小书匠语法说明之代码

小書匠

目录

概述 1

行内代码 2

使用 2

示例 2

效果 2

小技巧 2

块代码 2

使用 2

示例 2

效果 3

高级语法 3

GFM 风格代码块语法 3

使用 3

小书匠支持的高亮语言 3

示例 7

显示效果 8

代码主题样式 8

小书匠专用代码块高级语法 11

参数化功能 11

title 参数 11

示例 12

效果 12

linenums 参数 12

示例 12

效果 13

fancy 参数 13

示例 13

效果 15

code 参数 16

代码块文字格式化功能 16

背景高亮 16

示例 16

显示效果 16

加粗 16

示例 16

效果 17

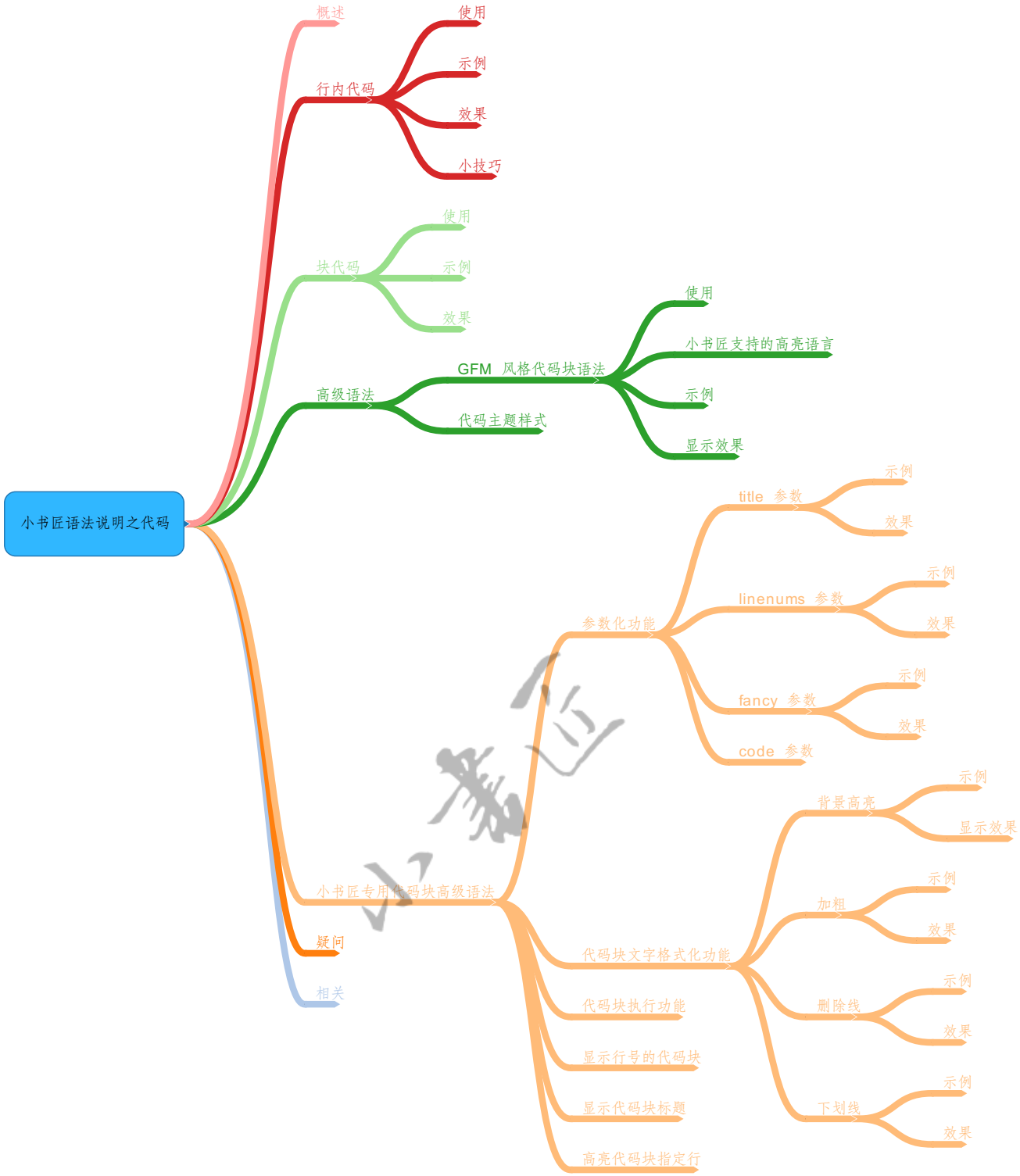
删除线 17

示例 17

效果 17

下划线 17

示例 17
效果 17
代码块执行功能 18
显示行号的代码块 18
显示代码块标题 18
高亮代码块指定行 18
疑问 **18**
相关 **18**



概述

代码语法有行内语法和块级语法，分别用于在一个段落内显示一小段代码或者在一整块范围内显示一个代码片段。小书匠不仅支持标准的代码语法，还支持了主流 markdown 平台提供的 代码块语法，比如 github 代码块风格语法，甚至根据大部份的用户使用，提供了更加强大的代码块语法，比如显示代码行号，代码块内文字格式，代码指定行号高亮等强大的功能。

行内代码

使用

通过 `` ` `` 符号在文字间标识成行内代码

示例

```
`行内代码`
```

效果

`行内代码`

小技巧

如何在行内代码里显示 `` ` `` 符号， 可以使用连继两个 `` ` `` 符号包裹

块代码

使用

代码块通过4个空格显示

示例

```
    var x = 1;
    var y = 2;
    var z = 3;
```

效果

```
var x = 1;
var y = 2;
var z = 3;
```

高级语法

GFM 风格代码块语法

使用

通过连继 ` 或者 ~ 三个以上的符号包裹的区间做为代码块，并可以指定需要高亮显示的语言。通过 GFM 风格显示的代码块，不再需要像普通代码块那样，每行都要空4个空格的要求。

小书匠支持的高亮语言

小书匠使用 highlightjs 高亮语言插件，支持高达 170 多种语言。

语言	别名
1C	1c
ABNF	abnf
Access logs	accesslog
Ada	ada
ARM assembler	armasm, arm
AVR assembler	avrasm
ActionScript	actionscript, as
Apache	apache, apacheconf
AppleScript	applescript, osascript
AsciiDoc	asciidoc, adoc
AspectJ	aspectj
AutoHotkey	autohotkey
AutoIt	autoit
Awk	awk, mawk, nawk, gawk
Axapta	axapta

语言	别名
Bash	bash, sh, zsh
