哈尔滨工业大学

**计算机科学与技术学院/国家示范性软件学院**

**2021年秋季学期**

**《软件过程与工具》课程**

**实验报告**

**Lab 1：增量模型应用+结对编程实践**

|  |  |  |
| --- | --- | --- |
| **姓名** | **学号** | **联系方式** |
| 马子豪 | 1190200908 | Email/手机号码:  1980209714@qq.com/18804608022 |
|  |  |  |
| 吴雄涛 | 1190200626 | Email/手机号码:  1368958396@qq.com/19825226985 |

目 录

[1 项目概述 1](#_Toc527037663)

[1.1 总体需求分析 1](#_Toc527037664)

[1.2 软件系统总体设计 1](#_Toc527037665)

[1.3 开发任务计划 1](#_Toc527037666)

[1.4 开发环境与工具 1](#_Toc527037667)

[2 第1轮迭代：基本系统 1](#_Toc527037668)

[2.1 迭代目标 1](#_Toc527037669)

[2.2 需求分析 1](#_Toc527037670)

[2.3 设计方案 2](#_Toc527037671)

[2.4 结对编程过程记录 2](#_Toc527037672)

[2.5 单元测试 3](#_Toc527037673)

[2.6 集成测试 3](#_Toc527037674)

[3 第2轮迭代：增量1 3](#_Toc527037675)

[3.1 迭代目标 3](#_Toc527037676)

[3.2 需求分析 3](#_Toc527037677)

[3.3 设计方案 3](#_Toc527037678)

[3.4 结对编程过程记录 4](#_Toc527037679)

[3.5 单元测试 4](#_Toc527037680)

[3.6 集成测试 5](#_Toc527037681)

[4 第3轮迭代：增量2 5](#_Toc527037682)

[5 项目总结 5](#_Toc527037683)

[5.1 增量模型应用总结 5](#_Toc527037684)

[5.2 结对编程总结 5](#_Toc527037685)

[文档全部完成之后，请在上述区域点击右键，选择“更新域”，在打开的对话框中选择“更新整个目录”]

# 项目概述

（1）项目名称：科学仿真计算器

（2）项目简要介绍：（200字之内）

本项目旨在参考Windows 系统提供的计算器功能， 运用Vue + elementUI制作一个自己的计算器，其中包括标准计算器功能和科学计算器功能，可以进行对于数字的加减乘除、乘方、取百分号、取相反数、三角函数计算、弧度角度制转换、阶乘运算，并最后将计算的结果以一个清晰明了的方式在网页端展现出来，同时允许客户可以在标准计算器和科学计算器之间切换。

## 总体需求分析

（1）项目要开发的软件的目的和意义（回答为什么要开发该项目）。

计算器作为传统计算机最基本的需求之一，在实现上经常存在诸多讨论，比如如何实现对多个括号的解析，如何完成多对重特殊数学运算符号的解析等。同时，此次项目旨在训练迭代开发能力和结对编程能力，适应两个人一起的开发环境。

（2）软件系统的涉众分析（列出该系统跟哪些相关人员有关，其关心的核心内容）。

该计算器的受众为普通大众，包括从小学到高中以及大学的学生，财会相关行业的人员，从事科学研究、需要使用计算器来简化计算过程的人员。本计算器关注的核心内容包括计算功能的基本实现，以及高级计算相关的函数等，包括对数运算、指数运算、阶乘运算、乘方运算等，能够使得日常生活中的计算变得更加简便快捷。

（3）软件系统的功能需求：可从用户视角来描述功能需求（即用户故事）。

从客户的角度来看，在满足基本的运算需求的基础上，需要计算器界面简单易读、见名知意，使得用户能够很快地上手。同时需要有满足基本的科学计算的能力，能够对常见的异常、错误情况进行友好的提醒或者处理。

