|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Subject | QPlay QisdaEIS APP - PIS | | Doc.: | S2-PIS | | Rev. | V1.0 |
|  | | | Release Date: | 2017/09/20 | | Revision Status: | |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |
| Approved By: | | Reviewed By: | | | Prepared By: | | |
| PM / | | QA / | | | SD / | | |
|  | | SE / | | | SE / | | |
|  | | SD / | | |  | | |
|  | |  | | |  | | |
|  | |  | | |  | | |
|  | |  | | |  | | |

[1.0](#_gjdgxs) 系統設計 (System Design) 4

[1.1](#_30j0zll) 系統架構圖 (System Framework Diagram) 4

[1.2](#_1fob9te) 系統流程圖 (System Flow) 5

[1.3](#_3znysh7) 類別圖 (Package/Class Diagram) 6

[2.0](#_2et92p0) 資料庫設計 (Database Design) 6

[2.1](#_tyjcwt) 資料庫物件清單 (Database Object List) 6

[2.2](#_1t3h5sf) 循序圖 (Sequence Diagram) 7

[2.3](#_4d34og8) 資料表結構 (Table Layout) 8

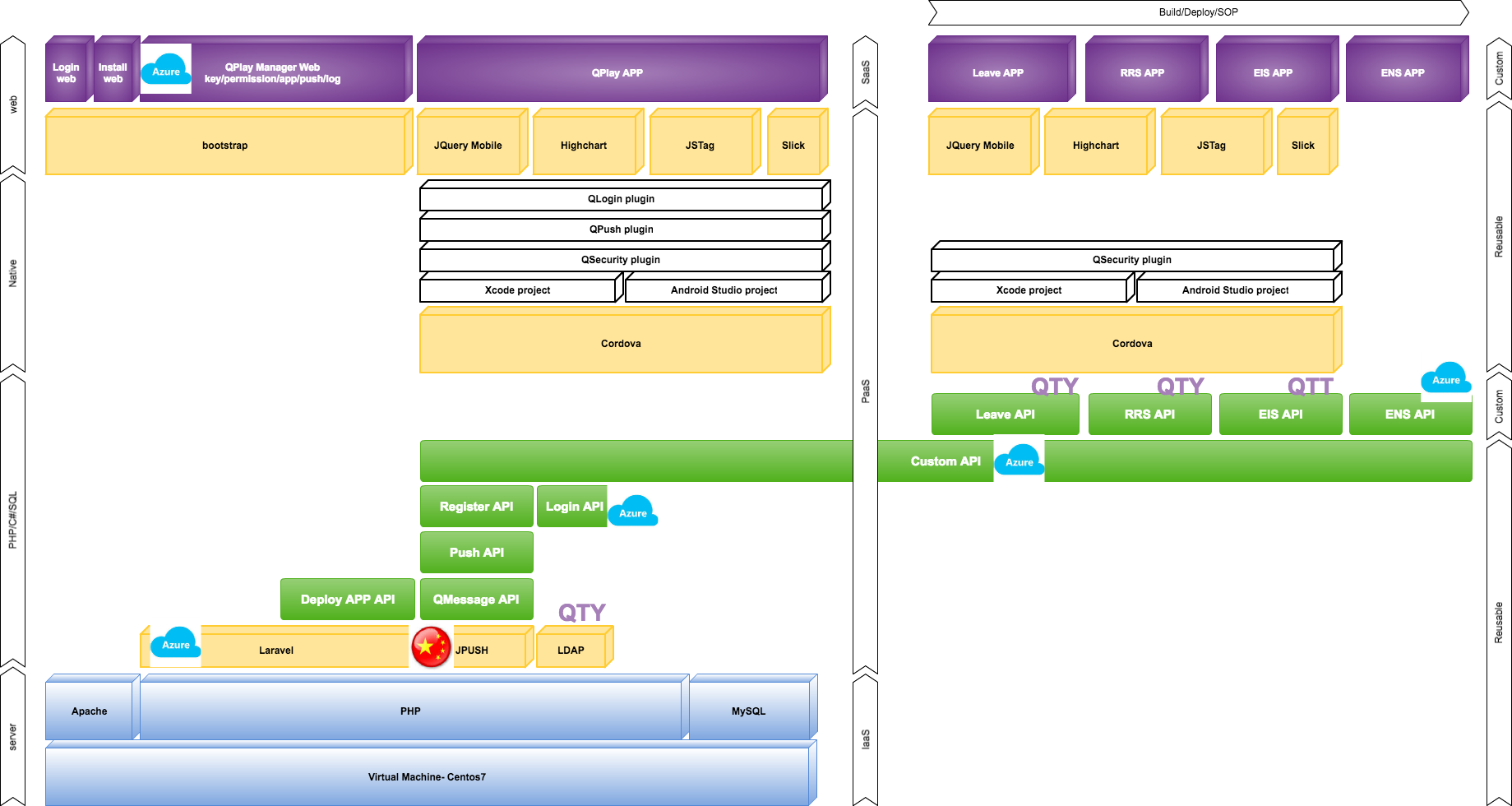
[3.0](#_2s8eyo1) Implementation 8

[3.1](#_17dp8vu) 公用元件 (Public Component) 8

[3.2](#_3rdcrjn) 共用界面 (Public Interface) 9

# 系統設計 (System Design)

## 系統架構圖 (System Framework Diagram)



## 系統流程圖 (System Flow)

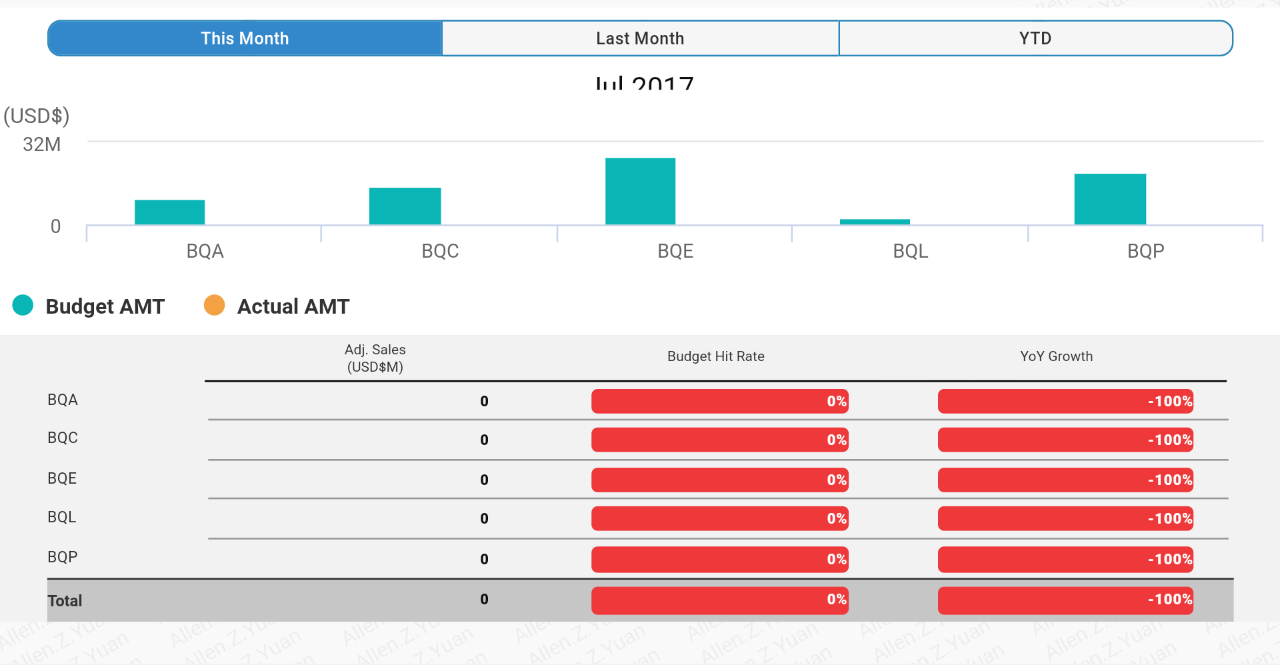
## 類別圖 (Package/Class Diagram)

