文本编辑系统

NEKOCheck

软件需求规格说明书

成员信息	朱康希	信安 171	41721604
	林鸿杰	信安 171	41721646
	程春亮	信安 171	41702025
	方帕思	信安 171	61724061
指导教师	崔晓龙		

2020 年 5 月 26 日

目录

一、 引言

1.1 目的

该文档根据 NekoCheck 的功能和性能，阐述了 NekoCheck 的概要设计，包括框架设计，功能模块设计，数据库设计，界面设计等部分。
本文档的预期读者包括：

- 设计开发人员
- 项目管理人员
- 测试人员
- 用户

1.2 范围

该文档的目的是解决整个项目系统的“怎么做”的问题。在这里，主要是根据用户提出的项目需求进行的全面设计。

1.3 引用标准

1.4 参考资料

1.5 版本更新信息

表 **1.1** 版本更新信息

修改编号	修改日期	修改后版本	修改位置	修改内容概述
001	2020.5.12	0.0	全部内容	添加基本信息
002	2020.5.19	0.0	功能模块	完善基本信息
003	2020.5.24	0.0	界面设计	增加基本信息
004	2020.5.26	0.0	总体修改	修改纰漏信息

二、 项目设计原则简介

在整个系统设计的过程中遵循以下的设计原则：

1. 实用性：实用性是系统的主要设计原则，系统设计必须最大可能地满足用户的需求，做到操作方便、界面友好、可即时更新，能适应不同层次用户的需求。

2. 先进性：信息技术发展迅速，系统设计尽可能采用先进的技术标准和技术方法。

3. 以用户为中心的处理：个性化服务充分体现了这一点，根据用户当前展业重点，配置页面功能布局及展现内容，贴合用户操作。

4. 使用便捷。系统要有设计良好的人机交互界面，即使系统的操作界面简单易用，又能具有较强的适用性，满足不同计算机使用水平的用户使用。

5. 灵活和易维护：采用开放的体系架构，基于开放源代码的技术框架和数据库系统，使用高效率的开源和免费开发工具，具备完整的文档说明。在维护方面，主要考虑两个层面，一是对于开发人员来讲，系统编码容易调整，可适应需求的变化和调整；二是对于系统管理维护人员来说，能够对系统进行便捷的维护和管理。

6. 安全可靠：选择安全可靠的软硬件运行平台，并在系统设计和实现的时候关注系统的安全控制和执行效率，提供相应的安全防护功能，保证系统具有较高的安全性和可靠性。安全性方面，要考虑系统的安全、数据管理的安全、网络安全。保证用户权限、数据安全和系统的稳定性。

7. 单一职责原则：我们系统在面向对象设计部分采取单一职责原则，其核心思想为：一个类，最好只做一件事，只有一个引起它的变化。单一职责原则可以看做是低耦合、高内聚在面向对象原则上的引申，将职责定义为引起变化的原因，以提高内聚性来减少引起变化的原因。从而最终提高我们系统的可修改性和可维护性。

本概要设计涵盖了体系结构设计、功能模块设计和界面设计。

三、 体系结构设计

NekoChcek 为了能够在多平台运行，采用 Eclectron 进行开发。Ecletron 框架使用渲染程序来展示我们编写的界面，并且使用只有一个的主进程来创建与管理其他渲染程序。NekoCheck 代码编辑器只有一个主界面，所以只需要使用一个主进程加渲染程序就足够。

四、 功能模块设计

概要：

(1) 目标:

该阶段目的在于明确系统的数据结构和软件结构，此外总体设计还将给出内部软件和外部软件之间的接口定义，各模块的功能说明，数据结构的细节以及软件的具体装配要求。

(2) 运行环境:
软件基本运行环境为 Windows、Linux 和 MacOS。

(3) 需求概述:
本系统要达到以下目标

(a) 系统必须支持跨平台使用

(b) 系统支持基本的文本编辑功能

(c) 系统支持语法高亮显示

(d) 系统支持不同编码的文本文件

(4) 条件与限制:
为了评价该设计阶段的设计表示的“优劣程度”，必须遵循以下几个准则：

(a) 软件设计应当表现出层次结构，它应巧妙地利用各个软件部件之间的控制关系。

(b) 软件设计应当表现出层次结构，它应巧妙地利用各个软件部件之间的控制关系。

(c) 设计最终应当给出具体的模块（例如子程序或过程），这些模块就具有独立的功能特性。

(d) 应该应用在软件需求分析期间得到的信息，采取循环反复的方法来获取设计。

(5) 系统结构的考虑:
软件系统结构必须模块化，各模块功能相互独立。

(6) 错误处理机制的考虑:
操作层提供用户操作指南和帮助文档。代码层设计完善的程序异常处理机制，将故障写入错误文件。

4.1 文本编辑

4.1.1 模块功能

文本编辑模块：实现打开文件，保存文件，新建文件，关闭窗口，剪切，复制，粘贴，查找，主窗口工作区的配置，菜单栏的设置，右键菜单栏。在工作区只能输入文本，打开只能打开文本类型的文件，不在工作区进行图片格式文件的处理。

