# vscode--Git Commit规范

**没有规矩不成方圆**规范化开发以及合理的库管理一直是我们所追求的~在了解规范之前，我们先来了解下git中tag的使用方法和应用场景；

### 一、前置知识（关于tag的使用）

#### 1.1 概念

tag即标签,git版本库的一个标记,指向某个commit的指针。一个库里面可以有很多分支，一个分支下可以有很多标签。tag主要用来发布版本的管理。一共有两种类型：1、轻量级标记；2、注释型标签。

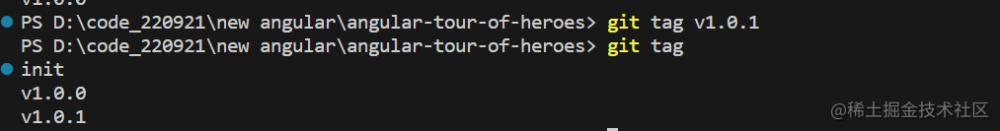
#### 1.2 用法

##### 1.2.1 新建&&查看

**sketch**

//1.新建tag  
 git tag <new tagName>  
 eg:  
 git tag v1.0.1  
 //2.查看所有的tag  
 git tag

**demo**

for 1&&2 

##### 1.2.2 筛选commit

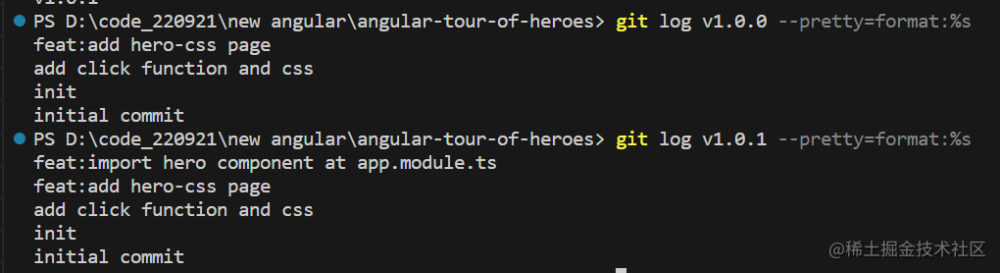
%s指的提交说明;其他想要了解看这里--> [Git format 格式](https://git-scm.com/docs/pretty-formats)

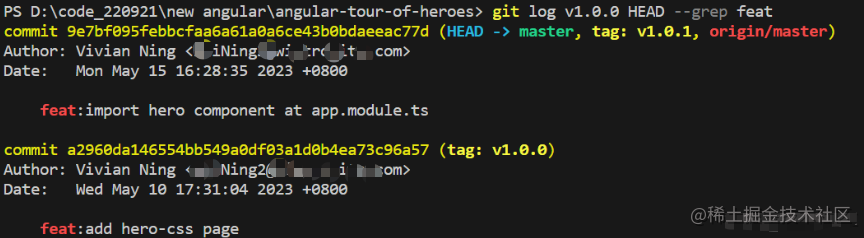
**sketch**

//3.查看指定tag中的所有commit  
 git log <tagName> --pretty=format:%s   
 eg:  
 git log v1.0.0 --pretty=format:%s   
  
 //4.查看最后一个tag中commit开头为feat的  
 git log <last tag> HEAD --grep feat  
 eg:git log v1.0.1 HEAD --grep feat

**demo**

for 3:

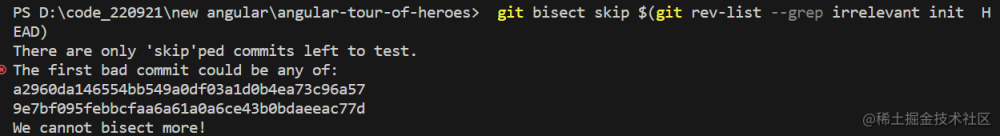
 for 4:



##### 1.2.3 忽略commit

**sketch**

//5.使用二分法时，可忽略部分历史commit  
 git bisect skip $(git rev-list --grep irrelevant <good place> HEAD)  
 eg:  
 git bisect skip $(git rev-list --grep irrelevant init HEAD)  
 //6.启动git的二分法  
 git bisect start