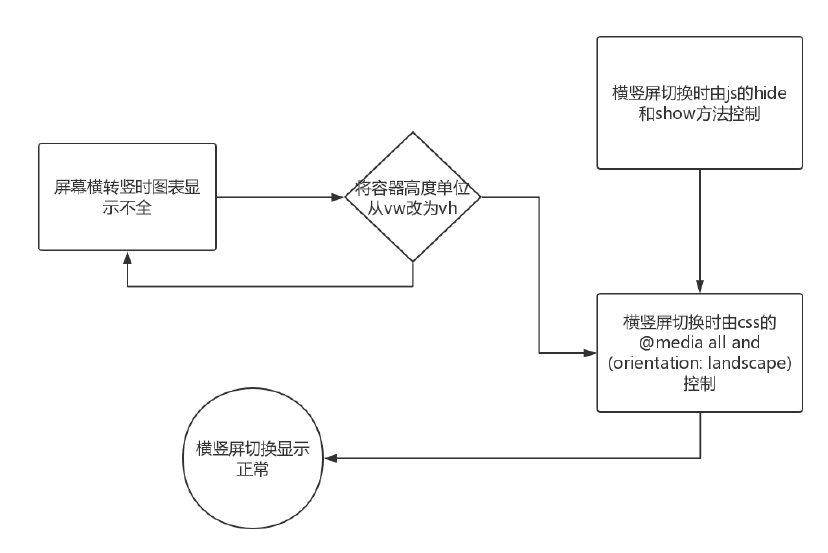
# 資料庫設計 (Database Design)

## 資料庫物件清單 (Database Object List)

## 循序圖 (Sequence Diagram)

### iOS屏幕翻转





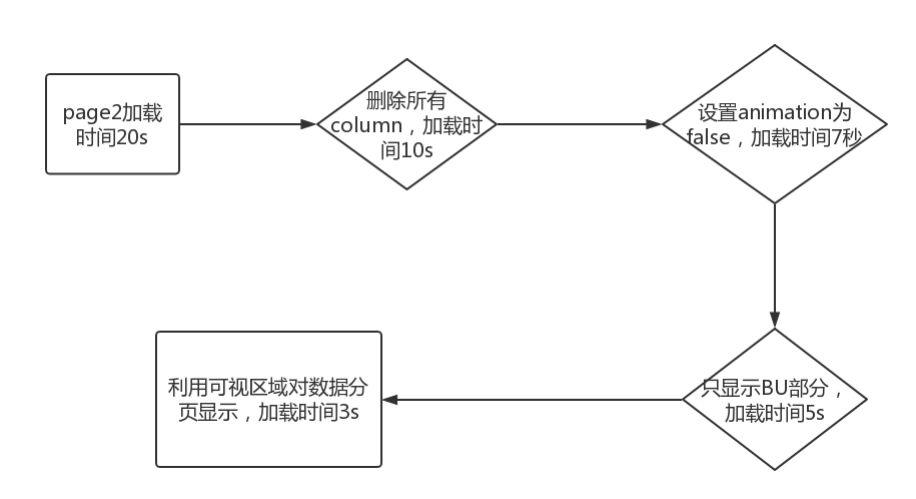
这个问题没有引起足够的重视，之前在做练习的时候也发生过相同问题，但都没有找到真正原因。因为在android下没

