中 国 地 质 大 学

本科生课程论文

课 程 名 称 生产实习

教 师 姓 名 胡成玉

学 生 姓 名 苏浩文、高天翔、常文瀚

学 生 班 级 191181

所 在 学 院 计算机学院

完 成 日 期 2022 年 2 月 20 日

目录

一、实习项目简介

1.1 快应用与 Kaldi 实践应用

课程内容主要涉及开源、快应用知识、代码编写以及 Kaldi 基础,主要为开发者介绍快应用的应用场景，并详细介绍快应用开发工具的使用，同时让开发者掌握开发快应用的必须点，包括组件、样式、脚本。

1、了解快应用的场景介绍。

2、熟悉快应用开发工具。

3、掌握开发快应用的必须点：组件，样式，脚本。

1.2 项目主要技术

1.2.1 Kaldi

Kaldi 是目前全球最流行的开源语音识别工具集，能让智能物联通过语音交互方式呈现的基础，是目前被业界公认的语音识别框架基石。Kaldi 在小米产品中的 ASR 中被广泛使用。目前国内外语音助手如小爱同学、苹果的 Siri、亚马逊的 Alexa 等的底层框架也都采用 Kaldi。

Kaldi 语音识别工具主要关注子空间高斯混合模型的建模，是一个开源的语音识别系统，主要用 C++实现。Kaldi 的目标是提供一个易于理解和拓展的更先进的识别语音系统。Kaldi 具备了如下几个重要的特性：整合了线性代数拓展、整合了有限状态转换器、代码设计易于扩展、开放开源协议、完整的语音识别系统搭建工具。

1.2.2 快应用

快应用简介

快应用是一种基于行业标准开发的新型免安装应用，其标准由主流手机厂商组成的快应用联盟联合制定。开发者开发一次即可将应用分发到所有支持行业标准的手机运行。

快应用具有以下特点与优势：

对开发者而言，成本低：使用相对简单的 JS 和 CSS 开发语言，代码量仅占安卓原生应用的 1/5。如有小程序，可直接转换，最快 2 天可完成；流量高：摆脱对某个 APP 的依赖，实现跨平台直接跳转；打通手机终端多场景；方便业务提升：直达服务/内容，体验好路径短，提升业务数据，拉新、促活、提留存。对用户而言，不用安装：即点即用，将应用图标添加至桌面即可秒开服务；节省空间：体积小，占用内存少，提升手机流畅度。入口便捷：桌面图标、负一屏、快应用中心、应用市场等均可找到快应用。