4.1.2 模块对象

1. 查找（使用 electron 自带的 FindInPage 类）约束：只能输入文本信息。
2. 新建文件：该组件在主进程中，向渲染进程发送消息，然后渲染进程中进行工作区的新建。约束：新建文件前得先查看当前工作区的文件是否已经保存好了，若没有则提醒用户是否保存；
3. 打开文件：该组件在主进程中，向渲染进程发送消息，然后渲染进程调用 opendialog 方法。约束：打开文件前得先查看当前工作区的文件是否已经保存好了，若没有则提醒用户是否保存；打开文件时点击取消不返回报错；
4. 保存文件：该组件在主进程中，向渲染进程发送消息，然后渲染进程调用 savedialog() 方法。
5. 粘粘，复制，剪切，关闭：使用 electron 文档中 menuitem() 的方法
6. 菜单栏：在主进程中进行配置。
7. 右键菜单：在主进程中进行配置。约束：只能在工作区里面使用。
8. 工作区：书写 css 文件进行属性的配置（长宽高，字体，背景颜色），渲染进程中调用 css 配置生成工作区。

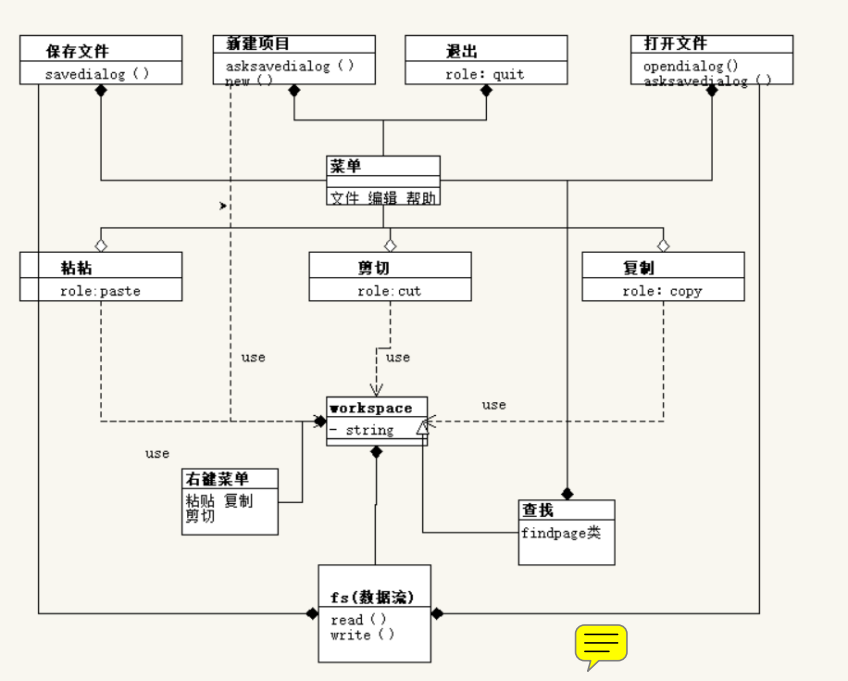


图 4.1 E-R 图

4.1.3 对象的触发机制

- 查找：用户点击查找按钮；跳出查找框，工作区相应的文本就会被加亮显示。

- 打开文件：用户点击打开文件按钮。工作区会打开相应目录下选择的文件

- 保存文件:：用户点击保存文件按钮。将相应工作区下的文档保存到指定目录中。

- 复制：用户点击复制按钮；复制相应文本。

- 粘贴：用户点击粘粘按钮；粘粘相应文本。

- 剪切：用户点击剪切按钮；剪切相应文本。

- 右键菜单栏：用户在工作区点击右键；返回配置好的菜单栏。

- 工作区：打开软件；主窗口中有一个编辑文本的工作区域。

4.1.4 对象的关键算法

在实现粘贴复制退出剪切等功能的时候，使用 electron 中自带的 role 角色属性，连同全局快捷键一起注册成功。具体功能参考electron 文档。在实现读取文件的时候，时候字符流数据传输, 引入 fs 模块。在实现查找功能的时候，使用系统自带的 findpage 函数进行默认属性的更改设置使用。在实现文件的打开，新建，保存功能的时候，采用同步通信的 showopendialog(),showsavedialog() 方法。

4.2 语法高亮与代码补全

4.2.1 模块功能

1. 语法高亮属于数据流处理加工的位置。文本从键盘输入或者从文件读入后，若选择某种语言就会对关键词显示不同颜色，并输出到工作区。

2. 代码自动补全也属于数据流处理加工的位置。文本从键盘输入就提示几个选项，用户可以选择一个即可补全代码，可加快工作。

3. 语法高亮显示支持几种流行的编程语言如:C,Java,python,HTML,CSS,JavaScript 和 SQL。而代码自动补全只支持 HTML，CSS，JavaScript 和 SQL 语言，用户可以在菜单选择使用什么语言对工作区进行处理。CodeMirror 参考网址