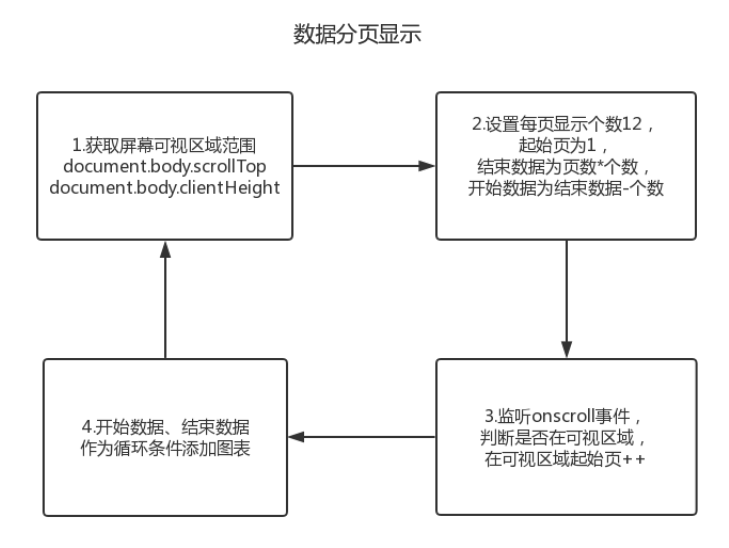
有这个问题，就认为是iOS自身的问题，也没有利用更多时间去分析。下次再做屏幕直转横时，横屏图表部分先不做，让竖屏下的元素在横屏状态下显示出来。等横竖屏切换没有问题之后，再做横屏的图表部分。另外还要学会在iOS上调试，这一点也是非常重要的。

可以将问题延伸，如果css和js同时控制某种样式，那js的优先级较高。但如果给样式添加！important，那就提高了css的优先级。所以能够用css解决的问题就使用css。

完成程度100%。

### 加载超时





调试这个性能问题花费了不少时间，虽然每次都有小幅提升，但显然没有完全解决问题，虽然方向是对的（减少每次图表显示），但没有找到重点，重点其实是可视区域问题。因为没有可视区域的概念，也没有使用过类似的方法，所以在调试性能的时候绕了一些弯路。

目前这种做法还有问题，当滚屏速度太快，中间会有某页的数据没有添加。因为这种方法是在网上查到的，它会在每页最后一笔数据下面增加一个“显示更多”的按钮，我没有加这个按钮。