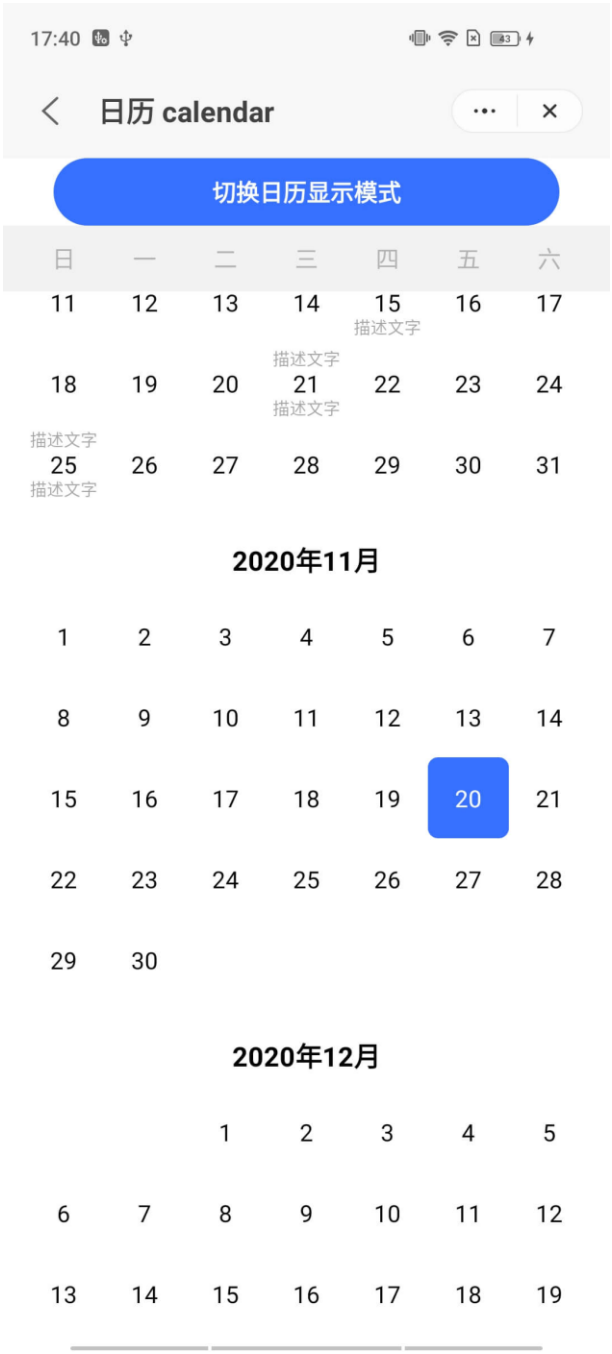
QAUI 组件介绍

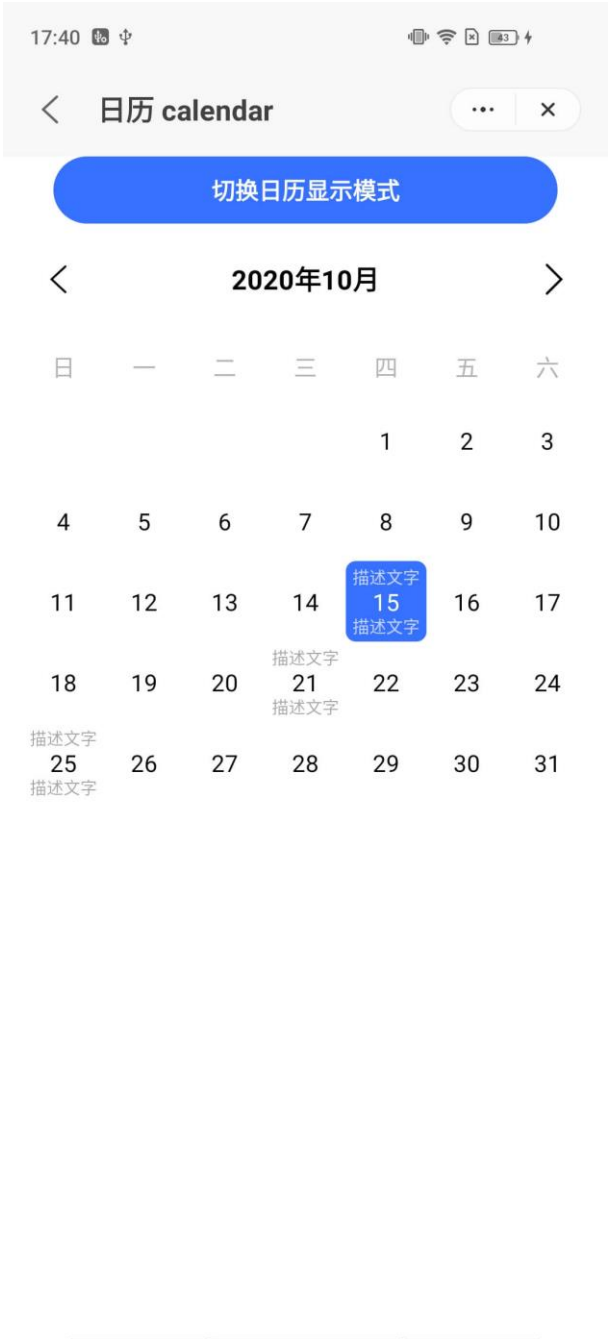
QaUI 组件库即快应用官方组件库。提供的基础组件包含图标、按钮、容器、标题栏、搜索栏、模块标题、分隔符、页脚；提供的表单组件包含单选按钮、单选按钮组、多选按钮、多选按钮组、输入框、多行输入框、筛选栏、验证码 、单选开关、滑动条、日历；提供的操作反馈包含通告栏、气泡菜单、异常页、结果页、加载进度条；提供的数据展示组件有列表步骤条、折叠面板、画廊；提供的高级组件包含导航栏添桌浮层、横向选项卡、纵向选项卡、字母检索表。本文选取部分项目中使用的组件进行介绍。

日历 calendar

描述：用于查看和选择日期

使用效果





组件属性

属性	类型	默认值	说明
type	String	'default'	日历显示类型，可选值有default,list
range	Array	['2020-10','2021-10']	日历显示范围
description	Array	[]	日期上显示相关描述文字
desc.date	String	‘’	要显示文字的日期，示例'2020-12-10'
desc.top	String	‘’	日期上面要显示的文字
desc.bottom	String	‘’	日期下面要显示的文字

组件事件

事件名称	事件描述	返回值
tap	日期点击	当前点击日期的数据

通告栏 notice

描述：适用性广泛的通知栏，可用于通知的静态展示以及动态展示。

使用效果

组件属性

属性	类型	默认值	说明
type	String	'normal'	通告栏类型，可选值有: 'normal' / 'warning' / 'transparent'
noticeText	String	''	通告栏文字
textColor	String	''	自定义通告栏文字颜色
bgColor	String	''	通告栏背景颜色,不建议使用 rgba，建议背景色为文字颜色乘以 0.15 透明度之后的 16 进制颜色值
leftIcon	Object	{}	左侧图标对象，具体说明见下方文档
rightIconType	String	'close'	通告栏右侧按钮图标，可选值有: 'close' / 'link'
rightIcon	Object	{}	右侧图标对象，具体说明见下方文档
scrollable	Boolean	false	是否滚动播放
speed	Number	20	滚动速度
scrollTimes	Number	5	滚动播放次数,当值为-1 时，则为无限滚动