（4）软件系统的非功能需求：包括性能要求、界面要求、技术指标等。

性能要求：需要在合理的时间内算出可行解，例如基本的算术运算的运算时间不超过2s。

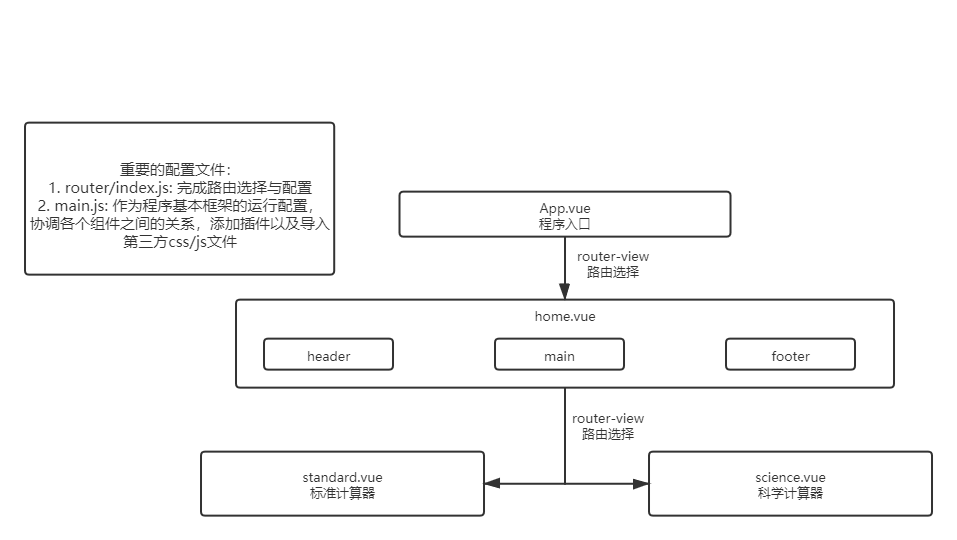
界面要求：界面简洁大方、容易理解、见名知意，能够给出友好的错误提示或者处理。

技术指标：能够达到基本的运算需求，在合适的时间内能够得到正确的答案。

## 软件系统总体设计

（1）软件系统功能结构设计方案：给出软件功能模块及模块间的关联关系。

本项目采用Vue.js框架，搭配ElementUI，旨在浏览器端呈现一个美观可用的计算器。其中程序框架的主要实现逻辑如下：



如图所示，软件主要分为四个模块和两个重要的配置文件：

App.vue: 作为程序的入口，通过router-view 进行路由选择，交给其他的components 去执行对应的业务逻辑。

router/index.js: 导入项目相关的组件，配置路由信息，完成页面路由的转发功能。

main.js: 负责导入App.vue入口文件，vue核心组件例如vue.js, vue-router, ElementUI，以及一些自定义的全局css。

home.vue: 作为首页，包含全局header，全局footer，以及页面主体main，其中main主体部分通过路由分发给两个vue组件进行渲染。

standard.vue: 标准计算器组件，包含了页面渲染以及逻辑运算功能，独立完成标准计算相关功能。

science.vue: 科学计算器组件，包含了页面渲染以及逻辑运算功能，独立完成科学计算相关功能。

（2）软件系统相关的数据结构设计：给出必要的数据结构设计，包括配置参数、缓冲数据结构等。

数据结构设计：由于整个项目体量不算大，整体的工作量体现在standard.vue和science.vue上，由于vue框架主要采用双向数据绑定的设计理念，在设计的时候主要涉及两个变量:

1. current 字符串类型，将当前用户的输入信息当做字符串进行存储

2. history 字符串类型，记录用户之前的计算结果，显示在界面上

## 开发任务计划

迭代过程的划分，各迭代计划完成的增量任务，进度安排。

（1）第1轮迭代的任务：

搭建项目框架，完成前端的设计和视图的渲染，实现路由选择两个计算器，同时完成基本的逻辑运算。

（2）第2轮迭代的任务：

针对第1轮迭代中的项目结果，进一步优化逻辑运算功能，使其具有一个完整的计算器的功能和体验。

（3）第3轮迭代的任务：

在逻辑完善的基础上，增加一个视图框展示实时计算结果，同时针对客户的错误输入进行友好的错误提示，并进一步完善前端UI和视图渲染，优化重构代码。

## 开发环境与工具

开发语言、开发环境与工具等。

开发语言为js + html + css， 其中js 采用了当下较为流行的前端框架Vue.js, css 主要采用ElementUI 进行美化

开发环境为Windows10 ，生产部署环境为云服务器 CentOS 7.6

工具为VScode+vue相关插件

# 第1轮迭代：基本系统

## 迭代目标

简要介绍本轮迭代的增量任务目标，所需完成的任务。

（1）任务1：利用Vue-cli 脚手架搭建一个完整的vue 项目。

（2）任务2：完成相关插件的导入，完成路由分发

（3）任务3：完成一个基本的计算器功能，包含标准计算器和科学计算器

## 需求分析

本轮迭代所开发模块的详细需求分析。

【注意】：仅仅针对本轮迭代的需求分析内容。

第1轮迭代主要是创建出一个基本可用的项目，包括基本的项目结构，UI 设计，基本的逻辑功能，将项目搭建好之后以方便后续迭代工作在此基础上舔砖加瓦。

## 设计方案

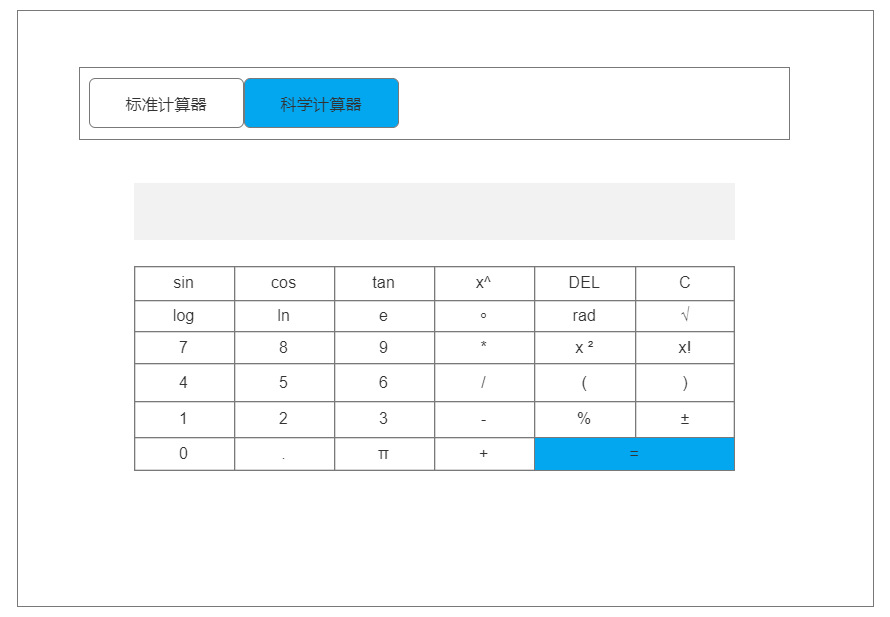
针对本轮迭代的开发内容给出设计方案：

（1）详细设计方案：模块功能结构设计，界面原型设计等。

模块功能结构设计与总体结构设计一致，这里不再多赘述。

界面原型通过Axure 设计如下：

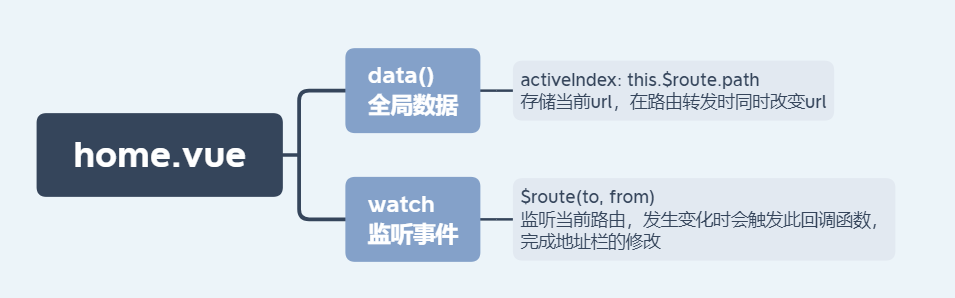




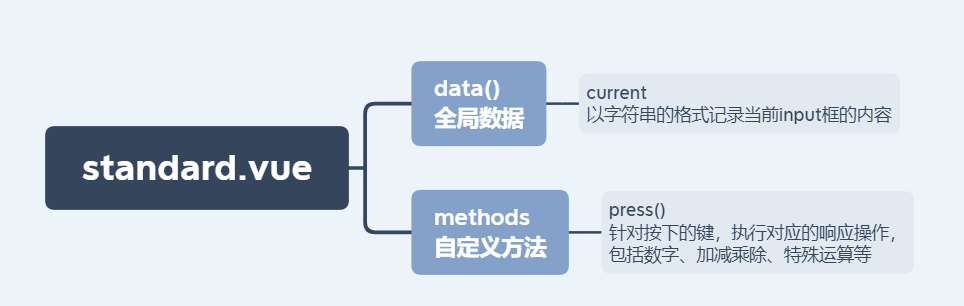
（2）模块相关的设计类（Design Class）：给出每个类的名字、属性、方法。

由于vue 文件中主要包括视图层和逻辑层，下面主要展示逻辑层的组织结构关系:

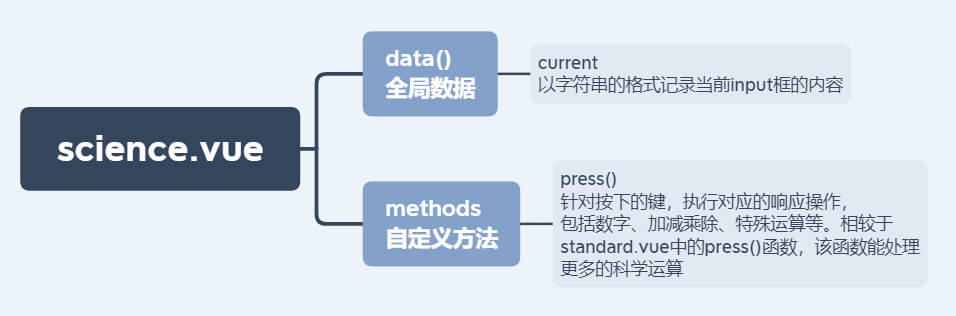
1. home.vue



2. standard.vue



3. science.vue



（3）所需的数据结构。

standard.vue和science.vue中主要涉及字符串current 变量来存储当前input 框的输入。

## 结对编程过程记录

**（1）角色切换与任务分工**

表1-1 第1轮迭代过程结对编程角色与任务分工

| 日期 | 时间(HH:MM - HH:MM) | 驾驶员角色 | 领航员角色 | 本段时间的任务 |
| --- | --- | --- | --- | --- |
| 10.13 | 13:00-17:00 | 马子豪 | 吴雄涛 | 编写程序入口与路由文件 |
| 10.14 | 9:00-11:30 | 马子豪 | 吴雄涛 | 编写首页与计算器组件 |
| 10.14 | 18:30-21:00 | 马子豪 | 吴雄涛 | 编写计算器的基础逻辑 |

【注意】该表格可自行增加更多的行。

**（2）工作日志**

由领航员负责记录，记录结对编程期间的遇到的问题、两人如何通过交流合作解决每个问题的。

表1-2 第1轮迭代过程结对编程工作日志

| 日期/时间 | 问题描述 | 最终解决方法 | 交流过程 |
| --- | --- | --- | --- |
| 10.14 | 组件调用出错 | 增加了组件逻辑关系描述 | 商讨了vue框架的使用技巧，理清各组件之间的逻辑 |

【注意】该表格可自行增加更多的行

**（3）结对编程工作现场照片**

请其他同学帮助拍摄结对编程现场照片至少2张。

第1次迭代结对编程现场照片1 第1次迭代结对编程现场照片2

## 单元测试

对本轮迭代所开发的每个类进行单元测试。

表1-3 第1轮迭代过程的单元测试记录

| 日期/时间 | 方法名 | 输入事件 | 触发事件 | 结论 |
| --- | --- | --- | --- | --- |
| 10.14 | route | 切换模式由标准到科学 | 模式切换为科学计算器 | 正确 |
| 10.14 | Press | 按下按键 | 触发对应事件 | 正确 |

【注意】该表格可自行增加更多的行

## 集成测试

对本轮迭代所开发功能模块进行测试。

表1-4 第1轮迭代过程的集成测试记录

| 日期/时间 | 功能名称 | 测试用例 | 测试结果描述 | 结论 |
| --- | --- | --- | --- | --- |
| 10.14 | 模式转换 | 按下科学计算器模式 | 切换为科学计算器 | 正确 |
| 10.14 | 按键功能测试 | 按下3、+、\*、/、- | 文本框内出现3+\*/- | 正确 |
| 10.14 | 基础运算测试 | 输入2+3-4\*4/2，按下= | 输出-3 | 正确 |
| 10.14 | 相反数测试 | 输入2，按下 | 输出-2 | 正确 |
| 10.14 | 平方测试 | 输入3和-3，按下 | 输出9和9 | 正确 |
| 10.14 | 删除测试 | 输入58，按下Del | 输出5 | 正确 |
| 10.14 | 清空测试 | 输入890+123，按下C | 输出空白 | 正确 |
| 10.14 | 括号功能测试 | 输入(1+2)\*3，按下= | 输出9 | 正确 |
| 10.14 | 三角函数测试 | 输入1，按下sin、cos、tan | 0.8414709848078965、0.5403023058681398、1.5574077246549023 | 正确 |
| 10.14 | 次方测试 | 输入2^2，按下= | 输出4 | 正确 |
| 10.14 | 对数测试 | 输入2，按下log与ln | 0.3010299956639812、0.6931471805599453 | 正确 |
| 10.14 | e测试 | 按下2，按下e | 7.38905609893065 | 正确 |
| 10.14 | 弧度度数转化检测 | 按下2，按下rad和度数 | 0.03490658503988659、114.59155902616465 | 正确 |

