### 2016.11.1-2016.11.17工作总结

这三周主要是学习阶段，根据项目中所用到的技术进行学习，学习的内容主要包括ionic和nodejs。另外就是对项目的环境配置和git的使用方法。

**一 、项目环境配置**

1 从propersoft-cn中找到项目ionic-sj，fork到自己的github下。

2 从自己的github下载到本地。

3 新建一个branch，并切换到此branch。

4 设置npm全局安装路径和缓存路径。

$ npm config set prefix "C:\Program Files\nodejs"

$ npm config set cache "C:\Program Files\nodejs\node\_cache"

5 安装全局命令行组件

$ npm install bower -g

$ npm install grunt-cli -g

6 安装 SCSS 编译环境

$ gem install sass

$ gem install compass

注：如果报错

ERROR:  Could not find a valid gem 'sass' (>= 0), here is why:Unable to download data from ....

可能是网站被墙了，可以用以下方式解决：

gem sources -a <http://ruby.taobao.org/>

或者gem sources -a http://gems.ruby-china.org/

7 配置对应nexus npm 库，以便nexus 下载npm依赖包(会修改~/.npmrc文件，以存储配置),npm login会提示你输入用户名、密码和邮箱，用户名密码使用分配给你的用户名密码，邮箱使用分配给你的公司邮箱：

$ npm config set registry https://server.propersoft.cn:8081/repository/npm-public/

$ npm login --registry=https://server.propersoft.cn:8081/repository/npm-public/

8 需要安装的软件:

$ npm install -g bower

$ npm install -g bower-nexus3-resolver

在 bower 的全局配置文件 (~/.bowerrc) 中配置 nexus 相关信息，便可在所有项目中共享 nexus 带来的便利（注意将配置中myusername 和 mypassword 需替换为为个人分配的账号, .bowerrc文件存放在c盘user的个人帐号下）：

~/.bowerrc

{

"registry" : {

"search" : [ "https://<myusername>:<mypassword>@server.propersoft.cn:8081/repository/bower-public" ]，

"register" : "https:// myusername: mypassword @server.propersoft.cn:8081/repository/bower-internal"

},

"resolvers" : [ "bower-nexus3-resolver" ],

"nexus" : {

"username" : "myusername",

"password" : "mypassword"

}

}

9 在项目路径下 ：

$ bower install

$ npm install

10 运行项目：输入

$ grunt serve:win

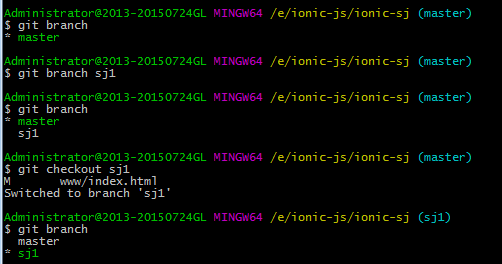
然后，在项目路径下，重新打开一个cmd，输入$ node proxy/proxy-server.js