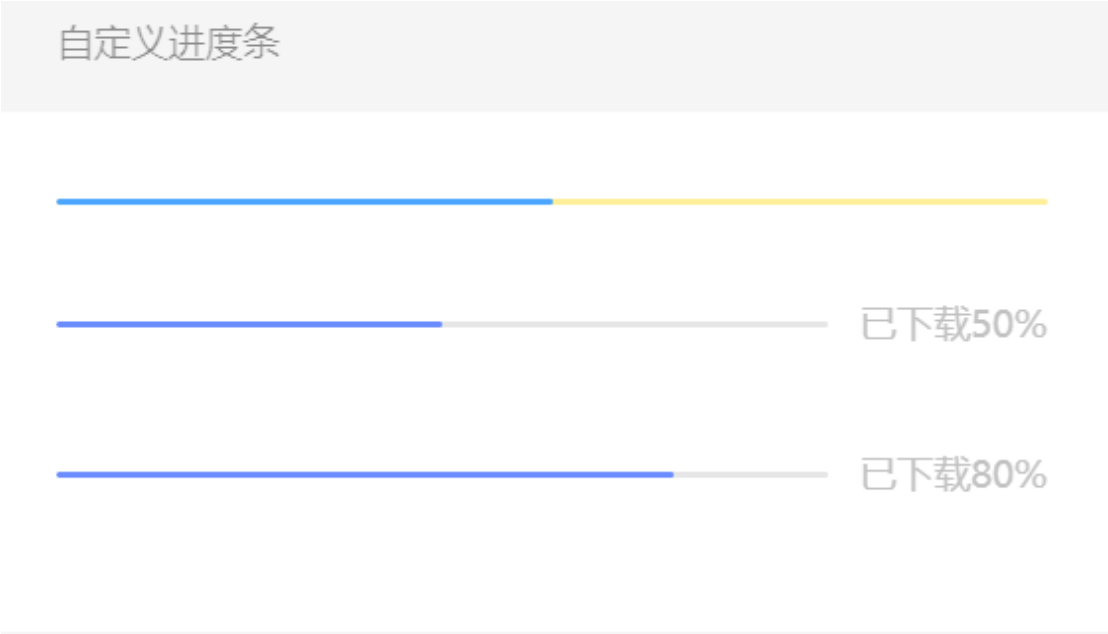
组件事件

事件名称	事件描述	返回值
linkTap	当 rightIconType 为'link'时，点击触发	event

进度条 progress

描述：用于展示加载进度，告知用户当前状态。

使用效果



组件属性

属性	类型	默认值	说明
percentage	Number	50	进度条百分比，可选值 0-100，可以为小数
progressWidth	String	'100%'	进度条组件的宽度(宽度包含文字的宽度，支持百分比和 px)
strokeHeight	Number	2	进度条的高度
isContentShow	Boolean	false	是否显示右侧文字

format	Function	(percentage) => percentage + '%'	右侧描述文字的模板函数
contentStyle	Object	{}	自定义右侧文字的样式
strokeColor	String	'#456fff'	进度条的颜色
trailColor	String	'rgba(0, 0, 0, 0.12)'	未完成分段的颜色

二、项目设计与实现

2.1 项目需求分析

基本需求：

1. 支持 Kaldi 语音输入
2. 支持修改代办事项
3. 支持添加完成时间

进阶需求：

1. 样式优化
2. Deadline 提醒
3. 增加统计界面
4. 代办事项状态修改

2.2 任务分工

高天翔

任务分工与项目代码合并

UI 设计、样式调整与优化

Demo 功能设计与用户体验优化

任务增加、修改、删除

PPT 的修改完善

实习报告的编排与撰写

苏浩文

引入 QaUI 快应用组件

增加日历查看功能

增加任务进度功能

实习报告的撰写

常文瀚

快应用开发模板&资料查询

答辩 PPT 的编写

实习报告的撰写与格式调整

2.3 项目平台

开发平台：Windows 10

快应用平台版本：1070

项目版本：2.2.1

涉及代码语言规范： JavaScript（ES6）、HTML 、CSS

2.4 前端 UI 设计

2.4.1 UI 设计模式

UE/UX，即 User Experience，翻译过来就是用户体验，它是指用户在使用产品过程中的个人主观感受，通俗地讲就是用户用得舒服不舒服。UE 设计师关注用户在使用前、使用过程中和使用后的整体感受，包括用户行为、情感和成就等各个方面，其目的是保证用户对产品的使用体验有正确的预期，了解他们的真实期望和目的，并以此作为依据对产品的核心功能设计进行修正，保证其与人机界面之间的协调工作。UED，即 User Experience Design，翻译过来就是用户体验设计，旨在发掘通过设计来提升用户使用产品的体验。用户体验是用户对产品产生的整体感受。

考虑到快应用的面向用户，采用 UCD(User Centered Design)，即以用户为中心的设计。 这种设计思维模式强调在产品设计过程中，从用户角度出发进行设计。让用户成为第一。

2.4.2 图标设计

图标设计原则

在移动应用中，图标通常分为两种：一种是应用图标，也就是我们在屏幕上看到的图标，点击它可以进入到移动应用中；还有一种是功能图标，它是存在于移动应用界面中，起到表意功能、取代文字或辅助文字的作用，例如搜索栏内的放大镜图形，即使不展示文字信息，用户也能知道它代表搜索功能。

由此可见，应用图标的设计风格可以有多种形式。就整个流行趋势来看，扁平化和轻质感的设计风格正在逐渐覆盖整个移动互联网，不管是应用界面还是图标，都很难再见到拟物化的身影。因此，快应用决定选用扁平化风格的纯平面面型图标。

扁平化设计是将拟物化拍扁，放弃一切装饰元素，呈现给用户最直接和最简洁的内容。它缺少拟物化设计那样华丽的视觉特效，却带给用户一个没有视觉噪音干扰的画面，以此减轻用户的视觉负担，使用户更加专注于内容本身。扁平化设计具有轻快、明亮、极简、高效等特点；纯平面图标通常指纯色的剪影图标，具有简洁大方、视觉识别度良好，色彩明朗和设计感强烈等特点；面形图标是平面的图形结构，因此它的视觉占比最大化，具有强烈的视觉表现力，适合用于应用界面的主要功能入口。

同时，图标不是单独的个体，它存在于应用界面的各个视图中，相同功能模块里展示的图标外观应该一致，例如，地图里定位与导航应统一使用同样尺寸、风格的图标。只有具有统一风格的系列图标，才能带给用户统一的视觉体验。当开始绘制图标，确立一个图标的设计风格后，接力的图标必须延展其风格设定，包括造型规则、国角大小、线框粗细、图形样式和个性细节等元素，用统一的设计语言贯穿整套图标设计。