### 1.2 commit 规范

#### 1.2.1 何为规范？

看看下面这些这些commit message 示例（来自Angular项目）：

* Fix small typo in docs widget (tutorial instructions)

* Fix test for scenario.Application - should remove old iframe

* docs - various doc fixes

* docs - stripping extra new lines

* Replaced double line break with single when text is fetched from Google

* Added support for properties in documentation

这些 messages 尝试说明提交更改的内容，但是它们没有遵循任何约定... 又比如：

* fix comment stripping

* fixing broken links

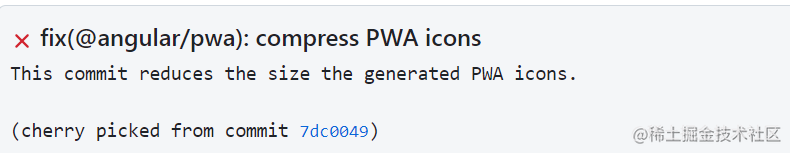
* Bit of refactoring

* Check whether links do exist and throw exception

* Fix sitemap include (to work on case sensitive linux)

从以上这些提交信息来看，去推敲当时到底改了哪部分非常费劲，虽然你可以通过查看被修改的文件来找到这些信息，但是这就比较花时间了。再查看git 历史的时候，是希望所有人都明确是什么地方发生修改，所以应该统一一下约束。

我们可以先看一个规范的例子，如下图：



**Format of the commit message**

<type>(<scope>): <subject>  
 <BLANK LINE>  
 <body>  
 <BLANK LINE>  
 <footer>

即

<类型>[可选的作用域]: <描述>  
  
 [可选的正文] --- 具体描述  
  
 [可选的脚注] --- 标注处理的是哪个issue

#### 1.3 type——类型

|  |  |
| --- | --- |
| 类型 | 详细介绍 |
| feat | 添加新功能 |
| fix | 修复bug |
| build | 构造工具或者外部依赖的改动，例如webpack,npm |
| docs | 更新文档，像readme文件的变动 |
| refactor | 代码重构的时候使用 |
| style | 不影响代码含义的改动，例如去掉空格、改变缩进、增删分号 |
| chore | bump version to [版本号]；构建过程或辅助工具的变动，例如package.json里面依赖的版本更新 |
| test | 添加测试或者修改现有的测试 |
| perf | 提高性能的改动 |

#### 1.4 scope——作用域

commit影响的范围 比如：route、component、utils、build.....。

如果没有更合适的范围，您可以使用\*。

#### 1.5 subject——描述

这是对代码修改的一个简短的描述。

写英文commit的时候

1. 使用祈使句，一般现在时：“change”不是“changed”也不是“changes”;

1. 不要将第一个字母大写;

1. 末尾没有点（.）。

#### 1.6 Message——主体

1. 就像在***subject***中使用祈使现在时一样:" change "不是" changed "也不是" changes "；

1. 包括改变的动机和与之前行为的对比

such as

Rather than having a commit with the title:*Renamed the iVars and removed the common prefix.*

Have one like this:*Rename the iVars to remove the common prefix.*

善用介词 ，就像of/to/for...

#### 1.7 Message——页脚

主要分为两部分：**Breaking changes**&&**Referencing issues**（突发变化以及引用的问题）；

##### 1.7.1 Breaking changes

所有的突发性变化都必须在页脚的break change块中提到，该块应该以break change:开头，用空格或两个换行符。提交消息的其余部分是对更改的描述、论证和迁移注释。such as:

BREAKING CHANGE: isolate scope bindings definition has changed and   
the inject option for the directive controller injection was removed.   
To migrate the code follow the example below:   
 Before:   
 scope: {   
 myAttr: 'attribute',   
 myBind: 'bind',   
 myExpression: 'expression',   
 myEval: 'evaluate',   
 myAccessor: 'accessor'   
 }   
 After:   
 scope: {   
 myAttr: '@',   
 myBind: '@',   
 myExpression: '&',  
 // myEval - usually not useful,  
 // but in cases where the expression is assignable, you can use '='   
 myAccessor: '='   
 // in directive's template change myAccessor() to myAccessor   
 }  
 The removed `inject` wasn't generaly useful for directives so there should be no code using it.

