscode-icons-team.vscode-icons
code --install-extension whatwedo.twig
code --install-extension Christophe Avonture.writing-documentation-
snippets
code --install-extension yatki.vscode-surround
code --install-extension yzhang.markdown-all-in-one
```

License

MIT