考虑到成本与绘制难度，快应用的图标选择在阿里巴巴矢量图标库（Iconfont，http://www.iconfont.cn）下载，Iconfont 提供阿里旗下所有产品的图标下载服务，每一个图标都支持多尺寸和多格式下载。

应用图标展示

在手机屏幕上，统一的外观形状能画面更加整齐，应用图标与应用图标之间和谐共处，而不规则外观的应用图堆砌在一起时就会让画面显得混乱。在快应用展示界面中，应用图标都有着统一的圆形外观，因此快应用的应用图标选择了扁平化的圆形图标，简洁美观大方。

2.4.3 状态栏

状态栏始终固定在整个屏幕的上边缘，它展示了关于设备及其周围环境重要信息，如移动运营商、网络信号、时间、电量等。快应用的默认状态栏为白色，适用于颜色较深的应用。大部分应用的状态栏与界面风格设计融为一体，将导航栏的颜色延伸至状态栏。快应用决定根据自身特点，选择与背景图片相近的蓝色作为背景色，形成较大的视觉反差，给人以视觉冲击。

三、项目功能模块

3.1 总体设计与代码结构

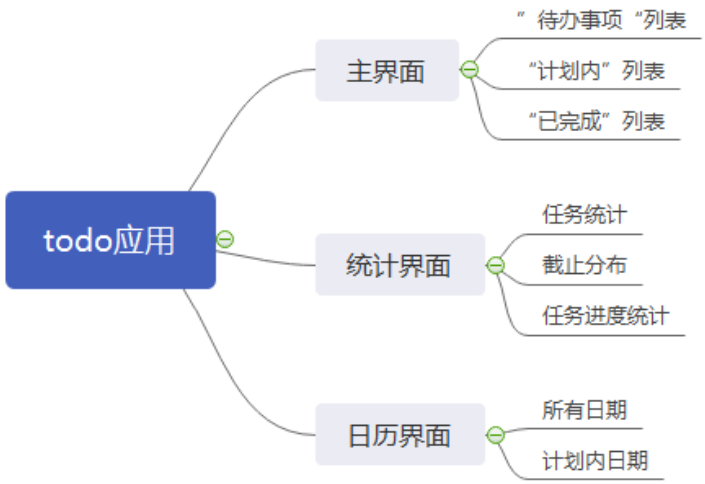


图 1 项目功能模块示意图

```
— src
│    ├── Common          # 公用的资源(Images/Styles/字体...)
│    │    ├──images      # 存储 png/jpg/svg 等公共图片资源
│    │    └──css         # 存放 less/css/sass 等公共样式资源
│    ├── Components      # 项目内使用的各种组件定义
│    │    ├──button      # 按钮组件
│    │    ├──icon        # 图表组件
│    │    ├──notice      # 告示栏组件
│    │    ├──progress    # 进度条组件
│    │    └──calender    # 日历组件
│    ├── Input           # 事项编辑页面代码
│    ├── MainPage        # 主页面代码
│    ├── app.ux          # 应用程序代码的入口文件
│    └── manifest.json   # 配置应用基本信息
└── package.json         # 定义项目需要的各种模块及配置信息
```

图 2 代码结构图

3.2 主页面

数据读取与展示

软件将数据保存在本地，在打开软件时读取数据，显示在主界面以及其他功能的界面。

相关代码

```
    calRange(){
        return '20220105'
      },
      //list 需做备份时调用，需传入 callback 回调
      saveLists(voidCallback = function () { }) {
        let that = this
        let list = { toDoList: this.toDoList, doingList: this.doingList, doneList:
this.doneList }
        storage.set({
          key: 'msg',
          value: list,
          success: voidCallback(),
          fail: function (data, code) {
            that.$app.$def.makeToast(`handling fail, code = ${code}`)
          }
        })
      },
```

页面跳转

软件主界面采用了滑动切换界面的方式，可以滑动切换到统计界面，对于切换到日历界面，我们添加了日历按钮，可以通过点击的方式切换到日历界面。

相关代码

```
//切换至新建页面
  openInput(name, start, end) {
    this.saveLists(function () {
      router.push({
        uri: '/Input',
        params: {
          pushName: name,
          pushStart: start,
          pushEnd: end,
          pushType: -1,
          pushIdx: -1
        }
      })
    })
  },
```

3.2 任务编辑页面

语音输入

点击添加任务界面的语音输入按钮，即可通过将语音转化为文字的方式设置任务名称。

任务添加、删除与修改

在主页面点击任务，进入添加任务界面，可对该任务的任务名/任务状态/截止时间进行修改。

设置任务截止时间

点击任务添加界面的日期一栏，即可通过弹出的日历，选择目标任务的截止日期。

相关代码

```
    let that = this
          storage.get({
            key: 'msg',
            success: function (data) {
              if (data != '') {
                // string 转数组
                let list = JSON.parse(data)
                // 修改事项时触发
                if (that.pushType == 0) list.toDoList.splice(that.pushIdx, 1)
                else if (that.pushType == 1) list.doingList.splice(that.pushIdx, 1)
                else if (that.pushType == 2) list.doneList.splice(that.pushIdx, 1)

                // 判断 todo/doing/done
                if(popType == 0) {
                  prompt.showToast({message: "已将任务添加至待办事项！"});
                  list.toDoList.push({ name: that.eventName, start: start, end:
end })
                }
                else if(popType == 1) {
                  prompt.showToast({message: "已将任务添加至计划内！"});
                  list.doingList.push({ name: that.eventName, start: start, end:
end })
                }
                else if(popType == 2) {
                  prompt.showToast({message: "已将任务添加至已完成！"});
```

```
                list.doneList.push({ name: that.eventName, start: start, end:
end })
            }
```

