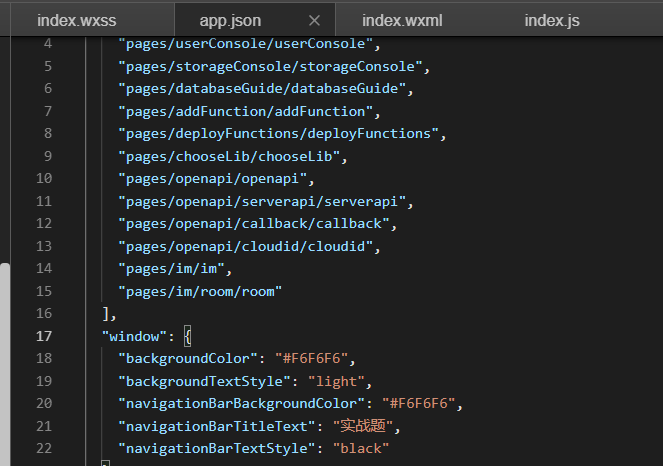
1. 基础题：

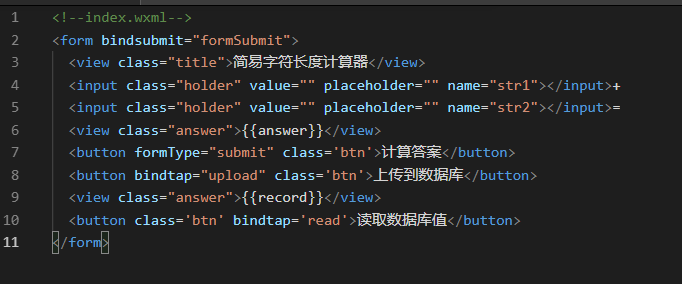
很简单，因为是整个小程序的页面配置项，所以我们可以在app.json里边儿 修改

"navigationBarTitleText"属性（如下图所示）

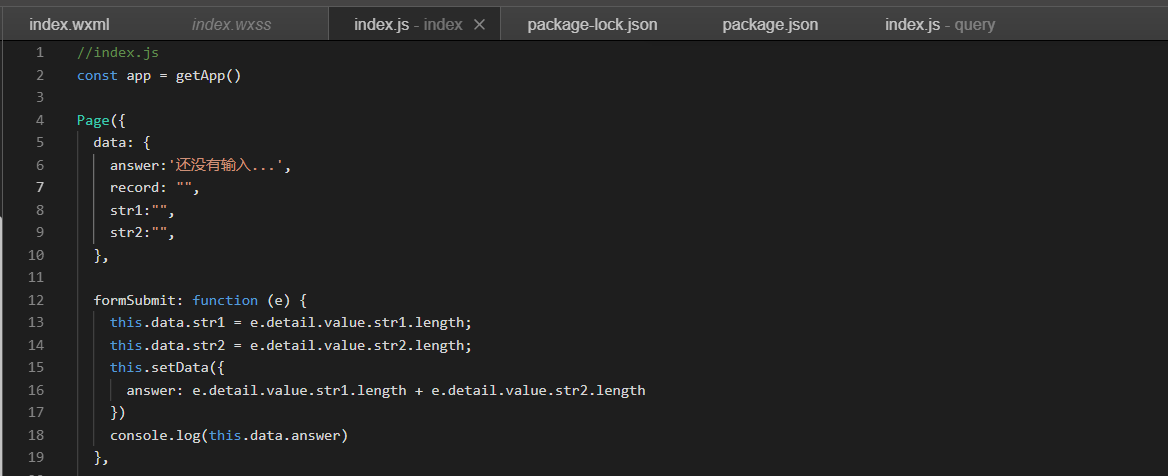


1. 这是一道比较综合的题目啦，主要是想大家在做的过程中能有所收获~

下边儿先放上我自己写的样例：







（这一部分仅仅是不涉及云函数的哦）

并没有放上源码，只放了截图，是想让大家自己回去尝试一下，得注意js并没有截图完整，一些括号什么的还得自己理理清楚

下边儿我简单地和大家讲解一下：

大家可以将wxml，wxss, js（其实wxml对应的是web中的html，wxss是对应的css，就是把人家的换成了微信的缩写…）分别理解为网页（程序）的骨骼，皮肤和神经

首先大家可以从wxml入手，因为做的是表单验证，所以可以用一个大的form组件包起来，然后呢加上bindsubmit属性绑定提交，接着用input来获取用户输入，值得注意的是需要用name属性来绑定我们获取的用户输入量，我用的是str1和str2。

还有比较关键的是button中的formType这可以用来提交这个form中input获得的值。这一步写完后，小程序的骨架就搭好了~

接下来再看wxss：首先page{}把整个页面的颜色更改一下，

下边的 .title 在题目中也说过，是类选择器，选择wxml中定义的类

当然还有其他的选择器，比如元素选择器

可以直接写成：

button{

xx: xxx;

}

少了那个“.”就直接找html中的标签（如本例中的button标签）

等等还有id选择器，大家都可以自己去了解

我在写的时候用了一些属性比如fontsize啊color啊都是很简单的字面属性，大家都可以自己尝试，而其中的边框：border、margin如何区分就稍微有那么一点点复杂，大家也可自己去查询，还有display属性，它设计到元素的浮动，大家也可以自己了解，前端的代码其实是非常直白的，大家别害怕，要用于尝试~好的，你已经完成小程序的皮肤了~

最后稍微难点的是网页的神经了：js

首先我们在wxml中用{{}}绑定了数据，编译器看见这个{{}}就会知道：哦，里边儿的东西不是善茬，它就会知道这是个数据，然后呢你还需要再page里面的data中加入它，至于很多眼花缭乱的大括号大家也别怕，这个{}代表的就是一个对象，其实你可以理解为就是一个小整体，咱们在data这个小整体中加入answer属性，方便我们之后修改，

