



谷歌支持教育部产学合作协同育人项目成果

华章科技

周庆国 崔向平 郭朋◎编著

模块化编程，  
轻松迈入程序设计大门

案例驱动式学习，  
在实践中体会编程思想

以解决问题为核心，  
培养计算思维和创新能力

# BLOCKLY 创意趣味编程



机械工业出版社  
China Machine Press

## 图书在版编目 (CIP) 数据

Google Blockly 作为一种可视化编程语言，允许您通过类似玩拼图玩具的方式，构建出一个程序。本书配有丰富的案例、图片，对 Blockly 的基础知识、程序结构以及高级使用进行了详细的介绍。在每一章结束后都搭配有一个游戏，帮助您巩固本章知识，反思学习效果，更快速地上手 Blockly 编程。此外，每一章的课外拓展资料则为计算机语言相关的小故事，可以帮助您了解计算机语言的发展历史。

本书的使用者既可以是完全没有编程经验的编程初学者，也可以是有了一定的编程基础、想要了解 Blockly 的编程爱好者，还可供中学信息技术教师向学生介绍编程相关知识。

出版发行：机械工业出版社（北京市西城区百万庄大街22号 邮政编码：100037）

责任编辑：赵亮宇

责任校对：殷虹

印 刷：中国电影出版社印刷厂

版 次：2019年 月第1版第1次印刷

开 本：186mm×240mm 1/16

印 张：10

书 号：ISBN 978-7-111-62900-9

定 价：69.00元

---

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010) 88378991 88379833

投稿热线：(010) 88379604

购书热线：(010) 68326294

读者信箱：hzjsj@hzbook.com

版权所有•侵权必究

封底无防伪标均为盗版

本书法律顾问：北京大成律师事务所 韩光/邹晓东

主编 周庆国 崔向平 郭朋

参编（按姓氏笔画排序）

尹航 邓文博 冉竹君

李丹 李鹏 李东辉

张宝仁 陆禹文 赵冲

赵义仓 胡轶凛 漆昱涛

魏小辉

# 前言

以计算机技术为先驱的科技革命深刻地影响着我们的生活生产方式、管理方式以及思维方式，推动着人类社会的蓬勃发展。有人说计算机技术就像人大脑的延伸，帮助人们计算、设计、创造、解决各种各样的问题。随着智能时代的到来，我们应尽早的学习和掌握计算机知识，同时也希望大家能拥有像写作一样普遍的编程技能。

编程难不难？这恐怕是每一个编程初学者都会提的问题。自从编程语言出现以来，经过几十年的发展，已有上千种不同的编程语言。如何选择适合自己、能够有效帮助自己解决实际问题的编程语言也是一个头疼的问题。此外，从头开始学习不同的编程语言也耗费编程人员许多的精力。

2012年6月，Google发布了完全可视化的编程语言Google Blockly。Blockly代码块由类似于积木的图形对象构成。您可以通过类似玩拼图玩具的方式，将它们拼接起来，创造出简单功能，然后将一个个简单功能组合起来，最终构建出一个程序。大家在创建程序的过程中只需要拖动鼠标，不需要键盘敲击，相较于其他种类编程语言，Blockly语言无须大家编写冗长的代码、考虑复杂的语法规则，趣味性更强，并且可以根据需要导出不同语言的代码，例如Python、JavaScript、PHP等，降低了学习成本。

本书旨在帮助您快速入门Blockly，掌握Blockly的使用方式，以便利用Blockly编写出您所需的程序。本书的使用者既可以是完全没有编程经验的编程初学者，也可以是有了一定的编程基础、想要了解Blockly的编程爱好者，还可供中学信息技术教师向学生介绍编程相关知识。

本书共设置了七章，每一章都包含详尽的案例，建议您按照目录顺序进行学习并亲手做一遍书中的案例，每一章的结尾都搭配一个小游戏，您在学习完一章的内容之后，可以通过打游戏的方式，巩固本章知识，反思学习效果。此外，每一章的课外拓展资料则为计算机语言相关的小故事，可以帮助您了解计算机语言的发展历史。

在本书编辑过程中不免有纰漏，欢迎读者批评指正。

# 目 录

前言 .....	IV
<b>第 1 章 Blockly 概述.....</b>	<b>1</b>
学习目标.....	1
知识图谱.....	1
1. 1 什么是 Blockly.....	1
1. 2 Blockly 编程环境.....	2
1. 3 Blockly 模块功能介绍.....	4
1. 4 小试牛刀——游戏：拼图.....	7
1. 5 本章练习.....	8
1. 6 课外拓展.....	8
<b>第 2 章 Blockly 编程基础与顺序结构.....</b>	<b>10</b>
学习目标.....	10
知识图谱.....	10
2. 1 数据类型.....	11
2.1.1 数据的含义.....	11
2.1.2 数据的表示形式.....	11
2.1.3 Blockly 中的数据类型.....	12
2. 2 变量的定义.....	16
2.2.1 变量的创建.....	16
2.2.2 变量的初始化.....	17
2. 3 运算符及其优先级.....	18
2. 4 顺序结构.....	21
2.4.1 赋值语句.....	22
2.4.2 输入与输出.....	22
2.4.3 顺序结构程序设计举例.....	25
2. 5 小试牛刀——游戏：电影.....	28
2. 6 本章练习.....	33
2. 7 课外拓展.....	34
<b>第 3 章 Blockly 选择结构.....</b>	<b>36</b>
学习目标.....	36
知识图谱.....	36
3. 1 基本概念.....	37
3. 2 单分支选择结构.....	38
3. 3 双分支选择结构.....	39
3. 4 多分支选择结构.....	40
3. 5 选择结构的嵌套.....	42
3. 6 小试牛刀——游戏：鸟.....	45
3. 7 本章练习.....	49
3. 8 课外拓展.....	49
<b>第 4 章 Blockly 循环结构.....</b>	<b>51</b>
学习目标.....	51

知识图谱.....	51
4. 1 基本概念.....	52
4. 2 次数重复循环结构.....	53
4. 3 条件重复循环结构.....	54
4. 4 步长循环结构.....	57
4. 5 列表循环结构.....	58
4. 6 循环的中断与继续.....	60
4. 7 循环结构的嵌套.....	61
4.7.1 内循环和外循环.....	62
4.7.2 非独立的内循环.....	62
4. 8 小试牛刀——游戏 1：迷宫.....	63
4. 9 小试牛刀——游戏 2：乌龟.....	68
4. 10 本章练习.....	75
4. 11 课外拓展.....	75
第 5 章 Blockly 列表 .....	77
学习目标.....	77
知识图谱.....	77
5.1 数组 .....	77
5.1.1 数组的定义.....	77
5.1.2 数组的性质.....	78
5.2 列表的基本操作.....	78
5.2.1 列表.....	78
5.2.2 列表的创建.....	78
5.2.3 列表数据的插入.....	79
5.2.4 列表的查找和修改.....	79
5.2.5 列表数据的删除.....	81
5.3 列表的使用.....	81
5.3.1 列表的简单应用.....	81
5.3.2 列表循环.....	82
5.4 小试牛刀——制作一个自动售货机.....	83
5.5 本章练习.....	86
5.6 课外拓展.....	86
第 6 章 Blockly 函数.....	87
学习目标.....	87
知识图谱.....	87
6. 1 基本概念.....	87
6. 2 实参与形参.....	88
6. 3 函数的创建与使用.....	88
6.3.1 无参函数.....	89
6.3.2 有参函数.....	90
6. 4 函数的返回值.....	91
6. 5 小试牛刀——游戏：池塘导师.....	93
6. 6 本章练习.....	99
6. 7 课外拓展.....	99