4.2.2 模块对象

- - 语法高亮与代码自动补全调用 ipcRenderer.js 的 setMode 函数，用户可在菜单选择一种编程语言或者由系统自动检查文件的扩展名来设置工作区的显示模式。

```
1  function setMode(language)
2  {
3    //对工作区设置语言
4  }
```

- - 设置主题调用 ipcRenderer.js 的 setThemes 函数来设置工作区的风格。

```
1  function setTheme(theme)
2  {
3    //对工作区设置某种主题
4  }
```

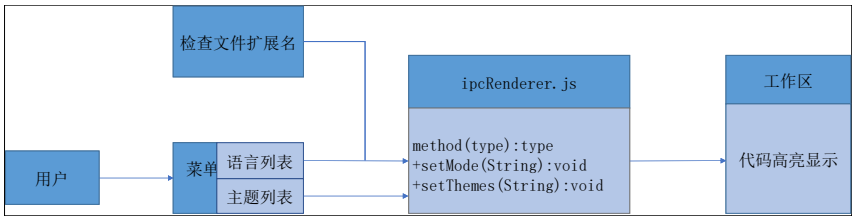


图 4.2 关系图

4.2.3 对象的触发机制

用户选择不同的编程语言，根据其语法格式和关键词对工作区提示几个选项供用户选择，并且将已输入的文本进行高亮显示。

4.2.4 对象的关键算法

Electron 框架使用 Web 前段技术，主要是由 HTML、CSS 和 JS 语言来开发。为了实现语法高亮与代码自动补全码自动补全功能，可选择使用开源的 CodeMirror 插件。使用 CodeMirror 必须在 HTML 代码引入 codemirror.css 和 codemirror.js。

```
1  <script src="src-nonconflict/ace.js" type="text/javascript" charest="utf-8"></script>
```

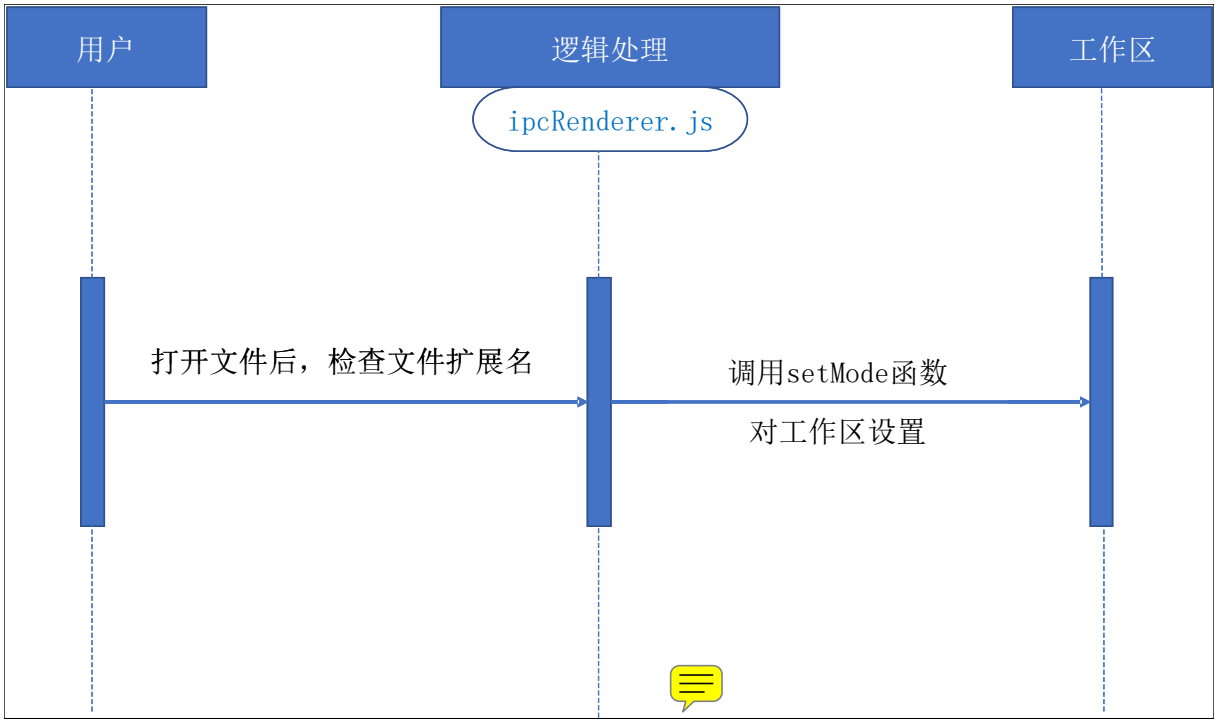
(1) 语法高亮与代码自动补全功能

```
<script src="src-nonconflict/ext-language_tools.js" type="text/javascript"
    charest="utf-8"></script>
```