**二 、git使用方法**

公司代码提交步骤：

1 切换到sj-fixbug上修改

（新建分支，切换分支，查看分支命令）



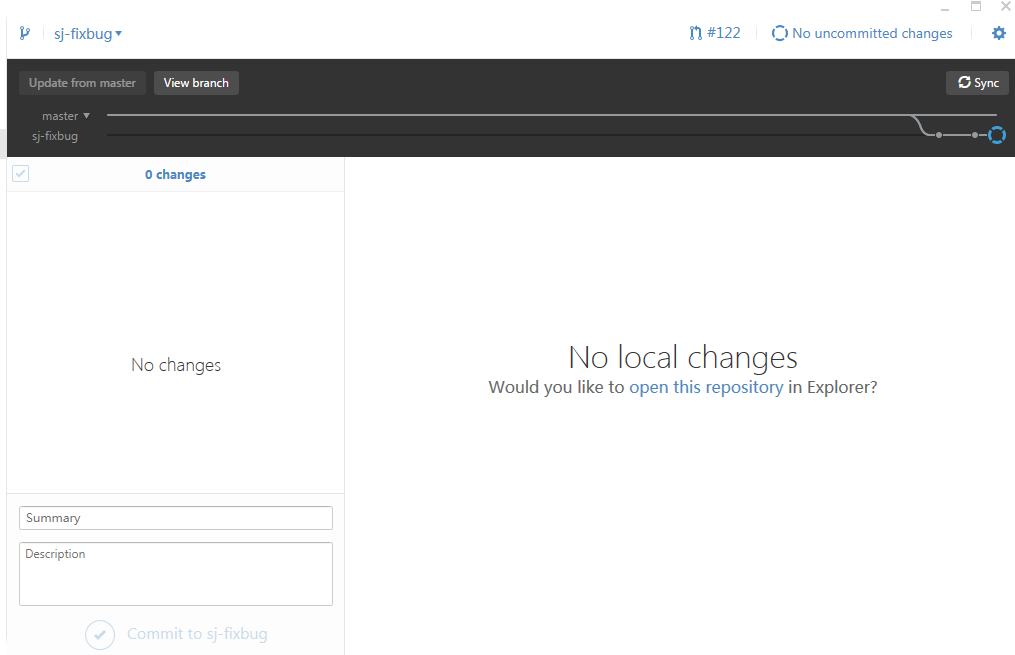
切换分支sj1

新建分支sj1



2 修改完成后，如果自己的github上没有这个分支，则先git push origin sj-fixbug：sj-fixbug

3 如果自己的github上有分支，则可以在github.exe中选到sj-fixbug，然后写好summary和description直接sync

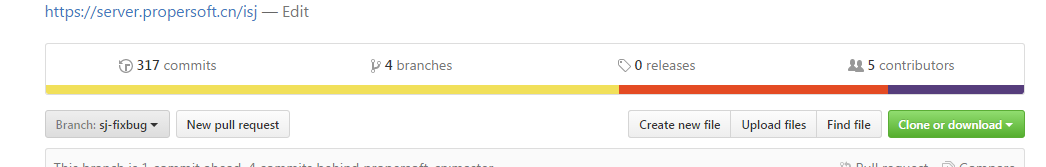


3

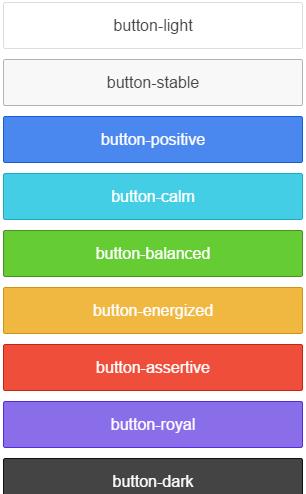
2

1

4 在github官网上点击new pull request



5 create pull request

**三 、 ionic学习总结**

1. ionic中 常用的关于颜色的class。

主要是后面这些颜色：light，stable，positive，calm，balanced，energized，assertive，royal，dark。前缀有很多，例如button，bar等不同的组件。