因为我之前绑定的时间是formSubmit,所以我们在js里继续添加formSubmit函数，

其中的e，this可能看得大家云里雾里，其实不用怕，e就代表的是传来的一个事件（函数中的括号里代表的是传参数嘛），而this指的就是它自己，等于这个函数叫了一声“嘿！我自己”（因为这在程序复杂的时候是很容易混乱的！）

这里再讲一下setData函数，我们想要的是实时更改页面，这一点其实设计到异步和同步的概念了，略显复杂，其实这就是微信小程序帮咱们做的事，不然实时更改其实是意见很复杂的事儿！但既然别人提供了帮助，我们只需调用这个函数就好了，把要改的数据在这个函数中修改就好了~

接下来就是最让同学们头疼的云函数了~

深吸一口气~

这是两个不同的函数，其中呢，upload代表上传，由于这一部分代码可能对于大家刚接触编程略显复杂，我在这儿帮大家理清一下思路~

首先：啥是云函数，啥是云数据库，可能很多同学对这个概念都云里雾里

大家可能在创建云函数的时候看见了一个比较奇怪的东西：Node.js，它又是啥呢？

做一个简单的比喻，Node.js就像一艘小船，它能帮咱们做很多事情，比如咱们写的小程序，它就像一座小岛，你本来就只能在这些页面内走来走去，但你如何访问外面的世界呢？你如何把你高高在上的客户的数据存储下来呢？页面可不具备这个能力，你会说存数据里边儿，但数据库就是另一座岛了！不过咱们可以通过node.js来划到这个岛上，去数据库之岛存储数据，所以，大家可以简单地把它理解为一座功能强大的小船~

而云函数又是什么呢，大家可以理解为它就是一个船夫，咱们想去数据库之岛，得先叫一声：“嘿！船夫！过来！我要给数据库岛的李栓蛋送封信，内容是’\*\*\*’”这里的“\*\*\*“大家可以理解为一个参数，咱们告诉这个云函数也就是这个船夫，咱们要传递的内容。

好了这样大家思路清晰了一些吧，而咱们这里多了一个云字，也就是云端，即服务器端，其实本质是相似的。而因为它在云端，所以它有些特别，比如query函数，我们费尽心思地把它放在cloudfuctions里边就是因为它要在云端，而且我们每次修改它都得要重新部署。但其实不用云函数也可以向云服务器发送数据的，咱们用一个函数也能实现一个船夫的功能嘛，船夫有所不同，但是船还是能开的对吧~

所以我的upload函数并没有新建一个search云函数，而是直接像2.1节中直接放在js中叫这个船夫就好啦~这里的注释其实是比较详细的，但其实很多东西若要全部细讲，我觉得同学们接受起来肯定很困难，但大家肯定可以就一行一行地读，每一行多有特定的解释，大家一行一行地接受、感悟也是一个不小的提高~

要注意的是我传入的content 和对应的tag 这都是待会儿还要用到的东西

//表单提交触发此函数

upload(){

if (!wx.cloud) {

console.error('请使用 2.2.3 或以上的基础库以使用云能力')

} else {

wx.cloud.init({ //初始化云函数

traceUser: true,

})

}

//初始化数据库

const db = wx.cloud.database()

//向数据库添加一条记录

db.collection('hello').add({

// data 字段表示需新增的 JSON 数据

data: {

content: this.data.str1 + "与" + this.data.str2,

tag: 2

},

success: function (res) {

// res 是一个对象，其中有 \_id 字段标记刚创建的记录的 id

console.log(res)

}

})

},

接下啦是读取数据库的值啦，咱们的船夫已经给数据库岛的李栓蛋发过信啦~，李栓但也受到了，但是我们想查一查咱们发的信息，怎么办呢？咱们可以调用query云函数，这里我又采取的是利用云函数，主要也是给大家做个区分，两种船夫都是相似的~

之不过云函数咱们相当于是单独地把这个云函数写在了cloudfunction里，而我们在index.js再去调用这个它罢了。

那么问题来了，为什么咱们要这么麻烦地做这件事呢？

微信的官方文档上这么写着：“云开发的云函数的独特优势在于与微信登录鉴权的无缝整合，当小程序端调用云函数时，云函数的传入参数中会被注入小程序端用户的 openid”，也就是说这个云函数它权力大！它能在我们做其他工作的时候提供不少方便！

而这一部分的代码大家也可以结合注释理解，

其中的 var num = \*\*\*\* 语句

其实是为了获得数据库中的最新数据，我们可以看到res.result.data[]其实是一个数组，而我们每次添加数据都会在后边加上一个数据，而我们为了获取最新的数据得获取到数组的长度，但对应下标则应该减一（因为数组是从0开始计数的）

read() {

//检查基础库版本

if (!wx.cloud) {

console.error('请使用 2.2.3 或以上的基础库以使用云能力')

} else {

wx.cloud.init({

traceUser: true,

})

}

//保存this变量

var \_this = this;

//调用云函数

wx.cloud.callFunction({

// 云函数名称

name: 'query',

// 传给云函数的参数

data: {

tag: 2

},

//success函数为调用成功后的回调函数

success: function (res) {

console.log(res.result);

var num = res.result.data.length - 1; //获取最新上传data的下标

\_this.setData({

record: res.result.data[num].content //设置record值，显示在页面上。setData函数会触发页面的重新渲染

})

},

//fail函数为调用失败后的回调函数

fail: console.error

})

},

整个解析写得比较长，其实是想大家真正学到东西，如果你是耐心地看我啰嗦地讲到这里，我相信验收对你来说真的没有一点问题了~给你点个赞~

By助教冯韵菱