3.3 日历页面

通过接口 router 切换页面和传递参数

router 接口在使用前，需要先导入模块。router.push(OBJECT)支持的参数 uri 与组件 a 的 href 属性完全一致 router.replace(OBJECT)的支持的参数 uri 不支持调起电话、短信、邮件，其他与 push 一致。

相关代码

```
    router.push({
          uri:'/Calendar',//切换页面的地址
          params: {//三种事项列表作为新页面的参数
          CalToDoList:tdEnd,
          CalDoingList:doingEnd,
          CalDoneList:doneEnd,
          nameList:namelist1
          }
        })
```

日历页面接收参数

组件 a 和接口 router 传递的参数的接收方法完全一致：在页面的 ViewModel 的 protected 属性中声明使用的属性。注意：protected 内定义的属性，允许被应用内部页面请求传递的数据覆盖，不允许被应用外部请求传递的数据覆盖。若希望参数允许被应用外部请求传递的数据覆盖，请在页面的 ViewModel 的 public 属性中声明使用的属性。

日历组件传递参数

在日历页面，将从上级页面接收的参数再次向下一级（日历组件）传递，通过组件 a 实现页面切换时，可以通过 '?key=value'的方式添加参数，支持参数为变量

相关代码

```
<q-calendar
      if="type==='default'"
      calDL="{{getcalDL1()}}"
      nameL="{{getnameL()}}"
      type="default"
      description="{{getDesc()}}"
    ></q-calendar>
```

日历页面显示设计

Zeller 公式

Zeller 公式是一种计算任何一日属一星期中哪一日的算法，由蔡勒（Julius Christian Johannes Zeller）推算出。

其中：W 是星期数；c 是世纪数减一，也就是年份的前两位；y 是年份的后两位；m 是月份，m 的取值范围是 3 至 14，因为某年的 1、2 月要看作上一年的 13、14 月，比如 2019 年的 1 月 1 日要看作 2018 年的 13 月 1 日来计算，d 是日数；[] 是取整运算；mod 是求余运算。

因为每个月最后一日最多不超过第四十二个格，故将日历以 7*6 的列表形式显示，使用 Zeller 公式确定每个月第一天星期，从而确定每个月的日期布局。

总结：

在日历组件设计中，主要解决通过快应用项目支持的两种参数传递方式（router 接口以及组件参数）逐级将主页面的事项数据向子页面传递，并通过 Zeller 公式确定日历的日期布局，最终通过日历组件显示添加的事项

3.4 统计页面设计

向进度条组件传递参数

在日历页面，将从上级页面接收的参数再次向下一级（日历组件）传递，通过组件 a 实现页面切换时，可以通过’?key=value’的方式添加参数，支持参数为变量

相关代码

```
<q-progress
   strokeColor="#e2d6bc"//进度条颜色
   percentage="{{ calPercent($item) }}"//进度条进度比例
   progress-width="{{ progressWidth }}"//进度条宽度
   is-content-show="{{ isContentShow }}"//是否显示
   format="{{ format }}"//标签文本格式
 ></q-progress>
```

函数将参数转换为进度条比例

计算输入事项的结束时间与当前时间的差值，计算所有事项在当前时间的时间进度。

相关代码

```
//计算事件进度百分比
   calPercent(e){
    let today=new Date()//获取当前时间
```

```
        let fullDate=new
Date(today.getFullYear(),today.getMonth()+1,today.getDate())
        let sta=e.start.match(/\d+/g)//提取参数中的数字
        let en=e.end.match(/\d+/g)
        let staFormat=new
Date(parseInt(sta[0]),parseInt(sta[1]),parseInt(sta[2]))//将转换为纯数字的参数转为
js 标准的时间格式，便于进行比较
        let enFormat=new Date(parseInt(en[0]),parseInt(en[1]),parseInt(en[2]))
        if(enFormat<fullDate) return 100
        else if(staFormat>fullDate) return 0
        else{
        let timeDiff=(enFormat.getTime()-staFormat.getTime())/(1000*60*60*24)
        let timeDiff1=(enFormat.getTime()-fullDate.getTime())/(1000*60*60*24)
        console.log(681,1-(timeDiff1/timeDiff))
        return (1-(timeDiff1/timeDiff))*100//返回进度条比例
        }
      },
```

总结：

在进度条模块的设计部分中，也出现了一些问题，经过查阅资料成功解决。页面向组件传递参数时，如果参数是数组变量，那么组件中接收到的参数会合并为字符串，需要根据逗号将字符串重新分割为数组。另外，因为存在某些事项的截止日期为空，之前没有考虑到，会导致报错，修改后使进度条忽略截止日期为空的事项，提高了程序的鲁棒性。

四、项目效果图

4.1 主页面

4.2 任务修改页面


添加任务
修改任务修改任务
长按语音输入
2022年1月14日
已设为计划内


添加任务
2022年
1月10日周一
2022年1月
日 一 二 三 四 五 六
1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
取消
确定
长按语音输入
2022年1月14日
已设为计划内

4.3 日历页面


09:02
计划内事项查看
查看全部计划内事项
蓝色通告栏有事项具体信息
2022年2月
无事项
日 一 二 三 四 五 六
1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28


22:45
计划内事项查看
查看全部计划内事项
2022年1月
无事项
日 一 二 三 四 五 六
1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31

