系统注意事项

1. 当前基于blockly原生文件之上更改的如下源文件：

block\_svg.js, blockly.js, field.js, field\_angle.js, field\_dropdown.js, flyout.js, scrollbar.js, toolbox.js, workspace\_svg.js, css.js, inject.js  
# block.js, block\_render\_svg.js, dropdowndiv.js

注意：下次blockly版本更新之后要注意将以上文件修改之处整理进入文档

1. 凡事修改过blockly中的core中文件之后，都需要定位到项目的根目录下运行python build.py命令将core文件重新编译压缩到对应的压缩文件中去，否则更改过的文件会在项目中不生效。压缩blockly源码的第二种方式，就是在具有java环境中（具有java jdk）运行，具体参考网址：

<https://developers.google.com/closure/compiler/docs/gettingstarted_app>

1. 监听项目文件变动并完成实时压缩命令 webpack –w （有格式压缩，方便调试），webpack -p（无格式压缩，系统上线时使用，不便于调试）

后期优化：A. 多程序块跳转可以使用webworker

结合程序在VM中运行

B. webpack优化，可以将变动库（eg：系统业务代码库）以及不变动库（eg：jquery，react等等）分开压缩

系统核心技术点参考文档网址

1. Blockly官网：

主页： <https://developers.google.com/blockly/>

开发者文档：<https://developers.google.com/blockly/guides/overview>

blockly git获取路径：<https://github.com/google/blockly>

2.Google closure

<https://developers.google.com/closure/>

3. React

<https://facebook.github.io/react/>

<http://www.ruanyifeng.com/blog/2015/03/react.html>

<http://m.blog.csdn.net/article/details?id=51034349>

<http://react-china.org/t/react-swipes/9042>

1. webpack

<http://blog.csdn.net/zaichuanguanshui/article/details/53610694>

<https://zhuanlan.zhihu.com/p/21346555>

<https://webpack.js.org/>

<http://webpack.github.io/docs/>

5.node

<https://nodejs.org/zh-cn/>

<http://nodejs.cn/>

系统环境配置

1. 安装node
2. 安装npm
3. 到项目根目录下使用npm install命令运行项目package.json文件，安装系统依赖包，自动创建node\_modules文件夹
4. 使用npm安装webpack 命令：npm install webpack –g
5. 使用npm命令安装项目中需要的框架

# npm install --save-dev style-loader css-loader  
# npm url-loader svg-url-loader --save-dev  
# npm install --save-dev webpack-dev-server  
# npm install --save-dev url-loader  
# npm install --save-dev file-loader  
# npm install --save-dev jsx-loader  
# npm install extract-text-webpack-plugin --save-dev  
# npm install --save-dev react-html5video  
# npm install --save-dev immutable

1. 到项目根目录下运行webpack –w命令实时监控并压缩项目文件
2. 到项目根目录下运行node web-server.js 运行web-server.js文件将项目发布至node服务器，就可以在本地访问项目了

系统核心技术

1. wepack 构建工具（webpack 既可以压缩js也可以压缩css，压缩文件可以压缩成有格式便于调试的文件，也可以压缩成无格式文件）
2. blockly
3. acorn\_interpreter blockly xml程序运行解析器也有第二种方式eval（code） 参考blockly官网
4. React 项目中各组件
5. Prettify 在项目中主要用于swift界面对代码块的布局渲染
6. immutable React复制数据类
7. lodash 数组对象控制类
8. Google

系统文件说明

1. 文件压缩

打开系统doc命令，到项目根目录项目根目录，eg：

cd /Users/hushuiping/Projects/Jimu/JimuBlockly/blockly

运行webpack –w命令 将初始化ubt\_blockly\_course.js或者ubt\_blockly\_app.js文件加载的所有js文件压缩到course.bundle.js ／course.min.css 或者app.bundle.js／ app.min.css 文件中，初始化html文件中会引用course.bundle.js ／course.min.css 或者app.bundle.js／ app.min.css 文件