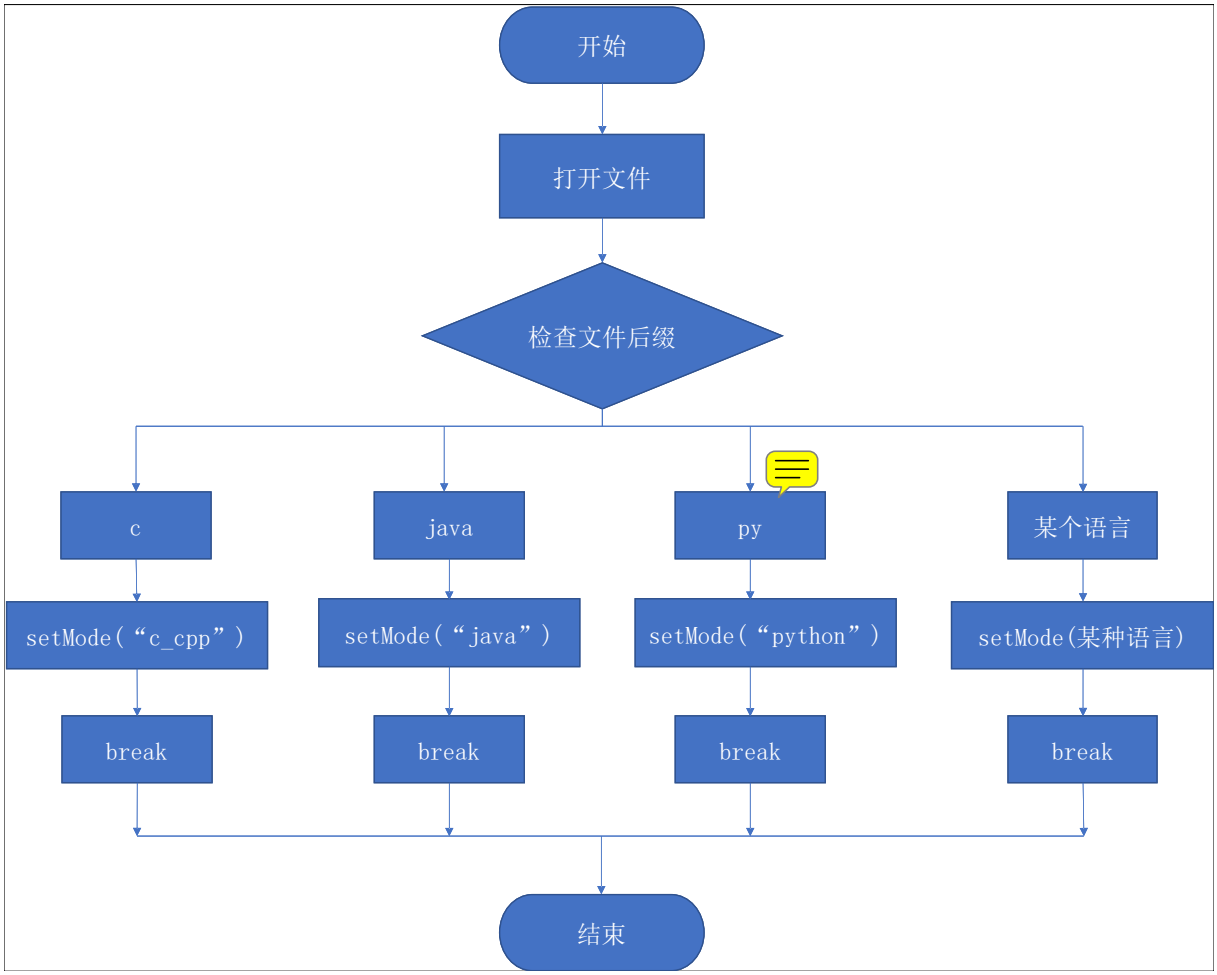
Ace-Editor-上手及调用方法

(2) 设置语言模式的方法：有 2 个触发机制：

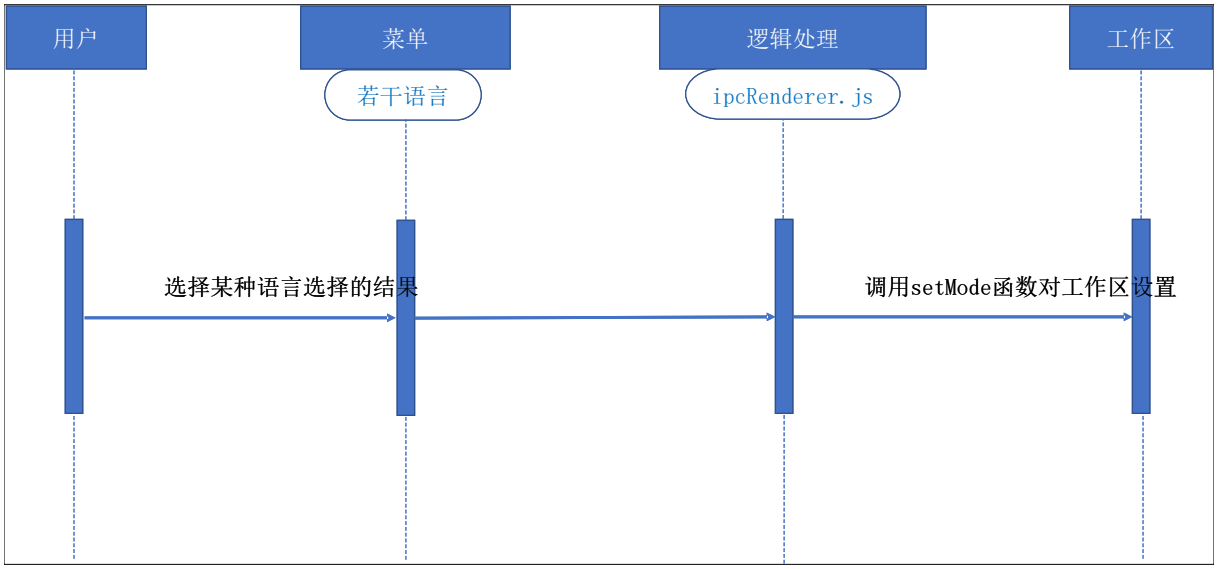
- 根据文件的扩展名：



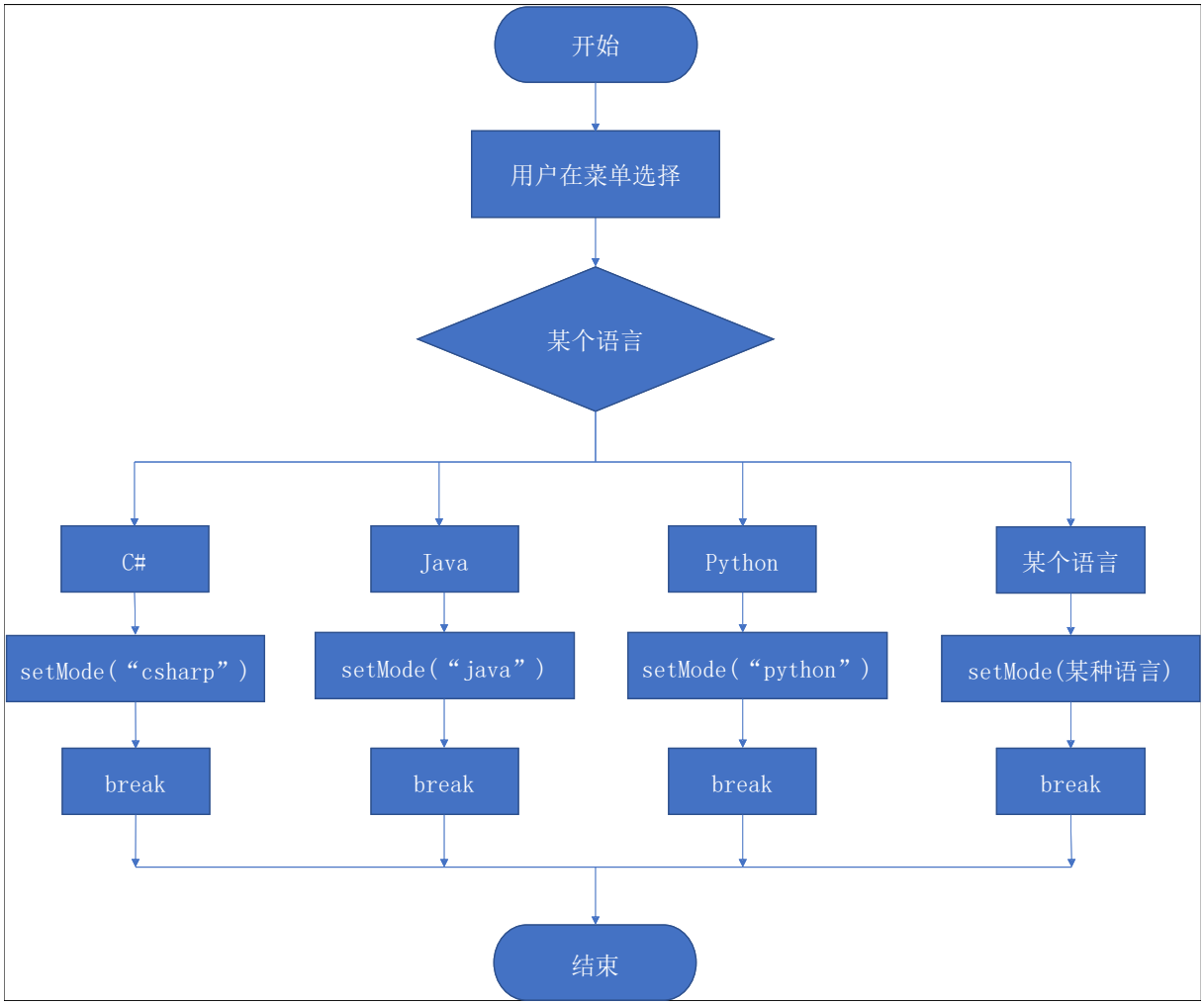
使用条件判断 switch case 语句来设置语言模式