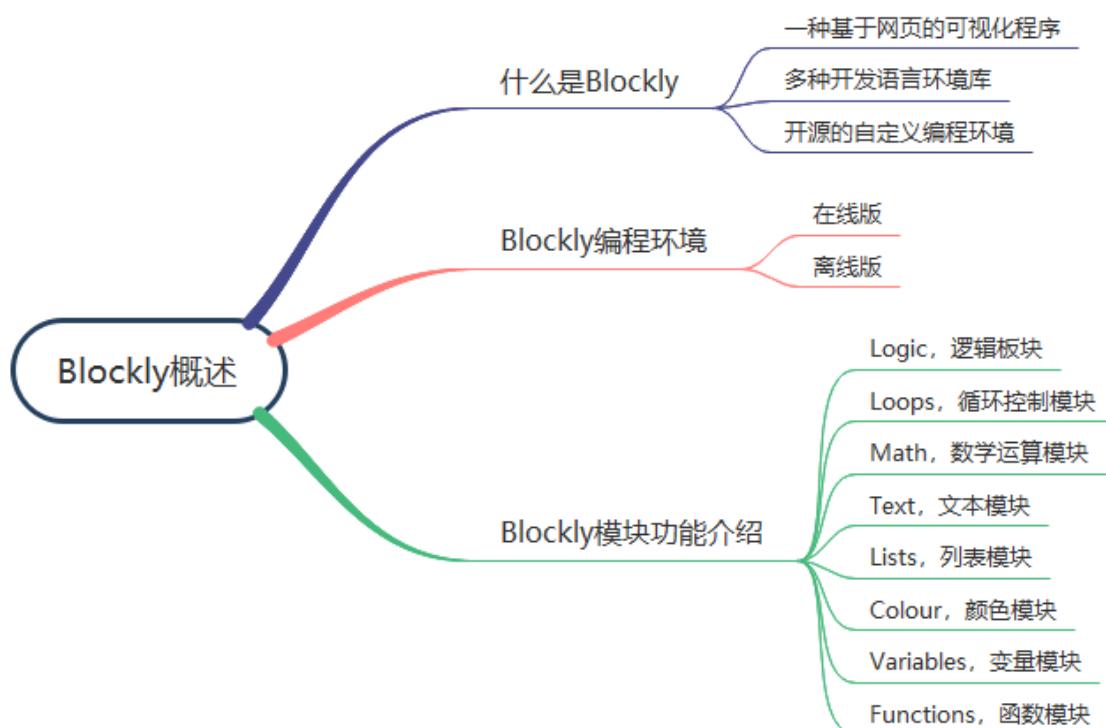
第7章 Blockly高级.....	102
学习目标.....	102
知识图谱.....	102
7.1 Blockly开发工具.....	102
7.1.1 定义一个块.....	104
7.1.2 管理库.....	112
7.1.3 导入和导出库.....	114
7.1.4 块导出器.....	114
7.1.5 工作区工厂.....	115
7.2 二次开发案例——Simple Blockly.....	119
7.2.1. 准备工作.....	119
7.2.2. 动手实践.....	120
7.3 Blocks 二次开发中的代码.....	123
7.3.1 Plane 中 raw 块的模拟.....	123
7.3.2 print-py 块的设计.....	126
7.3.3 Repeat-Do 块的复现 .....	128
7.4 二次开发案例-puzzle 游戏的制作.....	131
7.4.1 Simple Blockly .....	131
7.4.2 制作 puzzle 游戏 .....	137
7.5 Blockly 的高级使用.....	144
7.5.1 将 Blockly 作为代码生成器 .....	144
7.5.2 Blockly 的二次开发 .....	151
7.6 小试牛刀——游戏：池塘.....	153
7.7 本章练习.....	154
7.8 课外拓展.....	154

# 第1章 Blockly 概述

## 学习目标

1. 理解 Blockly 的概念、编程方式。
2. 掌握在线版和离线版 Blockly 的使用。
3. 掌握 Blockly 各个模块的功能。

## 知识图谱



在本章中，我们将学习到什么是 Blockly、了解 Blockly 的编程方式、在线版与离线版 Blockly 的使用方式以及各个模块的功能。在学习完本章知识内容后，我们将对 Blockly 有一个整体的了解。

### 1.1 什么是 Blockly

Blockly 是众多可视化编程工具之一。2012 年 6 月，Google 发布了完全可视化的编程语言 Google Blockly。Blockly 代码块由类似于积木的图形对象构成。使用者可以通过类似

玩拼图玩具的方式，将它们拼接起来，创造出简单功能，然后将一个个简单功能组合起来，最终构建出一个程序。在这个过程中，大家只需要拖动鼠标，不需要键盘敲击，相较于其他种类编程语言，Blockly 语言无须大家编写冗长的代码、考虑复杂的语法规则，趣味性更强，学习成本更低。

### (1) 一种基于网页的可视化程序

Google Blockly 是基于网页的可视化编程工具库。大家可以以离线或者在线的方式在 Windows、Linux、MC 和 Android 平台上的浏览器端进行编程操作。可以使用计算机端、手机或平板移动端进行随时随地的完成编程设计，教学编程方式多种多样。

### (2) 多种开发语言环境库

Blockly 基于图形化编程设计可以导出 JavaScript、Python、PHP、Lua、Dart 等多种语言。通过图形化编程完成程序设计，在 Blockly 中有一个类似语言转换器的工具箱，可以将图形化编程语言转化成多种编程语言代码。用图形化编程方式去理解多种程序语言。

### (3) 开源的自定义编程环境

Blockly 是开源的编程工具，用户可以根据自己编程的特点要求，对 Blockly 工具箱进行自定义设计。同时，Blockly 开发工具能让用户自定义块导出至工具箱，并在工作区工厂完成对代码的封装，如图 1-1 所示。

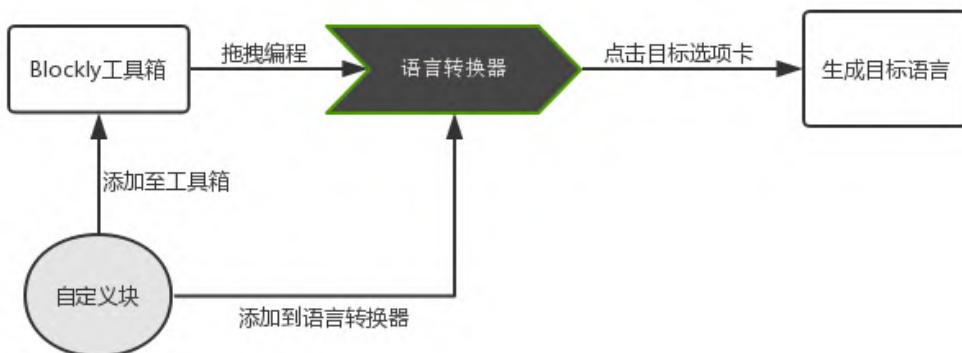


图 1-1 Blockly 使用流程图

## 1.2 Blockly 编程环境

Blockly 有在线版和离线版两个版本，在地址栏输入 <https://developers.google.com/blockly/>，访问 Blockly 官网，即可体验 Blockly 在线编程，如图 1-2 所示。



图 1-2 Blockly 在线版

离线版 Blockly 无须安装，只需 Clone 或解压后，进入 demos，打开 index.html，选择相应的选项，即可体验。Linux 系统，可下载 TAR Ball，在终端进行文件解压即可；Windows 系统，可下载 ZIP File，并继续解压即可。下载地址如下：

Github Blockly 地址：<https://github.com/google/blockly>

TAR Ball 地址：<https://github.com/google/blockly/tarball/master>

ZIP File 地址：<https://github.com/google/blockly/zipball/master>

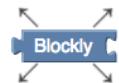
## [Blockly > Demos](#)

These demos are intended for developers who want to integrate Blockly with their own applications.



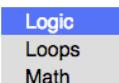
### [Fixed Blockly](#)

Inject Blockly into a page as a fixed element.



### [Resizable Blockly](#)

Inject Blockly into a page as a resizable element.



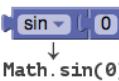
### [Defining the Toolbox](#)

Organize blocks into categories for the user.



### [Maximum Block Limit](#)

Limit the total number of blocks allowed (for academic exercises).



### [Generate JavaScript](#)

Turn blocks into code and execute it.

图 1-3 Blockly 离线版

## 1.3 Blockly 模块功能介绍

Blockly 语言总共分为 8 个模块，当我们学习了新的函数或者命令，就可以使用这些 Blockly 语句块进行练习，所有的模块被组织排放在左侧的列表中（如图 1-4 所示）。

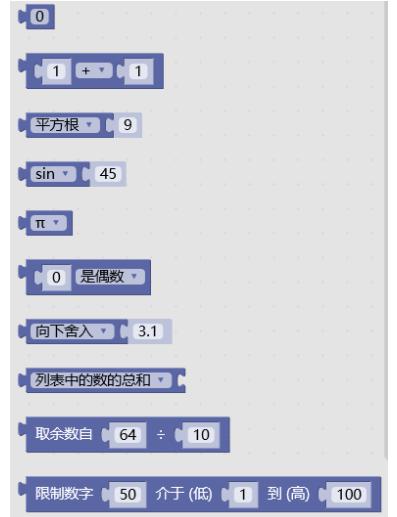


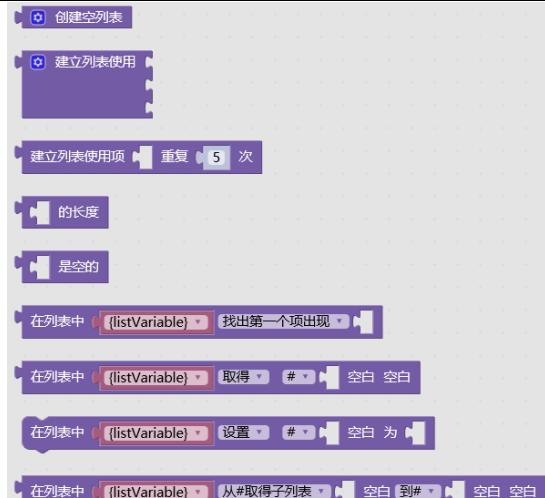
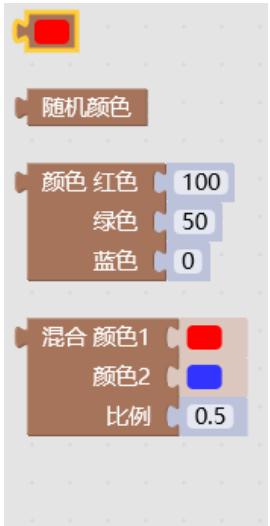
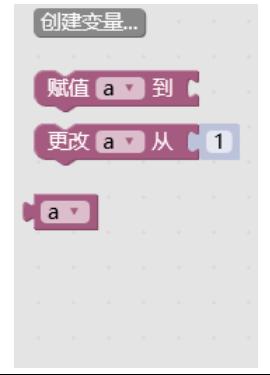
图 1-4 Blockly 模块列表

