grigoryvp / vscode-language-xi Public

Код Проблем 15 Запросов на извлечение Обсуждения Действия Проектов Безопасность

мастер vscode-language-xi / README.md Перейдите к файлу

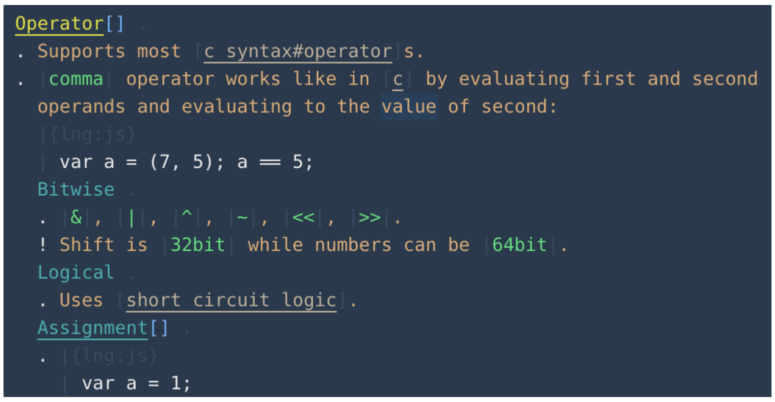
grigoryvpъ 4 месяца назад

213 lines (163 loc) · 10.3 KB

Предварительный просмотр Код Виноват

# Xi персональный вики-язык

"Xi" - это язык, подобный markdown, разработанный для персональной базы знаний. Цвет и отступ используются для выделения заголовков, абзацев, ссылок, примеров исходного кода и определений. Вы можете читать и записывать Xi из VSCode без необходимости "отображать" его в виде "визуальной" веб-страницы или PDF. Это делает Xi очень **быстрым** инструментом для ведения записей: вы что-то редактируете, просматриваете страницы Xi, добавляете или изменяете некоторый текст в том же редакторе, без необходимости переключаться между режимами "редактировать" и "предварительный просмотр". Пример Xi страницы из моей личной базы знаний с цветовой темой Memory:



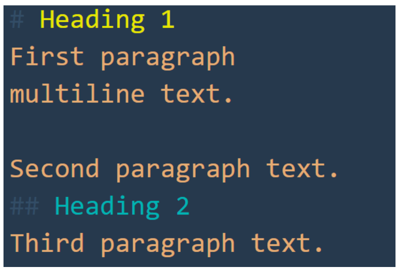
```
Operator[] .
. Supports most [c syntax#operator]s.
. |comma| operator works like in [c] by evaluating first and second
  operands and evaluating to the value of second:
  |{lng:js}
  | var a = (7, 5); a == 5;
  Bitwise .
  . |&|, |||, |^|, |~|, |<<|, |>>|.
  ! Shift is |32bit| while numbers can be |64bit|.
  Logical .
  . Uses [short circuit logic].
  Assignment[] .
  . |{lng:js}
    | var a = 1;
```

## VSCode extension installation guide

This extension is published to the "Extension Marketplace". To install it, click on the "Extensions" icon in the Activity Bar on the side of VSCode and enter "Xi markup language" in the search field. Click on the "install" button near the extension listed in the search results.

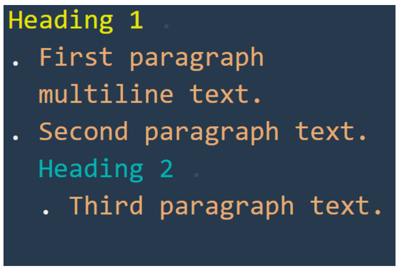
## General

The example below shows a simple markdown syntax with nested headings and paragraphs. With Xi, heading is marked with indentation and tail space-dot instead of indentation. So, heading 2 in markdown that is '## Heading 2' becomes ' Heading 2 .' in Xi (tail space-dot defines that it's a heading and leading two spaces defines second level. Third level will be 4 spaces and up to 10 spaces for a maximum nesting level 5). Paragraphs are marked with dot-space, effectively saving one empty line used in markdown:



```
# Heading 1
First paragraph
multiline text.

Second paragraph text.
## Heading 2
Third paragraph text.
```



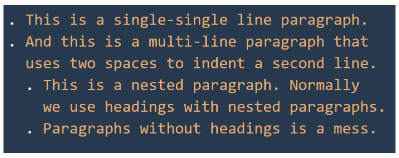
```
Heading 1 .
. First paragraph
  multiline text.
. Second paragraph text.
  Heading 2 .
  . Third paragraph text.
```

The extension provides a single command **extension.xi.lookup** that is binded to the **ctrl+k x** by default. This command brings file search for the directory with xi files, which is **~/.xi** by default and can be changed via the **xi.lookupPath** configuration option. It assumes workflow where if you are in any VSCode project and want to consult your personal knowledge base or add something to it, you hit "ctrl+k x", enter part of file name and open corresponding file for read and update. For example, writing some Python source code you want to remind yourself about API nuances. You know, that you store programming languages API records in files named "language_api.xi", so you hit "ctrl+k x", enter "python_a" that narrows search to "python_api.xi", hit enter, open file, check your notes and return to your code.