正确的做法应该是滚屏停止之后，判断哪些内容是在可视区域内，在可视区域内的才添加数据。

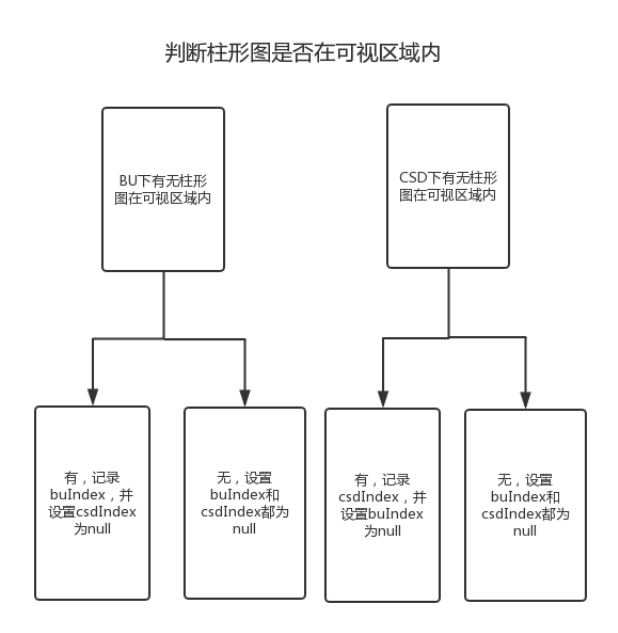
在学习别人方法时候不能只学一半，要先充分理解，再融会贯通。

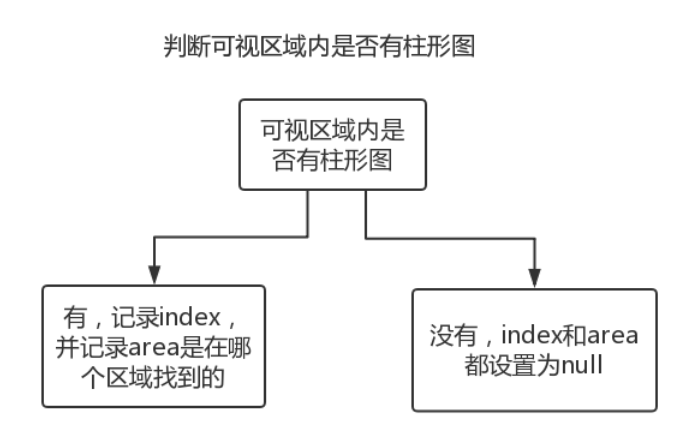
并且实现某些功能的方法和思路有很多种，每次再做之前先考虑有几种方法可以实现，再从性能、困难程度等方面去考量，哪个方法更适合我们的需求。

完成程度：60%，现在因为把facility为all的选项删除后，图表就没有这么多了，所以现在是将每页显示个数12改为50，数据剧本上都是在20个以内。

### 可视区域

横屏的逻辑是判断柱形图是否在可视区域内，在，则在横屏下显示当前柱形图。





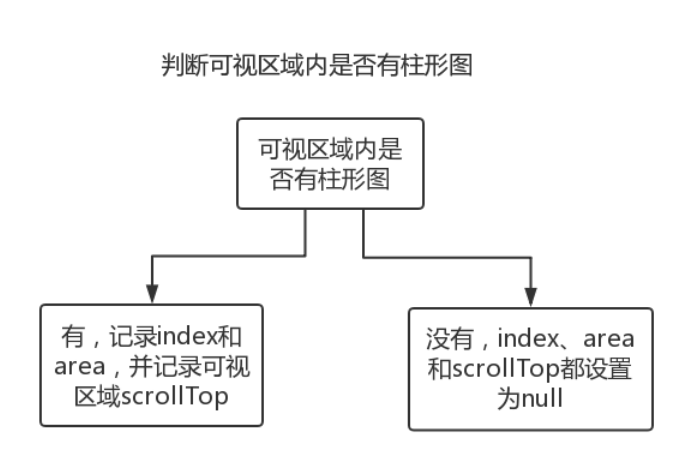
修改之前可视区域判断的时候会有盲区，修改之后不是在BU内就是在CSD内，并且它只会根据一个柱形图来判断横屏显示内容，没有必要分别在BU和CSD内分别设置一个index。

我对于可视区域的判断过于单一，不够灵活，尤其是在BU和CSD交界位置，其实只需要用可视区域高度，减去BU剩余部分的高度，然后看他是否大于1/2即可。然后可视区域的运用还有很多方面，要打开自己的视野，不能局限在一个地方。

完成程度100%。

### 页面定位

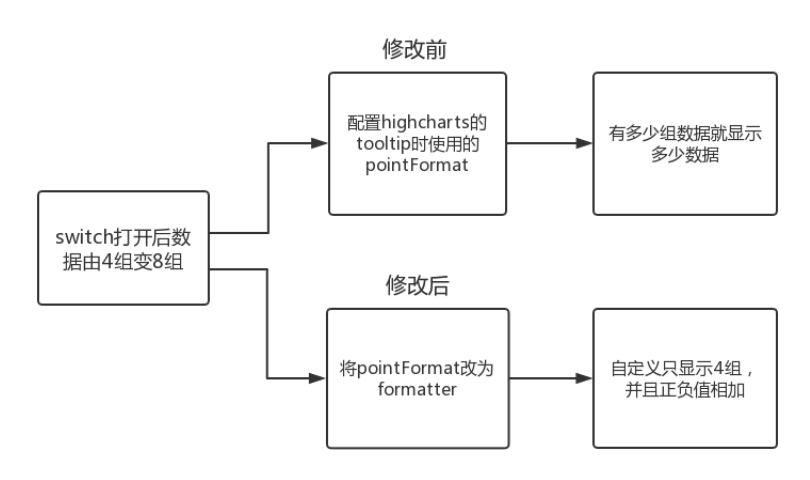
当竖屏show的时候，页面显示都在最顶端。如果滚动以后才展开柱形图，切换到横屏再切换回竖屏，它就不能定位到之前位置。