使用时根据正确的语法和适当的缺口对接就能实现预定的功能。因此，我们通过对模块进行适当的组织就能轻松的实现每一个新的想法和创意，如表 1-1 所示。

表 1-1 Blockly 模块功能

模块名称	模块内容	描述
Logic, 逻辑板块		表明数据间的逻辑关系

Loops, 循环控制模块		根据设定的条件，重复执行某项任务
Math, 数学运算模块		进行数据运算
Text, 文本模块		文本的输入输出、字符串的相关操作

Lists, 列表模块		创建列表、赋值、 列表的相关操作
Colour, 颜色模块		为元素设置颜色
Variables, 变量模块		创建变量、为变 量赋值
Functions, 函数模块		函数相关操作， 模块化编程

## 1.4 小试牛刀——游戏：拼图

学习完本章内容，大家对 Blockly 已经有了初步的认识。接下来我们通过一个游戏来熟练掌握这种类似积木拼接的编程模式，游戏地址如下：  
<http://cooc-china.github.io/pages/blockly-games/zh-hans/puzzle.html?lang=zh-hans>。

### 游戏规则：

- ① 每个动物都有自己独特的生理特征。拖动模块将动物与其特征进行拼接，并为每一个动物选择合适的腿数；
- ② 点击查看答案检查自己是否正确完成拼图，拼图匹配正确后，游戏结束，顺利通关。



图 1-5 游戏：拼图游戏

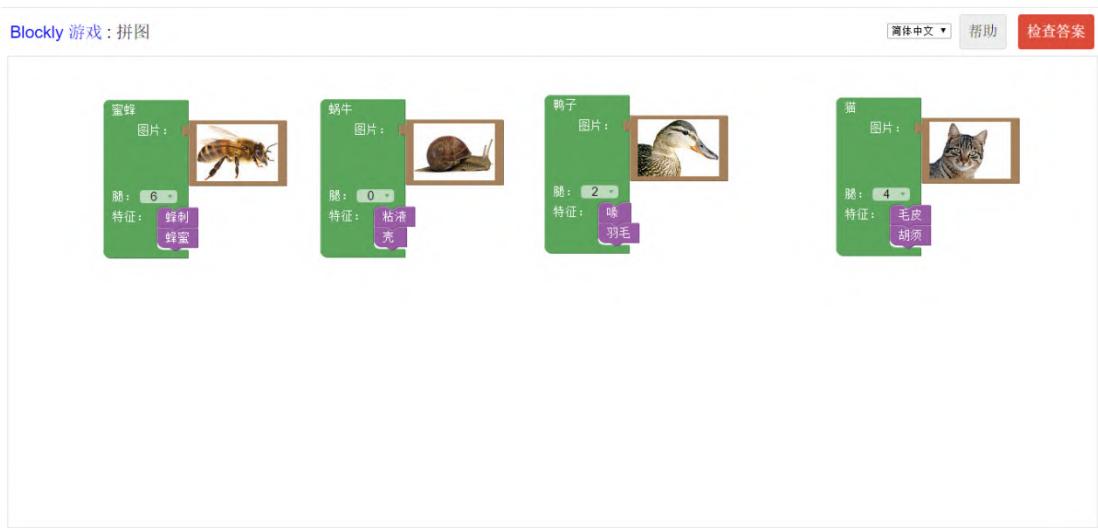


图 1-6 拼图游戏答案

## 1.5 本章练习

1. 进入 Blockly 官网，熟悉 Blockly，并使用在线 Blockly 输入“HelloBlockly”。
2. 在本地配置离线版 Blockly，并完成“Plane”游戏的练习。

## 1.6 课外拓展

### 计算机语言

计算机语言的种类非常的多，总的来说可以分成机器语言，汇编语言，高级语言三大类。计算机每做的一次动作，一个步骤，都是按照已经用计算机语言编好的程序来执行的，程序是计算机要执行的指令的集合，而程序全部都是用我们所掌握的语言来编写的。所以人们要控制计算机一定要通过计算机语言向计算机发出命令。通用的编程语言有两种形式：汇编语言和高级语言。

TIOBE 是开发语言排行榜，每月更新一次。其依据的指数是基于世界范围内的资深软件工程师和第三方供应商提供，其结果作为当前业内程序开发语言的流行使用程度的有效指标。

该指数可以用来检阅开发者的编程技能能否跟上趋势，或是否有必要做出战略改变，以及什么编程语言是应该及时掌握的。观察认为，该指数反应的虽并非当前最流行或应用最广的语言，但对世界范围内开发语言的走势仍具有重要参考意义。

Sep 2018	Sep 2017	Change	Programming Language	Ratings	Change
1	1		Java	17.436%	+4.75%
2	2		C	15.447%	+8.06%
3	5	▲	Python	7.653%	+4.67%
4	3	▼	C++	7.394%	+1.83%
5	8	▲	Visual Basic .NET	5.308%	+3.33%
6	4	▼	C#	3.295%	-1.48%
7	6	▼	PHP	2.775%	+0.57%
8	7	▼	JavaScript	2.131%	+0.11%
9	-	▲	SQL	2.062%	+2.06%
10	18	▲	Objective-C	1.509%	+0.00%
11	12	▲	Delphi/Object Pascal	1.292%	-0.49%
12	10	▼	Ruby	1.291%	-0.64%
13	16	▲	MATLAB	1.276%	-0.35%
14	15	▲	Assembly language	1.232%	-0.41%
15	13	▼	Swift	1.223%	-0.54%
16	17	▲	Go	1.081%	-0.49%
17	9	▼	Perl	1.073%	-0.88%
18	11	▼	R	1.016%	-0.80%
19	19		PL/SQL	0.850%	-0.63%
20	14	▼	Visual Basic	0.682%	-1.07%

图 1-6 2018 年编程语言热度排行

(来源: 百度百科

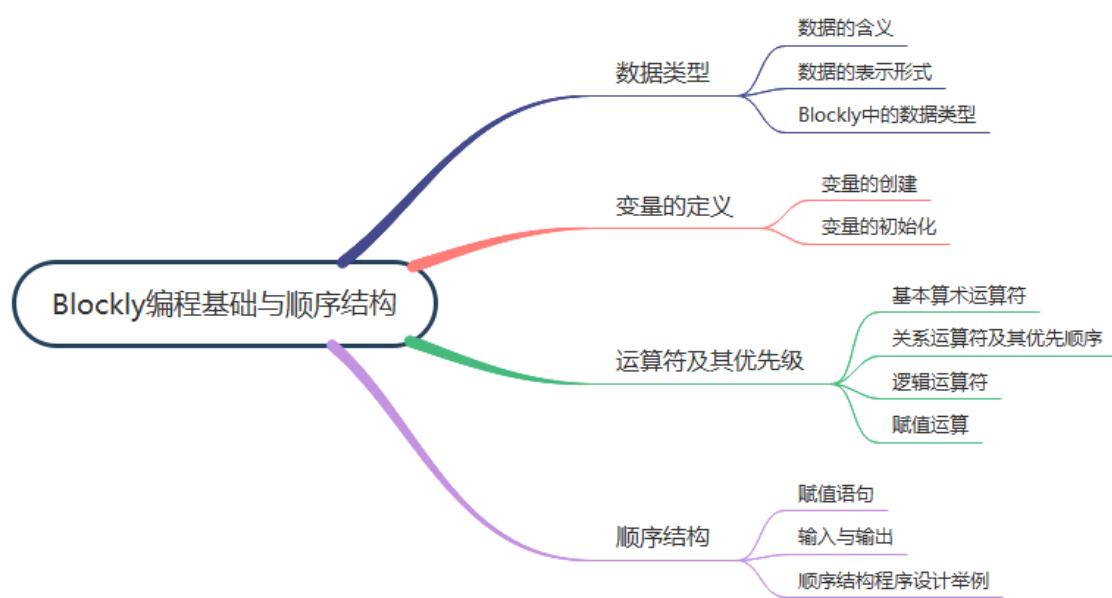
<https://baike.baidu.com/item/计算机语言/4456504?fr=aladdin>  
<https://baike.baidu.com/item/Tiobe/2830870?fr=aladdin>)