2 其他ionic css样式，可以直接查ionic手册。

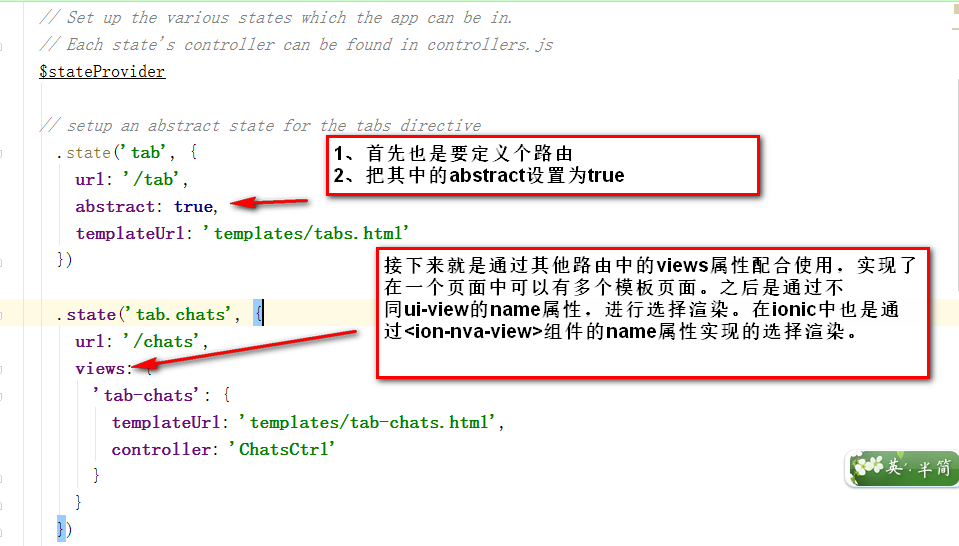
1. 项目中用到的ionic js。
2. Tabs选项卡

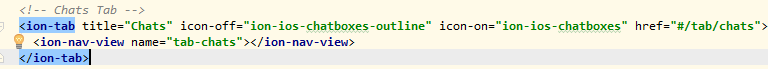
<**ion-tabs** class=**"tabs-icon-top tabs-light tabs-color-active-positive"**>  
 <**ion-tab** title=**"首页"** icon-on=**"ion-ios-medkit-outline"** icon-off=**"ion-ios-medkit-outline"** ng-click=**"tabRouter('tab.main')"**>  
 <**ion-nav-view** name=**"tab-main"**></**ion-nav-view**>  
 </**ion-tab**>  
 <**ion-tab** title=**"消息"** icon-on=**"ion-ios-bell-outline"** icon-off=**"ion-ios-bell-outline"** ng-click=**"tabRouter('tab.message')"**>  
 <**ion-nav-view** name=**"tab-message"**></**ion-nav-view**>  
 </**ion-tab**>  
 <**ion-tab** title=**"挂号单"** icon-on=**"ion-ios-paper-outline"** icon-off=**"ion-ios-paper-outline"** ng-click=**"tabRouter('tab.registration')"**>  
 <**ion-nav-view** name=**"tab-registration"**></**ion-nav-view**>  
 </**ion-tab**>  
</**ion-tabs**>

API：title：选项卡的标题。icon-on：被选中标签的图标。icon-off：没被选中标签的图标。

<**ion-nav-view** name=**"tab-registration"**></**ion-nav-view**>  
用来显示路由的页面。

1. Ui-router路由





（3）<**ion-nav-view** name=**"tab-registration"**></**ion-nav-view**>中内容的一般结构。

<**ion-view**>  
 <**ion-header-bar** class=**"bar-light"**>  
 </**ion-header-bar**>  
 <**ion-content** class=**"has-header content-gray message-content"**>  
 </**ion-content**>  
</**ion-view**>

**ion-view：ion-nav-view**的下级标签，是一个可以存放内容、导航栏和header信息的容器。

**ion-header-bar**：在内容顶部添加一个固定header栏。

**ion-content**：提供一个易用的内容区域，该区域可以用Ionic的自定义滚动视图进行配置，或浏览器内置的溢出滚动。在大多数情况下，建议使用Ionic的定制滚动功能，有时（出于性能原因）仅用浏览器原生的溢出滚动就足够了，因此我可以轻松地在设置了Ionic滚动和溢出滚动间切换。可以用[ionRefresher](http://www.ionic.wang/js_doc-index-name-ionRefresher/)指令实现拉动刷新，并可以用[ionInfiniteScroll](http://www.ionic.wang/js_doc-index-name-ionInfiniteScroll/)指令实现无限滚动。

（4）Lists列表

基本用法：

<ion-list>

<ion-item ng-repeat="item in items">

Hello, {{item}}!

</ion-item>

</ion-list>

<**ul** class=**"list personal-list"**>  
 <**li** class=**"item item-icon-left item-icon-right"** ng-click=**"itemRouter('familyMemberList')"**>  
 <**i** class=**"icon ion-ios-people-outline positive icon-left"**></**i**>  
 家庭成员  
 <**i** class=**"badge"**>{{**memberCount**}}</**i**>  
 <**i** class=**"icon ion-ios-arrow-right icon-right"**></**i**>  
 </**li**>  
 <**li** ng-if=**"isLogin"** class=**"item item-icon-left item-icon-right"** ng-click=**"itemRouter('settingIndex')"**>  
 <**i** class=**"icon ion-ios-gear-outline positive list-icon-left"**></**i**>  
 设置  
 <**i** class=**"icon ion-ios-arrow-right icon-right"**></**i**>  
 </**li**>  
</**ul**>

（5）ion-view的生命周期$ionicView

在controller中使用$scope.$on(‘$ionicView.XX’,function(){});调用。

$ ionicView.loaded    该视图已经加载。这个事件在视图被创建和加到dom中时发生且仅发生一次。如果视图离开，但被缓存，那么这个事件将不会再次发生。类似于Android的activity中的onCreate（）方法。  
$ ionicView.enter    该视图已经完全进入，并且是当前被激活的视图。此事件将触发，无论是第一次载入或已经在缓存中。类似于Android的activity中的onStart（）方法。  
$ ionicView.leave    该视图已经完全离开，不再被激活。此事件将触发，无论是缓存或销毁。类似于Android的activity中的onStop（）方法。  
$ ionicView.beforeEnter    视图是即将进入并成为被激活的视图。类似于Android的activity中的onResume（）方法。  
$ ionicView.beforeLeave    视图是即将离开，不再被激活视图。类似于Android的activity中的onPause（）方法。      
$ ionicView.afterEnter    该视图已经完全进入，并且是当前被激活的视图。  
$ ionicView.afterLeave    该视图已经完全离开，不再是激活的视图。  
$ ionicView.unloaded    该视图的控制器已经被销毁，它的元素已经从DOM中删除。类似于Android的activity中的onDestroy（）方法。

