数据库系统实验

Project 1: ToDoList

数据科学与计算机学院　17大数据与人工智能

17341015　陈鸿峥

由于以前完全没有前端开发经验，因此第一个实验开发一个APP就耗费了我大量时间。先是从基本的HTML、CSS和JavaScript学起，然后学vue.js学网页布局，最后再上UniApp进行实操。中间遇到了非常非常多的问题，但最后都耐心地将它们都解决了，整体来说还是十分愉悦的，因为学到了很多东西，也做成了一个还可以用的ToDoList，如图1所示。

图 1: 初始界面

我在ToDoList中实现了**添加、删除、编辑、保存、读取、查找、星标事项**等功能，下面将按照vue.js对应的三个模块模板、代码、样式分别介绍。

一、 架构及组件设计

这一部分对应的是vue.js中`template`的部分。

1. 交互组件

我创建了一个名为`search_box`的类，对用户输入框进行管理，并且采用了一个渐变颜色的

背景使得看上去更加舒服，即图1中所示的三个长方形框。

```
<view class="search_box">
    <span class="custom-font">&#xe8ef;</span>
    <input class="input" type="text" placeholder="查找一个日程"
        @confirm="find" v-model="text_find" />
</view>
```

图 2: 交互组件实例代码

该组件又包括两个部分：

- 一个小的图标
- 用户交互框，用input实现

2. 列表组件

这一部分是ToDoList的核心，所有功能都将在这里实现。最外层是一个list-container的类，然后采用v-for遍历每一个待办事项，并分别创建名为item的类。

```
<view class="list-container">
    <view class="item" v-for="(item, index) in arr" :key="index" >
        <checkbox :checked="item.done" @click="change(index)"
            style="transform: scale(0.7)"/>
        <view :class="[item.done ? 'done' : '']" v-if="!item.done">
            <!-- {{ index + 1 }}. {{ item.value }} -->
            <input v-model="arr[index].value" @confirm="edit(index)"/>
        </view>
        <view v-else class="done">{{ item.value }}</view>
        <span class="middle-space"></span>
        <view class="icon-pack">
        <view class="icon-right" @click="star(index)">
            <view v-show="item.star">&#xe86a;</view>
            <view v-show="!item.star">&#xe7df;</view>
        </view>
        <view class="icon-right" @click="del(index)">&#xe7c3;</view>
        </view>
    </view>
</view>
```

图 3: 列表组件实例代码

在一行的item类中又包括四个部分内容：勾选框checkbox、待办事项内容、中间空隙middle-space、右端图标icon-pack。

二、 界面设计

这一部分对应的是vue.js中style的部分。

为了让我的APP看上去没有那么糟糕，我还是花了一些时间在CSS的学习上。也通过查阅资料和查看网页源码等方式学会了一些基本的CSS布局方法（Flex布局）。

下面讲讲两个值得一提的点。

- 现在的APP中的内容都不可能只有文字，为了更加生动往往会增加一些图片或图标，这在手机端尤其是这样。因此为了插入几个常用的图标（垃圾桶、星标、搜索等），我从阿里妈妈的Icon网站上[1]注册下载了一些图标并嵌入到我的APP中，使得展示更加美观。
- CSS里面最难的恐怕就是布局了，我在这上面也折腾了很久，主要在于不知道如何做到一行内既有左对齐的组件，又有右对齐的组件，而且组件数目不一样多。最后的解决方案还是我自己摸索出来的，虽然也许不那么优美。即将一侧的组件打包成一个更大的类，然后在左右组件中间插入一个空隙类，采用Flex布局中的`justify-content: space-between`属性即可实现我想要的效果。

三、 功能实现及演示

这一部分对应的是vue.js中`script`的部分。

1. 添加删除编辑

每次在添加待办事项用户输入框键入回车，都会调用`schedule`函数，将用户输入的值添加到`arr`数组中。注意要判断用户输入是否为空，若为空则不添加。

若点击勾选框，则可以通过调用`change`函数，将未完成事项更改为完成。

删除则是检测到用户点击最右侧的垃圾桶图标时，调用`del`函数，然后利用`splice`函数将对应元素删除。

编辑则比较麻烦，考虑两种情况。一种是用户还没完成任务，则此时采用`input`输入框并将某一`item`的值初始化为一开始用户输入的值，这时用户就可以直接在每一项上面修改，按回车会调用`edit`函数进行更新。另一种情况则是用户已经完成了任务，则我将该事项锁死，即不再允许用户进行修改，除非用户将其状态切换回未完成。