## 第2章 Blockly 编程基础与顺序结构

### 学习目标

1. 了解数据的含义、表示形式。
2. 了解Blockly中的数据类型。
3. 了解变量的定义，掌握变量的创建和初始化。
4. 理解运算符及其优先级。
5. 掌握顺序结构。

### 知识图谱



在本章中，我们将学习Blockly编程基础知识，包括数据的类型、变量的创建以及常用的运算符，如算术运算符、关系运算符、逻辑运算符和赋值运算符。此外，我们还将了解什么是顺序结构，并学习到几种顺序执行的语句，如赋值语句、输入与输出语句。顺序结构是最常用的程序结构，对我们今后学习其他种类编程语言至关重要。

## 2.1 数据类型

### 2.1.1 数据的含义

在计算机程序的世界里，程序的基本任务就是处理数据，无论是数值、文字、图像、图形还是声音、视频等信息，如果要在计算机中处理的话，就必须将它们转换成所谓的数字信息，因为计算机中只能存储数字，甚至连计算机程序都是由数字组成的，所以在使用计算机程序解决问题的时候，第一步就是把需要处理的信息数字化，即使用数字表示需要处理的信息。如果我们要处理图像信息，可以把一副图像看作是由m行n列的点组成的，每一个点是一种颜色，每一种颜色可以使用三个数据（R、G、B）来表示，R表示红色的比例、G表示绿色的比例、B表示蓝色的比例，这样就可以用 $m \times n \times 3$ 个数据表示一副图像了。如果我们需要处理文字信息，例如英文，那么我们需要用数字来表示英文中出现的每个字母或标点符号，正如在 ASCII 编码标准中，用65表示字母“A”，用66表示字母“B”等，只要把使用到的每个符号都进行编码（数字化），我们就可以在计算机中处理文字信息了。总之，不论什么数据，如果要使用计算机程序来处理的话，就必须进行数字化，即将它们转换成数字表示的形式，才能够在计算机中处理。于是就出现了这样一些基本问题，在Blockly中可以使用哪些种类的数据？每一种类型的数据的书写形式是怎样的？本章的其余部分将详细讨论这些问题。

### 2.1.2 数据的表示形式

在计算机系统中，我们常见的数据表示形式有：二进制、八进制、十进制和十六进制。进制也就是进位计数制，是人为定义的一种计数方法，对于进制，有两个基本的概念：基数和运算规则。基数也称为底数，表示组成一种进制的基本数字的个数，例如二进制的基数为2，采用0和1两个数字，八进制的基数为8，采用0—7八个数字；运算规则规定了如何进位，例如二进制的运算规则为“逢二进一，借一当二”，十进制的运算规则为“逢十进一，借一当十”。

#### 1.二进制表示

众所周知，在计算机中，采用二进制代码表示字母、数字字符以及各种各样的符号、汉字等。在处理信息的过程中，可将若干位的二进制代码组合起来表示各种各样的信息。但由

于二进制数不直观，人们在计算机上实际操作时，输入、输出的数值数据都是十进制，而具体转换成二进制编码的工作则由计算机软件系统自动完成。字母和各种字符在计算机中的传输普遍采用 ASCII 码，即美国标准信息交换码，它用了7位二进制数来表达字母和各种常用字符。对于汉字信息的表示比较复杂，我国有汉字几万个，常用的汉字也有7000多个，为了统一，我国制定了汉字编码标准，规定了一、二级汉字共6763个，用两个字节来表示一个汉字。

## 2.十进制表示

十进制形式是我们最熟悉的表达形式，十进制数的书写规则是由正负号开头，后跟一个自然数的形式，如果是正数，正号可以省略。例如：-213、0、415、76、+83都是合法的整数。在Blockly中，如不特殊定义，所有数一般默认为十进制。

## 3.八进制形式

一些编程语言中，常常以数字0作为开头来表明该数字是八进制。那么用八进制表示一个整数的书写规则是：以数字0开头，后跟一个八进制形式的数，如果是负数，则以负号开头。例如：0123、-087、00、+0327等都是合法的八进制形式。

## 4.十六进制形式

十六进制的书写规则是以“0x”开头，后跟一个十六进制数，例如：0xFF03、0x123、0xAC7等都是合法的十六进制形式，而x37、287都是非法的十六进制形式。

### 2.1.3 Blockly 中的数据类型

程序中所有的数据都有特定类型，数据的表示方式、取值范围以及对数据可以使用的操作都由数据所属的类型决定。类型可以帮助编译程序生成高效的目标代码，编译程序在生成目标代码时，可按需分配存储空间和如何引用这个数据。一个数据属于某个特定的类型后，在数据上允许操作的运算也确定了。例如，整数可以做四则运算等；字符串则可以进行比较、连接、判断子串等，但不能做四则运算。下面就让我们一起来了解一下Blockly中的数据类

型。

## 1. 数字

在Blockly中，提供了数字输入模块 ，其默认一定的存储长度，默认数值为0。在一些计算公式中也提供了的数字输入模块，如图2-1所示。



图2-1 数字输入模块

各数据输入模块只区分数字、字符类型，也就是说在允许输入数字的模块中，可以输入任何数字，但不允许输入字符。在程序具体执行过程中，程序会对输入数字类型的合法性进行检查，如图2-2、2-3所示。



图2-2 非法输入字母



图2-3 非法输入标点

## 2. 字符

Blockly中的字符输入模块为 。字符在内存中存储的是该字符的ASCII编码值，由于字符数据存储的就是一个字符的编码数值，所以字符数据也可以当做一个整数。在Blockly中的基本表示形式是用“”引用起来，比如“A”、“Q”、“a”、“b”、“#”、“-”、“.”等。使用双引号将一个数字放在引号里面，其意义也表示该数字，和不用引号表示的意义相同，如“65”和65的意义相同。但是同一个字母的大写和小写对应的ASCII

编码值不同，因此为不同的字符。如“A”、“a”为不同字母。

同样的，在Blockly的字符输入模块中，允许输入任何形式的字符和数字，只要不超出特定的长度都是合法的。只有在程序执行的时候才检查输入是否正确。

### 3.字符串

在Blockly中字符串的表示和单个字符的表示形式是一样的，输入模块也是“”。

图2-4是Blockly提供的字符串输入模块。



图2-4 字符串输入模块

### 4.数据使用

前面讲过，在Blockly的数据和字符定义过程中，程序给了一定的存储空间，Blockly不计较输入数据或字符的类型和长度，程序员不需要考虑非法输入带来的麻烦，这给了程序员极大的方便。如图2-5、2-6、2-7所示：



图2-5 输入整数型数据



图2-6 输入浮点型数据



图2-7 输入负数

但是，在程序运行过程中，如果数据的长度超过程序可表示的范围或数据输入错误，那么运算结果将会出现差错。

下图进行了乘法运算，输入数据为11111111111111200和2，正确的输出结果应为22222222222222400，但实际的输出结果为22222222222222300。这是由于输入数据过长，16位后的数据将不再进行计算，并且输入数据超过16位后，运算结果将出现差错，如图2-8所示。

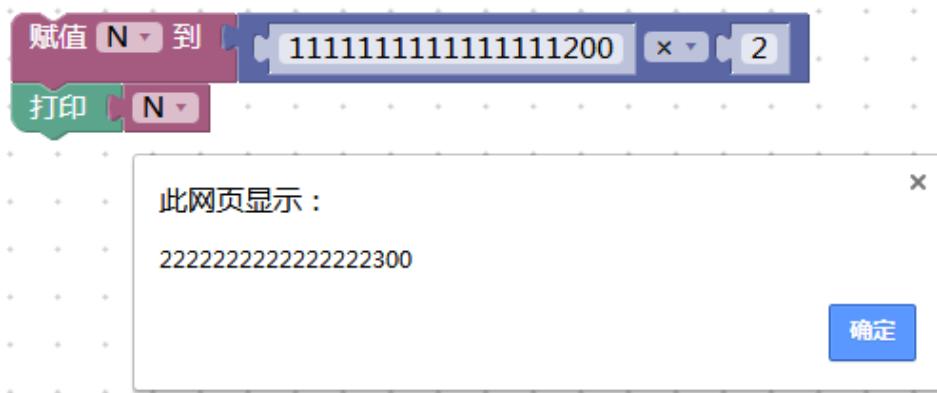


图2-8 输入过长数据

下图使用了循环语句，在该循环语句的重复次数输入模块，默认输入的数据为正整数。如果输入负数，程序默认为0；如果输入小数，默认在其整数部分上加1。如图2-9、2-10、2-11所示。