##### 1.7.2 Referencing issues

已关闭的issues应该在页脚的单独一行中列出，并以“close”关键字为前缀，如下所示

Closes #234

或在多个问题的情况下:

Closes #123, #245, #992

### 1.3 git commit message 例子

feat($browser): onUrlChange event (popstate/hashchange/polling)   
  
Added new event to $browser:  
 - forward popstate event if available  
 - forward hashchange event if popstate not available  
 - do polling when neither popstate nor hashchange available   
   
Breaks $browser.onHashChange, which was removed (use onUrlChange instead)

fix($compile): couple of unit tests for IE9   
  
Older IEs serialize html uppercased, but IE9 does not...  
Would be better to expect case insensitive, unfortunately jasmine does  
not allow to user regexps for throw expectations.   
  
Closes #392Breaks foo.bar api, foo.baz should be used instead

feat(directive): ng:disabled, ng:checked, ng:multiple, ng:readonly, ng:selected   
  
New directives for proper binding these attributes in older browsers (IE).  
Added coresponding description, live examples and e2e tests.   
  
Closes #351

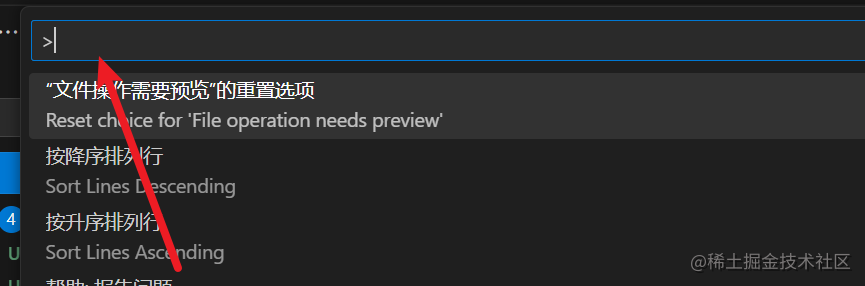
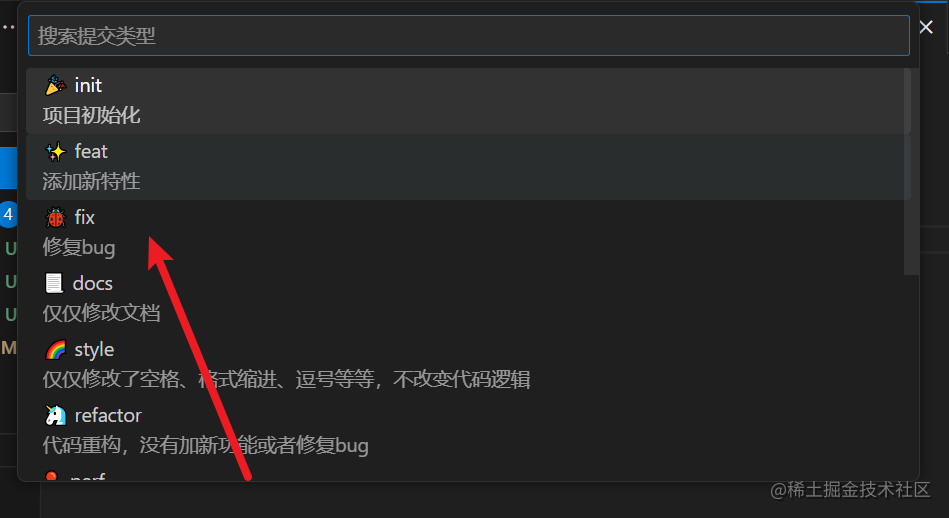
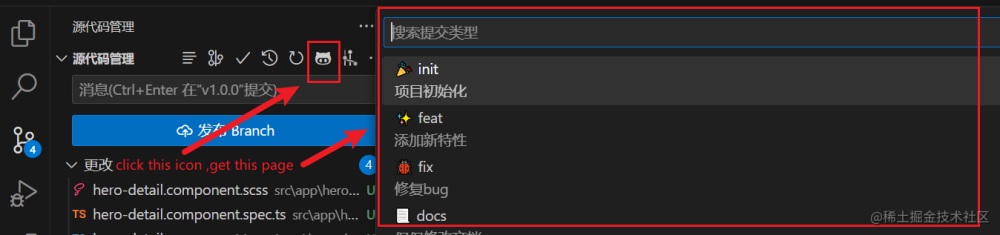
style($location): add couple of missing semi colons