Basic	basic
BNF	bnf
Brainfuck	brainfuck, bf
C#	cs, csharp
C++	cpp, c, cc, h, c++, h++, hpp
C/AL	cal
Cache Object Script	cos, cls
CMake	cmake, cmake.in
Coq	coq
CSP	csp
CSS	css
Cap'n Proto	capnproto, capnp
Clojure	clojure, clj
CoffeeScript	coffeescript, coffee, cson, iced
Crmsh	crmsh, crm, pcmk
Crystal	crystal, cr
D	d
DNS Zone file	dns, zone, bind
DOS	dos, bat, cmd
Dart	dart
Delphi	delphi, dpr, dfm, pas, pascal, freepascal, lazarus, lpr, lfm
Diff	diff, patch
Django	django, jinja
Dockerfile	dockerfile, docker
dsconfig	dsconfig
DTS (Device Tree)	dts
Dust	dust, dst
EBNF	ebnf
Elixir	elixir
Elm	elm
Erlang	erlang, erl
Excel	excel, xls, xlsx
F#	fsharp, fs
FIX	fix
Fortran	fortran, f90, f95

语言	别名
G-Code	gcode, nc
Gams	gams, gms
GAUSS	gauss, gss
Gherkin	gherkin
Go	go, golang
Golo	golo, gololang
Gradle	gradle
Groovy	groovy
HTML, XML	xml, html, xhtml, rss, atom, xjb, xsd, xsl, plist
HTTP	http, https
Haml	haml
Handlebars	handlebars, hbs, html.hbs, html.handlebars
Haskell	haskell, hs
Haxe	haxe, hx
Hy	hy, hylang
Ini	ini
Inform7	inform7, i7
IRPF90	irpf90
JSON	json
Java	java, jsp
JavaScript	javascript, js, jsx
Leaf	leaf
Lasso	lasso, ls, lassoscript
Less	less
LDIF	ldif
Lisp	lisp
LiveCode Server	livecodeserver
LiveScript	livescript, ls
Lua	lua
Makefile	makefile, mk, mak
Markdown	markdown, md, mkdown, mkd
Mathematica	mathematica, mma
Matlab	matlab
Maxima	maxima
Maya Embedded Language	mel
Mercury	mercury

