Spell Right

Bartosz Antosik | 48,896 installs | 443,114 downloads | ★★★★★ (28) | Free

Install Trouble Installing?

Overview Q & A Rating & Review

Multilingual, Offline and Lightweight Spellchecker for Visual Studio Code

Visual Studio Marketplace v3.0.22 installs 48.91K rating average: 4.64/5 (28 ratings) license MIT

Features

- Spells **plain text/markdown/LaTeX** documents, **comments/strings** parts of most **source code** (C++, C, Python, JavaScript, Batch, ..., D, Julia etc.) documents and **text/comment** nodes on **XML/HTML** class documents.
- Supports **every language** that can be used with either of the below mentioned native spelling engines (e.g. **all languages** that are available in Microsoft Office, see here, **multiple languages** in Windows Single Language editions, like here etc.)
- Supports use of multiple workspace plaintext file dictionaries which may be used for specialized vocabularies like **medical terms**, **trademark names** etc.
- Spelling documents' syntactic elements e.g. **comments/strings** in **different languages** (e.g. strings spelled in English and comments spelled in French).
- Spelling of **multiple languages in one document** either by selecting more than one language for spelling or using In-Document commands to switch between languages.
- **Case sensitive** which means that it will distinguish between *english* and *English*, *french* and *French* and which is critical in some western languages like e.g. German.
- Spells, among others, **short words**, **abbreviations**, **contractions** (*I*, *I'm*, *I'll*, *i.e.*, *doesn't*, *etc.*) and parenthetical plurals (*word(s)*, *process(es)*, etc.)
- Spells **CamelCase**, **snake_case** and **digit2inside** compound phrases respecting Unicode capital/small letters distinction (e.g.: *SuperŚlimak* is spelled as *Super Ślimak*) and capital letter adhesion (e.g.: *HTMLTest* is spelled as *HTML Test*).
- Unobtrusive **GUI/command** interface for switching spelling dictionary (language) and turning spelling ON/OFF for particular document class.
- **In-Document commands** allow to switch spelling **ON** or **OFF** despite global settings and **change spelling language** multiple times within the document.
- Small memory & CPU usage footprint - uses **offline**, **OS native** spell checking backends: Windows Spell Checking API (windows 8/10), NSSpellChecker (macOS) and Hunspell (Linux, Windows 7).
- Extension uses **background processing** (on idle) and **differential edit notifications** to minimize area spelled during editing only to lines touched by changes.

Installation

Search for *Spell Right* from the extension installer within VSCode or execute below line in the **command palette** (`F1` or `Ctrl+Shift+P`):

```
ext install spellright
```

Spell Right requires spelling back-end which is different for various platforms. Please read carefully the section below on how to provide dictionaries for Spell Right to work properly.

If you have any problem with installation or you see that Spell Right does not work with some type of document please read carefully the lengthy README file below and if nothing seems to be related to the problems you face post an issue here.

Dictionaries

Windows 8+

On Microsoft Windows from version 8 on Spell Right uses system spelling API. Follow system guidelines, like here or here, on how to install additional system spelling dictionaries.

macOS

On macOS Spell Right uses system spelling API.

Linux and Windows 7

On Linux and Windows 7 Spell Right uses built in *Hunspell* spell checker library. To use it a pair of Dictionary (*.dic) and Affixes (*.aff) files with **UTF-8 encoding** have to be downloaded (remember to download RAW files) e.g. from here (multiple languages), here (Portuguese) or here (French) and placed in `Dictionaries` subfolder of VSCode's user global configuration directory, located at:

- **Windows:** `%APPDATA%\Code\Dictionaries\`
- **Linux:** `$HOME/.config/Code/Dictionaries/`

`Dictionaries` subfolder does not exists there by default and has to be created manually.

In case you run *Insiders* edition of VSCode then `Code` part of the path has to be changed to `Code - Insiders` appropriately.

Spell Right cannot automatically match the dictionary name with system locale settings when using *Hunspell* dictionaries hence **at first run dictionary has to be selected manually** (e.g from status bar).