4.4 统计页面设计


08:53
To Do
任务统计:
4件 计划内
3件 待办
1件 已完成
死线分布:
7
未完成任务统计
计划内事项进度:
打豆豆
已度过100%
吃饭
已度过100%
睡觉
已度过25.7%


09:02
计划内事项查看
查看全部计划内事项
也可以仅查看计划内月份
无事项
日 一 二 三 四 五 六
2022年1月
2022年2月

五、实习心得与展望

5.1 结论与体会

课程感受：小米的老师们确实很热情，对我们的问题非常积极的响应，感觉老师们的积极性远高于同学，并且题目难度也不是很大 。

收获和评价：主要是对快应用的设计和实现有了很多了解，建议就是多提供一些项目实例参考，对于我们小组，本次实习除了 demo 还看了很多包括 qaui 的组件样例才大概掌握快应用，快应用的使用手册虽然全面，但感觉比较难理解，还是要配合大量实例才能掌握。

5.2 程序优化方向

目前快应用已基本实现所有需求，如任务的增删改，任务统计等。但受限于快应用的技术模型(WebView 魔改)，渲染耗时过长，在快应用冷启动时根据手机的配置需要有几秒钟左右的等待时间，造成用户体验不佳。

同时由于快应用文档更新维护较差，阉割了部分 JS 与 CSS 的 API，实际开发中只能使用快应用官方提供的 API，对于一些较复杂的功能会造成较大的开发困难，给开发和维护带来了很大的不便。如在界面设计时 CSS 原生的 box-shadow 在快应用中就未被支持，如需使用相关样式，需要在设计阶段以 hook 图片的方式提供。而自己在快应用上造一套相关组件，维护成本高，代码迁移与复用难。故下一步可以考虑使用 Flutter 技术进行重构，Flutter 是谷歌的移动 UI 框架，具有“UI 漂亮、像素级可控、性能流畅、可媲美原生性能”等特点，开发者可以使用相同的代码库快速在 iOS、Android、Windows、macOS 、Linux 以及 Web 端 构建高质量的原生用户界面，以此达到一次编写，处处运行的效果。

