目　录

1.Markdown常用语法

比较好的教程：https://doc.gsw945.com/docs/mindoc-docs/markdown-insert-html.md

MinDoc 的前身是 SmartWiki 文档系统。SmartWiki 是基于 PHP 框架 laravel 开发的一款文档管理系统。因 PHP 的部署对普通用户来说太复杂，所以改用 Golang 开发。可以方便用户部署和实用。

开发缘起是公司IT部门需要一款简单实用的项目接口文档管理和分享的系统。其功能和界面源于 kancloud 。

可以用来储存日常接口文档，数据库字典，手册说明等文档。内置项目管理，用户管理，权限管理等功能，能够满足大部分中小团队的文档管理需求。

我常用的语法

1 字体 颜色 背景色

Size：规定文本的尺寸大小。可能的值：从 1 到 7 的数字。浏览器默认值是 3。

```
<font face="STCAIYUN">我是华文彩云</font>
<font color=gray size=5>color=gray</font>
<font color=#0099ff size=5 face="黑体">color=#0099ff size=5 face="黑体"</font>
```

效果：
我是华文彩云
color=gray
color=#0099ff size=5 face=”黑体”

背景色

```
<table><tr><td bgcolor=#FF4500>这里的背景色是：OrangeRed， 十六进制颜色值：#FF4500， rgb(255, 69, 0)</td></tr></table>
```

效果：

这里的背景色是：OrangeRed， 十六进制颜色值：#FF4500， rgb(255, 69, 0)

呈现效果如下
这里的背景色是：OrangeRed， 十六进制颜色值：#FF4500， rgb(255, 69, 0)

2 列表

无序列表
语法：
无序列表用 - + * 任何一种都可以

- 列表内容
- 列表内容
- 列表内容

注意：- + * 跟内容之间都要有一个空格

效果如下：

有序列表
语法：
数字加点

1. 列表内容
2. 列表内容
3. 列表内容

注意：序号跟内容之间要有空格

用 `- [ ]` 来做未完成的计划列表，用 `- [x]` 作为完成的；下级在前面加 `Tab键` 即可。

- 历史记录再计算
- 小程序
  - 历史查询
  - 充值

3 表格

语法：

```
表头|表头|表头
--|:--:|---:
内容|内容|内容
内容|内容|内容
```

第二行分割表头和内容。

- 有一个就行，为了对齐，多加了几个
  文字默认居左
  -两边加：表示文字居中
  -右边加：表示文字居右
  注：原生的语法两边都要用 | 包起来。此处省略

示例：

```
这里要空一行
姓名|技能|排行
--:|:--:|--:
刘备|哭|大哥
关羽|打|二哥
张飞|骂|三弟
```

效果：注意前后要空行才行

姓名	技能	排行
刘备	哭	大哥
关羽	打	二哥
张飞	骂	三弟

4 流程图

```
```flow
st=>start: Start
op=>operation: Your Operation
sub=>subroutine: My Subroutine
cond=>condition: Yes or No?
io=>inputoutput: catch something...|current
e=>end: End

st->op->cond
cond(yes)->io->e
```
```

```
cond(no)->sub(right)->op
```

效果：
注意，要在上文的 `flow` 前加3个 ` 符号；

在最后面也加3个 ` 符号

st=>start: Start op=>operation: Your Operation sub=>subroutine: My Subroutine cond=>condition: Yes or No? io=>inputoutput: catch something...|current e=>end: End st->op->cond cond(yes)->io->e cond(no)->sub(right)->op

st=>start: index
op=>operation: 申请
op2=>operation: 结果页
op3=>operation: 查询本地
i1=>inputoutput: bid入库
i2=>inputoutput: 填写个人信息
c1=>condition: 检查登录
c2=>condition: 登录
c3=>condition: 查询本地记录
c4=>condition: 检测状态
c5=>operation: 风控审核
e=>end

st->op->c1()
c1(no)->c2(yes)->op()
c1(yes)->c3(no)->i1(right)->i2(right)->c5()->op2->e
c1(yes)->c3(yes)->c4(no)->i2
c1(yes)->c3(yes)->c4(yes)->op3->op2
c3()->e

5 顶部加上导航目录

语法

```
[TOC]
//这里要空一行
```

效果：

6 插入图片大小和位置

```
<img src="/uploads/md-001/images/m_53becc6e57206cc754de10e55294400d_r.png" width="100" align=right title="猫咪的图片-鼠标提示">
```

7 多级导航目录

理论上 MinDoc支持无限级的文档创建。
在 MinDoc 中没有目录概念，所有文档既是目录也是文档。默认情况下，点击左上角加号即可创建顶级文档：
如果想创建一个文档的下一级项目，可直接在项目上右键，即可弹出创建菜单：
同样，还可以通过右键菜单修改以及删除文档。

8 修改默认的封面

MinDoc 在创建项目时会通过浏览器的H5绘图接口生成一个默认的封面。
用户还可以在项目设置中修改项目封面：

第一步 打开我的项目

第二步 点击设置按钮

第三步 切换到项目设置

点击右侧项目图片，即可上传自定义封面。

其他见网络搜索结果
https://blog.csdn.net/qq_35164554/article/details/120013432
https://markdown.com.cn/basic-syntax/

9 下划线

```
<u>Honor </u>
```

<u>Honor</u>

实现下标与上标的三种方法：

方法1（符号）：
下标：θ~1~
上标：θ^2^

结果如下：
下标：θ1
上标：θ2

方法2（HTML标签）：

下标：θ_1
上标：θ^2

结果如下:
下标：θ1
上标：θ2

方法3（公式块）：

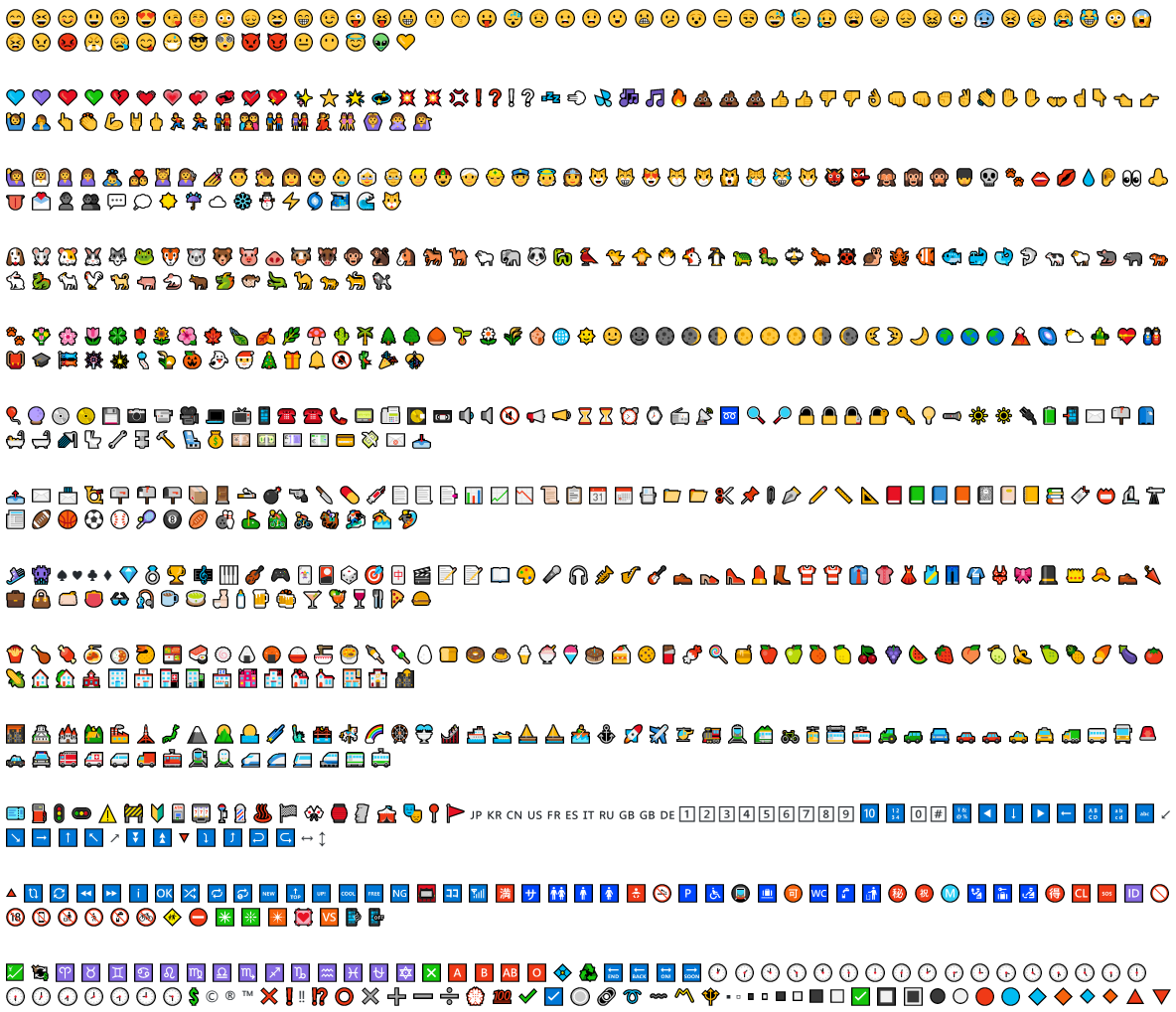
结果如下:
下 标： θ 1 ， 上 标： θ 2 下标：θ_1 ，上标：θ^2
下标：θ
，上标：θ

2.揭秘例子中的表达式写法:

θ1x1+θ2x2^2^+θ3x3^3^

结果：θ1x1+θ2x22+θ3x33

2.图标



3.思维导图（脑图mindmap）

支持3种思维导图方式
mindmap、mermaid和flow

1 mindmap

3个重音符号 ` ，波浪键上那个反单引号。
> 后面的数字是高度

```
```mindmap>400
# 水务院
## 水工
### 水工一室
### 水工二室
## 施工
### 施工一室
### 施工二室
## 预算
### 预算一室
### 预算二室
` ` `—这3个反单引号之间不要有空格，这里是为了显示需要，
加了空格，实际使用的时候不要加空格。
注意是键盘左上角波浪号下方那个反单引号
```

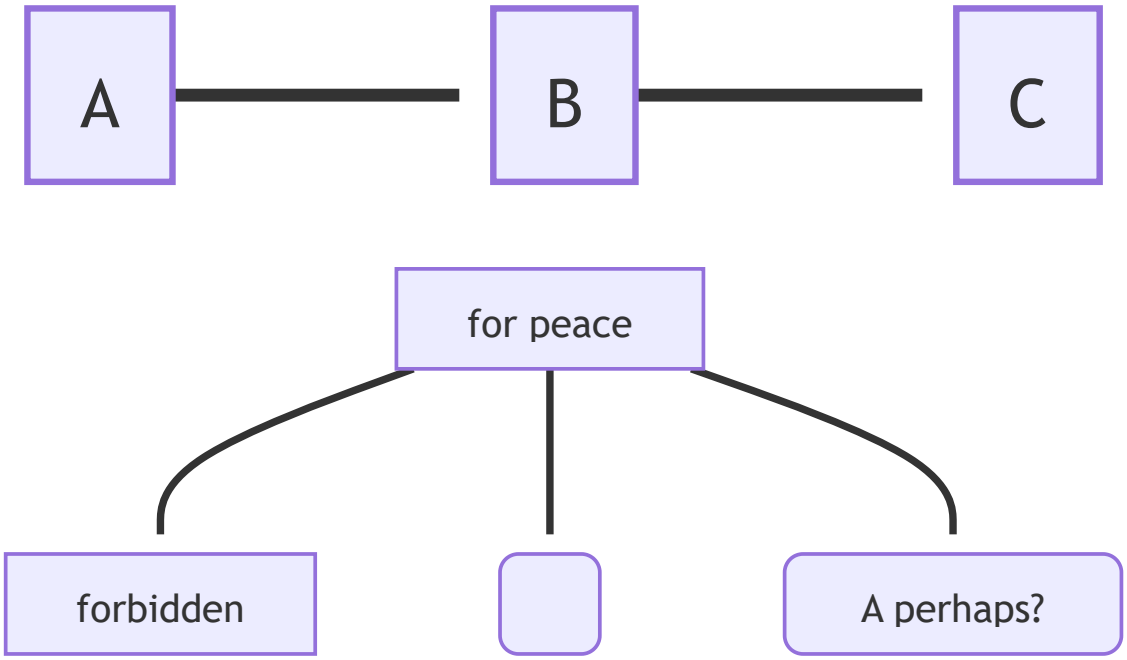
效果如下：

2 mermaid

https://doc.gsw945.com/docs/mindoc-docs
https://blog.csdn.net/wangyaninglm/article/details/52887045
https://mermaid-js.github.io/mermaid/#/

2.1 增加链接

```
```mermaid
graph LR
    A --> B
    B --> C
    click B "https://www.github.com"
` ` `
```