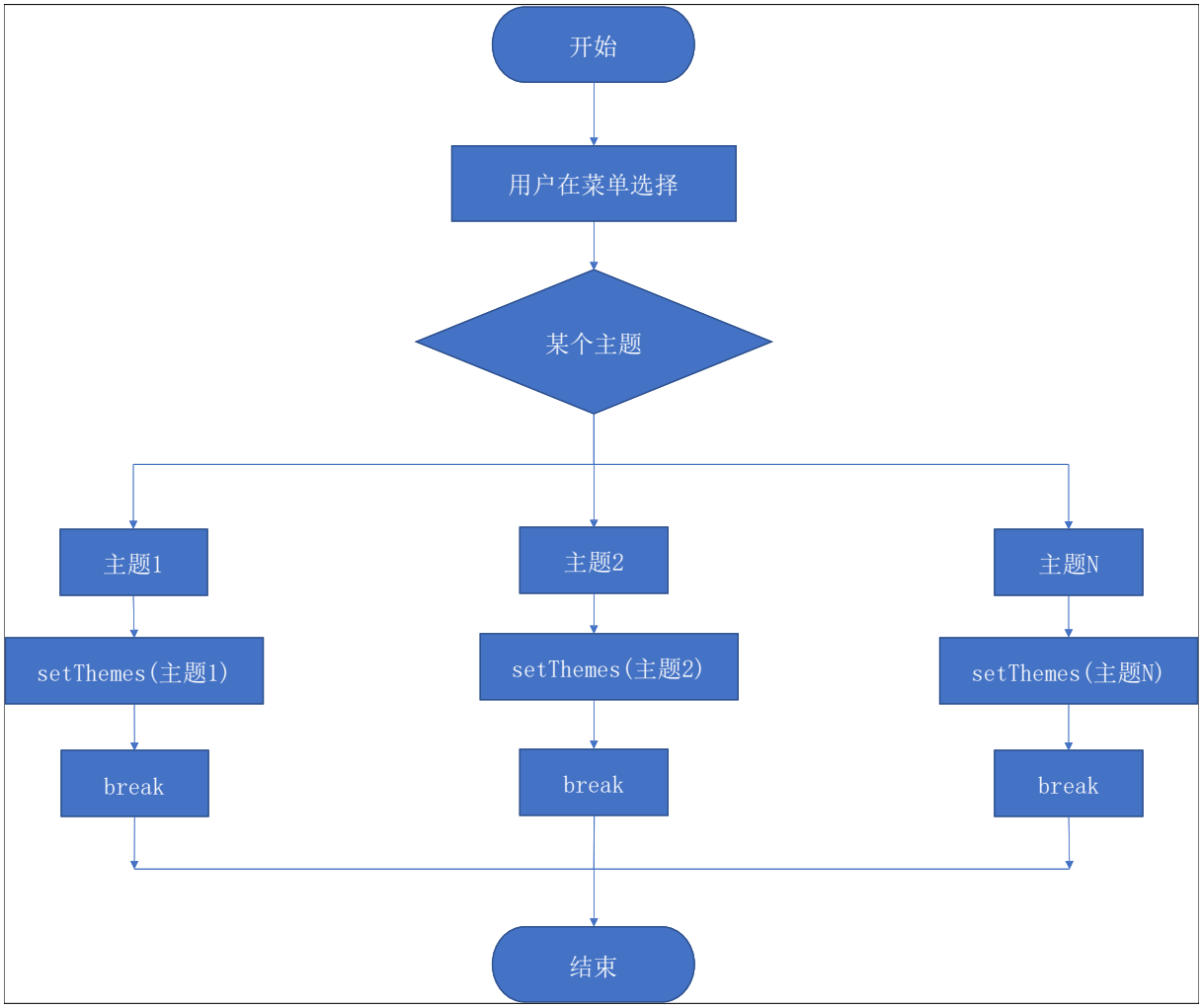
- 用户在菜单动手指定语言模式：



使用条件判断 switch case 语句来设置语言模式。



- 3. 切换主题的方法：
  使用条件判断 switch case 来设置不同的主题



4.3 文件编码格式与转换

4.3.1 模块功能

1. 文件编码属于数据流处理加工的位置。文本从键盘输入或者从文件读入后，若选择编码处理就会对文本做编码处理，并输出到工作区或将转码后的文件直接保存到本地。

2. 模块用途：
   用于处理编辑器对 GBK 和 UTF8 不同编码格式文本文件的处理，使编辑器按照实际使用情景选择 GBK 或者 UTF8 格式。编码模块属于数据流处理加工的位置。文本从文件读入后，根据所选的编码格式将文本文件读入到工作区（编辑栏）。同理，写入文件也是。