六、关键代码

```
<import name="main-page-item" src="../MainPage/main-page-item.ux"></import>


<!-- 界面组件 -->
<template>
  <div>
    <stack>
      <!-- 统计页面 -->
      <div class='analyze-page {{belowAnim}}' onswipe='aboveSwipe'>
        <stack>
          <!-- canvas 部分 -->
          <div class='canvas-container'>
            <text class='canvas-text'>任务统计:</text>
            <canvas id='line-canvas'></canvas>
            <text class='canvas-text'>死线分布:</text>
            <canvas id='time-canvas'></canvas>
          </div>
          <!-- mask 部分 -->
          <div class='mask-container'>
            <div class='mask-area {{maskAnim}}'></div>
            <div class='mask-area {{maskAnim}}'></div>
          </div>
        </stack>
      </div>
      <!-- 主页 -->
      <div class="main-page {{aboveAnim}}">
        <div class='header' onswipe='aboveSwipe'>

          <!-- <text class='span2'>———your event manager</text> -->
        </div>
```

```
<tabs class="tabs" onchange="changeTabactive" index="{{activeIndex}}">
  <tab-bar class="tab-bar">
    <text class="tab-text">待办事项</text>
    <text class="tab-text">计划内</text>
    <text class="tab-text">已完成</text>
  </tab-bar>
  <!-- type 0/1/2 → 待办事项/计划内/已完成 -->
  <tab-content class="tab-content">

    <div class="item-container">
      <list class='todo-list'>
        <block for='{{toDoList}}'>
          <list-item type="item">
            <main-page-item  item='{{$item}}'  idx='{{$idx}}'  type='0'></main-page-
item>
          </list-item>
        </block>
      </list>
    </div>

    <div class="item-container">
      <list class='doing-list'>
        <block for='{{doingList}}'>
          <list-item type="item">
            <main-page-item  item='{{$item}}'  idx='{{$idx}}'  type='1'></main-page-
item>
          </list-item>
        </block>
      </list>
    </div>
```

```
            <div class="item-container">
              <list class='done-list'>
                <block for='{{doneList}}'>
                  <list-item type="item">
                    <main-page-item  item='{{$item}}'  idx='{{$idx}}'  type='2'></main-page-
item>
                  </list-item>
                </block>
              </list>
            </div>
          </tab-content>

        </tabs>

       <div>
          <!--查看日历按钮 -->
        <div class='calender-btn' onclick="openCalender('','ok','不设置')"></div>
          <!-- 添加事项按钮 -->

        <div class='add-btn' onclick="openInput('','ok','不设置')"></div>

        </div>

      </div>
    </stack>
  </div>
</template>

<script>
import storage from '@system.storage'
import router from '@system.router'
import prompt from "@system.prompt";
```

```
export default {
  private: {
    aboveAnim: '',
    belowAnim: '',
    maskAnim: '',
    activeIndex: 1,
    toDoList: [
      { name: 'todo 事件 1', start: '2022 年 1 月 1 日', end: '2022 年 2 月 1 日' },
      { name: 'todo 事件 2', start: '2022 年 1 月 1 日', end: '2022 年 2 月 2 日' },
      { name: 'todo 事件 3', start: '2022 年 1 月 1 日', end: '2022 年 2 月 3 日' }
    ],
    doingList: [
      { name: '吃饭', start: '2022 年 1 月 1 日', end: '2022 年 1 月 2 日' },
      { name: '睡觉', start: '2022 年 1 月 1 日', end: '2022 年 2 月 8 日' },
      { name: '打豆豆', start: '2022 年 1 月 1 日', end: '2022 年 1 月 2 日' },
      { name: '豆豆哭了', start: '2022 年 1 月 1 日', end: '2022 年 12 月 13 日' }
    ],
    doneList: [
      { name: 'done 事件 1', start: '2022 年 1 月 1 日', end: '2022 年 1 月 29 日' },
      { name: 'done 事件 2', start: '2022 年 1 月 1 日', end: '2022 年 3 月 5 日' },
    ]
  },

  onInit() {
    // storage.clear()
    this.$page.setTitleBar({ text: 'To Do' })

    //将完成事件修改为待办事项
    this.$on('todoItem', (evt) => {
      if (evt.detail.type == 1) {
        this.toDoList.push(this.doingList[evt.detail.idx])
```

```
    this.doingList.splice(evt.detail.idx, 1)
  }

  if (evt.detail.type == 2) {
    this.toDoList.push(this.doneList[evt.detail.idx])
    this.doneList.splice(evt.detail.idx, 1)
  }
  this.saveLists()
})

//将事件修改为计划内
this.$on('doingItem', (evt) => {
  if (evt.detail.type == 0) {
    this.doingList.push(this.toDoList[evt.detail.idx])
    this.toDoList.splice(evt.detail.idx, 1)
  }

  if (evt.detail.type == 2) {
    this.doingList.push(this.doneList[evt.detail.idx])
    this.doneList.splice(evt.detail.idx, 1)
  }
  this.saveLists()
})

//完成事件（并存储）
this.$on('doneItem', (evt) => {
  if (evt.detail.type == 0) {
    this.doneList.push(this.toDoList[evt.detail.idx])
    this.toDoList.splice(evt.detail.idx, 1)
  }
  if (evt.detail.type == 1) {
    this.doneList.push(this.doingList[evt.detail.idx])
```

```
      this.doingList.splice(evt.detail.idx, 1)
    }
    this.saveLists()
  })

  //删除事件（并存储）
  this.$on('delItem', (evt) => {
    this.doneList.splice(evt.detail.idx, 1)
    this.saveLists()
  })
},

//每次显示页面时读取数据并绘制 canvas
onShow() {
  let that = this
  this.loadLists(function (data) {
    if (data != '') {
      let list = JSON.parse(data)
      that.toDoList = list.toDoList
      that.doingList = list.doingList
      that.doneList = list.doneList
      let nowDate = new Date()
      that.saveLists()
    }
  })
  this.drawLineCanvas()
  this.drawTimeCanvas()
},

// 切换 tab
changeTabactive(e) {
  this.activeIndex = e.index
```

```
},

//切换至新建页面
openInput(name, start, end) {
  this.saveLists(function () {
    router.push({
      uri: '/Input',
      params: {
        pushName: name,
        pushStart: start,
        pushEnd: end,
        pushType: -1,
        pushIdx: -1
      }
    })
  })
},
openCalender(todo, doing, done) {
  let tdEnd=new Array();
  let doingEnd=new Array();
  let doneEnd=new Array();
  let namelist1=new Array();
  //提取三个提醒事项表单中的截止日期
  for(let i=0;i<this.toDoList.length;i++){
    tdEnd.push(this.toDoList[i].end);
  }
  for(let j=0;j<this.doingList.length;j++){
    if(this.doingList[j].name!="未设置"){
      doingEnd.push(this.doingList[j].end);
      namelist1.push(this.doingList[j].name);
    }