2.2 样式类

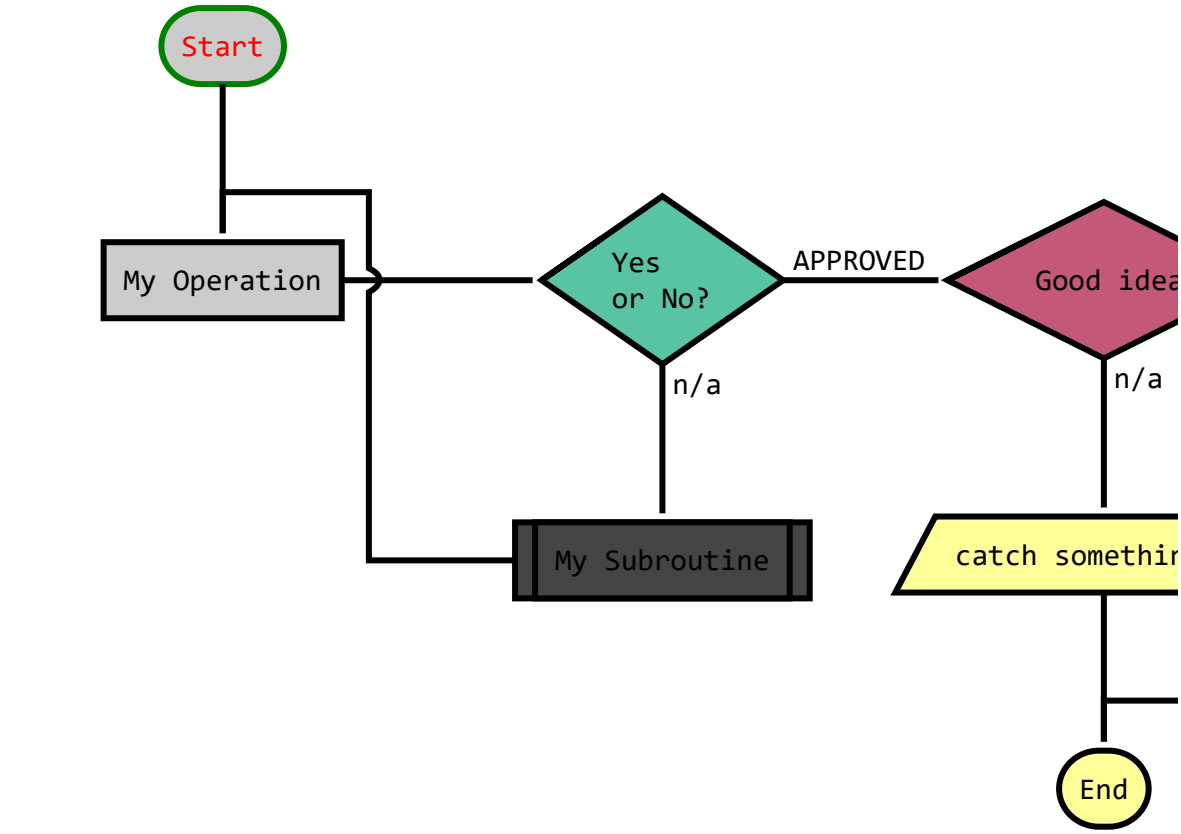
为了方便样式的使用，可以定义类来使用样式
类的定义示例：
classDef className fill:#f9f,stroke:#333,stroke-width:4px;
对节点使用样式类：
class nodeId className;
同时对多个节点使用相同的样式类：
class nodeId1,nodeId2 className;

3 带有色块的流程图

流程图是以 `flow ，开始并以` 结尾的区块中

```
st=>start: Start|past:>http://www.google.com[blank]
e=>end: End|future:>http://www.google.com
op1=>operation: My Operation|past
op2=>operation: Stuff|current
sub1=>subroutine: My Subroutine|invalid
cond=>condition: Yes
or No?|approved:>http://www.google.com
c2=>condition: Good idea|rejected
io=>inputoutput: catch something...|future

st->op1(right)->cond
cond(yes, right)->c2
cond(no)->sub1(left)->op1
c2(yes)->io->e
c2(no)->op2->e
```



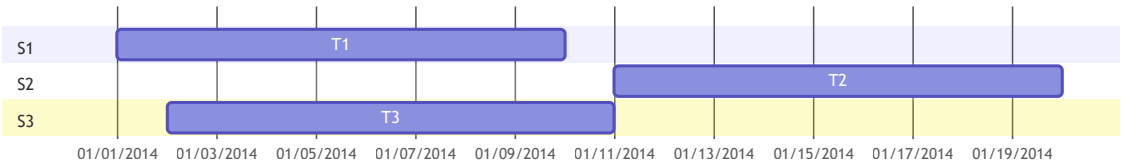
甘特图

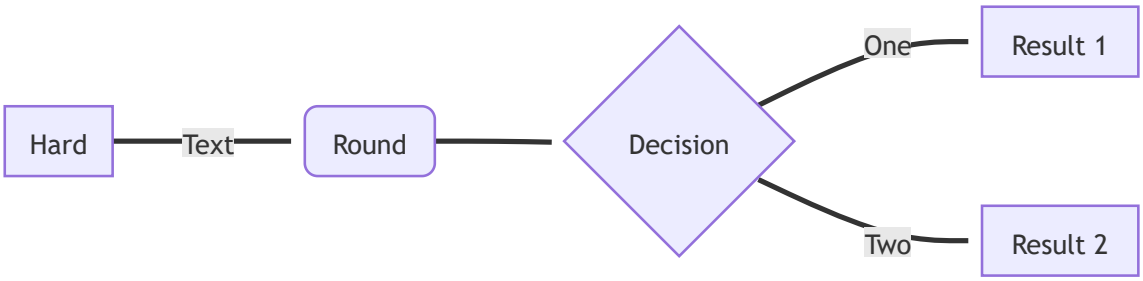
```
```mermaid
gantt
dateFormat YYYY-MM-DD

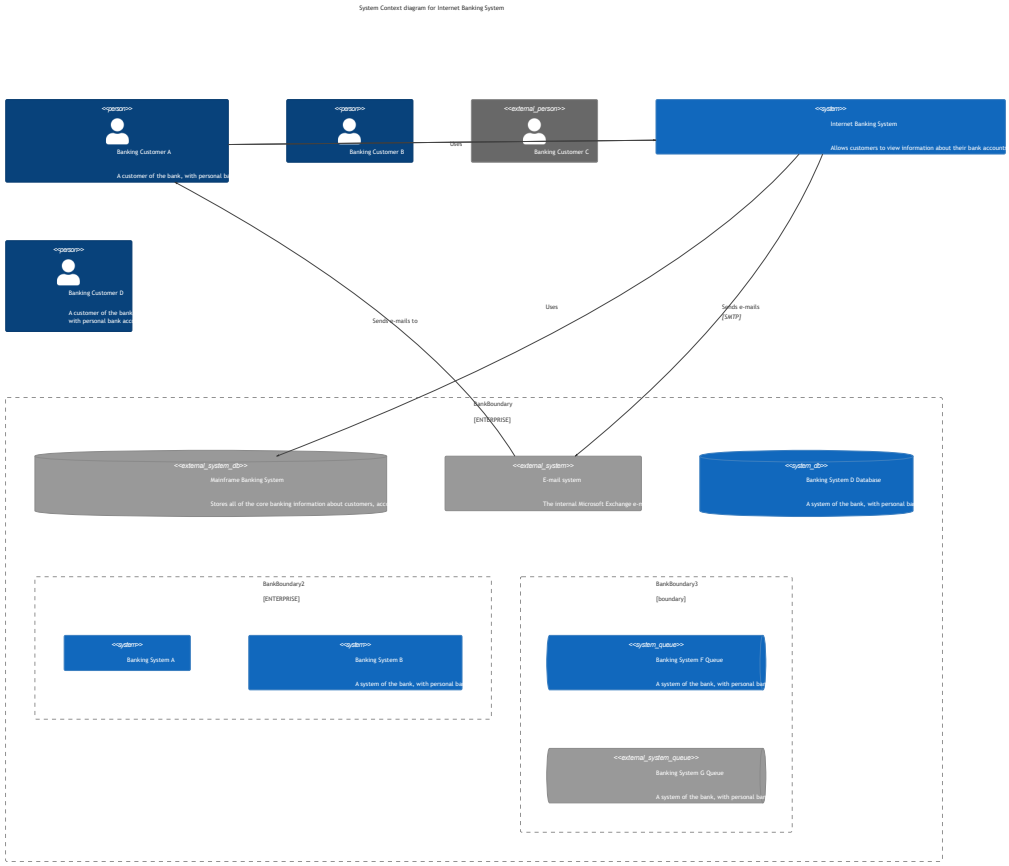
section S1
T1: 2014-01-01, 9d

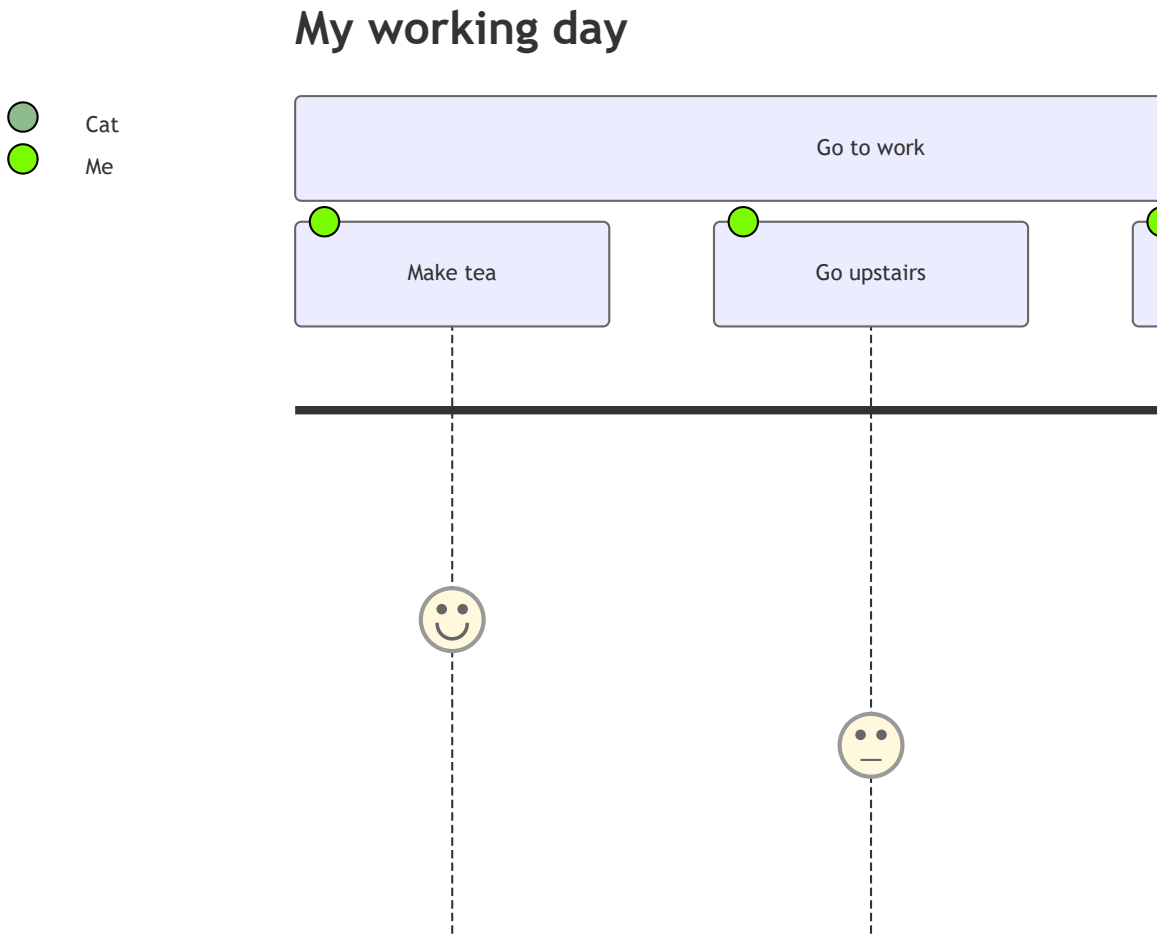
section S2
T2: 2014-01-11, 9d

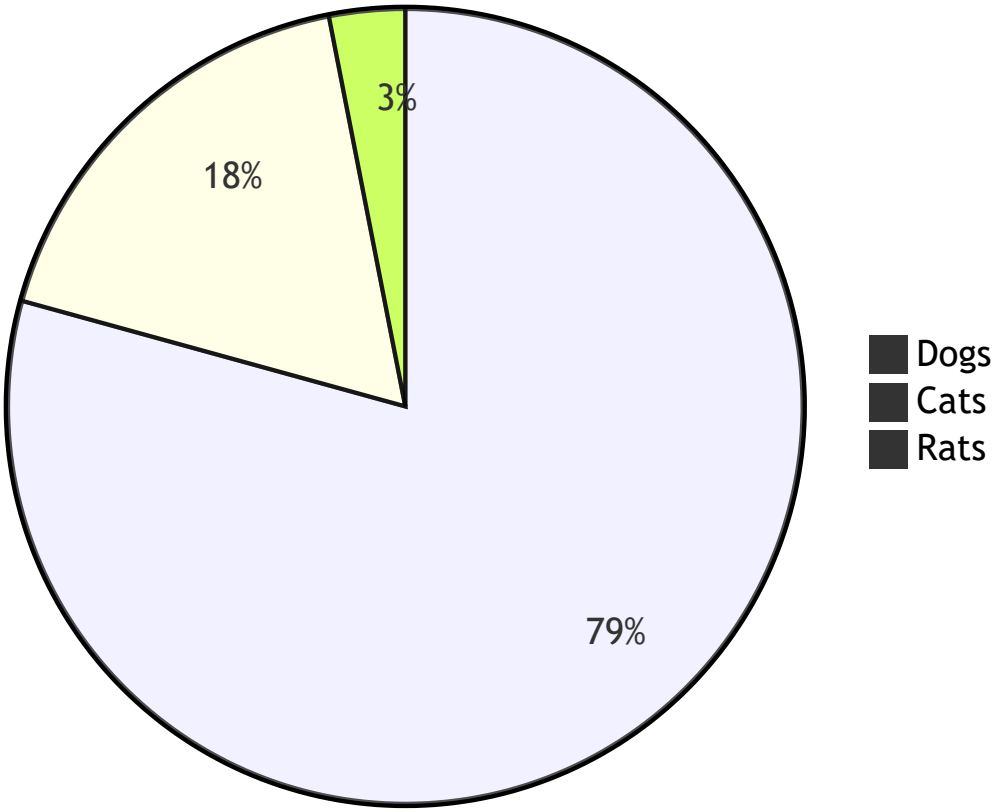
section S3
T3: 2014-01-02, 9d
```
```




Hard
Text
Round
Decision
One
Result 1
Two
Result 2


System Context diagram for Internet Banking System
<<person>>
Banking Customer A
<<person>>
Banking Customer B
<<external_person>>
Banking Customer C
<<system>>
Internet Banking System
Allows customers to view information about their bank account
<<person>>
Banking Customer D
Uses
Sends e-mails to
Sends e-mails
[SMTP]
BankBoundary
[ENTERPRISE]
<<external_system_db>>
Mainframe Banking System
<<external_system>>
E-mail system
<<system_db>>
Banking System D Database
BankBoundary2
[ENTERPRISE]
<<system>>
Banking System A
<<system>>
Banking System B
BankBoundary3
[boundary]
<<system_queue>>
Banking System F Queue
<<external_system_queue>>
Banking System G Queue