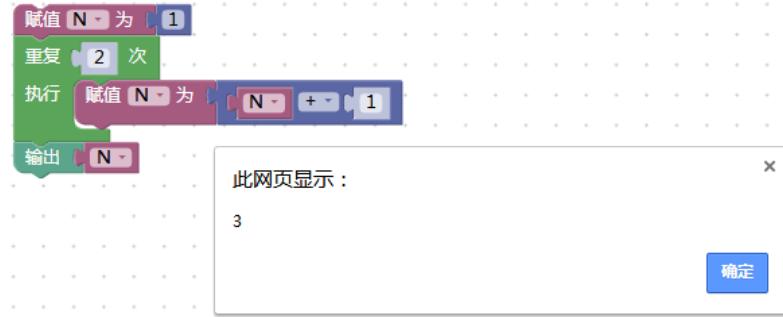


图2-9 输入数据全部为正整数



图2-10 输入数据包含负数

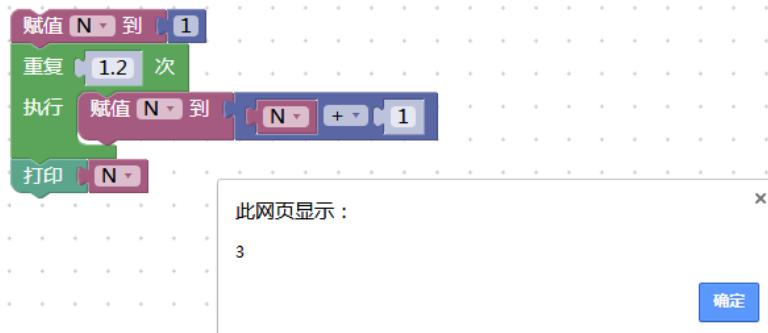


图2-11 输入数据包含小数

## 2.2 变量的定义

上一节讲的数据和字符，当给定一个值后，在程序中是确定的不能改变的量，我们称为常量，而与之对应的就是变量，顾名思义，就是在程序中可以根据需要改变的量。

### 2.2.1 变量的创建

初次打开Blockly，我们可以在“变量”模块中进行变量的创建，如图2-12所示：



图2-12 创建变量

点击创建变量后，会弹出定义变量名称的对话框，如图2-13所示，变量的命名方式比较随便，不受限于数字或字符，但是我们为了使用方便，尽量选用简单明了的字符，避免与程序中的其他名称重复。并且Blockly提供的变量定义不区分类型，只是在内存中分配一定的存储空间。



图2-13 定义变量名称

## 2.2.2 变量的初始化

初始化在计算机编程中的含义是指第一次为新创建的变量赋值的做法，如何初始化则取决于所用的程序语言以及所要初始化的对象的存储类型等属性。在汇编语言中，变量的初值即初始化后的变量的值，会占用一定空间，因此不必要的初始化会造成磁盘空间的浪费，但初始化变量在一定程度上可以降低漏洞出现的可能性。因此，是否对变量进行初始化操作需

要依情况而定。Blockly提供的变量初始化模块为“”，可以在规定的长度内输入任意数字、汉字、字母或符号。

## 2.3 运算符及其优先级

运算符是指用来表示在数据上执行某些特定操作的符号，而参与运算的数据称为操作数。根据参与运算的操作数的个数是一个、两个或三个，运算符分为一元运算符、二元运算符和三元运算符。使用运算符把常量、变量和函数等运算成分连接起来，组合成的有意义的式子被称作表达式。单个常量、变量和函数也都可以看成是一个表达式。表达式经过计算后都会得到一个确定的值，这个值就是表达式的值。每个表达式都具有唯一确定的值和唯一确定的类型。

Blockly中含盖了日常使用的所有运算符，我们主要认识一下常用的几类运算符：算术运算符、关系运算符、逻辑运算符和赋值运算符。

### 1. 基本算术运算符

基本算术运算有6 种运算符，如表2-1所示：

表2-1 基本算数运算符

运算符	描述
+	加法 - 相加运算符两侧的值
-	减法 - 左操作数减去右操作数
*	乘法 - 相乘操作符两侧的值
/	除法 - 左操作数除以右操作数
%	取余 - 左操作数除以右操作数的余数
^	幂运算 - 底数（底数的个数等于指数）相乘

算术运算表达式的值为其运算结果。如 $3+2$ 、 $5-6$ 、 $4*8$ ，值分别为5、-1、32。基本算术运算符的表达式格式为： $<\text{操作数}>\text{运算符}<\text{操作数}>$ 。Blockly中给出的模块如图2-14所示。



图2-14 算数运算表达式

## 2.关系运算符及其优先顺序

关系运算是逻辑运算中比较简单的一种，所谓的关系运算实际上就是比较运算，将两个值进行比较，从而判断比较的结果是否满足符合的条件，比如关系表达式 $a>5$ ，如果 $a$ 为6，那么表达式成立，结果就是真，反之，如果 $a$ 的值为-1，那么表达式不成立，结果就是假。

关系运算符有6 种，如表2-2所示：

表2-2 关系运算符

运算符	描述
=	等于 - 检查如果两个操作数的值是否相等，如果相等则条件为真
$\neq$	不等于 - 检查如果两个操作数的值是否相等，如果值不相等则条件为真
<	小于 - 检查左操作数的值是否小于右操作数的值，如果是那么条件为真
$\leq$	小于等于 - 检查左操作数的值是否小于或等于右操作数的值，如果是那么条件为真
>	大于 - 检查左操作数的值是否大于右操作数的值，如果是那么条件为真
$\geq$	大于等于 - 检查左操作数的值是否大于或等于右操作数的值，如果是那么条件为真

在这六种关系运算符中， $>$ 、 $<$ 、 $\geq$ 、 $\leq$ 的优先级相同， $=$ 、 $\neq$ 的优先级相同，且 $>$ 、 $<$ 、 $\geq$ 、 $\leq$ 的优先级顺序高于 $=$ 、 $\neq$ 的优先级。

关系运算表达式的值只有两个：1和0（真或假）。如 $4<2$ 、 $2>1$ 、 $1=2$ ，值分别为0、1、0。

关系运算符的表达式格式为：`<操作数>运算符<操作数>`。Blockly中给出的模块如图2-15所示。

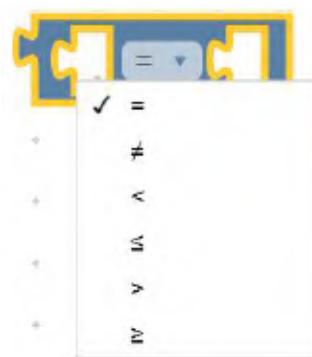


图2-15 关系运算表达式

### 3.逻辑运算

逻辑运算符有3种，如表2-3所示：

表2-3 逻辑运算符

运算符	描述
&&	逻辑与运算符 - 当且仅当两个操作数都为真，条件才为真
	逻辑或操作符 - 如果任何两个操作数任何一个为真，条件为真
!	逻辑非运算符 - 用来反转操作数的逻辑状态，如果条件为true，则逻辑非运算符将得到false

逻辑运算符包括与、或、非，在 Blockly 中分别表示为和、或、非。非在三种运算符中优先级最高，“和”和“或”优先级相等，且低于非逻辑。

逻辑运算表达式的值只有两个：1和0（真或假）。如 $(5<10) \mid\mid (5>20)$ 、 $!(3>2)$ ，其值为1、0。逻辑运算符的表达式格式为： $\langle\text{操作数}\rangle\text{运算符}\langle\text{操作数}\rangle$ 和 $\text{运算符}\langle\text{操作数}\rangle$ 两种形式。Blockly中给出的模块为：

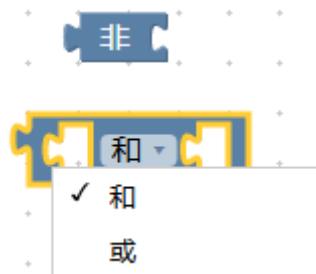


图2-16 逻辑运算表达式

#### 小提示

你如果看完了上文中关于逻辑运算符的介绍，还是不理解其中的含义的话，没关系，可以对应下表，来理解Blockly中的逻辑运算符。

a	b			
真	真	真	真	假
真	假	假	真	假
假	真	假	真	真
假	假	假	假	真



## 4. 赋值运算

赋值运算的值即为所赋的值。如 $a=3$ 、 $b=6$ ， $a$ 和 $b$ 值分别为3、6。Blockly中赋值运算与变量初始化的表达式相同。创建变量后使用变量赋值模块可以对变量进行赋值。例如创建了变量 $k$ ，使用下图的变量赋值模块，把10这个值赋给 $k$ 。