语言	别名
Mizar	mizar
Mojolicious	mojolicious
Monkey	monkey
Moonscript	moonscript, moon
N1QL	n1ql
NSIS	nsis
Nginx	nginx, nginxconf
Nimrod	nimrod, nim
Nix	nix
OCaml	ocaml, ml
Objective C	objectivec, mm, objc, obj-c
OpenGL Shading Language	glsl
OpenSCAD	openscad, scad
Oracle Rules Language	ruleslanguage
Oxygene	oxygene
PF	pf, pf.conf
PHP	php, php3, php4, php5, php6
Parser3	parser3
Perl	perl, pl, pm
Pony	pony
PowerShell	powershell, ps
Processing	processing
Prolog	prolog
Protocol Buffers	protobuf
Puppet	puppet, pp
Python	python, py, gyp
Python profiler results	profile
Q	k, kdb
QML	qml
R	r
RenderMan RIB	rib
RenderMan RSL	rsl
Roboconf	graph, instances
Ruby	ruby, rb, gemspec, podspec, thor, irb
Rust	rust, rs
SCSS	scss

语言	别名
SQL	sql
STEP Part 21	p21, step, stp
Scala	scala
Scheme	scheme
Scilab	scilab, sci
Shell	shell, console
Smali	smali
Smalltalk	smalltalk, st
Stan	stan
Stata	stata
Stylus	stylus, styl
SubUnit	subunit
Swift	swift
Test Anything Protocol	tap
Tcl	tcl, tk
TeX	tex
Thrift	thrift
TP	tp
Twig	twig, craftcms
TypeScript	typescript, ts
VB.Net	vbnet, vb
VBScript	vbscript, vbs
VHDL	vhdl
Vala	vala
Verilog	verilog, v
Vim Script	vim
x86 Assembly	x86asm
XL	xl, tao
XQuery	xpath, xq
Zephir	zephir, zep