Wikiwords and links are marked as VSCode "links" and can be opened via standard VSCode means, for example the **editor.action.openLink** keybinding.

## Paragraph

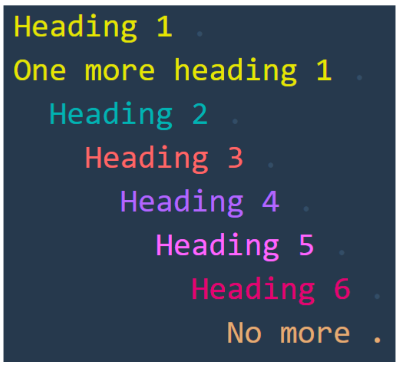
Simplest building block of a knowledge base, first paragraph line is prefixed with dot-and-space, while each next line is prefixed with two spaces. Paragraph itself can be indented with increments of two spaces in order to nest under different heading or other paragraphs. Compared to markdown notion of separating paragraphs with empty line, such layout saves a lot of vertical space, which is vital for documenting software-related things like programming lanugae syntax, APIs, frameworks etc. We tend to have short paragraphs that do not tend to follow each other, but tend to form a really complex nested strucure with lots of headings, meta information, code samples etc. Such paragrapah layout, used alongside colored headings and proper indentation, proved it's worth during my 10 years of using Xi as a personal knowledge base.



```
. This is a single-single line paragraph.
. And this is a multi-line paragraph that
  uses two spaces to indent a second line.
  . This is a nested paragraph. Normally
    we use headings with nested paragraphs.
  . Paragraphs without headings is a mess.
```

## Heading

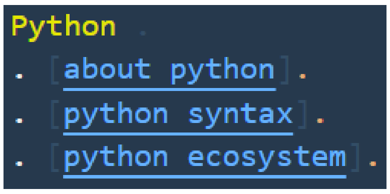
Alongside paragraph, heading is a second most used building block for a knowledge base. Heading type is denoted by space-dot suffix, while heading nesting level is denoted by increments of two spaces. In most cases headings are combined with paragraphs in order to create hierarchical structures that are easy to read **and modify**.



```
Heading 1 .
One more heading 1 .
  Heading 2 .
    Heading 3 .
      Heading 4 .
        Heading 5 .
          Heading 6 .
            No more .
```

## Links

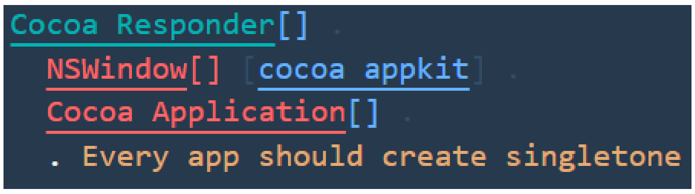
Here Xi starts to differenciate from Markdown. Simple "wikiword" link looks like one in Markdown, `[foo]` , but clicking on such link will try to open corresponding file, so everything works locally within VSCode. File name is created by lowercasing text between square braces, replacing spaces with underscores and appending `.xi` suffix. So clicking on the `[foo]` wikiword will instruct VSCode to open `foo.xi` in same folder, or ask to create one if it does not exist.



```
Python .
. [about python].
. [python syntax].
. [python ecosystem].
```

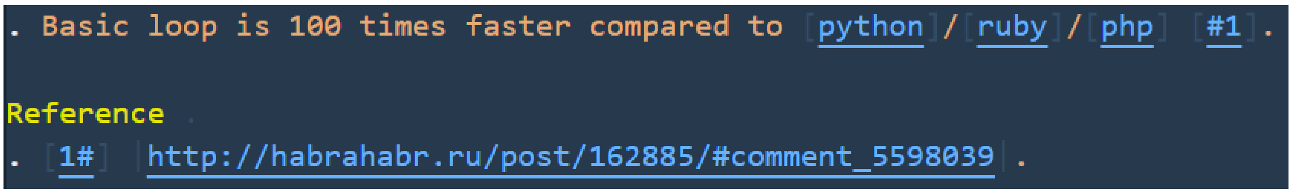
Wikiword links with anchors extend this concept by allowing to add heading name after `#` . Clicking on the like like `[js api#search]` will try to open `js_api.xi` file and scroll to the first `search` hading (that starts with spaces and ends with space-dot). Nested anchors are supported by chaining multiple '#' like `[js api#* String#- search]` .