3. 模块功能：
   通过用户确定编辑器的编码格式，编码模块可以以 GBK 或 UTF-8 编码格式打开

或者保存相对应的文本文件（即源代码文件）。避免了不同读取二进制的时候采用的编码和最初将字符转换成二进制时的编码不一致。

4. 特别约定：
用于处理编辑器对 GBK 和 UTF-8 不同编码格式文本文件的处理，使编辑器按照实际使用情景选择 GBK 或者 UTF-8 格式。编码模块属于数据流处理加工的位置。文本从文件读入后，根据所选的编码格式将文本文件读入到工作区（编辑栏）。同理，写入文件也是。

4.3.2 模块对象（组件）

编码模块 *Transcoding* 具有高度的松耦合性，模块独立化高，仅需要工作区文本 *WorkSpace* 和本地文本文件 *LocalTextFile* 作为输入源和输出源。文本内容作为输入数据经过编码模块处理后，返回新的编码格式文本数据到工作区或保存到本地。
模块的定义:
文本文件和其他文件一样都是以二进制保存在计算机中。但是区别于二进制文件，文本文件的二进制格式是通过字符集确定的。相同字符串在不同编码格式下保存的二进制内容就不同。读取二进制的时候采用的编码和最初将字符转换成二进制时的编码不一致时，就会产生乱码。本系统中编码模块的就是避免不同编码格式导致的乱码现象产生，支持国内比较常用的 GBK（GB2312）和普遍推荐使用格式 UTF8 两种编码。

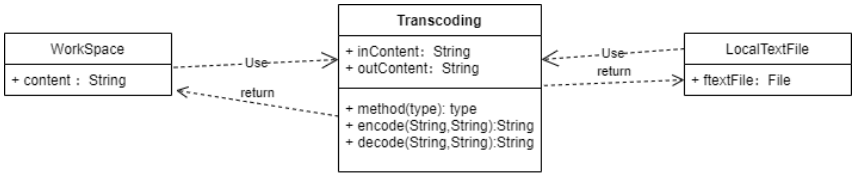


图 **4.3 E-R** 图

4.3.3 对象（组件）的触发机制

用户选择编码功能的不同格式，可以对工作区文本内容或本地文本文件执行相应编码操作。工作区文本内容和选定本地文本文件就会转码成之前选择的编码格式。
输入可分为文本文件的二进制流和编辑栏当前的文本字符串。输出同样分为存储二进制的文本文件和工作区的字符串。具体描述如下：

(1) 当输入为文件的二进制流时：经过编码模块判断后，采用选择的编码格式解码成字符串，输出到工作区。

(2) 当输入为工作区的文本时：编码模块先编码文本字符串为二进制流，然后再选择规定的格式解码为转码后的字符串，输出到工作区。

(3) 当输出为文件的二进制时：将编码后的二进制流写入到指定文件

(4) 当输出为工作区时：将转码后的字符串输出到工作区。输入和输出只能依照指定的编码格式才可以正常编解码，如 GBK 和 UTF-8。输入和输出的文件存储的二进制流都是依照写入时的编码的格式. 读取时如出现写入和读出的编码格式不一致时就会出现乱码。