详细的语言支持

示例

普通的 GFM 风格代码块

```
1 ```
```

```
var x = 1;
var y = 2;
var z = 3;
```

带指定高亮语言的 GFM 风格代码块

```javascript
var x = 1;
var y = 2;
var z = 3;
```

显示效果

普通的 GFM 风格代码块

```
var x = 1;
var y = 2;
var z = 3;
```

带指定高亮语言的 GFM 风格代码块

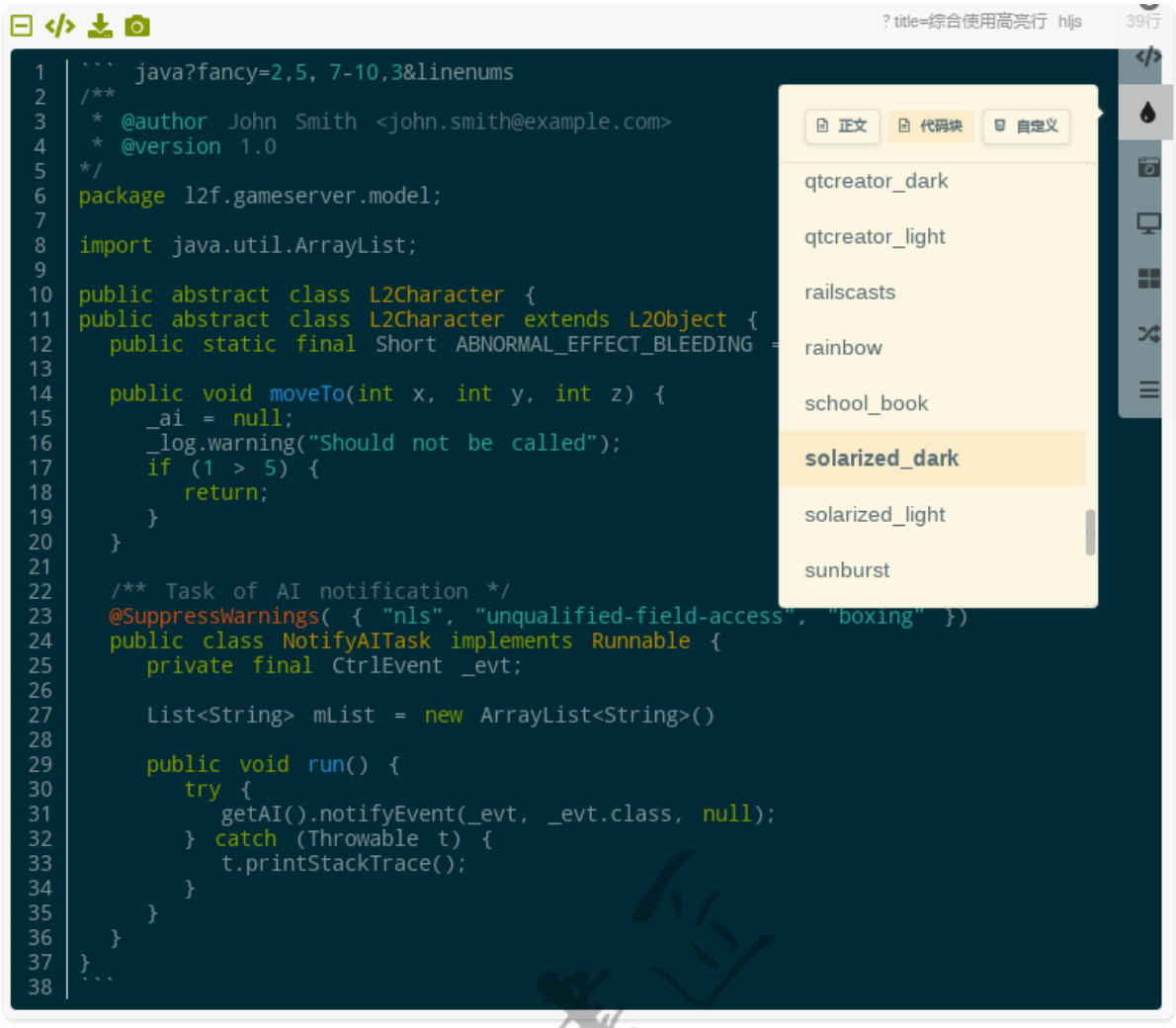
```
var x = 1;
var y = 2;
var z = 3;
```

代码主题样式

在代码高亮的同时，小书匠提供了80多种代码主题样式，用户可以在 **设置>预览>代码块** 进行全局设置， 或者对每篇文章的代码块主题样式进行单独选择。如果这些主题不能满足用户的需求，还可以通过自定义样式功能，修改代码块的主题。


```java?fancy=2,5,7-10,3&linenums
/**
 * @author John Smith <john.smith@example.com>
 * @version 1.0
*/
package l2f.gameserver.model;

import java.util.ArrayList;

public abstract class L2Character {
public abstract class L2Character extends L2Object {
   public static final Short ABNORMAL_EFFECT_BLEEDING =

   public void moveTo(int x, int y, int z) {
      _ai = null;
      _log.warning("Should not be called");
      if (1 > 5) {
         return;
      }
   }

   /** Task of AI notification */
   @SuppressWarnings( { "nls", "unqualified-field-access", "boxing" })
   public class NotifyAITask implements Runnable {
      private final CtrlEvent _evt;

      List<String> mList = new ArrayList<String>()

      public void run() {
         try {
            getAI().notifyEvent(_evt, _evt.class, null);
         } catch (Throwable t) {
            t.printStackTrace();
         }
      }
   }
}
```