【注意】该表格可自行增加更多的行

# 第2轮迭代：增量1

## 迭代目标

简要介绍本轮迭代的增量任务目标，所需完成的任务。

（1）任务1：增强标准计算器和科学计算器的计算能力，使其能够理解和计算更复杂的语句

## 需求分析

本轮迭代所开发模块的详细需求分析。

【注意】：仅仅针对本轮迭代的需求分析内容。

在第一轮迭代中，基本的项目框架已经搭建成功，但是在逻辑处理上仍然存在很多漏洞，具体体现在:

很多科学运算，例如x2 sin cos ln log , 只是对当前的current进行简单的数值转换计算，并不能处理一连串的input方框的内容，也不能对字符串中的嵌套特殊运算进行拆分，因此需要进一步优化。

## 设计方案

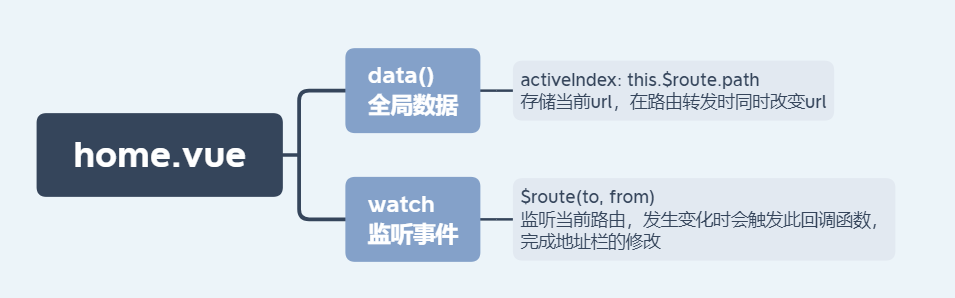
针对本轮迭代的开发内容给出设计方案：

（1）详细设计方案：模块功能结构设计，界面原型设计等。

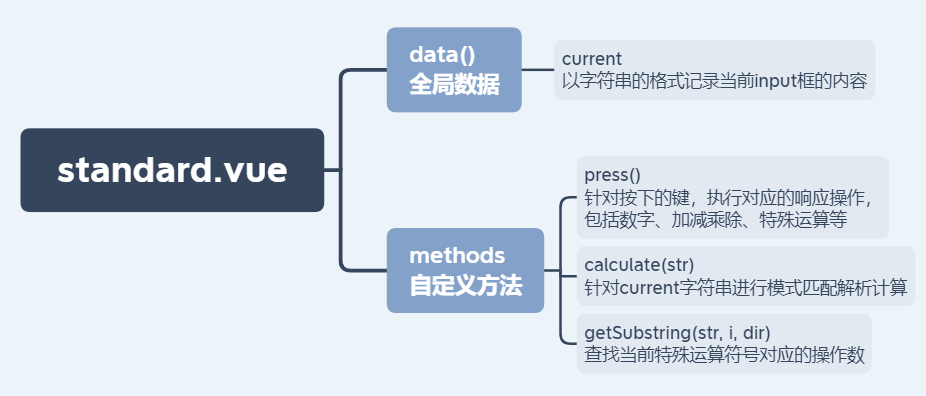
模块功能结构设计和界面原型设计与迭代1基本相同，某些具体类属性上发生了一些变化。

