**iApp笔记**



笔记说明 :

iApp是国产可视编程工具,支持iyu,java,lua,js等语言。(这里先讲iyu)

对个人纸质笔记的整理和分享,希望看到这个后发现问题时能纠正我,也希望新手能在这学到知识。

对于一些要点和细节,我写了一些通俗的解释(我的纸质笔记上没必要写那种太入门的东西),前期的序号标题是官方教程的标题,后期标题是自己加的。

这里为了直观,我把变量名标记了黄色背景,代码块标记了*加斜*,主要代码标记了下划线。

[]里的部分表示可有可无的内容,但是也是是很重要的,一定要看。

可以随意填补的代码块,或者可以用fn函数代替的地方我用蓝色字代码来代替。

如果发现有红色字体,则代表这个名词或代码在之前文章里没有出现过,但在以后会讲。

范例部分放在一个单独的框内

这里的一些实例在介绍里都会尽量给出,会多处引用帮助文件或者我自己写的实例,并加以分析(超详细)。

参考资料:iApp俗语言零基础系列教程(官方),百度贴吧(iApp),《俗语言》速成开发手册(官方)。

如果你是从这个笔记初学,还不会有关界面的任何东西,那么没关系"创建→创建新应用",之后自己熟悉一下,之后就可以看这个笔记学习iyu代码(只在iyu界面和myu界面使用)部分以及更高级的布局界面。

**by:洮羱芝闇**

IYU篇

1. s变量的用法

声明变量但不赋值 值为空 可以用来做初始变量

俗语言的变量不需要指示类型(能自动识别)

变量分区域性(以变量a为例) 分为 局部变量(a) 界面变量(ss.a) 全局变量(sss.a)

变量名能包括数字,字母,下划线等(现支持汉字) 但变量名不能是纯数字

语句: *s a ="123" ss a="123" sss a ="123"*

注:这里的 s是给局部变量赋值的代码 ss是界面变量赋值的代码 sss是全局变量赋值的代码

局部变量就是在一个代码串(页)运行结束后它就自动变成空值(null)了,也就是说生命周期随代码串(页)结束而结束,而界面变量一经创建,生命周期随界面的销毁而结束,全局变量就是可以跨界面的变量,不随界面关闭而消失,常用于界面交互

2. 注释与syso打印数据

用 //或.注释 进行注释 必须但独占一行 除此之外用*/. 代码 ./ 可以注释掉部分代码*

用*syso(a)* 可以打印出a的内容 打印出的内容可以在调试界面看到

3. f判断语句

判断语句主要的代码就是 *f(条件){代码} [else f(条件){代码} 或 else {代码}]*

判断语句的作用是对比数据,之后根据不同的数据执行不同的代码,简单来说就是达成了()里的条件,便会运行相应的{}里的代码

判断语句后可以用否则(else)来进行连续的判断比如 *else {代码}* 是表示不符合f()里的的条件就运行else的{}里的代码,此之外还有特别的*else f(条件){代码}* 是表示有条件的的否则(else),条件仍然是在()里填写

而且判断语句是可嵌套的,也就是说{}里还可以有判断语句

要注意不同数据类型的变量是不相等的

在()里 常会用到的判断符如下:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| == | != | >= | <= | > | < | ?\* | \*? | ? |
| 是否对等 | 是否不等于 | 是否大于或等于 | 是否小于或等于 | 是否大于 | 是否小于 | 字符串开通是否相同 | 字符串结尾是否相同 | 字符串是否被包含 |

()用到的逻辑符如下:

|  |  |  |
| --- | --- | --- |
| || | && | ! |
| 或者 | 并且 | 反意 |

实例1:判断2是否等于1

|  |
| --- |
| *s a = 2*  *//给a赋值为2*  *f(a == 1)*  *{*  *syso("等于1")*  *}*  *//判断a是否等于1,若等于1,则打印“等于1” 显然,这段代码运行后不会打印任何结果* |

实例2:判断1是否等于3,否则判断1是否等于2,如果仍然不对,那么输出1等于1,3等于3

|  |
| --- |
| *s a = 1*  *//给a赋值1*  *s b = 3*  *//给b赋值3*  *f(a == b)*  *{*  *syso("a等于b")*  *}*  *//判断a是否等于b,若等于,则打印"a等于b"*  *else f(a == 2)*  *{*  *syso("a等于2")*  *}*  *//若a等于b不成立,但a等于2成立,则打印"a等于2"*  *else*  *{*  *syso("a等于1,b等于3")*  *}*  *//若以上的条件都不符合,则打印"a等于1,b等于3" 显然,这个输出的结果是"a等于1,b等于3"* |