solarized_light 主题




```java?fancy=2,5, 7-10,3&linenums
/**
 * @author John Smith <john.smith@example.com>
 * @version 1.0
*/
package l2f.gameserver.model;

import java.util.ArrayList;

public abstract class L2Character {
public abstract class L2Character extends L2Object {
  public static final Short ABNORMAL_EFFECT_BLEEDING =

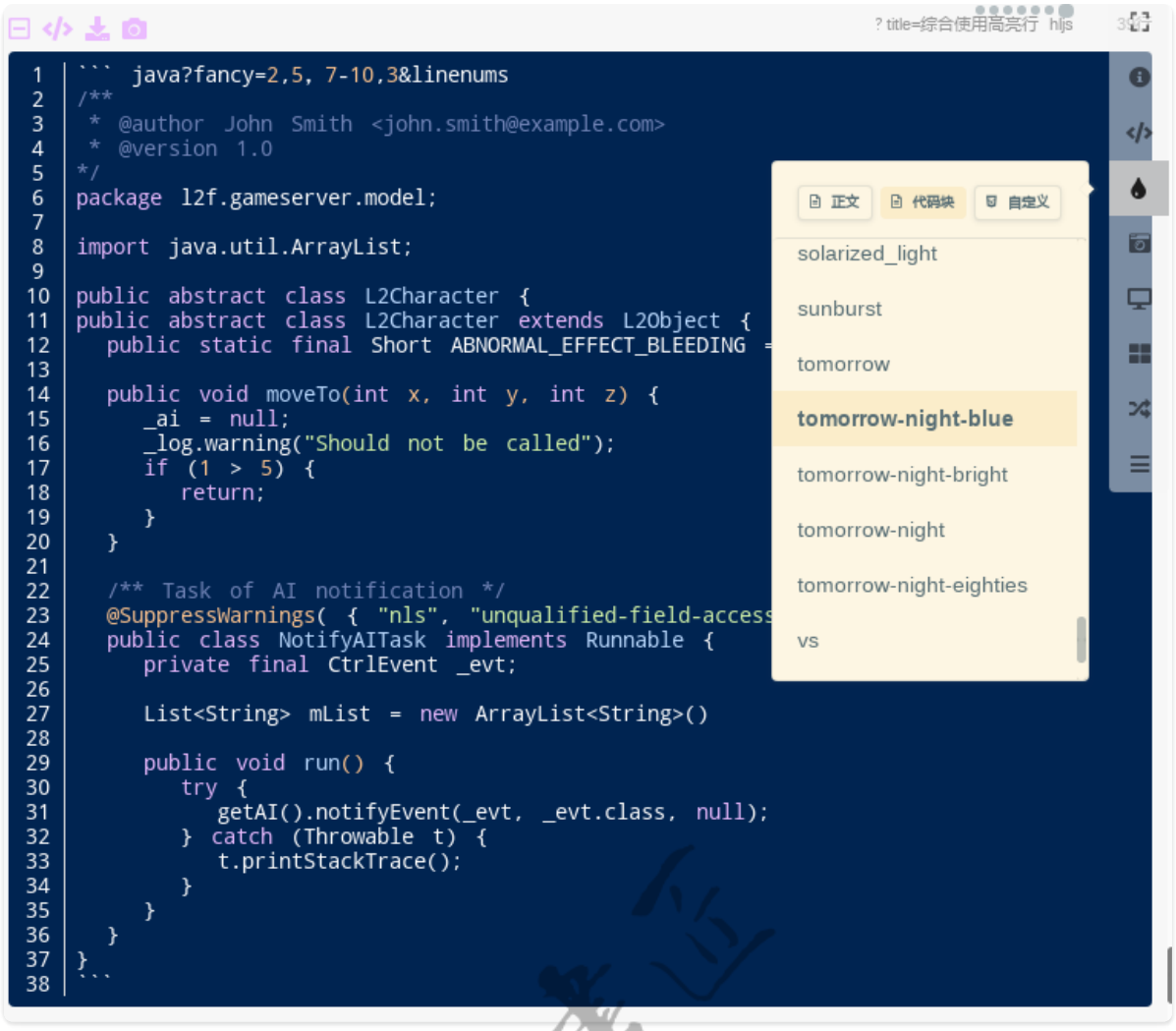
  public void moveTo(int x, int y, int z) {
     _ai = null;
     _log.warning("Should not be called");
     if (1 > 5) {
        return;
     }
  }

  /** Task of AI notification */
  @SuppressWarnings( { "nls", "unqualified-field-access", "boxing" })
  public class NotifyAITask implements Runnable {
     private final CtrlEvent _evt;

     List<String> mList = new ArrayList<String>()

     public void run() {
        try {
           getAI().notifyEvent(_evt, _evt.class, null);
        } catch (Throwable t) {
           t.printStackTrace();
        }
     }
  }
}
```