1. 目录engine下文件说明

API.js：程序块运行，blcok块高亮等程序处理工具类。Eg：查询陀螺仪传感器值，读取陀螺仪传感器值，读取触碰传感器值，查询触摸传感器值，读取红外传感器值，查询红外传感器值，程序跳转到条件start块等等

ubt\_blockly\_app.js：逻辑编程入口文件（国际化以及系统初始化文件加载）

ubt\_blockly\_course.js：课程入口文件（国际化以及系统初始化文件加载）

1. 目录engine/adapter说明：主要针对block快重写，以及工作空间菜单管理，具体文件详细说明：

block\_extension.js：Blockly的block块的重写管理，主要控制，eg：垃圾桶图片更改，播放音效更改等等

block\_trashcan.js：重写垃圾桶 暂时无用

field\_colour\_adapter.js：运动模块下，旋转舵机，执行舵机动作时长，旋转舵机360度这些block块的点击事件重写，主要用于设置点击之后弹出框内容

field\_default\_id\_adapter.js：未连接模型时，程序块中各传感器（红外，触碰，陀螺仪等）初始ID模拟

field\_device\_tilt.js：手机/平板倾斜block块点击事件重写，开始，事件菜单中含有该block块

field\_display\_emotion.js：显示表情block块点击事件重写

field\_dropdown\_adapter.js：block含有下拉框程序块点击事件重写

field\_gyroscope\_angle\_adapter.js：陀螺仪block块点击事件重写

field\_infrared\_sensor.js：红外传感器block块点击事件重写

field\_number\_adapter.js：涉及到数字弹出框程序块中点击事件重写，eg：旋转舵机中设置时长（80ms至5000ms），设置角度-188至188控制等

field\_play\_tune.js：设置音效／音调block块点击事件重写

field\_setting\_light.js:设置情景灯光block块点击事件重写

field\_textinput\_extension.js：重写block的Blockly.Field

文件，提供给其他重写组件调用或者重写

field\_time\_adapter.js：涵盖设置时长block块点击事件重写

field\_touch\_sensor.js：事件点击（单击，双击，长按）block块点击事件重写

field\_variable\_adapter.js；涵盖设置变量block块的点击事件重写

toolbox\_config.js：系统菜单生成管理

toolbox\_extension.js：系统左边菜单点击事件重写

xml\_extension.js： 系统Blockly.Xml处理

1. 目录engine/common 说明

events\_listener.js：事件处理工具类

condition文件夹：

program\_branch\_generator.js：生成程序块ID（多分支程序运行分支序数生成类），主要用于多个同样程序块跳转唯一ID，便于程序块运行指向。Eg：在开始菜单中手机/平板倾斜，多个手机/平板倾斜程序块时需要记录各自ID之后才能控制程序具体跳转运行

sensor\_condition.js：传感器条件类,涵盖各传感器条件，eg：手机/平板倾斜的branchID，红外传感器的

sensor\_condition\_swift\_code.js：主要提供给swift代码使用的传感器条件类,涵盖各传感器条件，eg：手机/平板倾斜的branchID，红外传感器的

sensor\_condition\_value.js：提供各传感器块获取值，核心方法：SensorConditionValue.prototype.initAllCondition

返回当前块中的数据信息

dialog文件夹：

popup\_show.js：系统提示信息管理文件，下拉显示信息方式

program文件夹：

command\_sequence.js：命令顺序执行模块

delay\_command.js：延迟执行命令的封装

delay\_command\_manager.js：延迟执行命令管理器的封装

interactive\_robot.js：控制程序单不执行，是否需要等待后台返回运行结果

program\_manager.js：保存或加载workspace程序，即workspace程序管理

program\_runner.js：程序块运行管理

program\_state.js：监控程序运行状态

program\_init.js：系统入口文件，主要初始化工作空间，初始化设置工作空间菜单栏样式，加载初始化react组件，以及载入国家化blockly内部语言

timer文件夹：

timer.js：记录程序运行时间的模块

utils 文件夹：

blockly\_utils.js：工作空间block块处理工具类，主要提供工作空间block块查找，返回工作空间所有block块集合，设置可删除block块，设置不可删除block块，设置可移动block块，swift代码生成，以及更新当前block块中最新数据等

create\_function\_string.js：将方法以及参数拼装成string工具类，主要提供各block块在workspace 的xml为字符串时自动调用所拼接方法

event\_utils.js：事件工具类，主要提供长按，单击，onTouchStart，onTouchMove，onTouchEnd事件归类处理

map.js：数组处理工具类

utils.js：系统方法提供工具类

validate 文件夹：

default\_number\_validate.js：默认数据验证基类，主要用于数据范围验证

emoji\_number\_validate.js：表情数据验证基类，主要用于数据范围验证

led\_number\_validate.js：灯光数据验证基类，主要用于数据范围验证

scenelight\_number\_validate.js：情景灯数据验证基类，主要用于数据范围验证

time\_number\_validate.js：时间数据验证基类，主要用于数据范围验证

times\_number\_validate.js：循环次数数据验证基类，主要用于数据范围验证

number\_validate.js：（提供外部调用）数字验证实现类主要提供数据范围验证包括表情数据验证，灯光数据验证，情景灯光数据验证，时间数据验证，循环次数数据验证（通过interface.js