On most Linux distributions system-wide dictionaries can be reused (for now only UTF-8 encoded dictionaries are supported, verify `SET` line in *.aff file) by soft linking the system folder e.g.:

Categories

Other

Tags

checker multi-language offline spell spellchecker

Resources

Issues
Repository
Homepage
License
Changelog
Download Extension

Project Details

bartosz-antosik/vscode-spellright
No Pull Requests
22 Open Issues
Last commit: a month ago

More Info

Version	3.0.22
Released on	4/4/2017, 10:17:11 PM
Last updated	12/31/2018, 4:52:34 PM
Publisher	Bartosz Antosik
Unique Identifier	ban.spellright
Report	Report Abuse

Spell Right stores words considered as spelled correctly and not existing in the main spelling engine (a.k.a. ignored words) in plaintext dictionary files. All the extension's commands add words to two main files, user and workspace dictionaries, both contained in `spellright.dict` files, one located in user settings folder and the other in workspace settings folder (`.vscode`). Besides these there can be any number of `*.dict` files in workspace settings folder e.g. dictionaries containing specialized vocabularies like **medical terms**, **trademark names** etc. User dictionary is always used and workspace dictionaries are used in the context of opened folder. All provided dictionaries are used in conjunction.

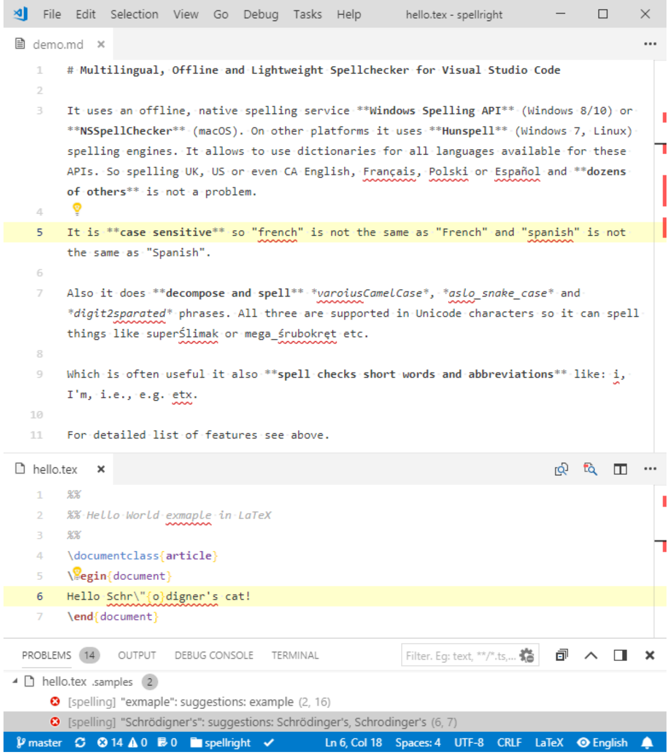
Symbols in Source Code Documents

When spelling source code class of document which provides symbol information for the document (e.g. the symbols which are used when `Ctrl+Shift+O` is pressed to perform GoTo symbol operation) the symbols are used for spelling of the document. It reduces the number of false spelling warnings mostly in comments which provide source code documentation on the usage of these symbols, e.g. variables, function parameters etc.

Screenshots

Documents

Spelling of **plain text/markdown/LaTeX** documents:



```
# Multilingual, Offline and Lightweight Spellchecker for Visual Studio Code

It uses an offline, native spelling service **Windows Spelling API** (Windows 8/10) or **NSSpellChecker** (macOS). On other platforms it uses **Hunspell** (Windows 7, Linux) spelling engines. It allows to use dictionaries for all languages available for these APIs. So spelling UK, US or even CA English, Français, Polski or Español and **dozens of others** is not a problem.

It is **case sensitive** so "french" is not the same as "French" and "spanish" is not the same as "Spanish".

Also it does **decompose and spell** *varoiusCamelCase*, *aslo_snake_case* and *digit2sparated* phrases. All three are supported in Unicode characters so it can spell things like superŚlimak or mega_śrubokręt etc.

Which is often useful it also **spell checks short words and abbreviations** like: i, I'm, i.e., e.g. etx.

For detailed list of features see above.
```

```
%%
%% Hello World exmaple in LaTeX
%%
\documentclass{article}
\begin{document}
Hello Schr\"{o}digner's cat!
\end{document}
```