My working day
Cat
Me
Go to work
Make tea
Go upstairs


3%
18%
79%
Dogs
Cats
Rats

4 插入视频

mindoc插入影音的办法(markdown)
插入本地视频

1. 上传视频并获取地址
   在mindoc目录的uploads中新建一个video文件夹
   把相关视频复制进去
   以大图标方式查看，如果该文件无法显示片头图片，则格式编码兼容性有问题，很大可能无法在线播放，
   需进行转码处理。
   视频绝对网址为 http://网站/uploads/video/文件名
   这里的http://网站可省略,省略后的视频相对网址为:
   /uploads/video/文件名
2. 插入视频
   iframe方式

```
<iframe width="600px" height="400px" src=""/uploads/video/ZT400系列更换打印头.mp4" >
</iframe>
```

这里的" "号如果偷懒也可以省略，不影响实际显示。

```
<iframe width=600px height=400px src=/uploads/video/ZT400系列更换打印头.mp4 >
</iframe>
```

video方式

```
<video width="600px" height="400px" controls>
<source src="/uploads/video/ZT400系列更换打印头.mp4 " type="video/mp4">
</video>
<video width=600 height=400 controls>
<source src=/uploads/video/ZT400系列更换打印头.mp4>
</video>
```

插入其他资源

1. 插入本地gif动图

```
<iframe width="500px" height="270px"scrolling="no"
src="/uploads/video/test.gif">
</iframe>
```

2. 插入本地音频

```
<video width="80%" height="80px" controls>
<source src="/uploads/video/su.mp3 " >
</video>
```

样式如下:

3. 插入网络视频
b站，视频下方点分享然后复制嵌入代码
效果如下：
同样，也可以省略成这样: 注:生产环境为稳定起见最好别省略4。
插入pdf
在uploads下新建一个book文件夹，把pdf文件放进去
方法一：iframe方式

```
<iframe width="1000" height="1200" id="iframepdf"
src="/uploads/book/zm600快速安装.pdf">
</iframe>
```

显示效果：

方法二：object方式

```
<object width="1000" height="1200"
data="/uploads/book/zm600快速安装.pdf" type="application/pdf">
<div>No online PDF viewer installed</div>
</object>
```

作为一个知识库，势必有大批PDF格式的厂家说明书，总不能全部上传为附件，然后在网页上下载下来本地看吧？在线看是必然的需求。
显示效果:

No online PDF viewer installed

注：以上两种方式都需要chrome或edge浏览器。IE如果未装PDF的插件是无法在线看PDF的。

5. iframe的其他应用
插入天气预报等……
到天气预报的插件网站选一个款式
https://www.tianqi.com/plugin/
插入doc
代码

```
<iframe src='https://view.officeapps.live.com/op/embed.aspx?
src=https://pymo.github.io/PYMO教程.doc'
width='100%' height='565px' frameborder='0'>
</iframe>
```

由于iframe本质是插入一个外部网页的子窗口,所以你还可以...
其实用途不大，原理是使用微软的office官网api读取网站上的doc文件然后在线显示出来，故此内网无法使用，因为微软无法读取内网里的doc。如果实在是有DOC要在线看，建议还是转成PDF后插入。
src后替换为要访问的doc绝对网址显示效果:

备注：
新版的配置文件conf\app.conf默认是禁止iframe的
默认:enable_iframe = “${MINDOC_ENABLE_IFRAME||false}”
改为:enable_iframe = “${MINDOC_ENABLE_IFRAME||true}”
无论是iframe或video插入的资源都可通过width和height调整长宽
width=“600px” height=”400px” 宽600像素，高400像素,这里的px可省略
width=“80%” height=”20%” 宽为页面的80%，高为页面的20%
需将false改为true，否则插入iframe代码无效。

5 展示三维模型

Mindoc里展示三维模型

mindoc v2.1支持iframe标签
网页嵌入(iframe)：
教程: `HTML <iframe>` 标签

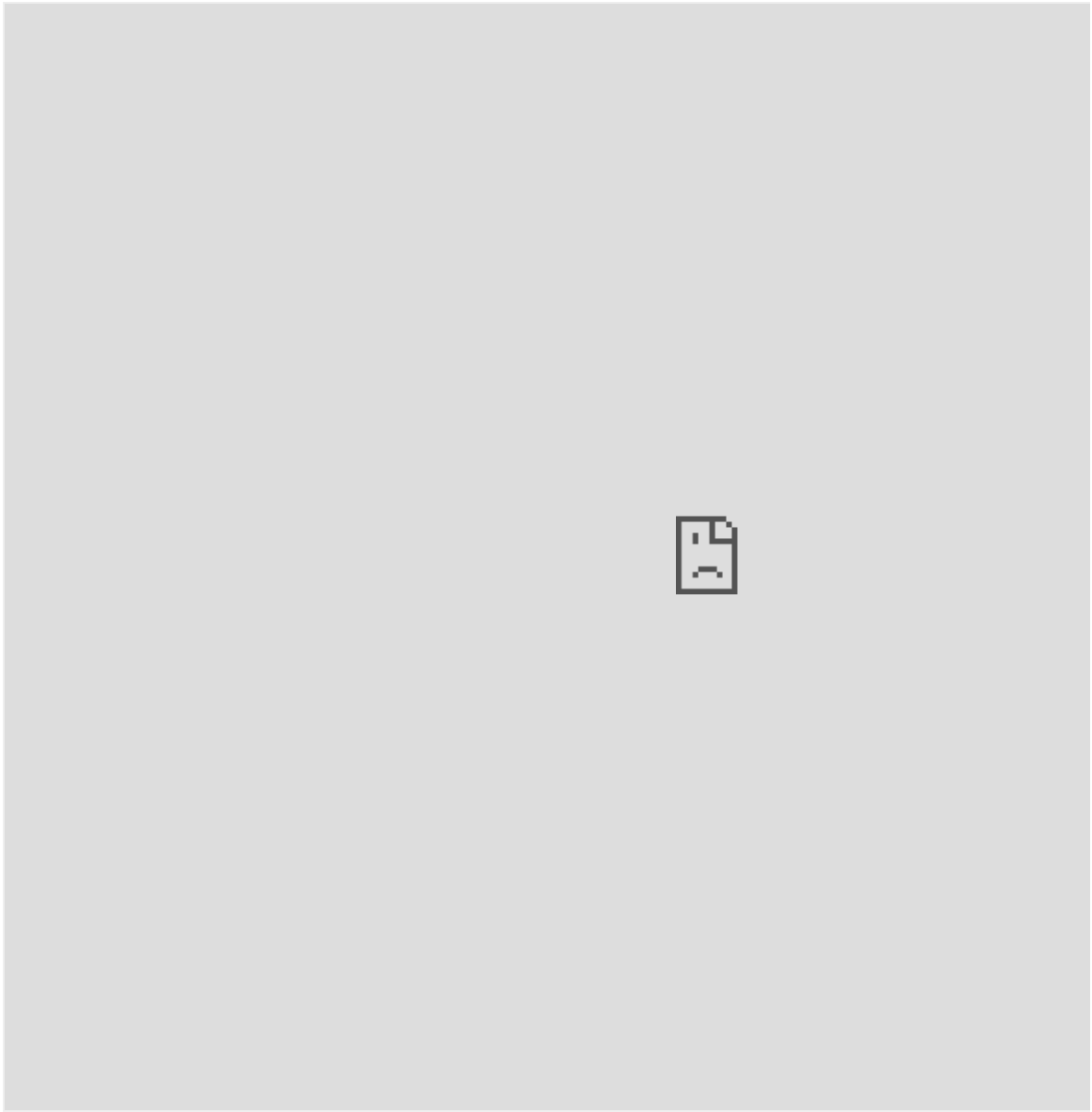
1.配置文件 conf/app.conf 需要启用iframe:

```
enable_iframe = "${MINDOC_ENABLE_IFRAME||true}"
```

2.markdown内容编辑里输入

```
<iframe src="https://pass.itdos.net/v1/freecad/online3dview#model=/static/online3dview/8.glb"
        width=800 height=600
        style="border:1px solid #eeeeee;">
</iframe>
```

当页面不显示时，添加: `is="x-frame-bypass"`
`</iframe>` 后面要空一行，否则会影响后面的文档格式



6 折叠内容

写法:

```
<details> <summary>点击展开</summary>

code ...这里是内容
各种内容

</details>
```

效果:

<details> <summary>点击展开</summary>

code ...这里是内容
各种内容

</details>

7 文档内跳转

使用 Markdown 扩展语法实现
MinDoc 扩展了 Markdown 的语法，支持创建锚标记。
第一步，在某个行设置锚点，一定要在单独行。设置完锚点，实际上是看不到的，展示的只是一个空行。

```
[](@first_link)
```

锚点行在下面一行
我在这里
解析器会将标签解析为：

```
<a name="first_link"></a>
```

第二步，在需要跳转的地方直接创建链接即可：

```
[第一个段落](#first_link)
```

第一个段落
作者：玖三伍 创建时间：2021-12-23 11:12

8.mindoc导出各种格式文档

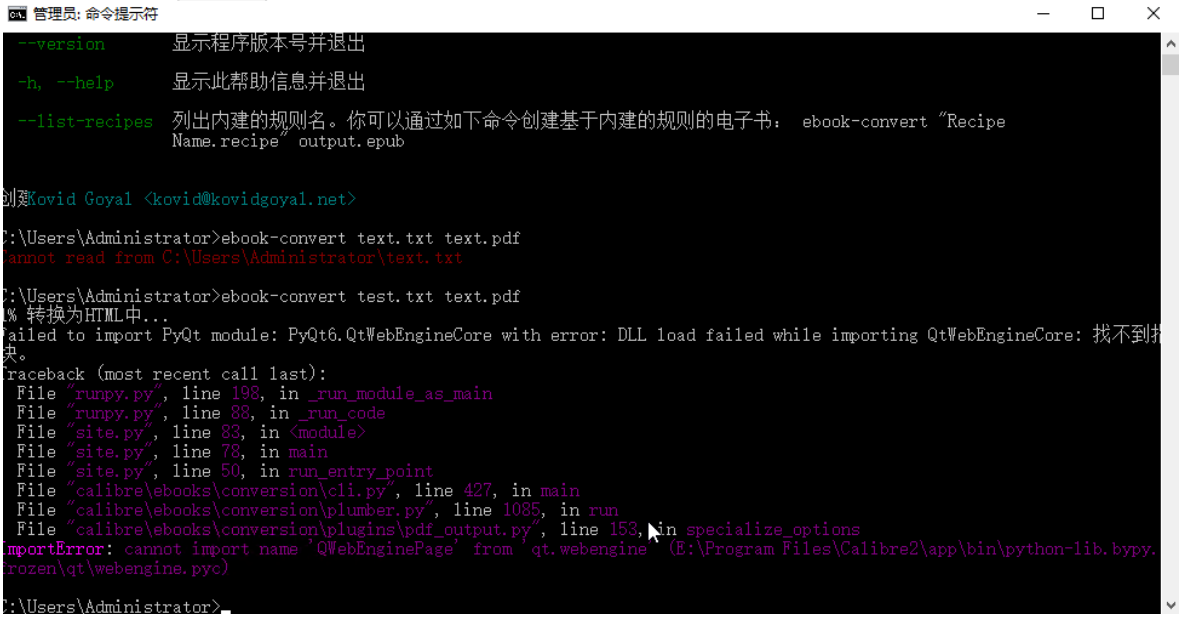
https://doc.gsw945.com/docs/mindoc-docs
https://doc.gsw945.com/docs/mindoc-docs/mindoc-export.md
https://doc.gsw945.com/docs/mindoc-docs/mindoc-calibre.md
https://www.cnblogs.com/gobyte/p/10953081.html

自己的项目右上角 设置 里，打开 导出 功能后点击 保存 设置

回到项目里，点击右上角 下载 按钮，选择下拉里的 pdf 即可。
下载后，记得关闭下载功能，除非你想别人也能下载。

mindoc如何设置下载多格式?