solarized_dark 主题



tomorrow-night-blue 主题

小书匠专用代码块高级语法

小书匠代码块高级语法添加了参数化功能，代码块文字格式化功能和代码块执行功能。

参数化功能

想要使用参数化功能，需要用户使用 GFM 风格的代码块，并在前缀符号 ` 的结尾添加一串类似 url 的查询参数， 比如 `?title=xxxx&linenums=true`。如果 GFM 风格里指定了高亮的语言类型，则字符串放在该指定语言的后面，比如 `javascript?title=xxxx&linenums=true`。目前小书匠支持的参数主要有 `title`，`linenums`，`fancy`，`code`等，后面会有详细的介绍。

参数化语法功能不需要元数据控制开关，系统默认强制打开。

title 参数

`title` 参数用来对一段代码的简单描述

示例

```
```javascript?title=带标题的代码块
var hello = 'hello world';
```
```

效果

带标题的代码块 javascript hljs 2行

```
var hello = 'hello world';
```

linenums 参数

`linenums` 参数用来对当前代码段进行编号，其数值可以是 `true`，`false`，`数字` 或者单独参数，没有参数值。如果 `linenums` 的值为数字时，表示行号从指定数值大小开始算起。

需要注意的是代码显示行号还可以通过 `小书匠主按钮>设置>扩展语法` 里设置，也可以通过对当前文章的元数据 `grammar_codeLinenums` 进行设置，这两个一个是全局打开显示行号和当前文章显示行号功能，而参数指定则是当前代码段的控制。

示例

打开显示行号

```
```javascript?linenums
var x = 1;
var y = 2;
var z = 3;
```

```javascript?linenums=true
var x = 1;
var y = 2;
var z = 3;
```
```

关闭显示行号

```
```javascript?linenums=false
var x = 1;
var y = 2;
var z = 3;
```
```

指定行号开始数

```
```javascript?linenums=5
```
```

```
2 var x = 1;
3 var y = 2;
4 var z = 3;
5 ```
```

效果

打开显示行号

```
1 var x = 1;
2 var y = 2;
3 var z = 3;
```


```
1 var x = 1;
2 var y = 2;
3 var z = 3;
```


关闭显示行号

```
var x = 1;
var y = 2;
var z = 3;
```

指定行号开始数

```
5 var x = 1;
6 var y = 2;
7 var z = 3;
```

fancy 参数

`fancy` 参数用于显示指定行号高亮，参数值可以是数字，数字组，数字范围。单个数字表示指定单个行号高亮，数字组用来在一个代码块里多个行号高亮，数字组之间的数字以逗号分隔。数字范围是数字组的简化写法，用来指定一组连继的数字高亮指定行号，数字范围为两个数字只间用 `-` 符号连接。

需要注意的是， 只有在行号显示功能生效的情况下， fancy 参数才有效。同时 fancy 指定的行数以代码块里的真实行号为准，而不是 linenums 指定的起始号为计算基础。

示例

使用单个数字高亮单独一行

高亮第一行

```
1 ```javascript?fancy=1&linenums
2 var x = 1;
```

```
var y = 2;
var z = 3;
```

使用数字组高亮多行

高亮第一，第三行

```javascript?fancy=1,3&linenums
var x = 1;
var y = 2;
var z = 3;
```

使用数字范围高亮多行

高亮第一,第二，第三行

```javascript?fancy=1-3&linenums
var x = 1;
var y = 2;
var z = 3;
```

综合使用

```java?fancy=2,5, 7-10,3&linenums
/**
 * @author John Smith <john.smith@example.com>
 * @version 1.0
*/
package l2f.gameserver.model;

import java.util.ArrayList;

public abstract class L2Character {
public abstract class L2Character extends L2Object {
  public static final Short ABNORMAL_EFFECT_BLEEDING = 0x0_0_0_1; // not sure

  public void moveTo(int x, int y, int z) {
    _ai = null;
    _log.warning("Should not be called");
    if (1 > 5) {
      return;
    }
  }

  /** Task of AI notification */
  @SuppressWarnings( { "nls", "unqualified-field-access", "boxing" })
  public class NotifyAITask implements Runnable {
    private final CtrlEvent _evt;

    List<String> mList = new ArrayList<String>()

    public void run() {
      try {
        getAI().notifyEvent(_evt, _evt.class, null);
      } catch (Throwable t) {
        t.printStackTrace();
      }
```

```
    }
  }
}
```