[spelling] "exmaple": suggestions: example (2, 16)

[spelling] "Schrödigner's": suggestions: Schrödinger's, Schrodinger's (6, 7)

Source Code & Markup Documents

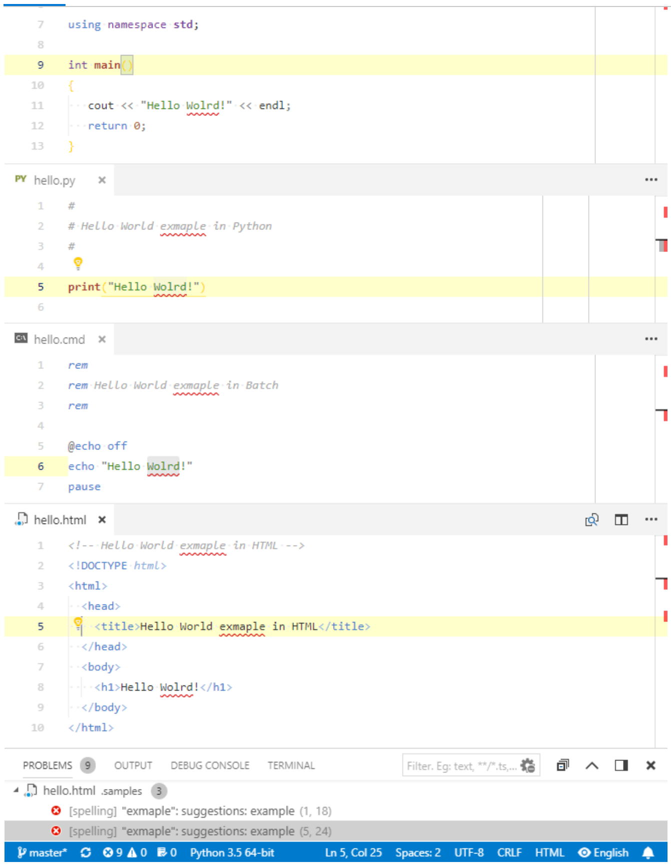
It can spell *strings/comments* parts of most **source code** (*C++*, *Python*, *JavaScript*, *Batch* and lots of others) and *text/comment* nodes on **XML**/**HTML** class documents:

Spell Right

Bartosz Antosik | 48,896 installs | 443,114 downloads | ★★★★★ (28) | Free

Install   Trouble Installing?

Overview   Q & A   Rating & Review



```
using namespace std;

int main()
{
    cout << "Hello Wolrd!" << endl;
    return 0;
}
```

```
#
# Hello World exmaple in Python
#

print("Hello Wolrd!")

```

```
rem
rem Hello World exmaple in Batch
rem

@echo off
echo "Hello Wolrd!"
pause
```

```
<!-- Hello World exmaple in HTML -->
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World exmaple in HTML</title>
  </head>
  <body>
    <h1>Hello Wolrd!</h1>
  </body>
</html>
```

hello.html .samples 3

[spelling] "exmaple": suggestions: example (1, 18)

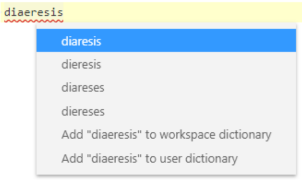
[spelling] "exmaple": suggestions: example (5, 24)

master* ⊗ 9 ⚠ 0 0 Python 3.5 64-bit Ln 5, Col 25 Spaces: 2 UTF-8 CRLF HTML English

Usage

Correcting Spelling Errors

Press `Ctrl+.` (Windows, Linux) ⌘`+.` or `Cmd+.` (macOS) or click *Lightbulb* to show a context **menu with suggestions**:



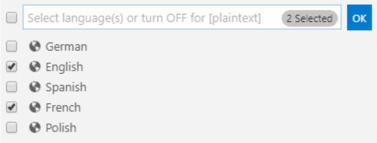
diaeresis

diaresis
dieresis
diareses
diereses
Add "diaeresis" to workspace dictionary
Add "diaeresis" to user dictionary

Use `F8` or `Shift+F8` to jump to **next**/**previous** spelling error.