（6）$ionicPopup

弹出框。常用于提醒，警告等。在用户响应之前其他交互行为不能继续。

$ionicPopup.XX({options}).then(function(){

//在弹出框关闭时被调用

});

XX : show alert confirm prompt

注：具体options可以到<http://www.ionic.wang/js_doc-index-id-52.html>查看。

(7)$ionicLoading

当进行耗时的操作时，可以使用载入指示器提示用户操作进行中，并暂时阻止交互。载入指示器通常会叠加一个半透明的幕布层以便阻止用户的交互。

$ionicLoading.show({options});

$ionicLoading.hide();

(8)ionic JS手势事件

长按on-hold：在屏幕同一位置按住超过500ms，将触发on-hold事件。

敲击on-tap：在屏幕上快速点击一次（停留时间不超过250ms），将触发on-tap事件。

双击on-double-tap：在屏幕上快速点击两次，将触发on-double-tap事件。

on-touch/on-release：在屏幕上按下手指，会立即触发on-touch事件，当手指抬起，会立即触发on-release事件。

拖拽on-drag：在屏幕上按住并移动时，触发on-drag事件。

根据运动方向的不同，可以细分为：

* on-drag - 向所有方向拖动时都触发此事件
* on-drag-up - 向上拖动时触发此事件
* on-drag-down - 向下拖动时触发此事件
* on-drag-left - 向左拖动时触发此事件
* on-drag-right - 向右拖动时触发此事件

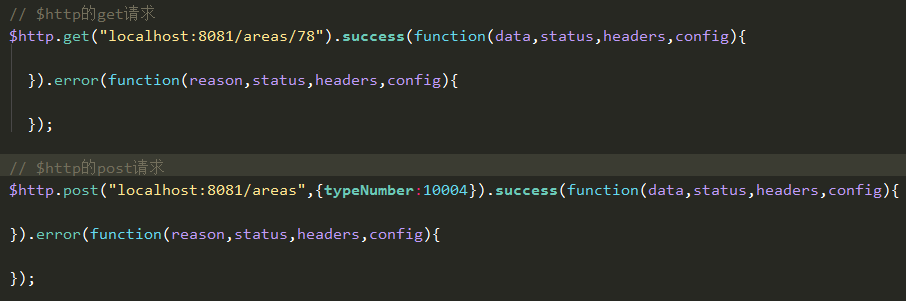
划动on-swipe：在屏幕上按住并快速拖动时，将触发on-swipe事件。

根据运动方向的不同，可以细分为：

* on-swipe - 向任何方向的划动都触发事件
* on-swipe-up - 向上划动时触发事件
* on-swipe-down - 向下划动时触发事件
* on-swipe-left - 向左划动时触发事件
* on-swipe-right - 向右划动时触发事件