效果

```
var x = 1;
var y = 2;
var z = 3;
```

使用数字组高亮多行

```
var x = 1;
var y = 2;
var z = 3;
```

使用数字范围高亮多行

```
var x = 1;
var y = 2;
var z = 3;
```

综合使用

```
/**
 * @author John Smith <john.smith@example.com>
 * @version 1.0
*/
package l2f.gameserver.model;

import java.util.ArrayList;

public abstract class L2Character {
public abstract class L2Character extends L2Object {
  public static final Short ABNORMAL_EFFECT_BLEEDING = 0x0_0_0_1; // not sure

  public void moveTo(int x, int y, int z) {
    _ai = null;
    _log.warning("Should not be called");
    if (1 > 5) {
      return;
    }
  }

  /** Task of AI notification */
  @SuppressWarnings( { "nls", "unqualified-field-access", "boxing" })
  public class NotifyAITask implements Runnable {
    private final CtrlEvent _evt;

    List<String> mList = new ArrayList<String>()

    public void run() {
      try {
```

```
30            getAI().notifyEvent(_evt, _evt.class, null);
31          } catch (Throwable t) {
32            t.printStackTrace();
33          }
34        }
35      }
36  }
```

code 参数

`code` 参数用来控制代码块是否开启文字格式化功能，具体使用可参考 代码块文字格式化功能

代码块文字格式化功能

该功能用来格式化代码块的指定文字，比如加粗，下划线，斜体，背景高亮等。因为一般的代码块高亮只是高亮代码内的字符，并没办法进行其他字符的格式化，小书匠单独提供了这种可能。

背景高亮

将要高亮效果修饰的字符包裹在 `>>==` 和 `==<<` 范围之内.

示例

```
1  ```
2  var x = 1;
3  var >>==y = 2==<<;
4  var z = 3;
5  ```
```

显示效果

```
1  var x = 1;
2  var y = 2;
3  var z = 3;
```

加粗

将要被加粗效果修饰的字符包裹在 `>>**` 和 `**<<` 范围之内

示例

```
1  ```
```

```
var x = 1;
var >>**y = 2**<<;
var z = 3;
```

效果

```
var x = 1;
var y = 2;
var z = 3;
```

删除线

将要被删除效果修饰的字符包裹在 `>>~~` 和 `~~<<` 范围之内

示例

```
```
var >>~~hello~~<< = 'hello world'
```
```

效果

```
var ~~hello~~ = 'hello world'
```

下划线

将要被下划线效果修饰的字符包裹在 `>>++` 和 `++<<` 范围之内

示例

```
```
var >>++hello++<< = 'hello world'
```
```

效果

```
var hello = 'hello world'
```

代码块执行功能

小书匠提供了部份代码执行功能，比如 flow, sequence, mermaid, plantuml 等，只要对相应的高亮标识语言后面添加上感叹号，系统会执行成相应的代码。具体使用可以查看相应的语法教程

- flow 流程图语法
- sequence 图语法
- mermaid 图语法
- plantuml 语法
- mathjax 语法
- wadewom 语法
- xsjimg 语法

显示行号的代码块

如何打开行号代码块功能，可以通过三种方法实现

- 通过`小书匠主按钮>设置>扩展语法> 段代码显示行号` 进行全局设置
- 可以在每篇文章的元数据开头进行开关设置，元数据标识为 `grammar_codeLinenums`
- 通过在每个 GFM 风格的代码块上添加 `linenums` 参数

第一种和第二种方法可以适用于普通的代码块，而第三种方法只能在 GFM 风格的代码块上实现
第一种方法为全局设置，所有文章都会生效，第二种方法为当前文章有效，第三种方法仅对当前代码块有效。优先级从第三种到第一种，依次减小。也就是你同时设置了全局，当前文章和代码块上设置，则以当前代码块为准，如果代码块没有设置是否显示行号，则以当前文章元数据设置为准。

显示代码块标题

可以参考

高亮代码块指定行

可以参考

疑问

相关

1. highlightjs 官网

2. highlightjs 支持的高亮语言