Changing Language and Turning OFF

Extension has a handy list interface for switching spelling dictionary (language) or turning spelling OFF for currently open document type:



□ Select language(s) or turn OFF for [plaintext] 2 Selected OK

□ German

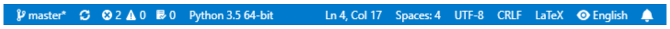
☑ English

□ Spanish

☑ French

□ Polish

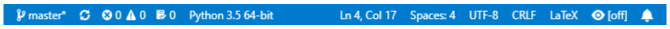
It can be reached by clicking on indicator field with an "eye" symbol in status bar:



master* ⊗ 2 ⚠ 0 0 Python 3.5 64-bit Ln 4, Col 17 Spaces: 4 UTF-8 CRLF LaTeX English

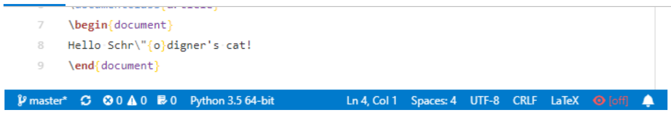
Alternatively same result can be achieved selecting command `SpellRight: Select Dictionary (Language)` or `SpellRight: Turn OFF for Current Document Type` from the **command palette** (`F1`/`Ctrl+Shift+P`).

Status bar indicator also shows when spelling for particular document class has been turned OFF:



master* ⊗ 0 ⚠ 0 0 Python 3.5 64-bit Ln 4, Col 17 Spaces: 4 UTF-8 CRLF LaTeX [off]

Or when it has been forced OFF by In-Document command (`spellcheck&#x2d;off`) or rule in `.spellignore`:



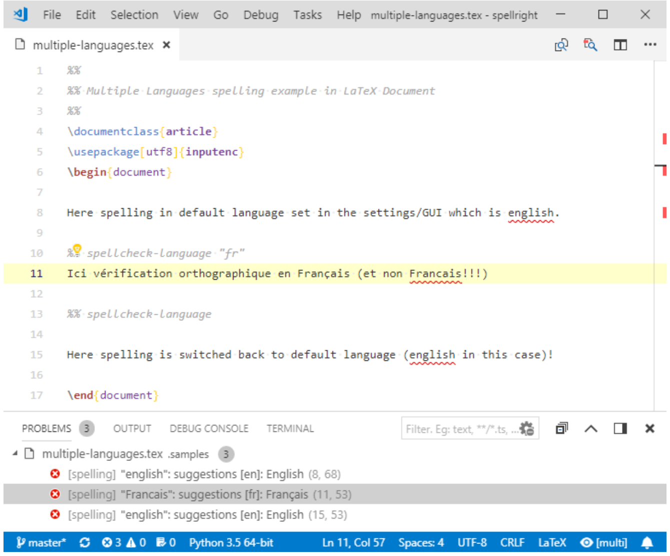
```
\begin{document}
Hello Schr\"{o}dinger's cat!
\end{document}
```

master* ⊗ 0 ⚠ 0 0 Python 3.5 64-bit Ln 4, Col 1 Spaces: 4 UTF-8 CRLF LaTeX [off]

Multiple languages in one document

Language selector allows to select more than one language. Document is then spelled using all of the dictionaries and when a token (word) is considered correct **in one of the dictionaries** it is considered as spelled correctly. When it is not, then suggestions for all selected languages are shown.