winserver2016开始使用 `calibre` v9.几版本，出现python提示错误。

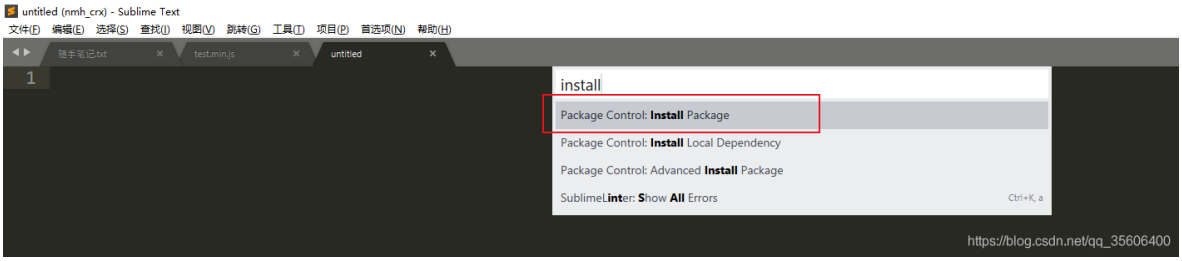


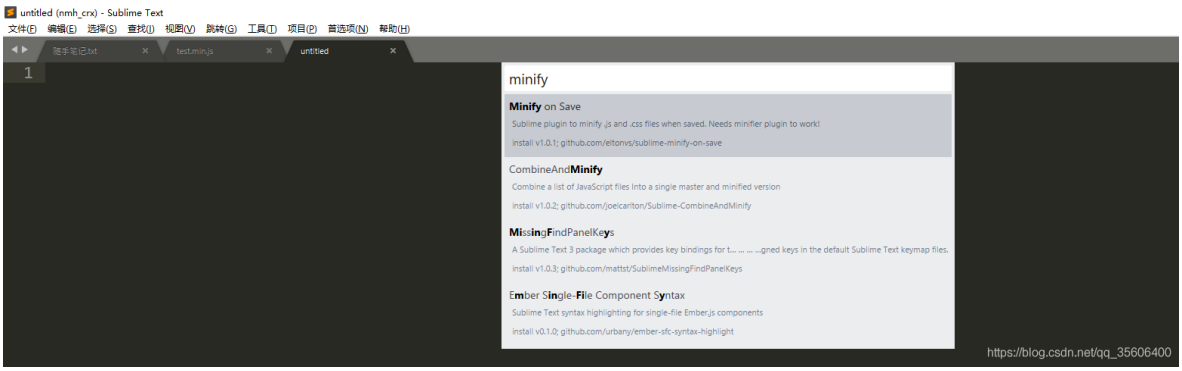
搜索后百度贴吧有人说下载portable版本，就是打包版，非安装版。然后还看到有一些使用7.16版本，不要用最新的9.几版。
将打包版解压到winserver2016我的服务器上后，在cmd里运行的engineercms提示仍然说path路径里没有找到ebook-convert文件。
关闭engineercms的cmd执行窗口，重新开启新的cmd窗口，运行engineercms后，就可以了。
原理是，我解压portable版后ebook-convert.exe文件所在的文件夹路径，重新在系统变量path里设置了，那么设置后，对于cmd窗口，是必须要重新启动一个新的cmd窗口，新的系统变量才有效。没想到engineercms的cmd窗口也是这个原理，万万没想到。

9.sublime minify

sublime3安装minify

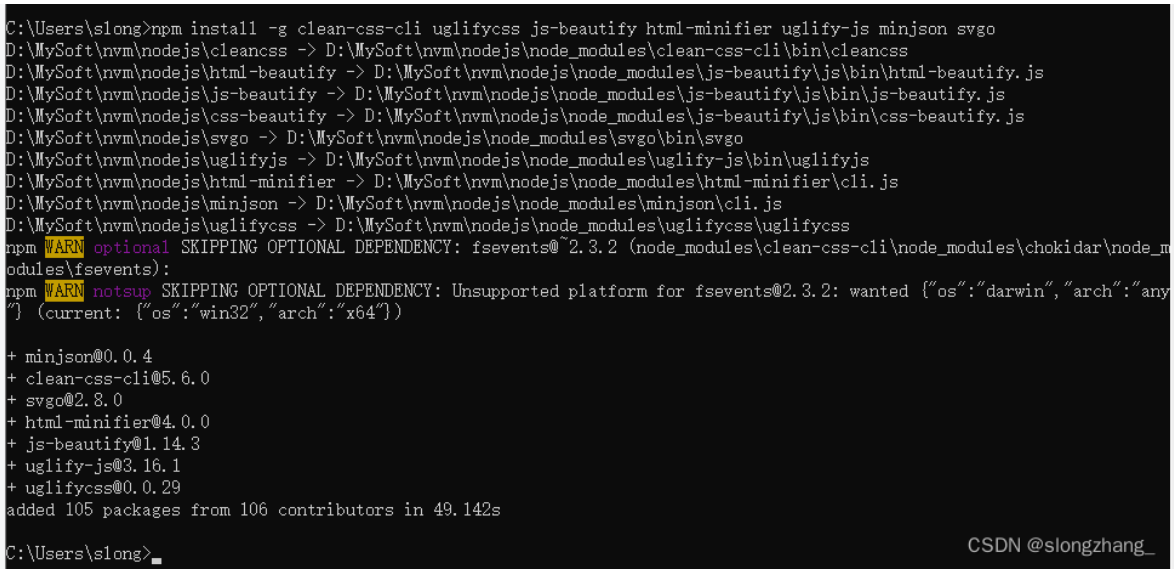
压缩js代码
打开SublimeText，快捷键ctrl+shift+p调出package control输入install package找到minify回车安装即可，安装结成功后该文档在编辑器中弹出。






```
    # 未安装过minify所需组件的命令（文档命令）
    npm install -g clean-css-cli@4.3.0 uglifycss js-beautify html-minifier uglify-js minjson svgo
    # 未安装过minify所需组件的命令（改装支持es6的命令）
    # npm install -g clean-css-cli@4.3.0 uglifycss js-beautify html-minifier minjson svgo
    # npm install -g --save-dev uglify-js@github:mishoo/UglifyJS2#harmony

    #已经安装minify所需组件的可执行下更新命令
    npm update -g clean-css-cli@4.3.0 uglifycss js-beautify html-minifier uglify-js minjson svgo
    # 注意如果已经安装了es5的uglify-js可以通过重新配置package.json 或者 uninstall -g uglify-js再重新安装--save-dev uglify-js@github:mishoo/UglifyJS2#harmony
    # 如果不懂移除请自行百度(或直接清空npm的根目录全部重新安装!!!最彻底，但代价大不推荐)
    # 附上 uglify-js文档一份 https://www.npmjs.com/package/uglify-js-es6#harmony
```



1. Install Node.js

Windows和Mac OS X用户只需要访问https://nodejs.org/ 并且点击install按钮安装即可

2. 安装完成后检查是否安装成功

打开命令行（Mac OS 下的终端/windows下的cmd.exe）输入以下命令
node –version

3. 安装所需依赖包

npm install -g clean-css-cli uglifycss js-beautify html-minifier uglify-js minjson svgo

如果sublime text无法运行install package

https://packagecontrol.io/installation#st3

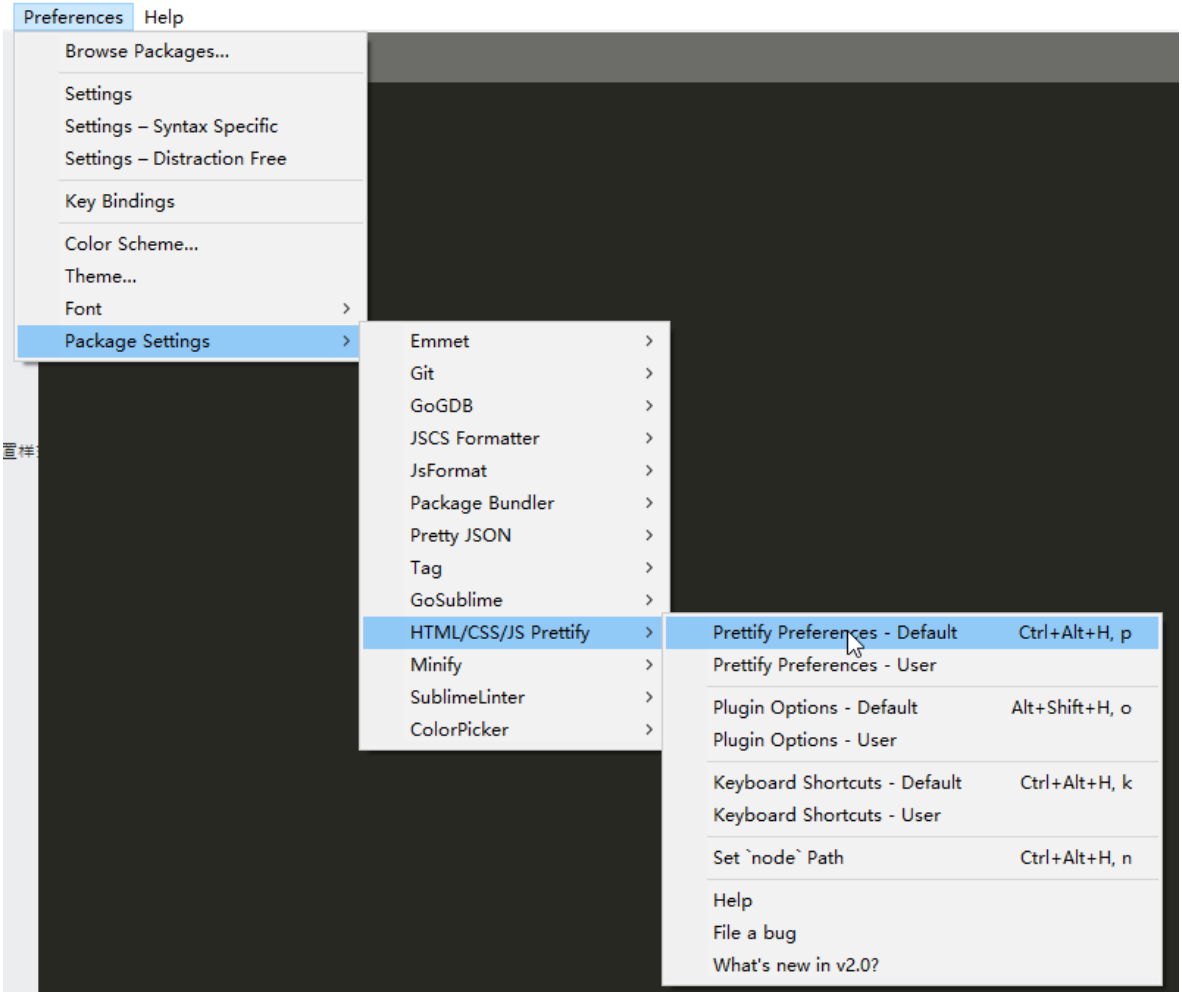
Manual
If the command palette/menu method is not possible due to a proxy on your network or using an old version of Sublime Text, the following steps will also install Package Control:

Click the Preferences > Browse Packages... menu
Browse up a folder and then into the Installed Packages/ folder
Download Package Control.sublime-package and copy it into the Installed Packages/ directory
Restart Sublime Text

安装html css json等格式化插件

Through Sublime Package Manager
Ctrl+Shift+P or in Linux/Windows/OS XCmd+Shift+P
type , select installPackage Control: Install Package
type , select prettifyHTML-CSS-JS Prettify

安装好后，进入




```
"all":
    {
        // These rules apply to any supported code that is going to be be prettified,
        // and have the lowest level of precedence (meaning any of the settings in
        // the 'html', 'css', 'js', 'json' and 'custom' categories override these).

        // You can add other .jsbeautifyrc rules in this section too.

        // End output with newline
        "end_with_newline": false,

        // Character(s) to use as line terminators.
        "eol": "\n",

        // Initial indentation level
        "indent_level": 0,

        // Indentation character
        "indent_char": " ",

        // Indentation size
        "indent_size": 2, // 这里缩进调整为2！！！
```