  }
```

```
    for(let k=0;k<this.doneList.length;k++){
      doneEnd.push(this.doneList[k].end);
    }

    console.log(669,namelist1)
    router.push({
      uri:'/Calendar',
      params: {
      CalToDoList:tdEnd,
      CalDoingList:doingEnd,
      CalDoneList:doneEnd,
      nameList:namelist1
      }
    })

},
  calRange(){
    return '20220105'
  },
  //list 需做备份时调用，需传入 callback 回调
  saveLists(voidCallback = function () { }) {
    let that = this
    let list = { toDoList: this.toDoList, doingList: this.doingList, doneList: this.doneList }
    storage.set({
      key: 'msg',
      value: list,
      success: voidCallback(),
      fail: function (data, code) {
        that.$app.$def.makeToast(`handling fail, code = ${code}`)
      }
    })
  },
```

```
  //list 需读取时调用，需传入 callback 回调
  loadLists(dataCallback = function () { }) {
    let that = this
    storage.get({
      key: 'msg',
      success: function (data) {
        dataCallback(data)
      },
      fail: function (data, code) {
        that.$app.$def.makeToast(`handling fail, code = ${code}`)
      }
    })
  },


  //曲线统计图
  drawLineCanvas() {
    //绘制策略为： 曲线高度 = （个数 - 最小值） / 最值差 * 画布高度
    let min = Math.min(this.toDoList.length, this.doingList.length, this.doneList.length)
    let gap = Math.max(this.toDoList.length, this.doingList.length, this.doneList.length) -
min
    let todo = 320 - (this.toDoList.length - min) / gap * 250
    let doing = 320 - (this.doingList.length - min) / gap * 250
    let done = 320 - (this.doneList.length - min) / gap * 250
    const canvas = this.$element('line-canvas')
    const ctx = canvas.getContext('2d')
    ctx.clearRect(0, 0, 750, 350)
    //绘制圆点
    ctx.arc(200, todo, 10, 0, 2 * Math.PI)
    ctx.fill()
```

```
  ctx.moveTo(375, doing)
  ctx.arc(375, doing, 10, 0, 2 * Math.PI)
  ctx.fill()
  ctx.moveTo(550, done)
  ctx.arc(550, done, 10, 0, 2 * Math.PI)
  ctx.fill()
  //绘制三阶贝塞尔曲线
  // ---------------------------->x
  // |           • cp2       • x2
  // |
  // |   • x1      • cp1
  // y
  ctx.moveTo(0, 175)
  ctx.bezierCurveTo(100, 175, 100, todo, 200, todo)
  ctx.bezierCurveTo(280, todo, 280, doing, 375, doing)
  ctx.bezierCurveTo(470, doing, 470, done, 550, done)
  ctx.bezierCurveTo(650, done, 650, 175, 750, 175)
  ctx.stroke()
  //绘制统计文字
  ctx.font = '28px sans-serif'
  ctx.fillStyle = 'white'
  ctx.fillText(`${this.toDoList.length}件 待办`, 200 - 45, todo - 30)
  ctx.fillText(`${this.doingList.length}件 计划内`, 375 - 45, doing - 30)
  ctx.fillText(`${this.doneList.length}件 已完成`, 550 - 45, done - 30)
},
//圆环统计图
drawTimeCanvas() {
  //判断任务区间
  let overtime = 0, day = 0, week = 0, month = 0, more = 0
  const nowDate = new Date()
  const dayDate = new Date()
  dayDate.setTime(dayDate.getTime() + 24 * 60 * 60 * 1000)
```

```
const weekDate = new Date()
weekDate.setTime(weekDate.getTime() + 7 * 24 * 60 * 60 * 1000)
const monthDate = new Date()
monthDate.setTime(monthDate.getTime() + 30 * 24 * 60 * 60 * 1000)
this.toDoList.forEach(function (value) {
  //没有 ddl 的算“还很远”
  if (value.end == '未设置') { more += 1 }
  else {
    let arr = value.end.match(/\d+/g)
    let endDate = new Date(arr[0], arr[1] - 1, arr[2])
    if (endDate.getTime() > monthDate.getTime()) more += 1
    else if (endDate.getTime() > weekDate.getTime()) month += 1
    else if (endDate.getTime() > dayDate.getTime()) week += 1
    else if (endDate.getTime() > nowDate.getTime()) day += 1
    else overtime += 1
  }
})
this.doingList.forEach(function (value) {
  if (value.end == '未设置') { more += 1 }
  else {
    let arr = value.end.match(/\d+/g)
    let endDate = new Date(arr[0], arr[1] - 1, arr[2])
    if (endDate.getTime() > monthDate.getTime()) more += 1
    else if (endDate.getTime() > weekDate.getTime()) month += 1
    else if (endDate.getTime() > dayDate.getTime()) week += 1
    else if (endDate.getTime() > nowDate.getTime()) day += 1
    else overtime += 1
  }
})
let total = overtime + day + week + month + more
//各自的百分比
let percentList = [overtime / total, day / total, week / total, month / total, more /
```

```
total]
    //用来计算弧度的百分比
    let circleList = [0, ...percentList]
    for (let i = 0; i < 5; i++) circleList[i + 1] += circleList[i]
    const canvas = this.$element('time-canvas')
    const ctx = canvas.getContext('2d')
    const color = ['#303841', '#d72323', '#f9906f', '#fdeff2', '#d6ecf0']
    for (let i = 0; i < percentList.length; i++) {
      percentList[i] = (percentList[i] * 100).toFixed(0)
      if (percentList[i].length == 1) percentList[i] = '  ' + percentList[i] //错个位使数据
整齐
    }
    let textList = [`已超时:  ${percentList[0]}%`, `一天内:  ${percentList[1]}%`, `一周内:
${percentList[2]}%`, `一月内:  ${percentList[3]}%`, `还很远:  ${percentList[4]}%`]
    ctx.clearRect(0, 0, 750, 350)
    ctx.lineWidth = 20
    ctx.font = '23px'
    for (let i = 0; i < 5; i++) {
      //绘制圆弧
      ctx.beginPath()
      ctx.strokeStyle = color[i]
      ctx.arc(220, 175, 120, Math.PI * 2 * circleList[i], Math.PI * 2 * circleList[i + 1])
      ctx.stroke()
      //绘制右侧圆点
      ctx.beginPath()
      ctx.fillStyle = color[i]
      ctx.arc(450, 75 + 50 * i, 10, 0, 2 * Math.PI)
      ctx.fill()
      //绘制右侧数据
      ctx.fillStyle = '#f0f8ff'
      ctx.fillText(textList[i], 475, 75 + 50 * i + 6) // +2 为字号修正
    }
```

```
    //绘制左侧总计数据
    ctx.textAlign = 'center'
    ctx.font = 'bold 55px'
    ctx.fillText(`${total}`, 220, 167)
    ctx.font = '22px'
    ctx.fillStyle = '#f0f8ff'
    ctx.fillText('未完成任务统计', 220, 205)
  },


  // 滑动切换统计界面
  aboveSwipe(dir) {
    if (dir.direction == 'left') {
      this.drawLineCanvas()
      this.drawTimeCanvas()
      this.aboveAnim = 'aboveForward'
      this.belowAnim = 'belowForward'
      this.maskAnim = 'maskForward'
    }

    if (dir.direction == 'right') {
      this.aboveAnim = 'aboveReverse'
      this.belowAnim = 'belowReverse'
      this.maskAnim = 'maskReverse'
    }
  },
}
</script>
```