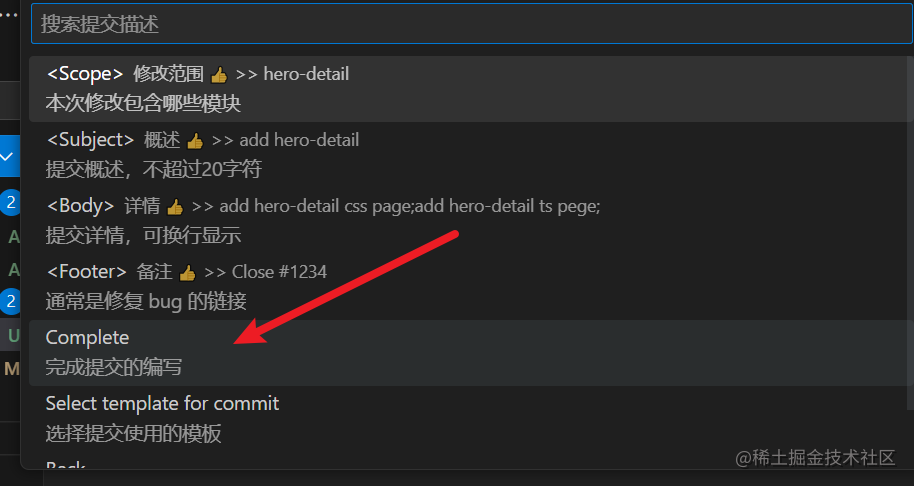
docs(guide): updated fixed docs from Google Docs   
  
Couple of typos fixed:  
 - indentation  
 - batchLogbatchLog -> batchLog  
 - start periodic checking  
 - missing brace

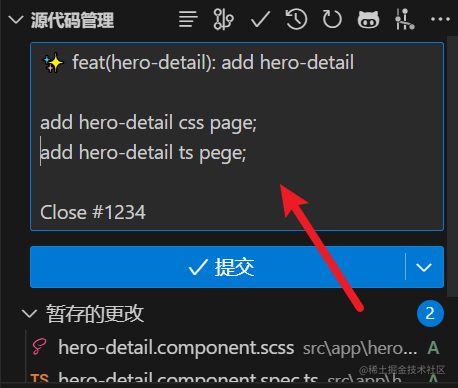
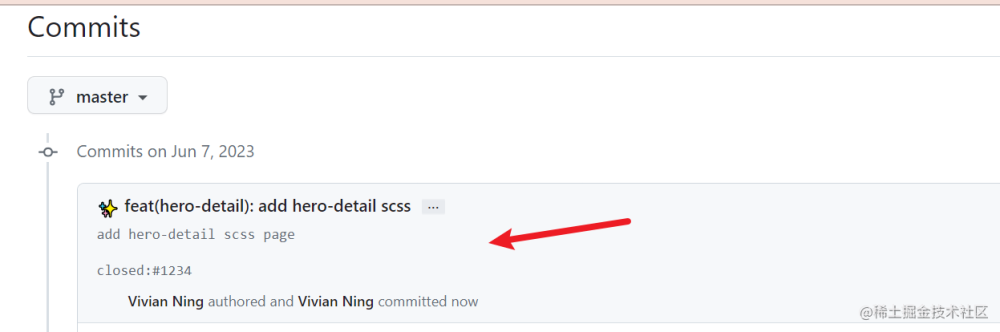
### 1.4 插件