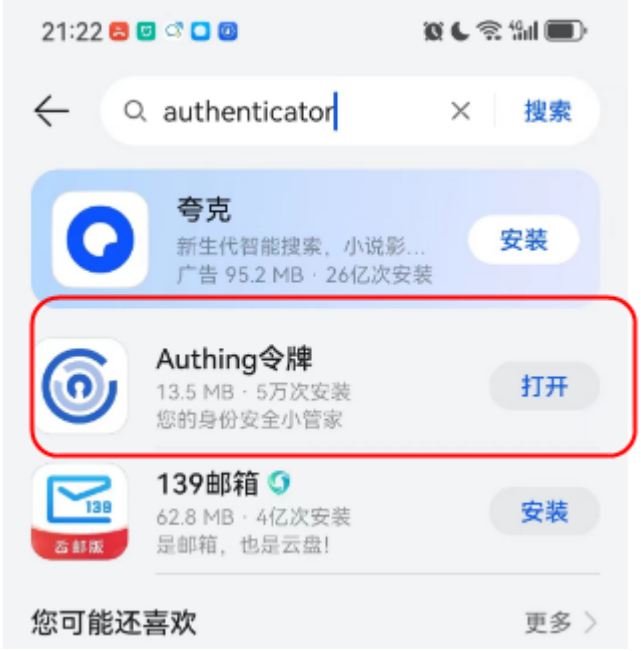
pip debug –verbose

我编译后放在网盘

10.github

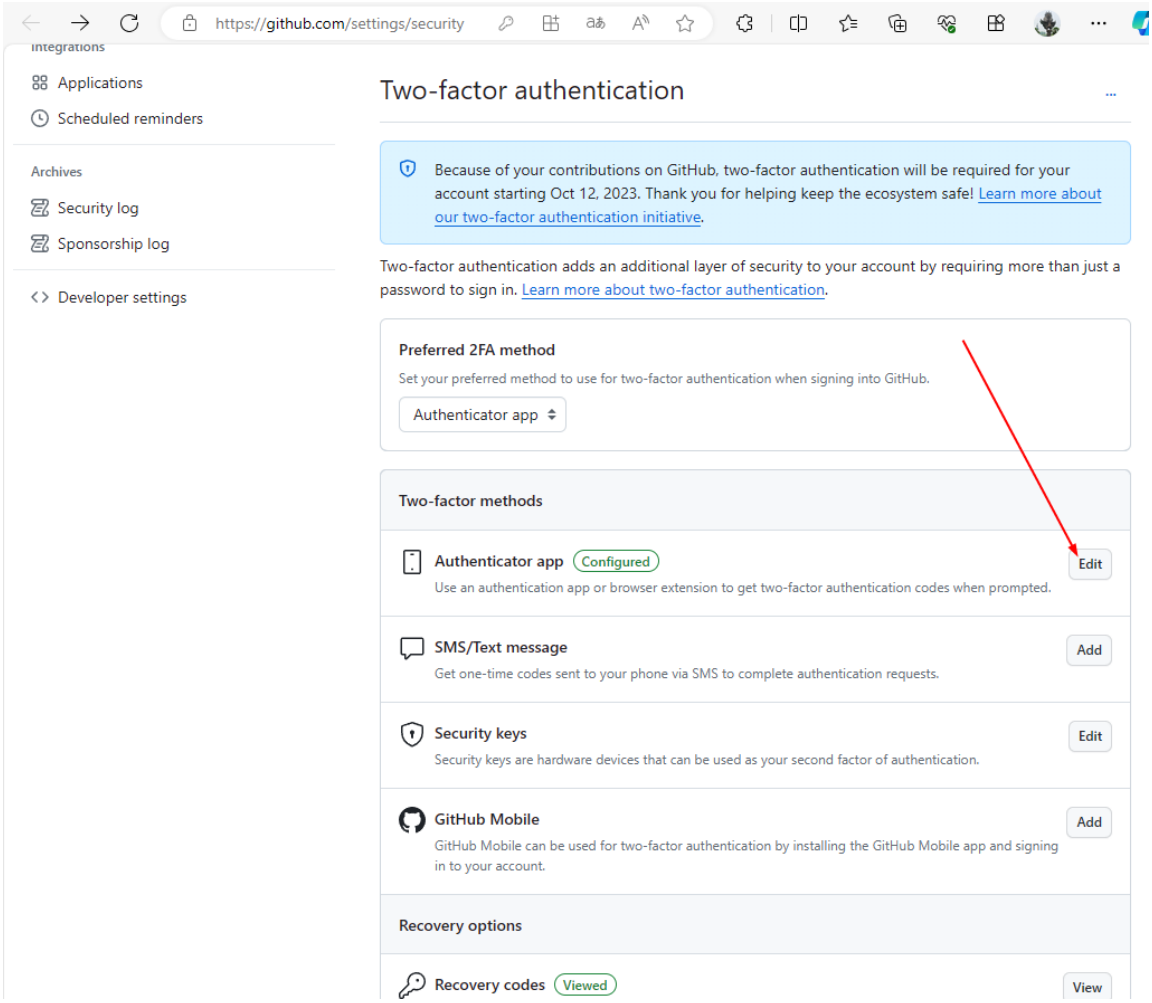
github的双重验证2FA登录方法

github启用2fa后，需要手机在应用市场里搜authenticator，下载那个叫authing令牌的app，打开

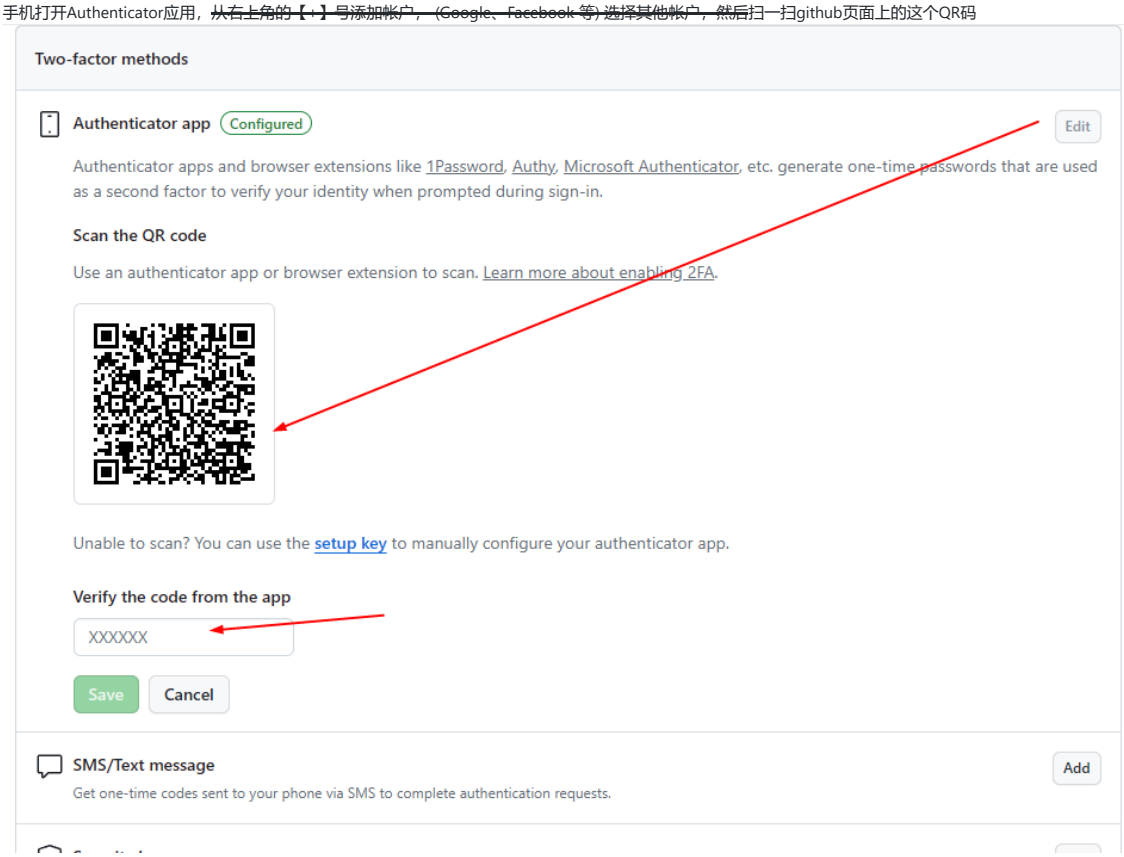


然后在github页面
注意，特别要保存下来 `2FA recovery code` **，见下图最下方那个** `recovery codes—view` **，并且保存的文件名就是这个名字，方便后面检索到。否则一旦更换手机，你就无法获得你的账户了。**

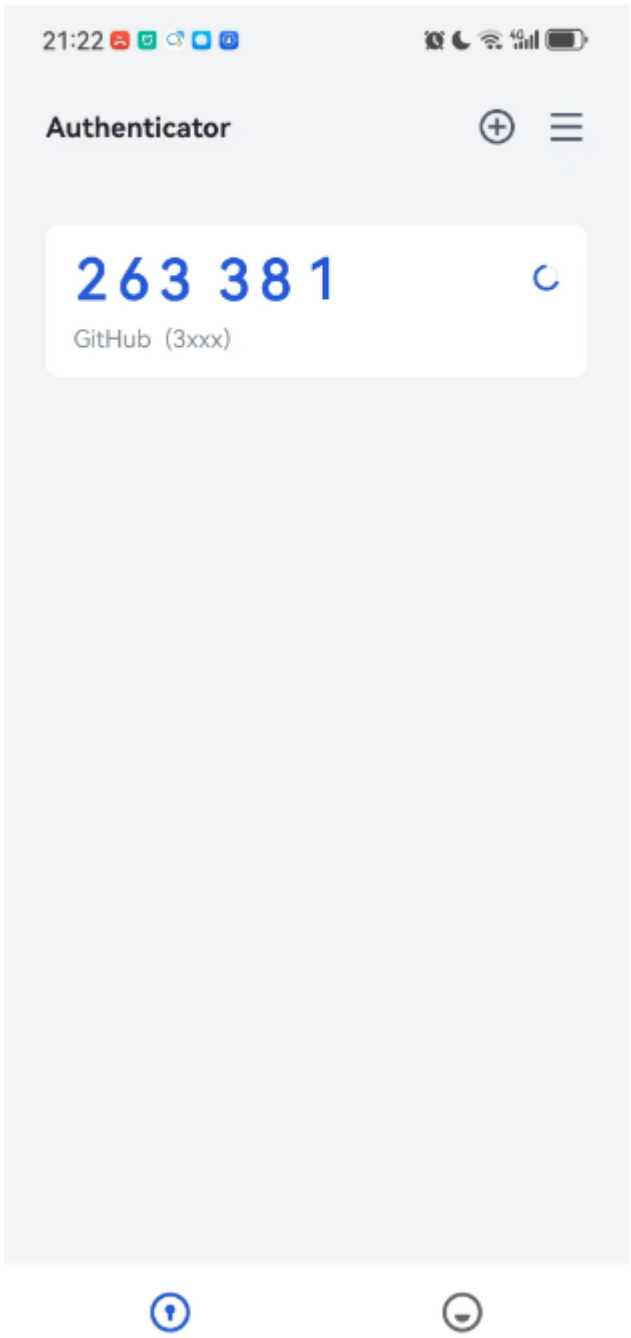
1. 打开GitHub，从右上角选择【设置】，或者点击https://github.com/settings/security
2. 从“个人设置”中选择【password and authentication】，然后往下拉，来到【Two-factor authentication】区域 —— 点击 “Authenticator app” 右侧的 Edit，就会弹出 显示 QR 代码， 请勿关闭此页面。


https://github.com/settings/security
Integrations
Applications
Scheduled reminders
Archives
Security log
Sponsorship log
Developer settings
Two-factor authentication
Because of your contributions on GitHub, two-factor authentication will be required for your account starting Oct 12, 2023. Thank you for helping keep the ecosystem safe! Learn more about our two-factor authentication initiative.
Two-factor authentication adds an additional layer of security to your account by requiring more than just a password to sign in. Learn more about two-factor authentication.
Preferred 2FA method
Set your preferred method to use for two-factor authentication when signing into GitHub.
Authenticator app
Two-factor methods
Authenticator app
Configured
Use an authentication app or browser extension to get two-factor authentication codes when prompted.
Edit
SMS/Text message
Get one-time codes sent to your phone via SMS to complete authentication requests.
Add
Security keys
Security keys are hardware devices that can be used as your second factor of authentication.
Edit
GitHub Mobile
GitHub Mobile can be used for two-factor authentication by installing the GitHub Mobile app and signing in to your account.
Add
Recovery options
Recovery codes
Viewed
View

3. 手机打开Authenticator应用，~~从右上角的【+】号添加帐户，(Google、Facebook 等) 选择其他帐户，然后扫一扫~~github页面上的这个QR码



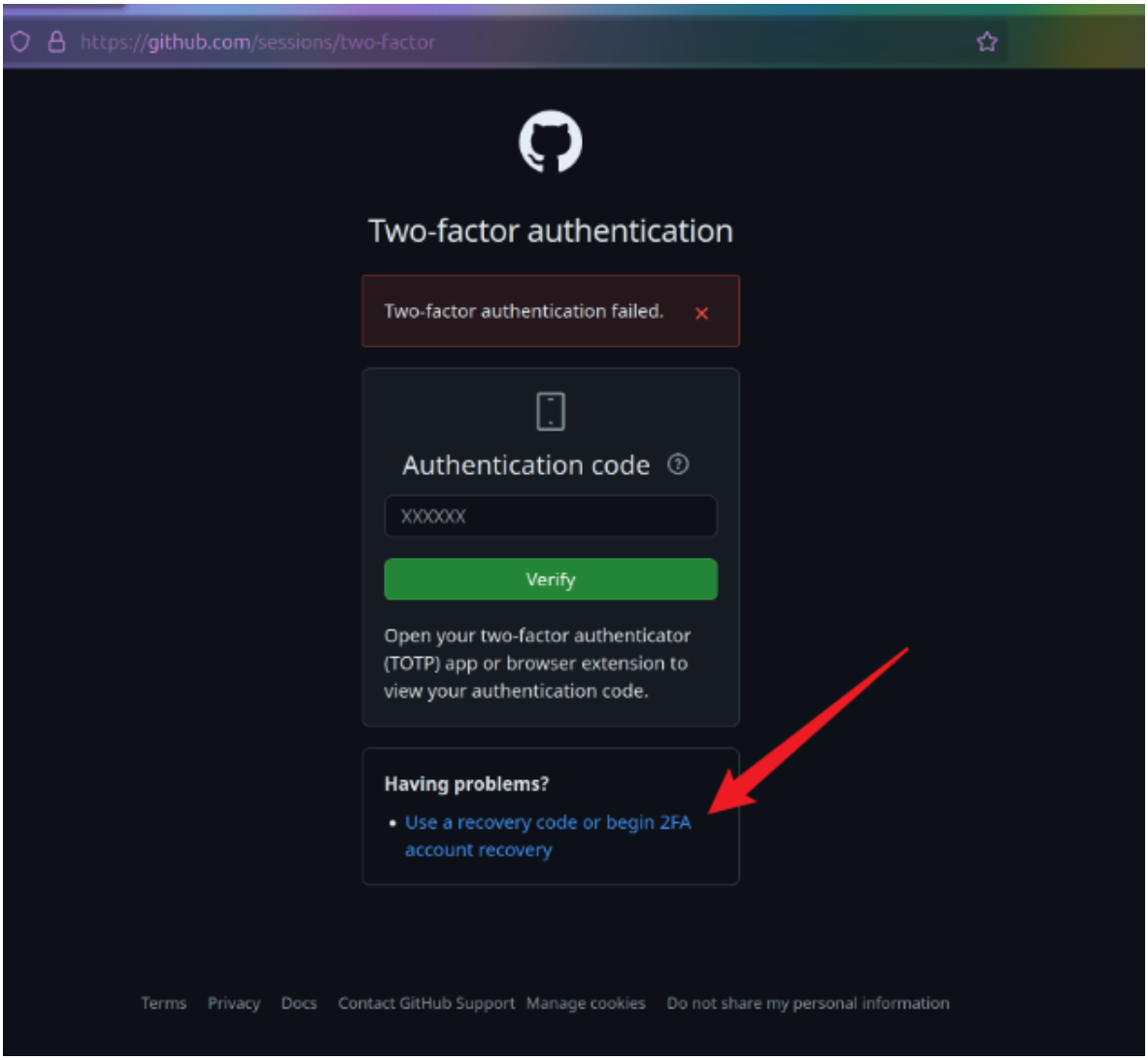
4. 扫描完成之后出现6个数字，将这6个数字输入到页面上，帐号就添加到Authenticator中了。