图2-17 赋值运算表达式

### 小提示

如果你以前接触过编程，看到这里你也许会疑惑，为什么判断两个操作数是否相等的表达式与赋值运算的表达式一样？计算机又是如何区分两者呢？

细心的同学可能已经发现了，Blockly可以把我们搭建好的模块转换成代码，而在代码当中，判断两个操作数是否相等与赋值是有区别的。“=”表示赋值语句，比如 $a = 5$ ，是把5 赋值给变量 $a$ ；而“==”是逻辑判断，比如 $a == 5$ ，是表示变量 $a$ 的值是否和5 相等，如果相等就返回真，反之返回假。Blockly中把逻辑判断“==”写成了“=”，是为了方便大家去理解，但实际上我们需要知道“=”和“==”是不一样的。



Blockly中与其他编程语言不同，不需要太多的考虑运算符的优先级问题，因为Blockly将不同的运算符集成在不同的模块中，在使用中以模块嵌套的形式出现，因此其运算顺序只能是由里到外。

## 2.4 顺序结构

顺序结构是最简单的程序结构，也是最常用的程序结构，只要按照解决问题的顺序写出相应的语句就可以了，它的执行顺序是自上而下，依次执行，顺序结构的流程图如图2-18所示。在本章的学习中，我们将学习到几种顺序执行的语句，在这些语句的执行过程中不会发生流程控制的转移，比如赋值语句，输入输出语句。

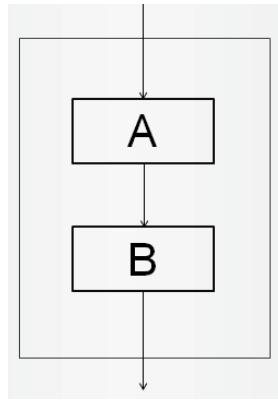


图2-18 顺序结构流程图

### 2.4.1 赋值语句

在上一节，同学们学习了赋值运算，是不感觉赋值特别简单呢？

在Blockly中，赋值语句是由一个语句块构成： 。其中i指的是一个变量，也可以用其他字母代替，在“为”后面紧跟着的是要赋给i的值。同样的，这个赋值表达式也可以包括在其他表达式中，例如：if后面跟着的是一个条件，其作用是当i大于零时，将一个值赋给i，如图2-19所示。



图2-19 嵌套赋值表达式

### 2.4.2 输入与输出

当计算机被用于和外界交互时才是最有趣的，所谓的输入与输出是以计算机主机为主体而言的。输入就是将数据从输入设备(如键盘，磁盘，光盘，扫描仪等)带入计算机，输出就是将数据从计算机发送到外部输出设备(如显示屏，打印机，磁盘等)，输入输出有时候并称为I/O。目前I/O的种类有很多，包括人机界面，网络接口，存储设备接口和自动机器接口。那么在Blockly中如何进行输入与输出呢？

#### 1. Blockly 的输出模块

Blockly中的输出模块为 。输出模块后面可以拼接各种各样的模

块从而输出不同的数据，输出模块后拼接运算表达式，如图2-20所示，输出模块后拼接了 $1+1$ ，那么运行后输出的答案就为 $1+1$ 的结果2。



图2-20 输出模块后拼接运算表达式

输出模块后拼接字符串，如图2-21所示，输出模块后拼接了字符串“Hello World!”，那么最终将会在屏幕上输出这一段文本。

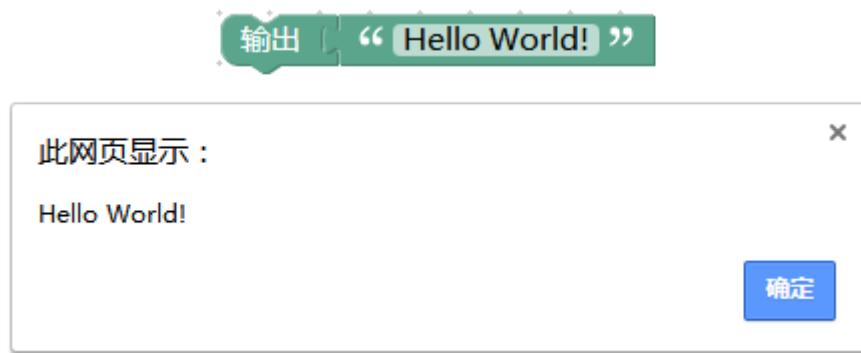


图2-21 输出模块后拼接字符串

输出模块后也可拼接多个模块的组合，之前我们输出了一段文本，在这一段文本的前面加上另一个模块将会得到另外一种效果，如图2-22所示，拼接了计算字符串长度的模块，最终输出的结果为这段文本的长度。



图2-22 输出模块后拼接多个模块的组合

### 小提示

使用不同的浏览器打开 Blockly 的离线版，部分模块的表述可能会有出入，例如使用 360 浏览器和火狐浏览器打开 Blockly 的离线版，输出模块为上面我们介绍的样子，使用 IE 浏览器打开，则为 “abc”。



## 2. Blockly 的输入模块

在Blockly中，输入模块如图2-23所示，在输入模块中输入的既可以是文本，也可以是数字，通过模块后面的下拉按钮可以进行不同的选择。



图2-23 输入模块

当我们运行这个模块的时候，屏幕会弹出一个文本框，这时我们就能在文本框里面输入我们想输进计算机的数据，点击确认后，我们所输入的数据就将会进入我们所设置的变量里面，图2-24便是我们输入数据的界面：



图2-24 输入数据界面

通过上面简单的介绍，同学们可能对输入的理解还不够深刻，下面让我们来举一个具体的例子。

首先，让我们设置一个变量a，然后再将上面的输入语句块连接在设置变量语句块的后面，点击运行，在出现的为文本框里面输入我们想要输入的数据，点击确认以后，数据就会被赋值给a了，如果同学们想确认a的值是不是真的是我们所输入的数据，可以在这段搭建好的模块下面加上输出数据块，将a的数据输出到屏幕上，这样我们就能确认a的值了。



图2-25 利用输入模块为变量赋值

### 2.4.3 顺序结构程序设计举例

现在，同学们对顺序结构、赋值语句以及输入输出已经有了一定的了解，接下来就让我们一起来学习两个顺序结构的例子来巩固一下学到的基础知识吧。

例1 从键盘输入一个大写字母，要求改用小写字母输出。

看到这个题目，同学们首先想到的是什么呢？在第一章介绍的几个 Blockly 模板中大家会首先想到哪个语句块？或许记忆力好的同学已经想到，在我们介绍的文本模块中就有一个

语句块是用来转换大小写的：‘转为大写’ ‘“abc”’。

这个语句块使用起来相当简单，只需要将你需要转换的文本连接在此语句块的后面就行了，此语句块同样能根据需求不同产生三种不同的效果，我们可以根据需要选择。

既然已经找到了这个问题所需要的核心语句，那么后面的问题就简单了，不难看出这个题目同时用到了输入和输出，所以我们只需要设置一个变量用来存放我们所输入的数据，然后将输入的数据转化成小写并输出，这个问题就解决了。所组成的模块及运行结果如图2-26所示：

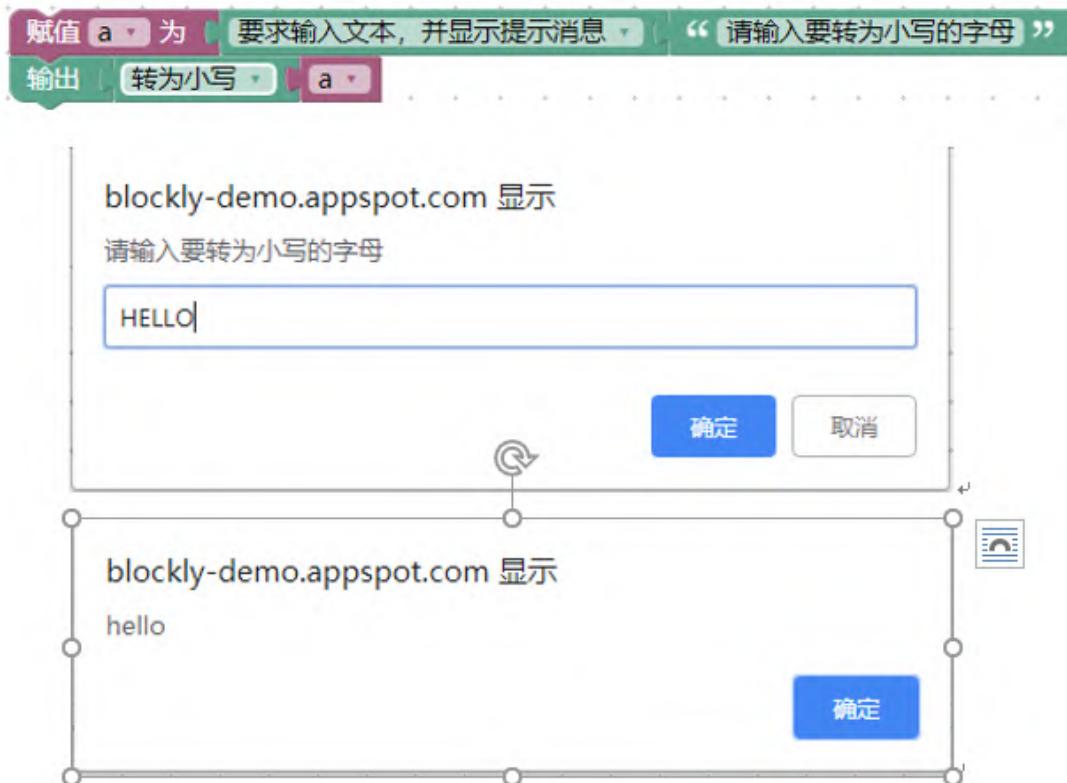
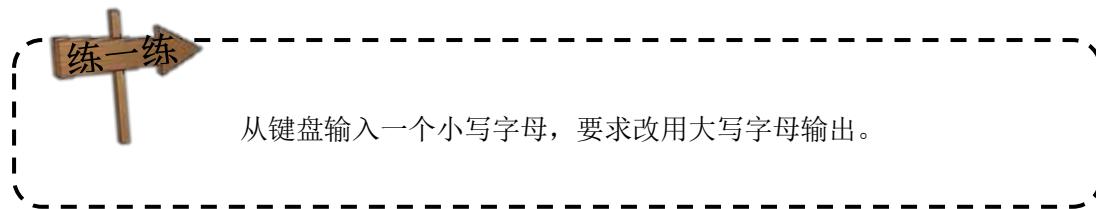