[1] https://www.iconfont.cn

图 4: 添加、完成、删除、编辑示例

2. 存储与加载数据

存储数据则可以直接调用UniApp提供的接口，如下所示。

```
uni.setStorage({
  key: 'storage_key',
  data: this.arr,
  success: function(){
    console.log('success');
  }
});
```

将上面一段代码插入到每一段添加修改删除的代码后面[2]，即可实现数据的及时保存。

载入数据则采用下面一段代码，这里注意采用了JavaScript的箭头函数，否则this的指向会出错，导致无法正常读出数据。

```
uni.getStorage({
  key: 'storage_key',
  success: (res) => {
      console.log(res.data);
      this.arr = res.data;
  } // binding
});
```

结果如图5所示，可以见到刷新后，本地数据能够被重新读出并正确加载。

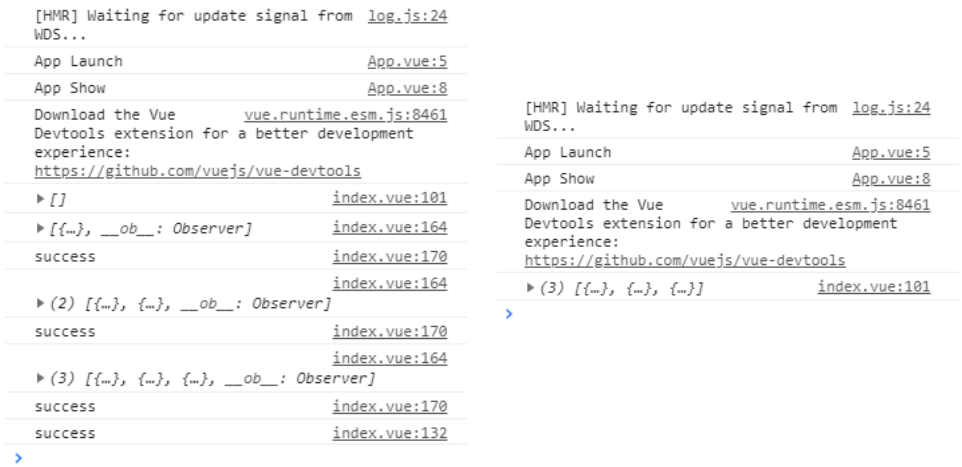

图 5: 保存及重加载数据

3. 查找

通过调用find函数实现，这里引入了一个全局变量tmp，当tmp不为-1时即找到对应的元素，此时会将该条目进行输出。这里的条目都是共享对象的，即在查找出来的条目中修改数据

[2]可能会很慢，这里还未进行优化

也会导致源数据被修改。同时注意tmp的数据要及时修改回-1，如在添加删除或其他操作后面都要将tmp值恢复。

图 6: 查找

4. 星标事项

星标事项也是我的ToDoList的一个亮点，通过点击最右侧的星星图标，就可以把某一代办事项加入到星标事项中（调用star函数，和修改待办事项状态的方法类似）。这里的输出会比较麻烦，因为vue.js不建议将v-for和v-if混合使用，因此需要动态将星标事项筛选输出。这里在App的生命周期中添加了动态计算computed的部分，用starArr函数将星标事项筛选出来后再遍历输出。具体实现则采用了JavaScript的函数式编程功能，采用filter对列表进行筛选，如下。

```
computed: {
  starArr: function () {
    return this.arr.filter(function (item) {
      return item.star
    })
  }
}
```

星标事项的演示如图7所示，可以看到三个条目全部都被删除、标星，说明是紧密联系在

一起的对象。

图 7: 星标事项

参考文献

[1] Vue.js生命周期，https://www.cnblogs.com/chenzeyongjsj/p/8093789.html

[2] Vue.js风格指南，https://cn.vuejs.org/v2/style-guide

[3] 阮一峰，Flex布局教程，http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html

[4] Uni-app Iconfont的引入，https://www.jianshu.com/p/c2b100636006