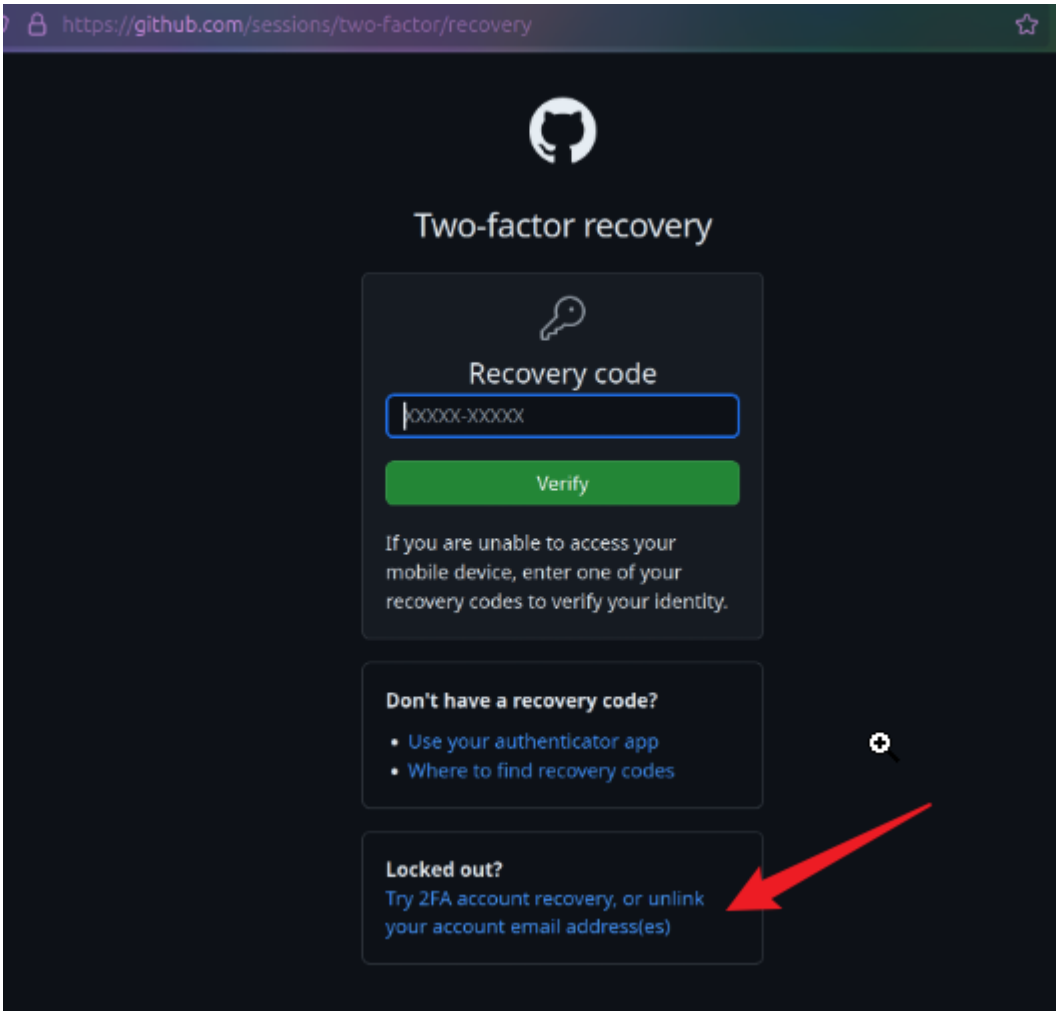


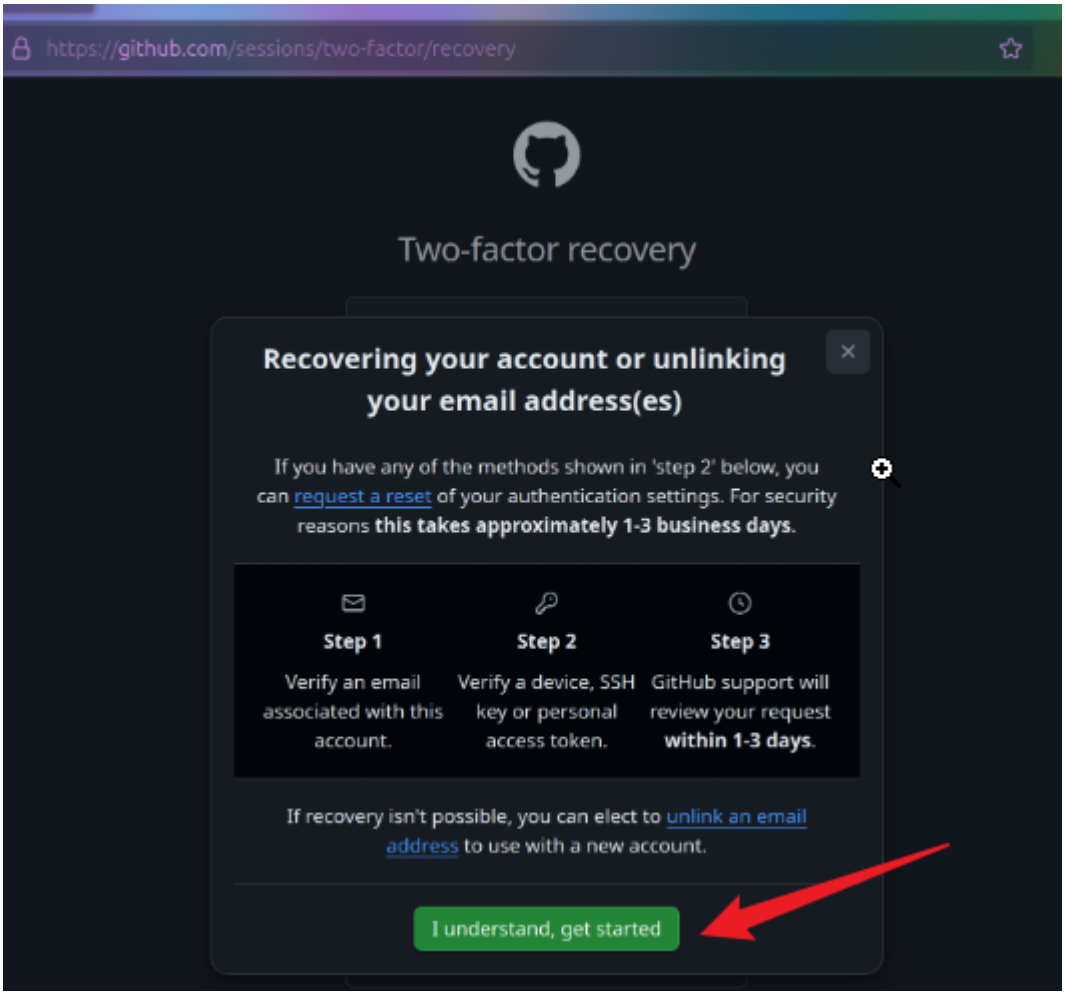
5. 随后手机app上不断刷新这6个数字，切到github登录页面，即可将这个动态验证码输入登录页面里了。
   注意！要将app里2FA的账户导出并保存，就是导出一个二维码保存，下次换手机要导入这个账户。

记录我自己丢失2FA账户申请的经历。

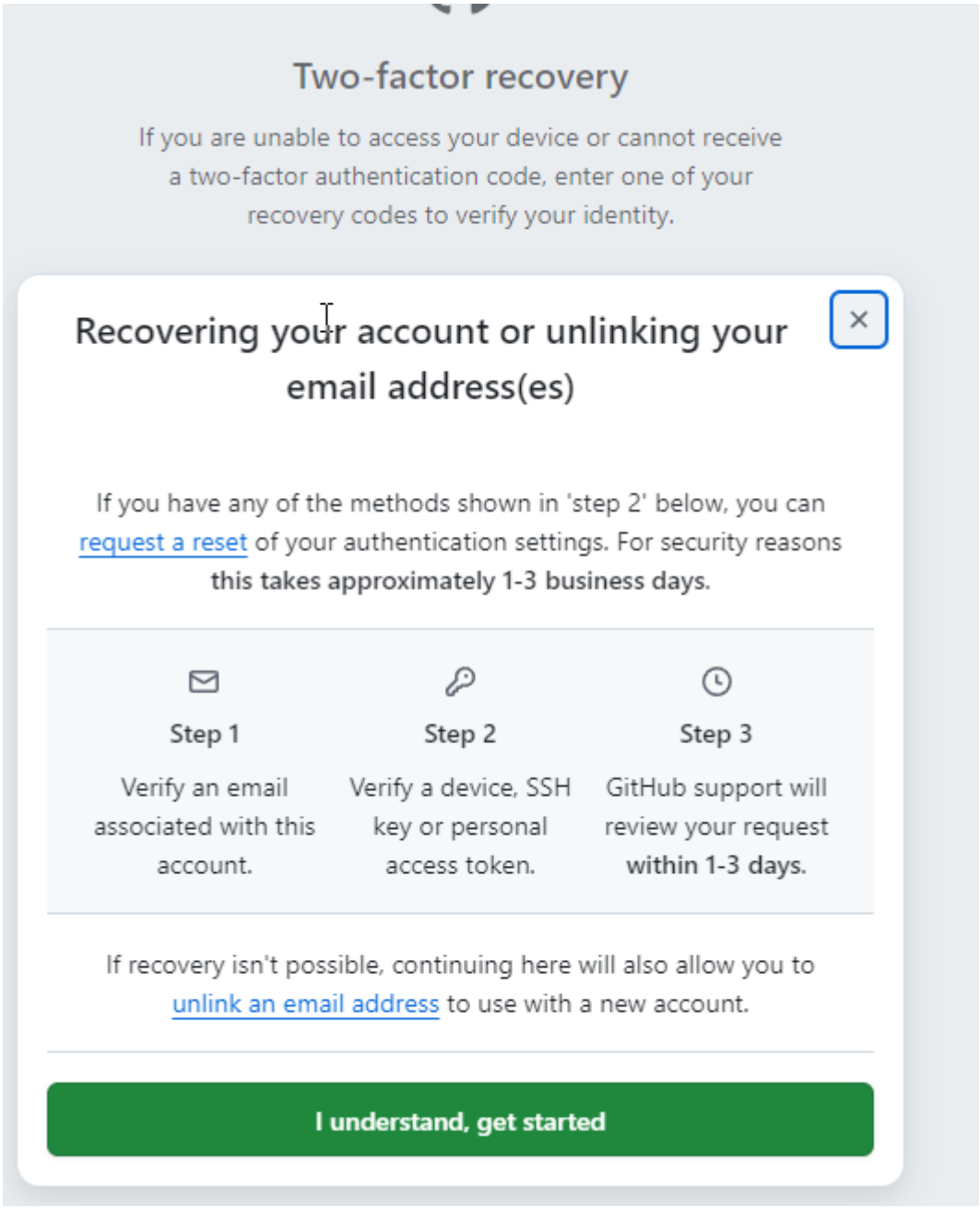
换手机了，authing这个app的账户丢失了。
于是按照
https://blog.csdn.net/fuxily/article/details/140164639


https://github.com/sessions/two-factor
Two-factor authentication
Two-factor authentication failed.
Authentication code
XXXXXX
Verify
Open your two-factor authenticator (TOTP) app or browser extension to view your authentication code.
Having problems?
Use a recovery code or begin 2FA account recovery
Terms
Privacy
Docs
Contact GitHub Support
Manage cookies
Do not share my personal information