横屏转竖屏后使用window.scrollTo(0, scrollTop)方法定位到之前的可视区域。因为竖转横后，容器的大小发生变化，但scrollTop定位是绝对距离px，所以横再转竖时就定位不准。然后我的办法是使用setTimeout，等屏幕包括容器完全配适竖屏状态后，再scrollTop，如果翻转屏幕太快，也会定位不准，即使定位准了，因为加了setTimeout，也会有一些延迟，视觉效果上也体验不佳，页面先显示顶部，再跳到指定位置，添加动画效果后也是一样。

目前翻转功能去掉，完成程度60%。困难点是，屏幕翻转时会触发滚屏事件，一滚屏就会改变scrollTop的距离，定位就会不准。如果锁住竖屏，它又不会触发屏幕翻转事件。

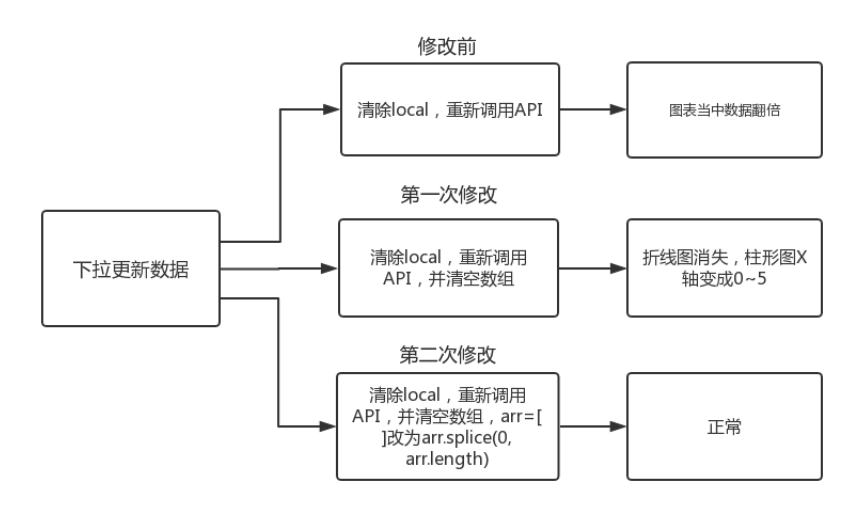
### 图表提示框



Highcharts配置tooltip时的pointFormat，pointFormatter，formatter这3个属性看起来很像，但它们之间是有差别的。因为pointFormat用法最简单，直接接字符串，所以我使用的最多，导致另外2个属性都没有使用过，但另外2个是function，功能更加强大。所以以后都使用formatter，它能解决所有提示框数据显示的问题。