Language can also be switched by using In-Document command `spellcheck-language` placed within document:

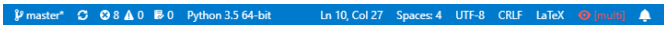


```
%%
%% Multiple Languages spelling example in LaTeX Document
%%
\documentclass{article}
\usepackage[utf8]{inputenc}
\begin{document}

Here spelling in default language set in the settings/GUI which is english.

%% spellcheck-language "fr"
Ici vérification orthographique en Français (et non Francais!!!)

%% spellcheck-language

Here spelling is switched back to default language (english in this case)!

\end{document}
```

```
multiple-languages.tex .samples 3
  ⊗ [spelling] "english": suggestions [en]: English (8, 68)
  ⊗ [spelling] "Francais": suggestions [fr]: Français (11, 53)
  ⊗ [spelling] "english": suggestions [en]: English (15, 53)
```

master* ⊗ 3 ⚠ 0 0 Python 3.5 64-bit Ln 11, Col 57 Spaces: 4 UTF-8 CRLF LaTeX [multi]

And when at least one language spelled is missing dictionary:



master* ⊗ 8 ⚠ 0 0 Python 3.5 64-bit Ln 10, Col 27 Spaces: 4 UTF-8 CRLF LaTeX [multi]

Commands

This extension contributes the following commands:

`SpellRight: Select Dictionary (Language)`

Pops dictionary selection list. Selecting language also turns spelling ON. The list also allows to turn spelling OFF for currently open document type. Can also be reached by clicking indicator field in status bar.

`SpellRight: Turn OFF for Current Document Type`

Turn spelling OFF for currently open document type.

`SpellRight: Open User Dictionary File`

Open in editor `spellright.dict` file from user settings folder.

`SpellRight: Open Workspace Dictionary File`

Open in editor `spellright.dict` file from workspace settings folder (`.vscode`) appropriate for currently open document in (multi root) workspace.

`SpellRight: Add Selection to Workspace Dictionary`

Add text selected in editor to main workspace dictionary appropriate for currently open document in (multi root) workspace.

`SpellRight: Add Selection to User Dictionary`

Add text selected in editor to main user dictionary.

`SpellRight: Update Configuration`

Saves current state of GUI (that is language and map of documents classes spell checked) to appropriate configuration file. Works only if `"spellright.configurationUpdate"` is set to `false`.

Settings

This extension contributes the following settings (with default values):

`"spellright.language": [ "" ]`

Default language (dictionary/country name) used for spelling. Typically in a LANGUAGE (e.g.: "en", "fr", when `"spellright.groupDictionaries"` is `true`) or LANGUAGE-COUNTRY format (e.g.: "en-US", "en-GB", "fr-CA", Enable/disable including suggested corrections in hints. Disabling suggestions significantly speeds checking up. May be useful in case of large, often switched or saved documents.

`"spellright.addToSystemDictionary": false`

When `true` words added to user dictionary are stored in system default custom spelling dictionary instead.

`"spellright.groupDictionaries": true`

Enable/disable grouping of dictionaries by language. Disabling grouping results in displaying dictionaries for all regional variants (e.g. en-US, en-GB, en-CA etc.) as separate entries. When enabled regional dictionaries are displayed as single dictionary under common language name (e.g. "English"). Works only on native Windows & macOS spelling APIs.

`"spellright.recheckOnSave": false`

Enable/disable re-checking of entire document after file is saved.

`"spellright.documentTypes": [ "plaintext", "markdown", "latex" ]`

Document types for which spelling will be turned ON by default.

`"spellright.ignoreRegExps": []`

Regular expressions ignored in spelling. Allows to ignore/consider as spelled correctly generalized expressions. Works on raw document **before** separating words to spell which allows to ignore larger parts of the document. Regular expressions have to be in double quoted JavaScript regular expression format. That is backslash has to be quoted as well e.g.: `"/(\\.?)(gif|png)/g"` to ignore file extensions like `".gif"` and `".png"`.