**四 、 angularjs中$http使用**

$http服务，返回一个promise对象，具有success和error两个方法。



$http响应对象：

data ： 这个数据代表转换过后的响应体

status ： 响应的http状态码

headers ：头信息

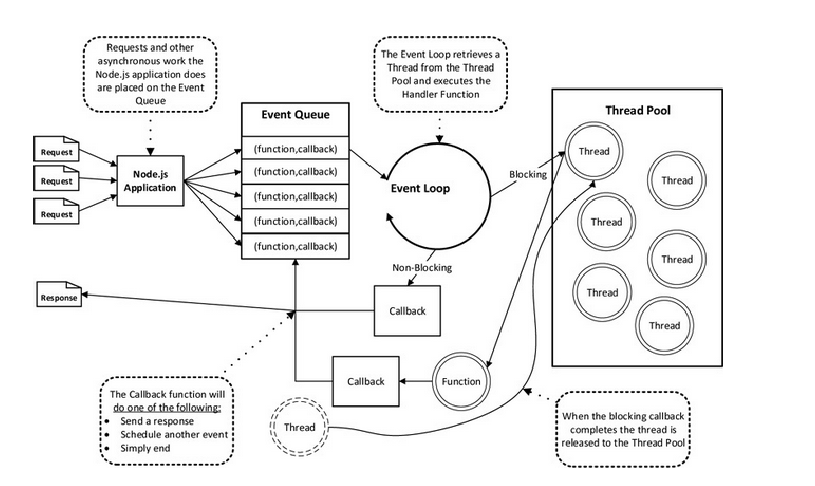
config ：这个对象用来生成原始请求的完整设置对象

statusText ：这个字符串是响应的http状态文本

**五 、node.js**

1 nodejs运行机制





1. 内置模块

核心模块的意义

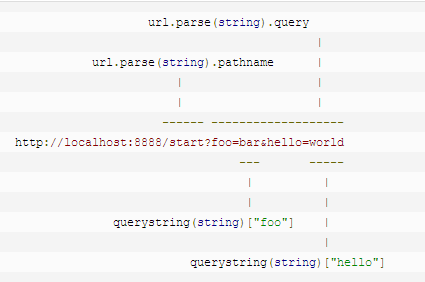
* 如果只是在服务器运行JavaScript代码，意义并不大，因为无法实现任何功能（读写文件，访问网络）。
* Node 的用处在于它本身还提供的一系列功能模块，用于与操作系统互动。
* 这些核心的功能模块在 Node 中内置。

内置如下模块：

* [path](http://nodejs.org/api/path.html)：处理文件路径。
* [fs](http://nodejs.org/api/fs.html)：操作文件系统。
* [child\_process](http://nodejs.org/api/child_process.html)：新建子进程。
* [util](http://nodejs.org/api/util.html)：提供一系列实用小工具。
* [http](http://nodejs.org/api/http.html)：提供HTTP服务器功能。
* [url](http://nodejs.org/api/url.html)：用于解析URL。
* [querystring](http://nodejs.org/api/querystring.html)：解析URL中的查询字符串。
* [crypto](http://nodejs.org/api/crypto.html)：提供加密和解密功能。
* [其他](https://nodejs.org/api/)

1. Path模块

*//获取当前脚本所在路径***console**.log(\_\_dirname);  
*//获取文件路径***console**.log(\_\_filename);



1. fs模块

（1）同步或异步调用

Fs模块对文件的几乎所有操作都有同步和异步两种形式，例如readFile()和readFileSync()，他们的区别是：同步调用会阻塞代码执行，异步不会。异步调用会将读取任务下达到任务队列，直到任务执行完成才会回调。异常处理方面，同步必须使用try catch，异步可以通过回调函数的第一个参数。



1. 文件读取

fs.readFile(file[, options], callback(error, data))

fs.readFileSync(file[, options])

fs.createReadStream(path[, options])

1. 文件写入

fs.writeFile(file,data[, options], callback(error))

fs.writeFileSync(file,data[, options])

fs.createWriteStream(path[, options])

fs.appendFile(file,data[,options],callback(err))

相比较之前文件流的方式，这种方式不会占用文件资源，append完成就会释放

fs appendFileSync(file,data[,options])

1. 获取文件信息

fs.stat(path,callback(err,stats))

fs.statSync(path) // => 返回一个fs.Stats实例

1. 移动文件或重命名文件或目录

fs.rename(oldPath,newPath,callback)

fs.renameSync(oldPath,newPath)

1. 删除文件

fs.unlink(path,callback(err))

fs.unlinkSync(path)

1. 创建一个目录

fs.mkdir(path[,model],callback)

fs.mkdirSync(path[,model])

1. 删除一个空目录

fs.rmdir(path,callback)

fs.rmdirSync(path)

1. 读取一个目录

fs.readdir(path,callback(err,files))

fs.readdirSync(path) // => 返回files

1. http模块



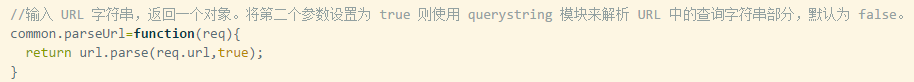


设置相应头：

res.setHead(200,{"Content-Type":"text/plain;charset=UTF8"});

req：

最关键的就是req.url属性，表示用户的请求URL地址。所有的路由设计，都是通过req.url来实现的。识别URL，用到两个新模块，第一个就是url模块，第二个就是querystring模块。



Post：