完成程度100%。

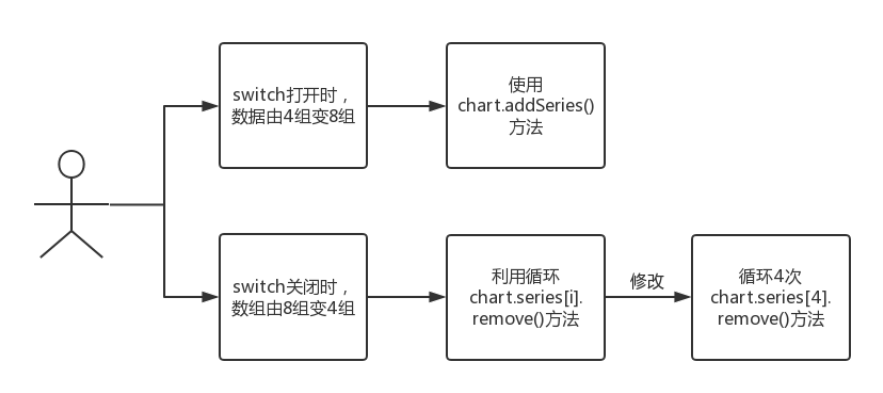
### 下拉更新



我疏忽的地方是当数据发生改变之前，就应该在调用该函数的时候就进行数组初始化的动作。并且清空数组的方法有很多种，当一种方法有问题的时候可以换一种方法去解决。

完成程度100%。

### 切换开关

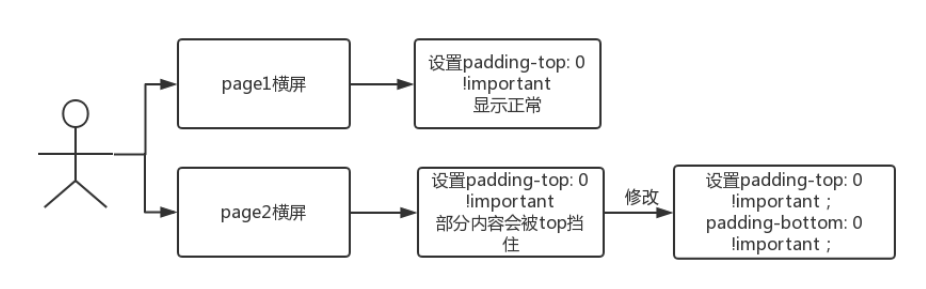


Highcharts在对series进行删除操作的时候，不能利用循环依次删除，只能利用循环控制次数，对数组某一个索引进行多次删除。比如要删除数组索引为4，5，6，7的数据，不能依次删除4，5，6，7。只能对索引为4的数据删除4次。

对数组进行删除操作的时候要注意，循环添加是可以的，但删除时会改变数组元素的索引位置。

完成程度100%。

### 横屏图表



Page1横屏图表没有问题，也不能上下滑动。相同设置page2横屏图表的标题会被遮挡，并且可以上下滑动。

在使用Chrome浏览器F12调试css的时候发现，它还设置一个padding-bottom：45px，但在page1下没有起作用。所以同时添加了padding-bottom: 0 ! important。

我犯的错误和iOS屏幕翻转问题一样，都是认为某个地方设置的都一样，不可能是这里的错误，就认定是它自己的错误。不管怎样，遇到问题都要试着去解决，这才是正确处理问题的方式。

Chrome浏览器F12调试功能十分强大，不光可以调整css，也可以给js代码添加断点，debug remote device模式还能调试移动设备APP。所以查找问题的时候，还是要寻根问底，从最底层开始，逐层排查，依靠工具提供的帮助，快速解决问题。

### 总结

这次QisdaEIS开发对于我来说是一次非常宝贵的经验，让我了解了整个QPlay APP的开发流程，以及APP背后的运作方式，包括如何使用github进行版本控制和团队合作，如何使用jenkins进行daily build，如何使用gulp合成文件或者压缩文件，如何在PC浏览器上debug手机APP等等。

也使我认识到自己的不足，一些编程思想需要转变，比如某些已经加载过的数据或html如果没有变化就不需要再次加载，应该记录它的状态。还有一些状态改变或者变量初始化的时机问题，有时候在做一些操作的时候，已经修改了，数据或样式却没有变化。还包括代码的可维护性、方法的封装，什么情况下是引用，什么情况下需要传参，还需要提高代码的复用性。还有一些技巧和经验需要通过平时练习的积累，比如遍历数组的方法、性能的优化、相容性问题等等。

争取在下次开发中做的更好。

## 2.3資料表結構 (Table Layout)

# Implementation

* Index.js
  + Define pageList
  + Define appKey, appSecretKey
  + Define window.initialSuccess()
  + Define onBackKeyDown() for android platform
  + Define onPause() / onResume() when APP work in background / foreground
* viewMain.html
* viewMain.js
* viewDetail.html
* viewDetail.js

## 公用元件 (Public Component)

## 共用界面 (Public Interface)

* App.css
* App.html
* App.js