Header links transforms a header into link by adding `[] .` instead of space-and-dot. Mostly useful while describing APIs or frameworks with complex tree structures where some headers points to articles of their own. Header can also start and/or end with wikiword links.



```
Cocoa Responder[] .
  NSWindow[] [cocoa appkit] .
  Cocoa Application[] .
    . Every app should create singletone
```

Anchor links starts with `#` and jumps within same document (mostly used for "Reference" section at bottom). Anchor target is specified by placing `#` as a last symbol of a wikiword link, so clicking on the link `[#1]` will jump into a line with `[1#]` anchor.



```
. Basic loop is 100 times faster compared to [python]/[ruby]/[php] [#1].

Reference .
. [1#] [http://habrahabr.ru/post/162885/#comment_5598039] .
```

## Text highlight

Xi supports a number of ways to highligh important parts of text. Pipes are used to highlight "special text" which are terms, API method names etc. Backticks are used to highlight something important with bright color. Please note that if memory color theme is used, highlight syntax is dimmed:

```
. We `highlight` part of text and |Term| within it.
```

## Code samples

Xi was created with the main goal to keep all my notes about different programming languages, framework and APIs I use. It is tailored to represent code samples in multiple ways. Most common is a "block code sample" which is a paragraph (dot-space prefix) that starts with a pipe character, followed by optional meta information inside curly braces. Meta syntax is a list of key-value pairs, a key separated from a value with the colon character and pairs separated from each other with the semicolon character. I use the 'lng' key to denote a programming language type, with a file extension as a value. So the block code sample for a Python programming language will start with `. |{lng:py}` , which will be dimmed out by a syntax highlighter. Each line of the sample is prefixed with a pipe, indented to match the position of the pipe in the `|{lng:py}` .

```
. |{lng:py;file:foo.py}
  | print("paragraph")
```

Code sample can be a paragraph of it's own or appear as a part of paragraph with text:

```
. Paragraph with a
  |{lng:py;file:foo.py}
  | print("code sample")
```

Code sample can also be inlined into a paragraph text by starting code right after `{}` meta inormation and terminating code with a pipe. for multiline code paragraph the optional meta information line should not contain any symbols after the meta:

```
. Inline |{lng:py}"code"| sample.
```

## Named parameters

This Xi feature is more specialized for programming languages. While taking notes on APIs it's often required to reference method parameters inside of a method description. While this can be done via the text highlight feature, doing so will not distinguish parameters among other highlighted words. Specialized syntax allows to include named parameter between curled braces. In addition, char-space prefix can be added after opening curled brace to mark input or output parameters. It's also possible to include parameter type wihin round braces after a mark character: all with syntax highlighting for a visual navigation:

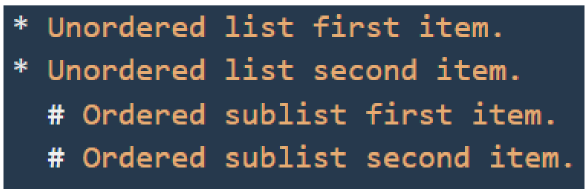
```
. {param-name} is not a |term|.
. Input {i named_param}.
. Output typed {o(str) param}.
```

Usage example that describes an "insert" method that takes two named and positional parameters, "indice" and "item".

```
- insert .
. After specified {i indice} inserts specified {i item}.
```

## Lists

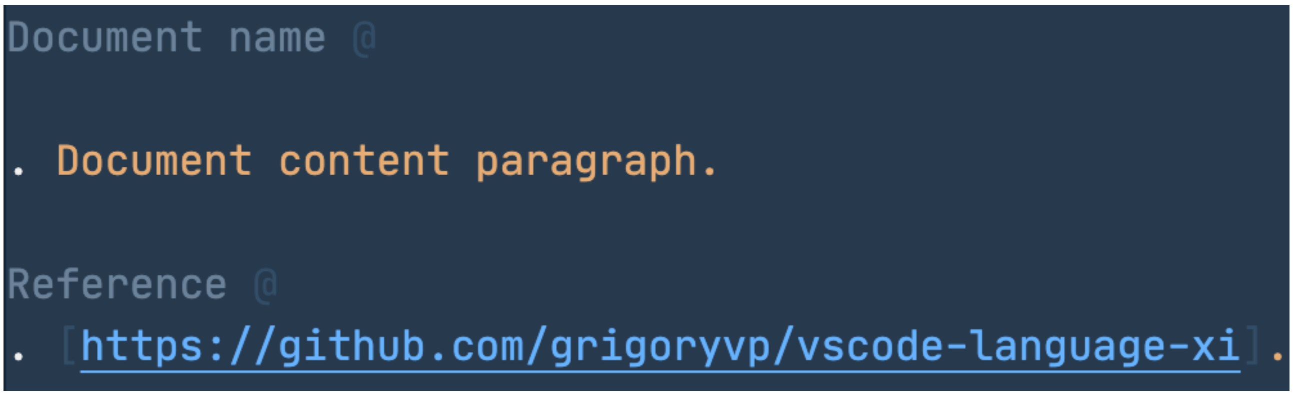
Like in Markdown, lists in Xi are denoted with `#` , `*` or `-` characters. The syntax highlighting behaviour is the same as with paragraphs (that start with the `.` character).



```
* Unordered list first item.
* Unordered list second item.
  # Ordered sublist first item.
  # Ordered sublist second item.
```

## Meta heading

Sometimes a document contains some headers that denote meta information about the document. Two common examples of such headers are document name at top and the "reference" section at bottom. For them a special coloring is provided if they are terminated with the "@" character instead of the dot:



```
Document name @

. Document content paragraph.

Reference @
. [https://github.com/grigoryvp/vscode-language-xi].
```

## Examples

- Lua syntax
- Neurophysiology article

## Todo

- Re-take all screenshots.
- Outline API support.

## License