`"spellright.ignoreRegExpsByClass": {}`

Extends setting of `"spellright.ignoreRegExps"` per document type. Accepts object of key-multi-value pairs. For example following settings:

```
"spellright.ignoreRegExpsByClass": {
    "markdown": [ "/&amp;/g", "/&nbsp;/g" ],
    "cpp": [ "/#include\\s+\\\".+\\\"/g" ],
    "html": [ "/<script>[^]*?</script>/gm" ],
    "latex": [ "/\\\\begin{minted}[^]*?\\\\end{minted}/gm" ]
}
```

- avoid spelling of `&amp;` and `&nbsp;` literals in markdown documents;
- avoid spelling of strings in `#include "file"` construct in CPP documents;
- avoid spelling of multiline `<script></script>` tag content in HTML documents;
- avoid spelling of "minted" code blocks in LaTeX documents.

Please mind the fact that both `"spellright.ignoreRegExps"` and `"spellright.ignoreRegExpsByClass"` **may have serious impact on performance**. They are applied on whole document before every, even smallest spell check, has to reapply these filters so they may cost time in complicated expressions.

`"spellright.ignoreFiles": [ "**/.gitignore", "**/.spellignore" ]`

Set of file patterns to globally, silently exclude files from being spelled. Files described with this setting will not be reported as forced OFF spelling (red indicator in status bar). Patterns defined as for gitignore.

`"spellright.notificationClass": "error"`

Allows to change class of diagnostic messages produced by Spell Right which changes in turn underline color. Possible values (with corresponding underline color) are: `"error"` (red), `"warning"` (green), `"information"` (green), `"hint"` (invisible).

`"spellright.notificationClassByParser": {}`

Extends `"spellright.notificationClass"` to allow change class of diagnostic messages per parser. For example following settings:

```
"spellright.notificationClassByParser": {
    "code": "information"
}
```

will produce `"information"` class diagnostics for documents spelled with `code` parser. Possible values on left side of the association are: `plain`, `markdown`, `code`, `latex` and `xml`.

`"spellright.spellContext": "body comments strings"`

Allows to enable (present in string) or disable (absent in string) spelling of syntactic parts of the documents. Currently supported are:

- `body` - body of document (e.g. LaTeX, Plaintext, Markdown etc.);
- `code` - code blocks in Markdown type documents (spelled as whole, not syntactically);
- `comments` - comment (block & line) sections in programming languages, also LaTeX;
- `strings` - strings in programming languages.

`"spellright.spellContextByClass": {}`

Same as `"spellright.spellContext"` but per document type. Accepts object of key-value pairs. For example following settings:

```
"spellright.spellContextByClass": {
    "latex": "body",
    "cpp": "comments",
    "python": "strings"
}
```

- disable spelling of comments in LaTeX documents;

```
        "strings": [ "en-US" ],
        "comments": [ "en-GB" ]
    }
```

will spell strings in *American English* and comments in *British English* of course if the `"spellright.groupDictionaries"` flag is set to `false`.

Configuration item `"spellright.languageContext"` is more important than `"spellright.language"` but less important than In-Document commands.

`"spellright.languageContextByClass": {}`

Same as `"spellright.languageContext"` but per document type. For example following settings:

```
    "spellright.languageContextByClass": {
        "latex": {
            "body": [ "fr" ],
            "comments": [ "en" ]
        }
    }
```

will spell body of `latex` documents in *French* and comments in *English*.

Configuration item `"spellright.languageContextByClass"` is more important than `"spellright.language"` and `"spellright.languageContext"` but less important than In-Document commands.

`"spellright.configurationUpdate": true`

If set to true then each operation on the GUI (change of language, turning spelling OFF for particular document type) is automatically saved in appropriate configuration settings (workspace if workspace is open and user if not). Setting to false requires using of `"spellright.configurationUpdate"` command to save the changes performed in GUI.

`"spellright.configurationScope": "workspace"`

Allows to decide which configuration gets updated when `"spellright.configurationUpdate"` is set to true. Possible values are `user` and `workspace`.