4. w循环与for循环

w循环的语法是*w(条件){代码}* 这个的条件注释方法和f判断语句一样,当符合条件时就运行{}里的代码,但要注意如果出现了不是在线程中出现的死循环,程序会崩溃

w循环实例:让变量a依次减一,从而使循环99次,并且每次都把a打印一遍

|  |
| --- |
| *s a = 99*  *//给a赋值99*  *w(a > 0)*  *{*  *syso(a)*  *s(a - 1, a)*  *}*  *//w循环条件是a是个大于0的数,循环内容是先打印a的值,之后让a的值减一,直到循环到第99次跳出循环* |

for循环的语法是*for(条件){代码}*  这个功能的条件不仅支持数值,还支持数组数据(这个以后会详解),参数可以给予另个参数,一个为初始循环的值,一个是最大循环值。 而且这个条件格式在括号里是类似*(1; 20)*这个样子的,看实例:简单for循环实例:打印10次"循环10次"

|  |
| --- |
| *s a = 1*  *//给a赋值1*  *s b = 10*  *//给b赋值10*  *for(a; b)*  *{*  *syso("循环10次")*  *}*  *//a从1变到10 打印10次"循环10次"* |

5. t线程与tw提示语句

创造一个线程的语法是*t(代码)* 线程用来执行一些要运行很久的代码(下载文件，获取网页源码，大量的文件操作),线程中如果要更新界面(改变界面上一些控件的属性或弹出窗口和提示字符),就必须用*ufnsui(){}* 来设置空间数据,否则会报错,用线程的目的是防止处理量太大而导致界面卡住,而且在线程里允许出现死循环(否则程序会崩溃),也就是说可以一直循环(随时刷新控件得以实现),启用新的线程帮你处理代码，这样不会影响到主线程

t线程实例:

|  |
| --- |
| *t()*  *{*  *syso("新线程里执行代码")*  *}*  *//创建一个新线程,线程内执行的代码打印"新线程里执行代码"* |

tw提示语句,嗯,*tw()* 可以直接在屏幕上提示()里的字符,用于提醒用户,界面显示时长大约为 2秒钟,弹出代码中的文字,来提醒用户,这个字符可以是变量,而且这个()可以除弹出的内容以外另外加一个参数()里用逗号分开的两个数值,特殊名词或变量啥的),第二个参数可以是0或1(1表示显示的时间长久；0表示显示的时间短暂),在()中的内容可以用 \n 来换行

tw提示语句例一:

|  |
| --- |
| *tw("你好")*  *//在界面上弹出"你好"* |

tw提示语句例二:

|  |
| --- |
| *tw("你好\n吗？", 1)*  *//在界面上长久(其实差不多)弹出"你好"* |

6. ssj 设置或修改控件事件代码

除了在可视性编程时编辑控件事件执行的代码,我们还可以用ssj语句后改这些代码,用ssj语句会直接替换原来的代码,再次触发的时候会执行新的代码,这个代码可以作用于触发代码的控件本身,也就是说可以实现每次以不同形式触发同一控件的时候可以改变不同方式触发(比如长按和点击)同一控件所执行的代码,这样便可做成每次连续点击触发效果不同的控件了

格式是这样的*:ssj(){代码}*  这个()里面有两个参数 第一个参数是要更改的控件的控件id("控件→空间管理"里可以看到,还有就是生成控件的时候上面标得号也是),第二个是要更改触发执行得触发方式

触发方式对照表如下:

|  |
| --- |
| clicki=单击事件  touchmonitor=触屏监听事件  press=触屏长按事件  keyboard=键盘触发事件  pressmenu=触屏长按菜单事件  editormonitor=框编辑监听事件  ontextchanged=文本内容已改变  beforetextchanged=文本内容改变之前  aftertextchanged=文本内容改变之后  focuschange=获得焦点事件  onscrollstatechanged=滚动状态已改变  onscroll=滚动  clickitem=单击项目事件  onprogresschanged=加载过程进度改变  shouldoverrideurlloading=加载网址之前  ondownloadstart=文件下载事件  onpageselected=滑动切换界面事件  onpagescrolled=滑动切换界面过程  onpagescrollstatechanged=滑动操作过程  ondrawerclosed=侧滑关闭事件  ondraweropened=侧滑展示事件  onoptionsitemselected=项目选择  onitemselected=选择项目事件 |