（2）模块相关的设计类（Design Class）：给出每个类的名字、属性、方法。

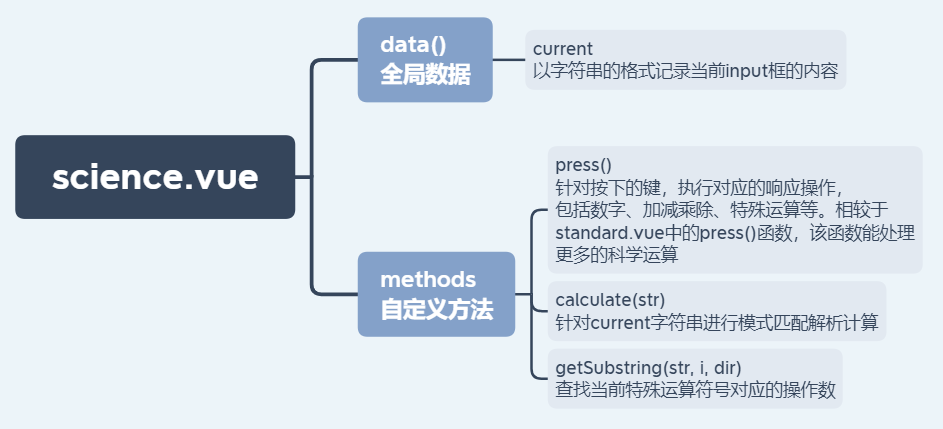
1. home.vue



2. standard.vue



3. science.vue



（3）所需的数据结构。

standard.vue和science.vue中主要涉及字符串current 变量来存储当前input 框的输入。

## 结对编程过程记录

**（1）角色切换与任务分工**

表2-1 第2轮迭代过程结对编程角色与任务分工

| 日期 | 时间(HH:MM - HH:MM) | 驾驶员角色 | 领航员角色 | 本段时间的任务 |
| --- | --- | --- | --- | --- |
| 10.15 | 13:00-17:30 | 吴雄涛 | 马子豪 | 完成getSubString函数功能 |
| 10.15 | 19:00-22:00 | 吴雄涛 | 马子豪 | 完成calculate函数的功能 |
| 10.16 | 10:00-11:30 | 吴雄涛 | 马子豪 | 测试 |

【注意】该表格可自行增加更多的行。

**（2）工作日志**

由领航员负责记录，记录结对编程期间的遇到的问题、两人如何通过交流合作解决每个问题的。

表2-2 第2轮迭代过程结对编程工作日志

| 日期/时间 | 问题描述 | 最终解决方法 | 交流过程 |
| --- | --- | --- | --- |
| 10.15 | 抽出子串会包含左侧多余字符 | 在切割时多切割了左侧字符，左侧加一 | Debug过程中一步步查看 |
| 10.15 | 在计算过程中^和%同时出现时会出错 | 转换计算优先级，将%放在最优先的地方 | 交流之后认为在不同计算符号之间应该有优先级问题，所以在计算时需要按顺序进行对应计算 |
| 10.16 | 当出现很小的数字用e表示时会出错 | 将e作为E输入 | 判断是因为e会判定为计算的e而直接转换为数字 |

【注意】该表格可自行增加更多的行

**（3）结对编程工作现场照片**

请其他同学帮助拍摄结对编程现场照片至少2张。

第2次迭代结对编程现场照片1 第2次迭代结对编程现场照片2

## 单元测试

对本轮迭代所开发的每个类进行单元测试。

表2-3 第2轮迭代过程的单元测试记录

| 日期/时间 | 类名：方法名 | 输入数据 | 输出数据 | 结论 |
| --- | --- | --- | --- | --- |
| 10.16 | Calculate | (100-10)^2 | 8100 | 正确 |
| 10.16 | getSubString | (100-10)^100, 8, -1 | (100-10) | 正确 |

【注意】该表格可自行增加更多的行

## 集成测试

对本轮迭代所开发功能模块进行测试。

表2-4 第2轮迭代过程的集成测试记录

| 日期/时间 | 功能名称 | 测试用例 | 测试结果描述 | 结论 |
| --- | --- | --- | --- | --- |
| 10.16 | 不带括号的复杂运算 | cos+lnE+2^2+100% | 5 | 正确 |
| 10.16 | 带括号的复杂运算 | (1+1)^(1+100%) | 4 | 正确 |
| 10.16 | 嵌套的复杂运算 | ((-2)^2-2)^(1+100%) | 4 | 正确 |

【注意】该表格可自行增加更多的行

# 第3轮迭代：增量2

## 迭代目标

简要介绍本轮迭代的增量任务目标，所需完成的任务。

（1）任务1：优化界面UI，重构优化代码，增加相关文档描述

（2）任务2：增加一个显示框，展示当前实时计算结果，增强用户体验感。同时对于用户的错误输入输出合适的报错信息

## 需求分析

本轮迭代所开发模块的详细需求分析。

【注意】：仅仅针对本轮迭代的需求分析内容。

在第二轮迭代中，一个功能完整的计算器已经基本完成，但是在用户体验上仍然有可以优化的地方，例如

1. 增加一个实时显示框，展示当前的计算结果。2.对于用户的错误输入进行友好的错误提示

## 设计方案

针对本轮迭代的开发内容给出设计方案：

（1）详细设计方案：模块功能结构设计，界面原型设计等。

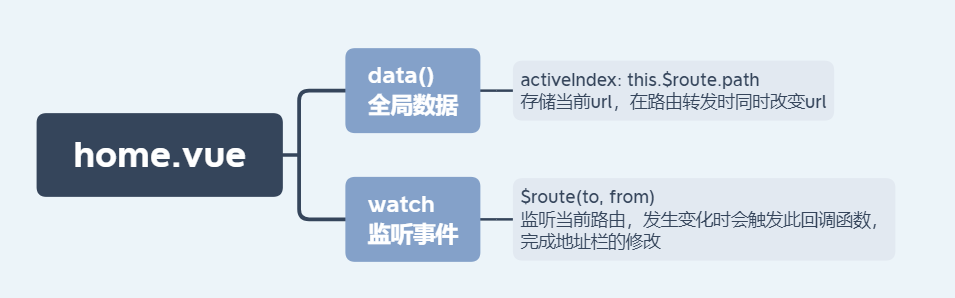
模块功能结构设计与迭代1基本相同。在原型图的设计上有一些变化，具体为在input 框上增加一个小的实时结果展示框