`"spellright.latexSpellParameters": (see below for default value)`

Defines LaTeX commands that should have both mandatory (`[]`) and optional (`{}`) parameters spelled. Other commands are removed from spelling. Default value:

```
    "spellright.latexSpellParameters": [
        "author",
        "title",
        "date",
        "chapter",
        "section\\*?",
        "subsection\\*?",
        "subsubsection\\*?",
        "part",
        "paragraph",
        "subparagraph",
        "text(rm|sf|tt|md|bf|up|it|sl|sc|normal)",
        "underline",
        "emph",
        "item",
        "footnote(text)?",
        "caption(of)?",
        "multicolumn",
        "href",
        "hyperref",
        "begin\\{frame\\}"
    ]
```

`"spellright.parserByClass": {}`

Allows to assign or override generic parser for particular document class. For example following settings:

```
    "spellright.parserByClass": {
        "perl": {
            "parser": "code"
        }
    }
```

assigns parser of generic type `code` (Source Code Parser) to `perl` (Perl) document class. Possible values are:

- `plain` - spells entire content of the document;
- `markdown` - spells everything except code blocks;
- `code` - spells comments and strings;
- `latex` - spells everything except LaTeX commands;
- `xml` - spells comments and everything outside markup.

`"spellright.useDocumentSymbolsInCode": true`

If set to true Spell Right will use document symbols (variable, function names etc.) when spelling source code documents. Significantly reduces number of misspelled words in doc-strings and in comments whenever a symbol used in code is used and the symbol does not disassemble to properly spelled parts using CamelCase, snake_case etc. separation.

In-Document Commands

Beside global settings following commands can be embedded inside spelled parts of the document (e.g.: comments, strings etc.):

`"spellright.languageContextByClass"` and `"spellright.languageContext"` and `"spellright.language"` configuration items.

`spellcheck&#x2d;off` (alternative syntax: `spellcheck: off`)

Forces spelling **OFF** for the entire document despite global settings.

`spellcheck&#x2d;on` (alternative syntax: `spellcheck: on`)

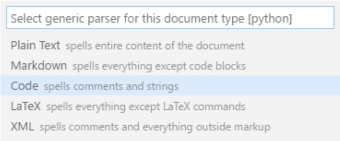
Forces spelling **ON** for the entire document despite global settings. Has higher priority than turning spelling off with both *In-Document* `spellcheck&#x2d;off` command and `.spellignore` patterns.

Ignore file

`.spellignore` file located in workspace root directory, appropriate to currently open file in (multi root) workspace, can be used to disable spelling for files described by gitignore syntax file patterns.

Unknown document type

Spell Right must have an idea of the document structure in order to be able to spell it. Most document types have parts that do not need to be spelled (e.g. code blocks in Markdown, commands in LaTeX documents etc.) When an unregistered document type is encountered then Spell Right proposes a list of generic parsers to choose from:


Select generic parser for this document type [python]
Plain Text spells entire content of the document
Markdown spells everything except code blocks
Code spells comments and strings
LaTeX spells everything except LaTeX commands
XML spells comments and everything outside markup

Select appropriate class to be able to spell the document. If `spellright.configurationUpdate` is set to `true` assignment will be written to the configuration settings either user or workspace according to the value of `spellright.configurationScope` setting.

Known Issues

- Hint box associated with "Bulb" Code Action has an ugly habit of wrapping text at certain width not at white/punctuation character thus suggestions got cut in weird places (this is probably more of a VSCode's issue).
- There is a limit, imposed by VSCode, on the number of diagnostics that an extension (Spell Right among) can provide for one file. The number is 1000 and Spell Right cannot display more spelling errors. Once the spelling errors from the head of the file are corrected or added to the dictionaries more issues will appear at the end.
- Due to bug in `NSSpellChecker` layer of macOS Spell Right cannot currently use dictionaries installed in `~/Library/Spelling` folder, just those that came with the system.
- There are rare situations when dictionaries are not reported correctly on Windows using Windows Spelling API (Windows 8+). They can result in no dictionary or only some dictionaries available. So far the reason for this is unknown but seems to be outside of the extension (see #106 and #53 for details).

Notice

This extension can be considered a Work In Progress. Please report all the errors and/or annoyances that you see on the issues page.

Release Notes

Changelog