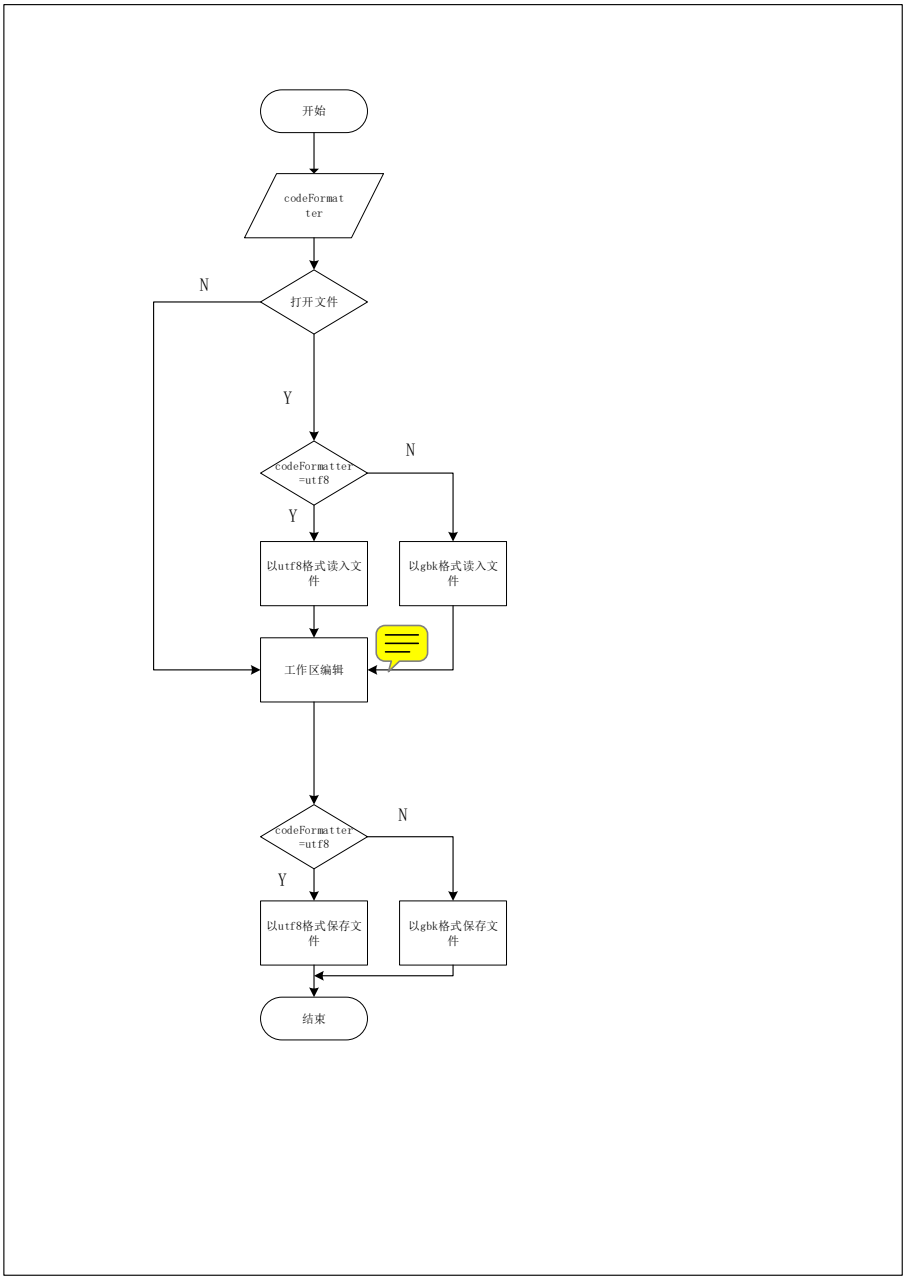


图 **4.4** 流程图

在 ipcRenderer.js 中使用 codeFormatter 表示编码格式，默认为 UTF-8。用户点击编码菜单栏，弹出下拉按钮 UTF 和 GB”。选择后，会在标签栏显示当前的编码格式。

系统通过 codeFormatter 来判断文件保存和打开的格式选择。因为系统默认的编码格式就是 UTF8，所以为 UTF 格式时，直接读取源文本文件并显示。若读取的文件为 GBK 格式就必须经过 GBK 格式的解码。由于 Node 环境自身不提供 GBK 编码，所以使用 iconv-lite 第三方库实现 GBK 的解码。将依赖引入工程后，使用 require() 获取 iconv-lite，即 const iconvLite = require("./iconv-lite") 在对 GBK 文件的读取时，选择 iconvLite.decode(fsData, "gbk") 解码。这样 GBK 文本文件即可正常显示。同理要保持写入文本的流的格式时，使用 iconvLite.encode(getContent(), "gbk") 或 iconvLite.encode(getContent(), "utf8") 将文本流编码后写入。

4.3.4 对象（组件）的关键算法

node.js 由于不支持 GBK 编码形式，所以通过引入 iconv-lite 模块解决中文乱码问题。const iconv = require('iconv-lite') 系统引入 iconv-lite 模块后，调用 iconv.encode() 和 iconv.decode() 即可完成转码操作。

五、 界面设计

工具栏界面与通常软件的工具栏设计大致相同。为了界面简介与方便考虑，我们将总体页面设计为工具栏界面与文本编辑界面，设计效果如下所示

图 **5.1**　总体界面设计

5.1　工具栏设计

工具栏主要为用户提供各种功能按钮，是各种函数的接口。

图 **5.2** 工具栏界面设计

(1) 文本管理按钮
对文本进行各种操作

- 打开按钮：弹出窗口打开文件
- 保存按钮：保存修改文件
- 打印按钮：将文件打印出来
- 退出按钮：退出该程序

(2) 编辑管理按钮
基本的文本操作

- 恢复按钮：恢复上一步的改动
- 剪切按钮：撤销上一步的改动
- 复制按钮：复制所选择的文本
- 粘贴按钮：将文本粘贴出来
- 删除按钮：删除所选文本

- 全选按钮：全选所有文本

(3) 视图管理按钮
对视图的操作

- 加载按钮：加载窗口
- 缩小按钮：对窗口进行尺寸的缩小
- 放大按钮：对窗口进行尺寸的放大
- 重置缩放按钮：重新定义窗口尺寸
- 全屏按钮：将窗口设置为全屏

(4) 语言选择按钮
此处列出候选语言，供用户选择，如图所示

(5) 主题选择按钮此处列出候选主题，供用户选择，如图所示

(6) 帮助按钮

5.2 文本编辑界面设计

文本编辑界面是提供语法高亮以及代码补全功能的区域。

1. 语法高亮：再选择合适的代码格式后，软件能够根据语法的规则自动实现语法高亮。
2. 代码补全：在选择合适的代码格式后，软件能够根据输入的关键字自动列出相应的补全候选项，共用户进行使用。

图 **5.3**　文本编辑界面

5.3　文件管理界面设计

此功能暂定，根据开发时间来判断是否要进行。