图2-26 将输入数据转为小写并输出



例2 输入一个两位数，如果这两位相乘大于两位相加，则输出“大”这个字。

在同学们第一次见到这个题目时，可能会感到有点手足无措，但其实我们一步步分析就会发现这个题目并不难，在解决这个问题前，我们首先要搞清楚怎么得到这个两位数的个位数和十位数，如果大家曾经接触过其他语言，就会知道两位数除以10得到的商就是十位上的数字，而得到的余数就是个位上的数字，弄清楚这个问题后，这个题目是不是就简单了许多呢。具体数据块如图2-27所示：

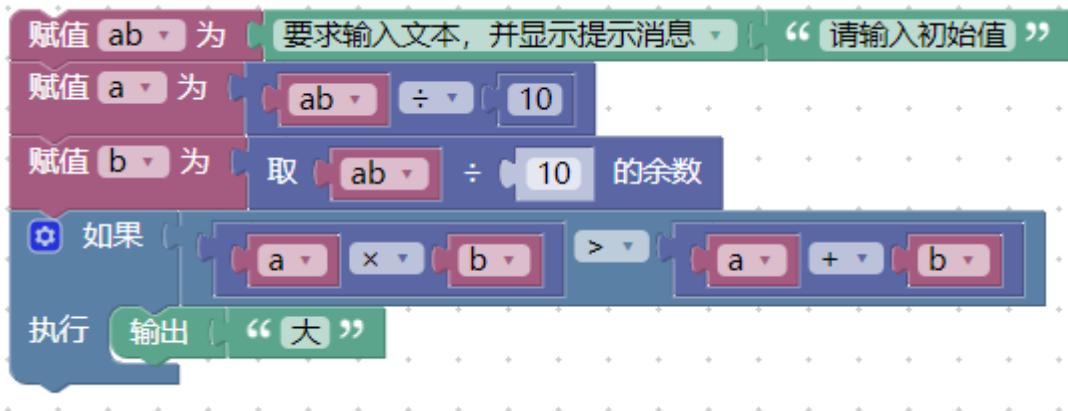
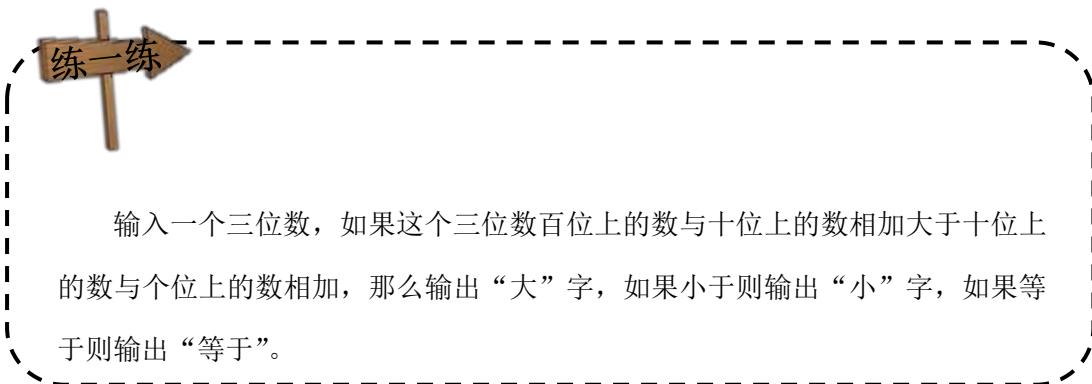


图2-27 例2模块拼接

在这个组好的模块里，我们首先将输入的两位数存到ab这个变量里面，然后将计算得出的个位数和十位数分别赋值给b和a，再利用我们前面提到的逻辑模块里面的if语句块判断大小，最后输出。运行过程与结果如图2-28所示：



图2-28 例2运行结果



通过本章的讲解，相信大家对Blockly语言的顺序程序设计有了大概的了解，也对输入

输出有了清晰的认识，语言的顺序程序设计在同学们今后的语言学习中起着相当重要的作用，希望能引起大家的重视。

## 2.5 小试牛刀——游戏：电影

通过这章的学习，同学们对Blockly的基础知识以及顺序结构有了一定的了解，接下来，就让我们通过一个游戏，进一步加深同学们对本章知识的掌握。游戏地址如下：

<http://cooc-china.github.io/pages/blockly-games/zh-hans/movie.html?lang=zh-hans>。

### 游戏规则：

- ① 游戏开始前，需要观看示例。游戏任务是编写代码，达到与示例相同的视觉效果；
- ② 点击Run Program按钮后，执行玩家搭建好的代码块，当达到与示例相同的视觉效果后，游戏结束，顺利通关。