类实现emoji\_number\_validate.js，led\_number\_validate.js，scenelight\_number\_validate.js，time\_number\_validate.js，times\_number\_validate.js类中的validate方法

）

courses 文件夹：主要用于课程内容配置

jsx\_blockly文件夹：系统各弹出框组件类（若当前系统为逻辑编程入口，那么当前文件夹涵盖所有弹出框组件；若当前系统为课程入口，那么课程会使用到当前文件夹中部分组件不是全部，课程独有的弹出框组件类放置在jsx\_course文件夹中）

jsx\_common 文件夹：系统弹出框组件服务类，主要提供给jsx\_blockly，jsx\_course两个文件夹中的弹出框组件类调用

jsx\_course 文件夹：课程独有弹出框组件类

service 文件夹：

blockly\_datas.js：block块数据工具类，主要用于组装初始化数据，初始化各传感器数据，以及各传感器数据设置等（eg：各传感器下拉列表数据设置以及模拟）

colours.js：当前系统颜色工具类，eg：左边菜单颜色设置

course\_data.js：课程数据处理工具类，eg：根据课程ID获得单个课程内容，json转字符串，获得课程索引，返回单课课程

data\_service.js：逻辑编程命令管理工具类

jsx\_manage\_service.js：封装创建所有的jsx 入口

robat\_command.js：于ios/android接口对接入口工具类

storage 文件夹：

base\_storage.js：数据存储实现类提供外部调用，主要通过interface.js（接口注入实现管理类）实现file\_storage.js，local\_storage.js中数据存储调用

file\_storage.js：数据文件存储基类

interface.js：面向接口变成，接口注入实现工具类

local\_storage.js：本地数据存储基类，主要包括保存程序到本地存储，删除程序，修改程序到本地存储，载入程序到工作空间，从本地存储中载入程序等等

5. project 文件夹下文件说明

blocks 文件夹：当前系统菜单管理

css 文件夹：当前系统自定义所有样式管理

fonts 文件夹：当前系统所有外部字体引用管理

images 文件夹：当前系统所有外部图片引用管理

ionicPopup 文件夹：逻辑编程相关业务管理，其中包含系统音效播放文件

project/ionicPopup/common/infrared\_sensor.js: 功能 ：js与IOS或者android交互的模块(主要针对红外传感器数据对接管理)

project/ionicPopup/common/interactive\_methods.js: 功能 ：js与IOS或者android交互的模块(主要用于IOS或者android回调 js)

msg 文件夹： 当前系统除去blockly自带翻译之外其他所有翻译文件管理

swift 文件夹：各菜单swift代码转换

系统核心流程

1. 系统文件压缩流程
2. blockly文件压缩，通过python build.py命令压缩（在doc控制台cd到项目的根目录下运行该命令即可关于该部分说明以上“系统注意事项”中第二点有详细说明），主要将blockly的core文件夹下的blockly文件压缩到blockly\_compressed.js，blockly\_uncompressed.js，blocks\_compressed.js，javascript\_compressed.js等文件中去
3. 系统业务代码压缩，主要压缩当前engine下的文件以及系统css文件，即系统入口文件（ubt\_blockly\_app.js／ubt\_blockly\_course.js加载的所有文件），通过webpack –w 命令压缩（该压缩条件在以上“系统注意事项”中第三点有详细说明）
4. gitLib上Blockly.zip中文件生成说明

通过远程服务器10.10.1.14中D://java/Jenkins/workspace/jimu\_blockly\_dash的build.xml配置从git上Blockly路径下获取对应的文件压缩成Blockly.zip并推送至dash的git服务器中

代码分支以及Jenkins打包管理

1. git上远程分支：

A：develop 涵盖angular分支

B：master develop\_removeng备份分支（去除angular）

C：develop\_removeng 分支（去除angular）

D: Jimu\_Blockly\_Release分支（对应分支develop\_removeng提供测试以及上线使用，每次发布测试以及上线版本前都需要将该分支的代码与develop\_removeng开发分支代码取得同步）

1. JenKins打包：

A：开发打包 Jimu\_blockly\_removeng对应git分支develop\_removeng

B: Jimu\_blockly\_dash对应git分支Jimu\_Blockly\_Release（为测试以及上线使用）

1.Google，lodash操作数据集合对象

2.自己重写core类

3.运行blockly解析器两种方式：acorn\_interpreter

以及第二种eval（code）

4.方法绑定作用（\_initApi）提供给程序块拼接成string后可以直接解析运行

5.运行块流程

6.程序运行模拟器

7.生成可执行代码块

\_addSensorConditionListener : function() { //注册传感器条件监听器 注册分支

window.sensorCallback ：分支跳转ios，android调用