以vscode为例，主要介绍这个**git-commit-plugin**插件，commit提交模板直接调用即可。

* type 1 下载插件 

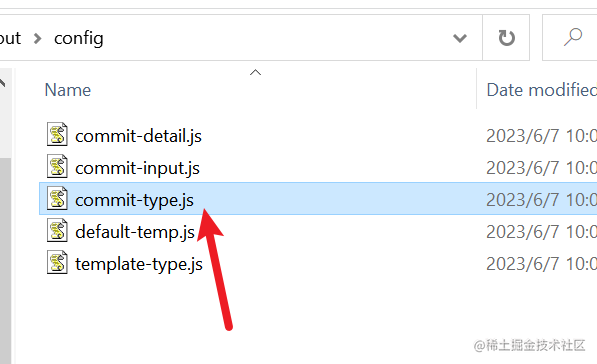
* type 2 下载成功之后，直接用快捷键找到：Control + Shift + P打开，输入show git commit template用快捷键Control + Shift + P打开如下图：  输入 show git commit template 回车如下图  或者 *type 2* 还可以点击小图标进入上述插件页面 

* type 3 填写具体信息，点击complete 

* type 4 点击complete 可得到下图 点击提交，可以在代码库中看到提交的history

*注：下载插件之后可以修改文件配置修改，更改commit提交的提示信息内容*

C:\Users\.vscode\extensions\redjue.git-commit-plugin-1.1.2\out\config  
//以上路径取决于你本地的安装路径

 内容如下：

let CommitType = [  
 {  
 label: 'init',  
 key: 'init',  
 detail: vscode\_nls\_i18n\_1.localize('extension.commitType.init.detail'),  
 icon: '🎉',  
 },  
 {  
 label: 'feat',  
 key: 'feat',  
 detail: vscode\_nls\_i18n\_1.localize('extension.commitType.feat.detail'),  
 icon: '✨',  
 },  
 {  
 label: 'fix',  
 key: 'fix',  
 detail: vscode\_nls\_i18n\_1.localize('extension.commitType.fix.detail'),  
 icon: '🐞',  
 },  
 {  
 label: 'docs',  
 key: 'docs',  
 detail: vscode\_nls\_i18n\_1.localize('extension.commitType.docs.detail'),  
 icon: '📃',  
 },  
 {  
 label: 'style',  
 key: 'style',  
 detail: vscode\_nls\_i18n\_1.localize('extension.commitType.style.detail'),  
 icon: '🌈',  
 },  
 {  
 label: 'refactor',  
 key: 'refactor',  
 detail: vscode\_nls\_i18n\_1.localize('extension.commitType.refactor.detail'),  
 icon: '🦄',  
 },  
 {  
 label: 'perf',  
 key: 'perf',  
 detail: vscode\_nls\_i18n\_1.localize('extension.commitType.perf.detail'),  
 icon: '🎈',  
 },  
 {  
 label: 'test',  
 key: 'test',  
 detail: vscode\_nls\_i18n\_1.localize('extension.commitType.test.detail'),  
 icon: '🧪',  
 },  
 {  
 label: 'build',  
 key: 'build',  
 detail: vscode\_nls\_i18n\_1.localize('extension.commitType.build.detail'),  
 icon: '🔧',  
 },  
 {  
 label: 'ci',  
 key: 'ci',  
 detail: vscode\_nls\_i18n\_1.localize('extension.commitType.ci.detail'),  
 icon: '🐎',  
 },  
 {  
 label: 'chore',  
 key: 'chore',  
 detail: vscode\_nls\_i18n\_1.localize('extension.commitType.chore.detail'),  
 icon: '🐳',  
 },  
 {  
 label: 'revert',  
 key: 'revert',  
 detail: vscode\_nls\_i18n\_1.localize('extension.commitType.revert.detail'),  
 icon: '↩',  
 },  
 ];

Tips:插件还有很多，感兴趣的自行了解

that's all ~

*个人整理，实属不易，望君珍惜！*

#### 参考文献

[Git Tag](https://www.atlassian.com/git/tutorials/inspecting-a-repository/git-tag)

[Git Commit Message Conventions](https://docs.google.com/document/d/1QrDFcIiPjSLDn3EL15IJygNPiHORgU1_OOAqWjiDU5Y/edit#)

[vscode插件-git-commit-plugin使用](https://blog.csdn.net/weixin_41056807/article/details/127202536)