实例:

|  |
| --- |
| *s id = 3*  *//这里假设控件3是一个按钮,我们把控件id也就是3赋值给变量id*  *ssj(id, "clicki")*  *{*  *tw("ok")*  *}*  *//这里我们修改了单击控件id是3的控件(也就是id是3的按钮)时触发的时间为弹出提示"ok"* |

7. 项目结果与设置文件路径

iApp这个软件在软件制作阶段的项目会有实际的项目文件,路径一般是/storage/emulated/0/iApp/ProjectApp/

打一个开工程文件,可以看见有几个个文件夹:files,res,src(,apk,bin),还有一个png图标,一个xml配置文件,一个.project

这里AndroidManifest是配置文件,src是界面的源码,res是项目用到的资源,files是项目的私有数据,在/storage/emulated/0/iApp/ 下有一个叫做Userimg的文件夹是用户资源库,所有项目都可以引用其中的内容

注意:如果在直接SD卡上其他位置选择资源,它们不会自动复制到资源目录里,不能打包走,也就是说软件会出问题(图片无法显示,资源无法引用什么的)

项目的资源在打包后会出现在安装包内的assets文件夹里,在程序中可以引用

系统图标是通用的,也就是说系统图标引用后就不用做别的事了,在其他手机上也是可以正常显示的.

下表是代码中引用资源位置的标识,将来会用到.

|  |  |  |  |
| --- | --- | --- | --- |
| @ | $ | % | (留空) |
| 项目资源 | 应用私有文件 | SD卡上的文件 | SD卡上的文件 |

8.文件操作

文件操作的代码有这么几个,我先用表格展示,之后再逐一说明.

|  |  |  |
| --- | --- | --- |
| 代码 | 参数数量 | 功能 |
| fw() | 2或3 | 写入操作文本文件 |
| fe() | 2 | 判断文件是否存在 |
| fs() | 2 | 获取文件大小 |
| fr() | 2 | 把文本内容读出来 |
| fc() | 3或4 | 复制文件 |
| ft() | 3 | 文件转移 |
| fl() | 3 | 读取一个目录的文件列表 |
| fdir() | 1或2 | 获取根目录路径或目录的绝对路径 |
| fd() | 2 | 删除文件 |
| fj() | 3或4 | 压缩文件夹至zip |
| fuz() | 4或5 | 解压zip部分文件 |
| fuzs() | 3或4 | 解压整个zip |
| fo() | 1 | 打开文件 |

咱们先来说说*fw()*这个代码,将文本数据写入到文件里面,然后将这个文件保存到指定的路径(文本型),如果原本没有这个文件,它能自动的创建这个文件,如果有这个文件,那么这句代码能将文本内容自动改成设定的结果.这句代码如果有两个参数,那么第一个参数是保存的路径加上要创建的文件名,第二个参数是txt文件(把这个任意文件看做txt文件)的内容.如果这句代码有三个参数,前两个参数是和两个参数的时候一样的,这第三个参数是写入文本的编码. 以上的参数都是文本型的,所以都要加上双引号

实例1(两个参数):

|  |
| --- |
| *s a = "%abc.txt"*  *//给a 赋值"%abc.txt"*  *s b = "我是一个txt文件的内容"*  *//给b 赋值"我是一个txt文件的内容"*  *fw(a, b)*  *//将文本b数据写入至SD卡根目录的a文件里面* |

实例2(三个参数):

|  |
| --- |
| *s a = "%abc.txt"*  *//给a 赋值"%abc.txt"*  *s b = "我是一个txt文件的内容"*  *//给b 赋值"我是一个txt文件的内容"*  *s c = "utf-8"*  *//给c赋值"utf-8"*  *fw(a, b, c)*  *//将文本b数据以c编码写入至SD卡根目录的文件里面* |

接下来说说*fe()* ,它是用来判断文件是否存在的,这个代码就两个参数,一个是输入的,一个是输出的,这第一个输入的是要判断是否存在的文件所在路径加上文件名(文本型),这第二个参数是输出的判断结果,类型是布尔型(false或true)

注:false为不存在,true为存在

实例:

|  |
| --- |
| s a = "%abc.zip"  //给a赋值"%abc.zip"  fe(a, b)  //判断SD卡跟目录上是否存在名为abc.zip的文件,并获得结果b  tw(b)  //把判断结果b在界面上弹出 |