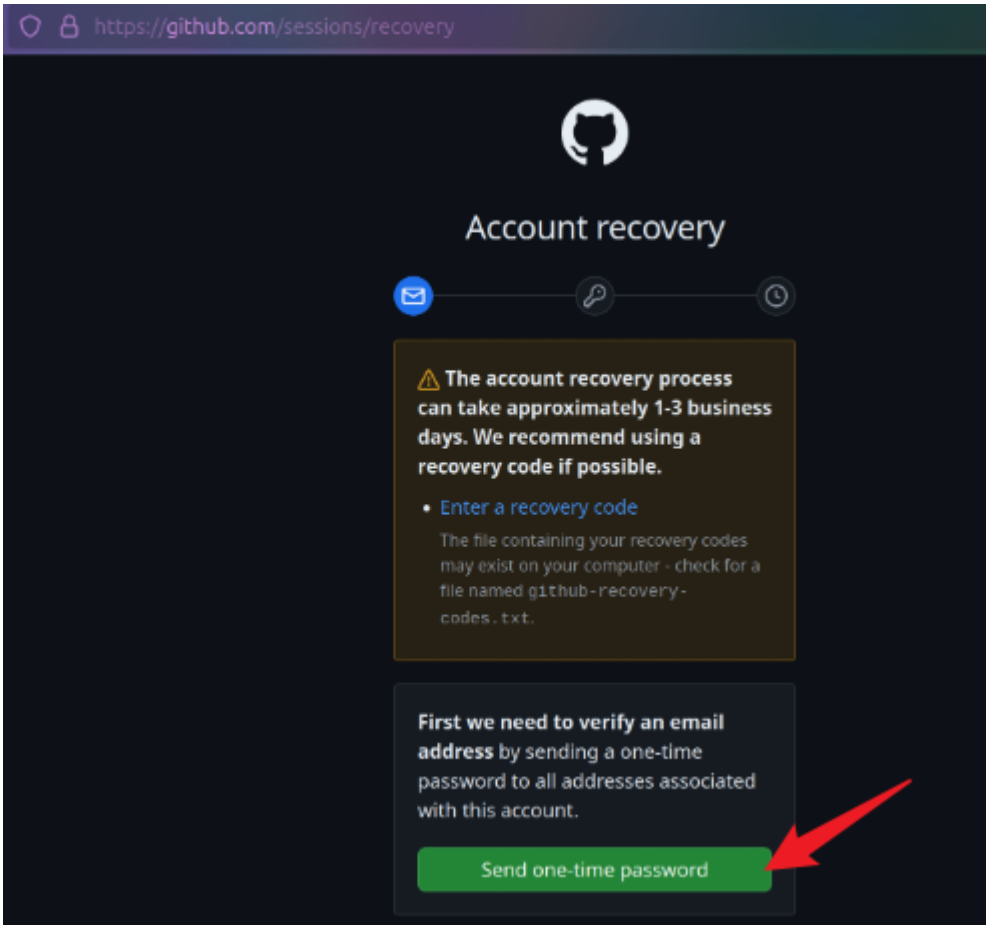

https://github.com/sessions/two-factor/recovery
Two-factor recovery
Recovery code
xxxxx-xxxxx
Verify
If you are unable to access your mobile device, enter one of your recovery codes to verify your identity.
Don't have a recovery code?
Use your authenticator app
Where to find recovery codes
Locked out?
Try 2FA account recovery, or unlink your account email address(es)


https://github.com/sessions/two-factor/recovery
Two-factor recovery
Recovering your account or unlinking your email address(es)
If you have any of the methods shown in 'step 2' below, you can request a reset of your authentication settings. For security reasons this takes approximately 1-3 business days.
Step 1
Verify an email associated with this account.
Step 2
Verify a device, SSH key or personal access token.
Step 3
GitHub support will review your request within 1-3 days.
If recovery isn't possible, you can elect to unlink an email address to use with a new account.
I understand, get started

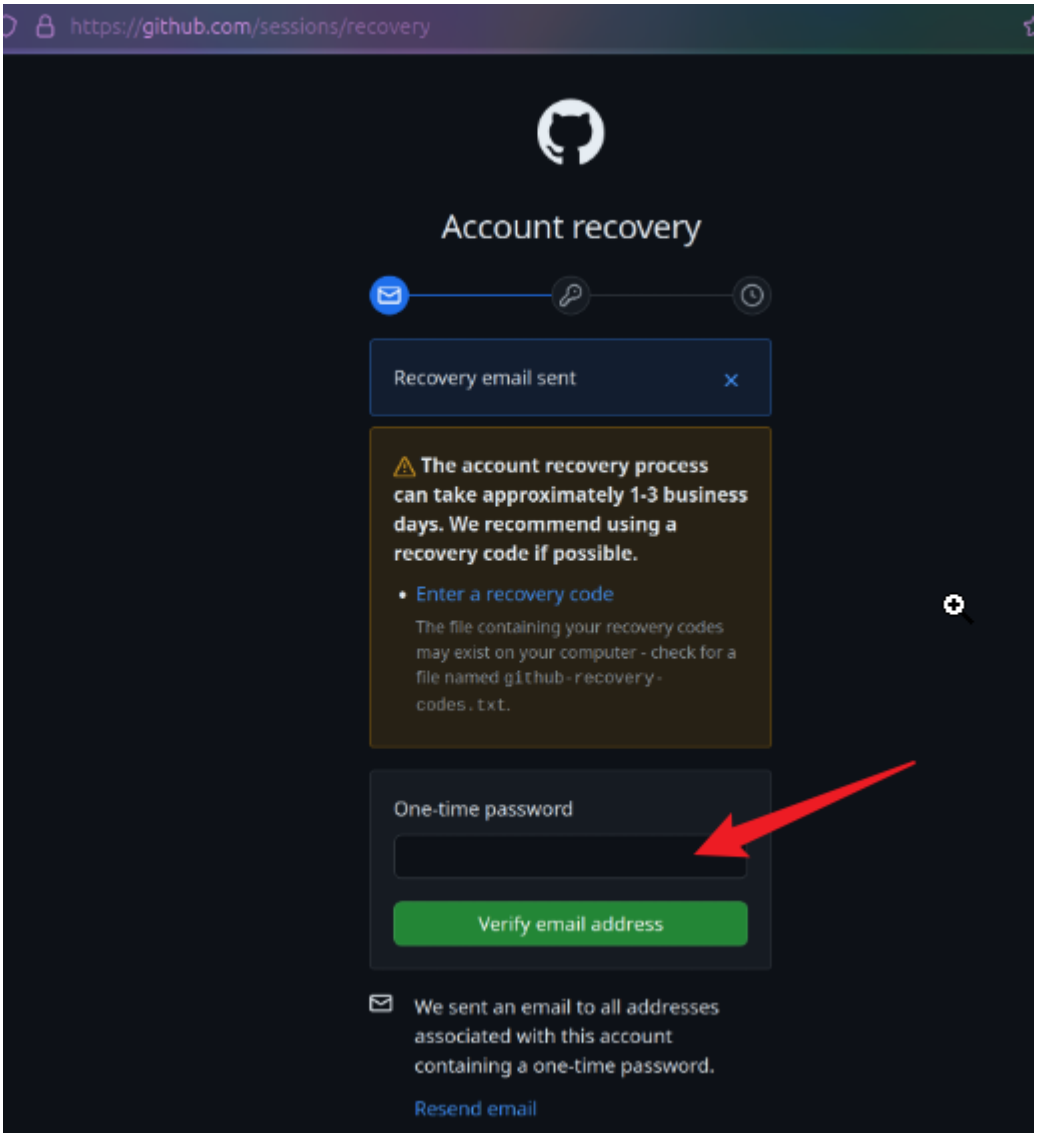
我的是这个样子



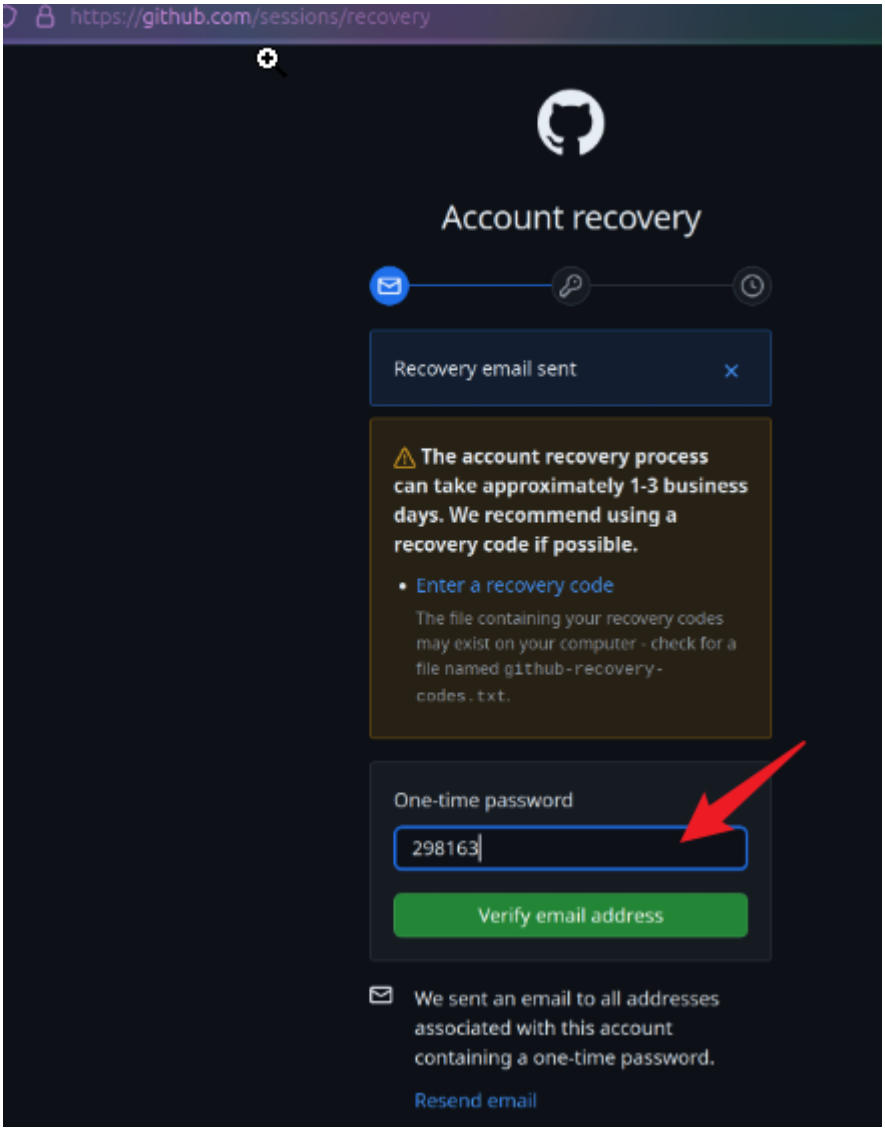
根据上图可知，在没有Recovery code的情况下我们可以通过3个步骤来恢复我们的github*账号
验证github账号绑定的邮箱
可以使用以下三种方式之一验证：常用的主机、github中设置的ssh key、personal access token
github客服将在1-3天内人工审核上述信息，如果审核成功，则会发送邮件通知我们
点击绿色的I understand, get started按钮



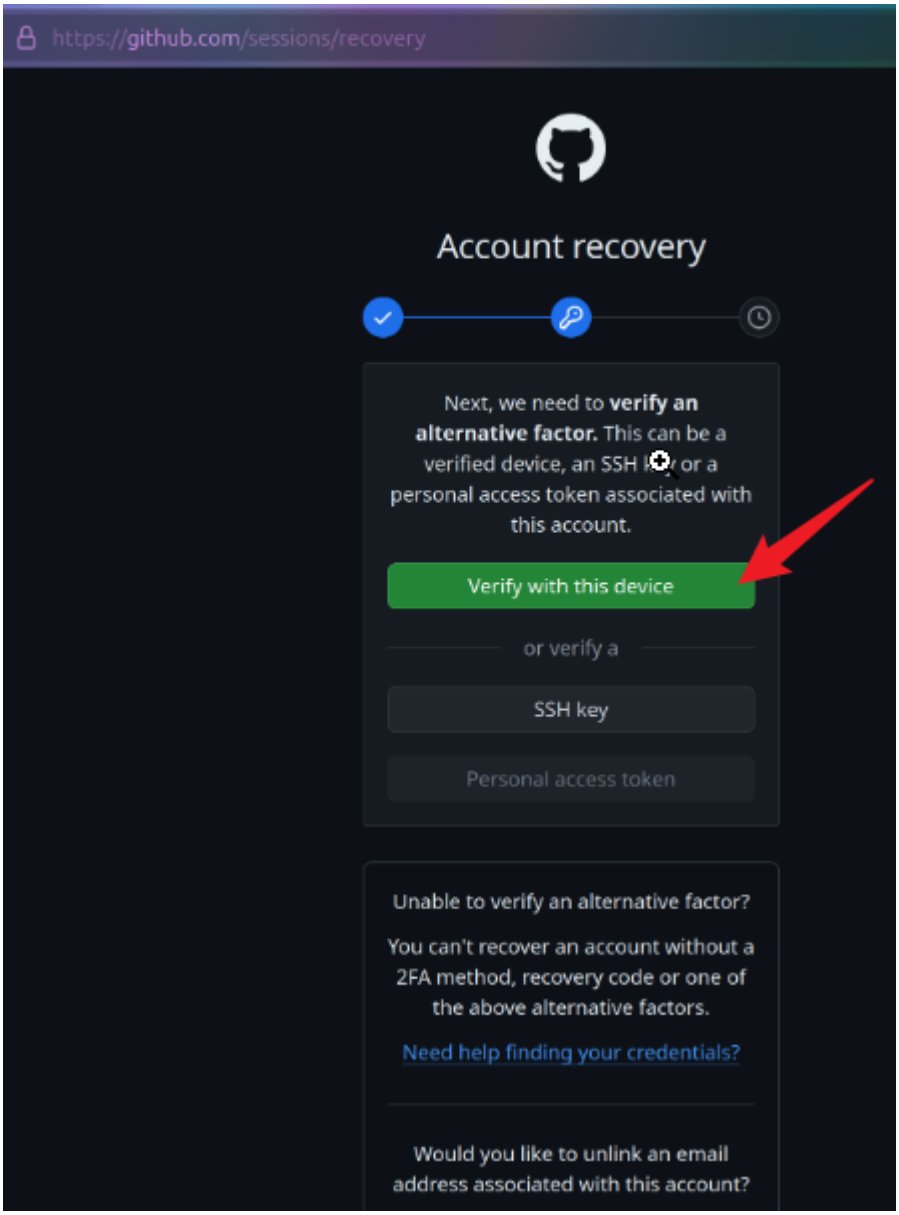
点击绿色的Send one-time password按钮，向github账号绑定地邮箱发送验证码