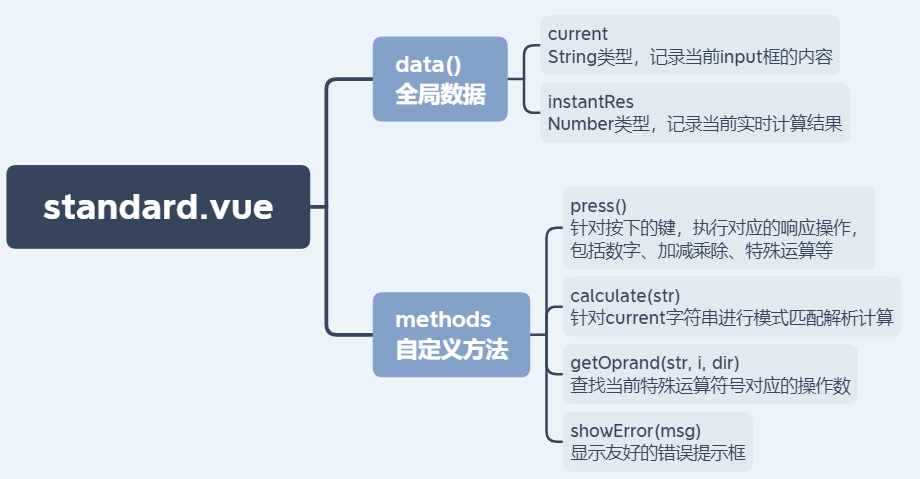


（2）模块相关的设计类（Design Class）：给出每个类的名字、属性、方法。

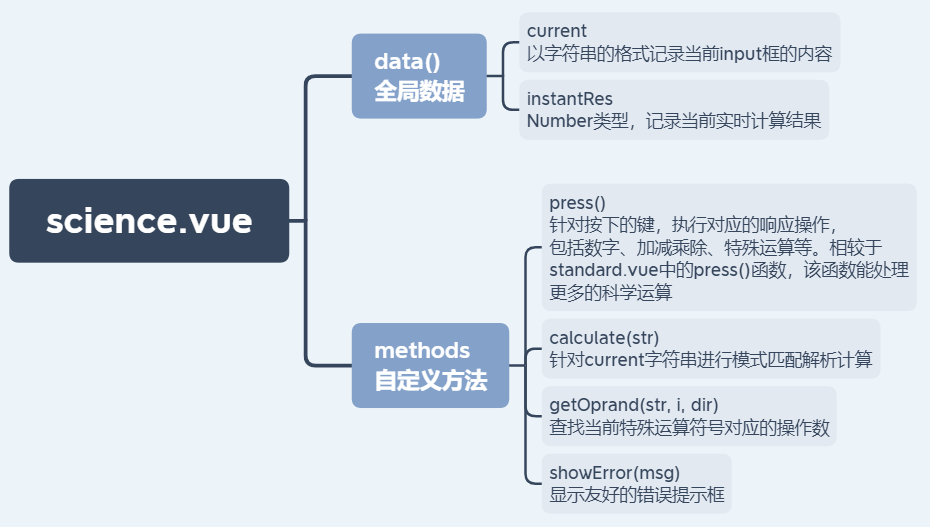
1. home.vue



2. standard.vue



3. science.vue



（3）所需的数据结构。

standard.vue 和science.vue 中主要涉及字符串current 变量来存储当前input 框的输入，以及Number类型变量instantRes存储当前即使运算结果

## 结对编程过程记录

**（1）角色切换与任务分工**

表2-1 第2轮迭代过程结对编程角色与任务分工

| 日期 | 时间(HH:MM - HH:MM) | 驾驶员角色 | 领航员角色 | 本段时间的任务 |
| --- | --- | --- | --- | --- |
| 10.17 | 19:00-21:00 | 马子豪 | 吴雄涛 | 编写showError()方法 |
| 10.18 | 10:00-11:30 | 马子豪 | 吴雄涛 | 增加实时显示框 |
| 10.18 | 15:00-17:30 | 马子豪 | 吴雄涛 | 测试功能 |