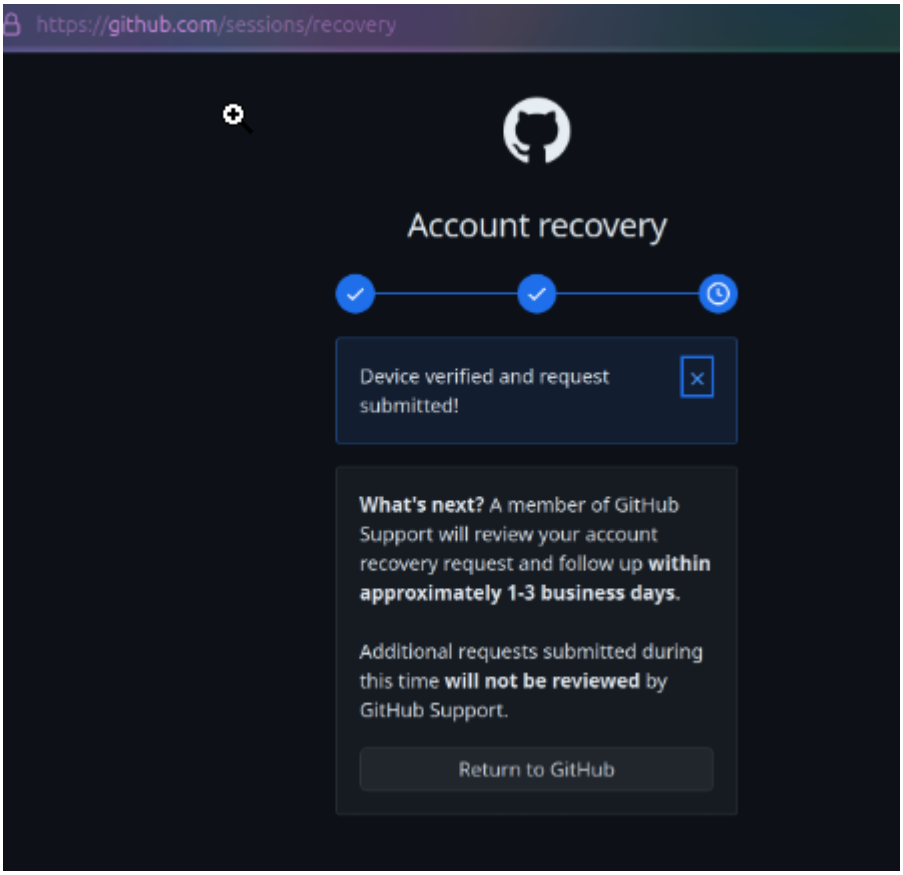
打开电子邮箱，输入邮件中的验证码



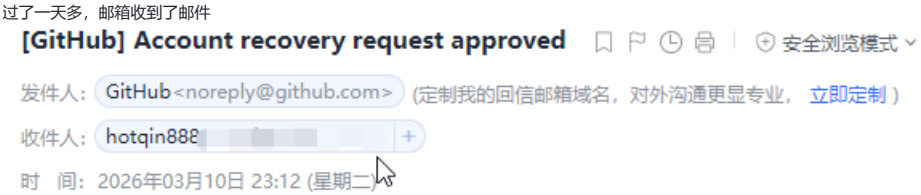
方式一、使用经常登录或最后一次登录github的设备验证



点击上图中的Verify with this device 按钮，采用这种方式必须使用自己经常登录或最后一次登录成功的电脑执行这些操作。



等待接收github发送地邮件，大概需要1-3个工作日
过了一天多，邮箱收到了邮件



Your request to disable two-factor authentication for the account **@3xxx** has been approved by GitHub staff.

To proceed, click this link: Complete account recovery. This link will only be valid for 72 hours.

If you wish to cancel this request, click this link: Cancel Account recovery request

Once you have regained access to this account, please follow our guidance to re-enable 2FA, resecure your account and ensure that it remains protected.

Sign into GitHub · Terms · Privacy · Contact GitHub Support

意思是暂时停止了2FA验证，点击里面的链接，输入账户和密码就能登录，然后显示2FA的二维码账户，手机authing app扫码就行了，然后提示下载2FA recovery code文件，里面有好几组的恢复码，保存好。

给github做贡献

```
Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx
$ git clone https://github.com/3xxx/mindoc.git
Cloning into 'mindoc'...
remote: Enumerating objects: 11960, done.
remote: Counting objects: 100% (823/823), done.
remote: Compressing objects: 100% (595/595), done.
remote: Total 11960 (delta 284), reused 675 (delta 213), pack-reused 11137
Receiving objects: 100% (11960/11960), 90.08 MiB | 10.69 MiB/s, done.
Resolving deltas: 100% (6172/6172), done.

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx
$ cd mindoc

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindoc (master)
$ git remote add upstream https://github.com/mindoc-org/mindoc.git

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindoc (master)
$ git remote -v
origin  https://github.com/3xxx/mindoc.git (fetch)
origin  https://github.com/3xxx/mindoc.git (push)
upstream        https://github.com/mindoc-org/mindoc.git (fetch)
upstream        https://github.com/mindoc-org/mindoc.git (push)

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindoc (master)
$ git fetch upstream master
From https://github.com/mindoc-org/mindoc
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> upstream/master

这里先更新一下
Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindoc (master)
$ git pull upstream master
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 1 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 859 bytes | 859.00 KiB/s, done.
From https://github.com/mindoc-org/mindoc
 * branch            master     -> FETCH_HEAD
   397dced..db620be  master     -> upstream/master
hint: Waiting for your editor to close the file...
/usr/bin/bash: line 1: q: command not found

shell returned 127

Press ENTER or type command to continue
Merge made by the 'ort' strategy.


Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindoc (master)
$ git add .

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindoc (master)
$ git commit -m "update images upload"
[master af88388] update images upload
 2 files changed, 306 insertions(+), 260 deletions(-)

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/3xxx/mindoc (master)
$ git push
warning: ----------------- SECURITY WARNING ----------------
warning: | TLS certificate verification has been disabled! |
warning: ---------------------------------------------------
warning: HTTPS connections may not be secure. See https://aka.ms/gcm/tlsverify for more information.
warning: ----------------- SECURITY WARNING ----------------
warning: | TLS certificate verification has been disabled! |
warning: ---------------------------------------------------
warning: HTTPS connections may not be secure. See https://aka.ms/gcm/tlsverify for more information.
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 20 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 3.21 KiB | 3.21 MiB/s, done.
Total 9 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), completed with 7 local objects.
To https://github.com/3xxx/mindoc.git
   397dced..af88388  master -> master
```

1 win下编译mindoc

修改conf里
注释掉mysql
取消sqlite注释
拷贝一个sqlite数据库到database文件夹里，更名为mindoc.db
cmd进入mindoc文件夹
因为用sqlite需要cgo，go env -w CGO_ENABLED=1
https://blog.csdn.net/qq_43625649/article/details/134488353

D:\gowork>cd src\github.com\mindoc
D:\gowork\src\github.com\mindoc>bee run

2 Fork 这个项目

3 Clone 已经 fork 的项目

当你 fork 一个项目之后，你需要把它克隆到你的电脑上，这样你才能开始工作。
要 clone 这个项目，你先打开你自己的 GitHub 主页，找到 fork 过来的项目，打开后点击右上角的 “Clone or download” 按钮，得到复制的地址。
GitHub 为克隆项目提供了两种传输协议： HTTPS 和 SSH 。关于这个主题的更多内容可以看这里。这里假设你决定使用 HTTPS 。
当你复制了项目 URL，你可以在 Git 客户端或者 shell 里 clone 项目：

```
$ git clone https://github.com/mindoc-org/mindoc.git
```

Clone 一个项目会在你的硬盘上创建一个文件夹，里面有项目的所有文件，还有跟踪文件变化的 git 文件夹。
注意：

1. clone到本地的项目文件夹作为将来提交的文件夹，因此要保持干净，不要在这里面进行开发调试。
   本地开发调试要另外再clone这个项目，完成开发后把个别修改的文件拷贝到前面clone的文件夹里，然后在它里面进行git push等提交代码操作。
2. 提交到自己fork的项目里后，再进行pull request操作。

4 设置克隆过来的项目

进入克隆过来的项目文件夹，将原来项目的 URL 添加到你的本地代码仓库，这样你就可以随时从原来的项目 pull 最新的修改:

```
$ git remote add upstream url
```

我用 upstream (上游) 作为远程仓库的名字，这是 GitHub 的风格，但是你可以用任何名字。

现在远程仓库列表是这样的:

```
Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)
$ git remote add upstream https://github.com/mindoc-org/mindoc.git

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)
$ git remote -v
origin  https://github.com/engineercms/mindoc.git (fetch)
origin  https://github.com/engineercms/mindoc.git (push)
upstream        https://github.com/mindoc-org/mindoc.git (fetch)
upstream        https://github.com/mindoc-org/mindoc.git (push)
```

这里创建了一个叫做upstream的与源代码origin关联,关联之后，就可以同步源代码库的其他分支到自己的仓库了
在你把自己的修改 push 到你的 fork 之前，从上游拉回一次新的变动。

```
Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)
$ git fetch upstream master
From https://github.com/mindoc-org/mindoc
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> upstream/master

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)
$ git pull upstream master
From https://github.com/mindoc-org/mindoc
 * branch            master     -> FETCH_HEAD
Already up to date.
```

5 推送

```
Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)
$ git add .
warning: in the working copy of 'conf/app.conf.example', LF will be replaced by CRLF the next time Git touches it

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)
$ git commit -m "add prev&next in default read.tpl"
[master 2988fab] add prev&next in default read.tpl
 11 files changed, 1776 insertions(+), 95 deletions(-)
 create mode 100644 conf/app.conf.example
 delete mode 100644 uploads/202404/cover_17c7b2c6745688e0.png
 create mode 100644 uploads/202404/cover_17c8260e47e06f18_small.png
 create mode 100644 uploads/202404/cover_17c8286be57a8600_small.png
 create mode 100644 uploads/202404/cover_17c829abdef5fdc8.png
 create mode 100644 uploads/books/2/book.zip
 create mode 100644 views/document/test.tpl
 delete mode 100644 "win\344\270\213\347\274\226\350\257\221.md"

Administrator@DESK-XDIGIWSFLS MINGW64 /d/gowork/src/github.com/mindoc (master)
$ git push
warning: ----------------- SECURITY WARNING ----------------
warning: | TLS certificate verification has been disabled! |
warning: ---------------------------------------------------
warning: HTTPS connections may not be secure. See https://aka.ms/gcm/tlsverify for more information.
warning: ----------------- SECURITY WARNING ----------------
warning: | TLS certificate verification has been disabled! |
warning: ---------------------------------------------------
warning: HTTPS connections may not be secure. See https://aka.ms/gcm/tlsverify for more information.
Enumerating objects: 33, done.
Counting objects: 100% (33/33), done.
Delta compression using up to 20 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (21/21), 249.03 KiB | 27.67 MiB/s, done.
Total 21 (delta 11), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (11/11), completed with 11 local objects.
To https://github.com/engineercms/mindoc.git
   6054b5f..2988fab  master -> master
```

1. 问题一

甲乙两个程序员正在合作开发一款程序，甲晚上10点push了自己编辑的test.txt文档到GitHub。
乙11点也想要push自己编辑过后的test.txt文档到GitHub,因为乙本地仓库不是最新，乙push时会报错，乙如果使用git pull拉取最新版本，那自己修改的东西丢失了怎么办?
解决办法:
先把自己修改好的代码存放在缓存里，等最新代码拉取下来以后再恢复缓存里的自己修改的代码,乙再push。
前提是两人不要修改相同的地方，可能会产生冲突!
步骤如下:

```
git stash save "这里是注释"
git pull
git stash pop
git stash list
git stash show
```

2. 问题二

乙在上个问题背景时不知道甲做了提交，直接commit了自己的test.txt文件，想取消自己的commit怎么办?
解决办法:

```
git reset --soft HEAD^
```

原文链接：https://blog.csdn.net/m0_61502213/article/details/120620168

git-拉取代码不覆盖本地修改的代码

1 先将本地代码放在暂存区:

```
git stash
```

2 将远程gitlab上代码拉取下来：

```
git pull
```

3 将1中暂存区的代码放回本地：

```
git stash pop
```

4 接下来将代码合并到master：

4.1 提交到远程库：

方法一：提交tracked的修改

```
git commit -am 'comment code1'
git push
```

方法二：提交未tracked的修改

```
git add -u
git commit -m 'comment code1'
git push
```

4.2 codereview:

```
arc diff origin/master
```

4.3 合并到master：

```
arc land --onto master
```

其他：

```
git add -u <==> git add –update
```

提交所有被删除和修改的文件到数据暂存区

```
git add .
```

提交所有修改的和新建的数据暂存区

```
git add -A <==>git add –all
```

提交所有被删除、被替换、被修改和新增的文件到数据暂存区

【gitHub】在提PR（pull request）时提交指定的commit

一、背景
今天在提PR的时候，发现以前一些旧的commit也一起被提交上去了，这样比较不友好，在网上研究了一些方法，在这里总结记录一下。

二、方法
对于方法博主研究后有两种，一种是通过版本非强制回退，然后再重新commit一次，这样就能够合并之前的所有commit。另外一种是通过使用cherry-pick选择commit进行提交。博主推荐使用第二种。

2.2.使用cherry-pick进行提交
cherry-pick是在新分支下选择本地分支的commit记录进行合并的操作，因此只需要在本地新建一个上游仓库（注意不是远程仓库）的本地分支，再合并想要提交的commit即可。

首先，配置上游仓库地址并同步。

```
git remove add upstream XXX.git // XXX.git是上游仓库地址
git fetch
```

在github上或者在本地运行git log选取commit记录。
注意：需要选择上次PR之后的除了merge操作的所有commit的id。

在本地从上游仓库中拉取新的分支。
注意：upstream是上游仓库，在第一步配置过；origin是自己的远程仓库；main是主分支名。

```
git checkout -b new_branch upstream/main
```

在new_branch的分支上cherry-pick所有commit记录。
注意：要选择所有未提交PR的所有commit，这样才不会出错。

```
git cherry-pick commit_id1 commit_id2 commit_id3 ......
```

将新分支推送到远程仓库。

```
git push origin new_branch
```

在github提PR的时候选择新推送的分支即可。

PR被接受后，可选择删除新建立的本地分支和远程分支。

```
git branch -D new_branch   // 在main分支删除new_branch分支
git push origin --delete new_branch   // 删除远程仓库分支
```