【注意】该表格可自行增加更多的行。

**（2）工作日志**

由领航员负责记录，记录结对编程期间的遇到的问题、两人如何通过交流合作解决每个问题的。

表2-2 第2轮迭代过程结对编程工作日志

| 日期/时间 | 问题描述 | 最终解决方法 | 交流过程 |
| --- | --- | --- | --- |
| 10.18 | 计算次方时出错 | 转化数值为String | 在计算次方时，计算的值是对应的计算结果为数值型，而Calculate函数需要以String型输入 |

【注意】该表格可自行增加更多的行

**（3）结对编程工作现场照片**

请其他同学帮助拍摄结对编程现场照片至少2张。

第2次迭代结对编程现场照片1 第2次迭代结对编程现场照片2

## 单元测试

对本轮迭代所开发的每个类进行单元测试。

表2-3 第2轮迭代过程的单元测试记录

| 日期/时间 | 类名：方法名 | 输入事件 | 输出事件 | 结论 |
| --- | --- | --- | --- | --- |
| 10.18 | showError | 输入错误 | 弹出错误信息的提示 | 正确 |

【注意】该表格可自行增加更多的行

## 集成测试

对本轮迭代所开发功能模块进行测试。

表2-4 第2轮迭代过程的集成测试记录

| 日期/时间 | 功能名称 | 测试用例 | 测试结果描述 | 结论 |
| --- | --- | --- | --- | --- |
| 10.18 | 实时显示结果 | 1+1+5 | 实时框里显示7 | 正确 |
| 10.18 | 错误提示 | 1+(1+1 | 弹出错误提示 | 正确 |

【注意】该表格可自行增加更多的行

# 项目总结

## 增量模型应用总结

对增量模型应用过程进行总结，包括但不限于以下几方面的内容：

* 增量划分是根据工作量划分还是根据用户需求来划分？

我们的增量划分时根据用户的实际需求划分，以优化用户的实际体验。

* 第1个增量你们团队是如何确定的？

我们想制作一个可以实现基本功能的计算器，搭载基本的项目结构与 UI 设计，让用户可以进行一定的计算

* 你认为增量模型的好处在哪里？有什么不好之处吗？它的适用场合是什么？

好处：

1. 开发人员分配灵活，在项目开始阶段不用投入大量人力资源；

2. 如果核心产品很受欢迎，则可增加人力实现下一个增量；

3. 可先发布部分功能给客户，对客户起到镇静剂的作用。

缺点：

1. 并行开发构件有可能遇到不能集成的风险，软件必须具备开放式的体系结构；

2. 增量模型的灵活性可以使其适应这种变化的能力大大优于瀑布模型和快速原型模型，但也很容易退化为边做边改模型，从而是软件过程的控制失去整体性。

适用场合：

1. 进行已有产品升级或新版本开发；

2. 设计与完成对完成期限严格要求的产品；

3. 对所开发的领域比较熟悉而且已有原型系统。

## 结对编程总结

对结对编程过程进行总结，包括但不限于以下几方面的内容：

* 与2人分别编程相比，是否体验到编程效率的提高、编程质量的提高？

结对编程与2人分别编程相比，更容易使两人相互监督，相互促进，相互学习。结合优势与劣势，互相学习编程技巧，提升编程效率。同时两人一起发现问题，减少bug ，提升编程质量。

* 你认为，你们2人本次合作是否真的提高了效率和质量？有什么成功的体验和需要反省的不足或教训？

合作确实提高的编程的效率和质量。

成功的体验：在结对编程过程中，如果进行合理的分工，可以扬长避短，结合双方优势，推进编程进度。同时在结对编程时，由两人同时检查，可以更好地避免bug ，提升编程质量。

不足与教训：在结对编程时，需要有合理的计划。编程中的沟通与交流也是必须的。

* 你认为结对编程的优势在哪里、有什么不适应之处？它的适用场合是什么？

优势：

1. 团队工作能增加成员的工作积极性。在面对问题的时候，一起分担，共同尝试新的策略。

2. 两个人一起工作可以互相配合，互相监督。

3. 互相学习编程技巧。在编程中，相互讨论，可以更快更有效地解决问题，互相请教对方，可以得到能力上的互补。

4. 让资深开发者和新手一起工作，可以让新人更快上手。

5. 两人互相监督工作，可以增强代码和产品质量，并有效的减少 BUG。

劣势：

1. 与合不来的人一起编程容易发生争执，不利于团队和谐。

2. 经验丰富的老手可能会对新手产生不满的情绪。

3. 一山不容二虎，开发者之间可能就某一问题发生分歧，产生矛盾，造成不必要的内耗。

4. 开发人员可能会在工作时交谈一些与工作无关的事，分散注意力，造成效率低下。

适用场合：规模小，时间紧，需求变化多，质量要求高的软件开发