### 通关详解：

**第一关：**调节参数，画出与示例形状、大小相同的图形。



图2-29 第一关示例与答案

**第二关：**半径为10的圆，圆心的初始位置在X=0，Y=50的位置，使其移动到X=100，Y=50的位置。



图2-30 第二关示例与答案

**第三关：**尝试与第二关相反的移动路径，使圆从X=100, Y=50的位置移动至X=0, Y=50。

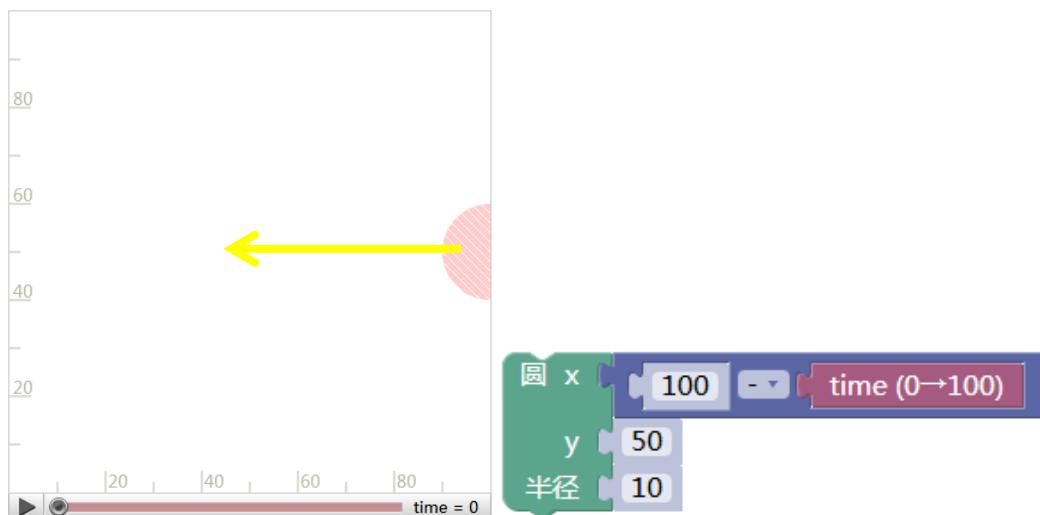


图2-31 第三关示例与答案

**第四关：**按照示例在上下左右画出4个圆，分别按照各自箭头所指的方向移动，最终位置为对应方向圆的起始位置，例如，右边圆的起始位置是左边圆的最终位置。

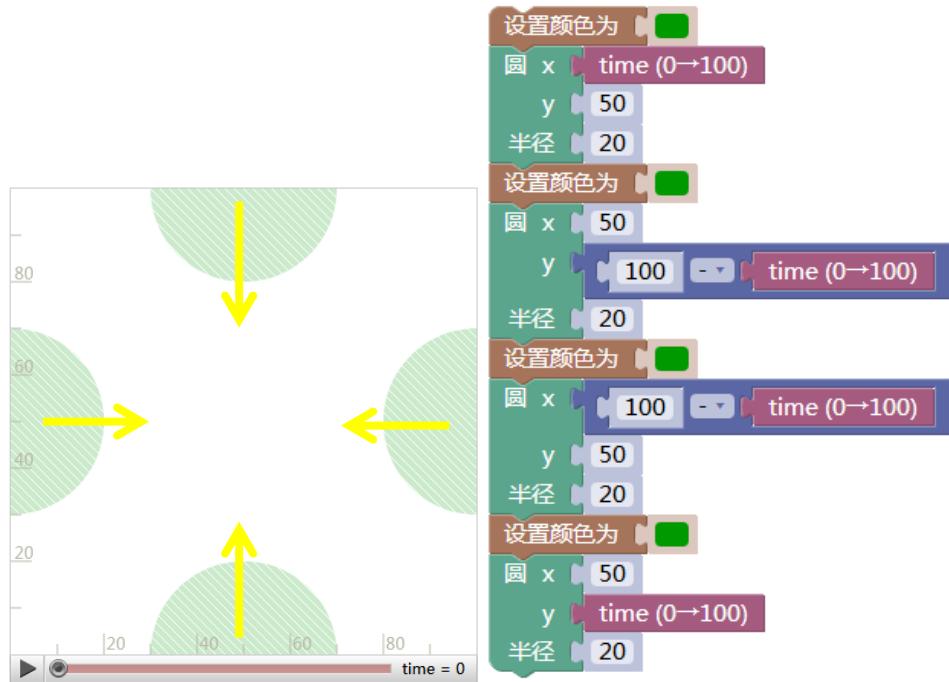


图2-32 第四关示例与答案

**第五关：**按照示例用三个圆拼出米老鼠的头型，按照箭头所指方向，向上移动。

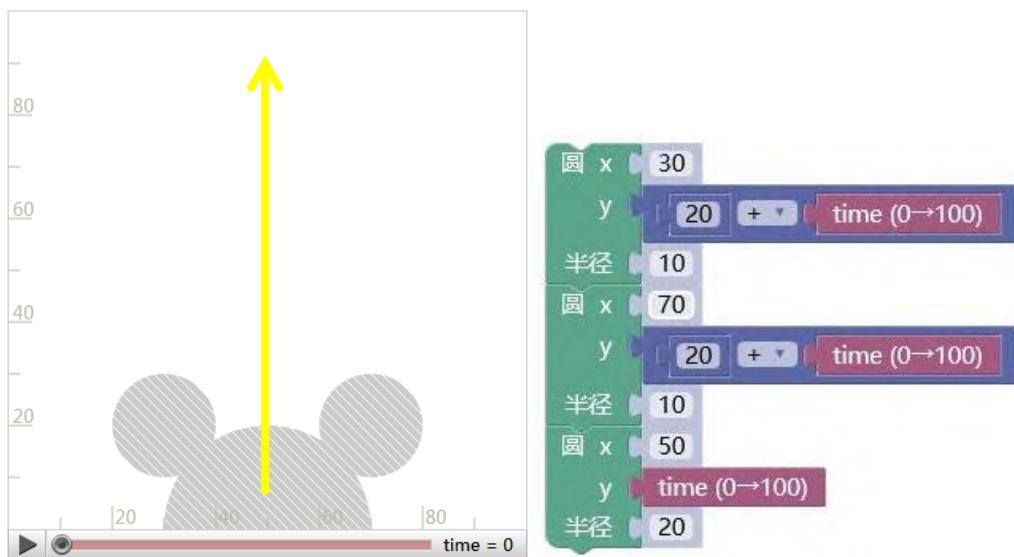


图2-33 第五关示例与答案

**第六关：**起始时刻一条线与X轴重叠，另一条线在界面的对称轴上，仔细观看两条线的运动轨迹，拼接代码块，达到与示例相同的效果。

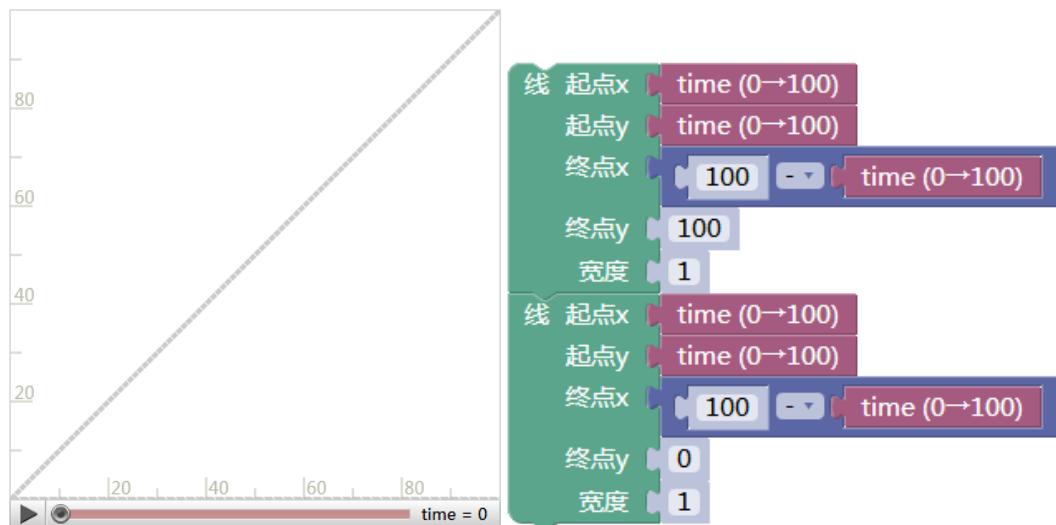


图2-34 第六关示例与答案

**第七关：**观看示例，使圆按照示例中的抛物线轨迹移动。同学们可以先计算出抛物线方程，这样更容易拼接代码块。

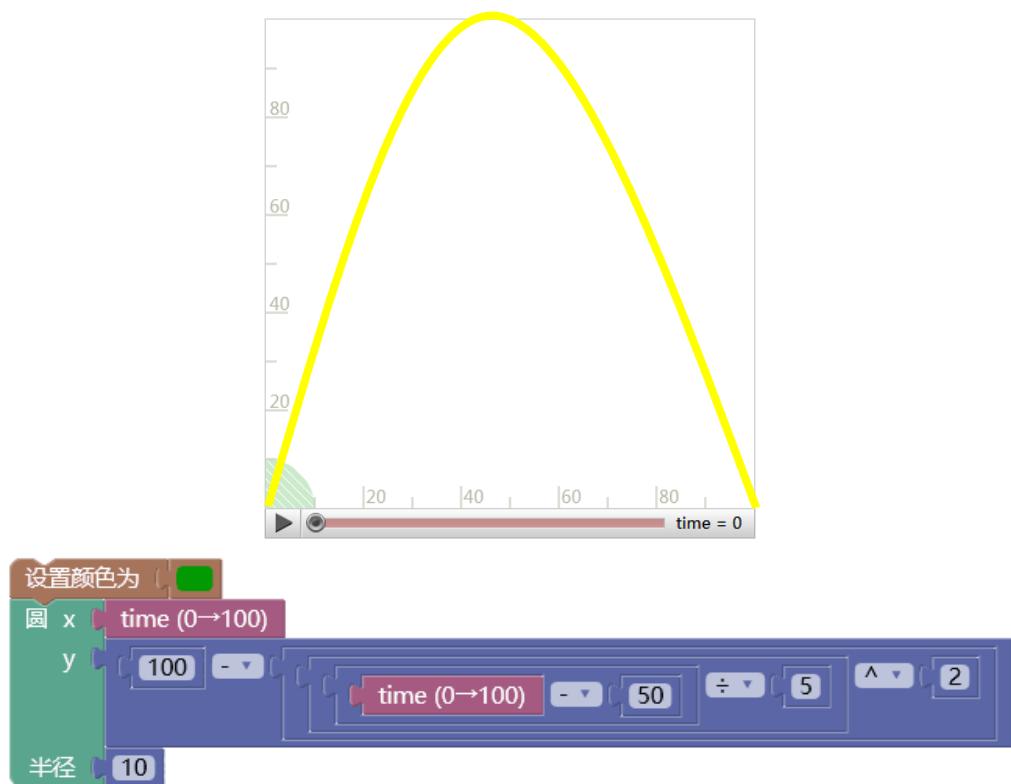


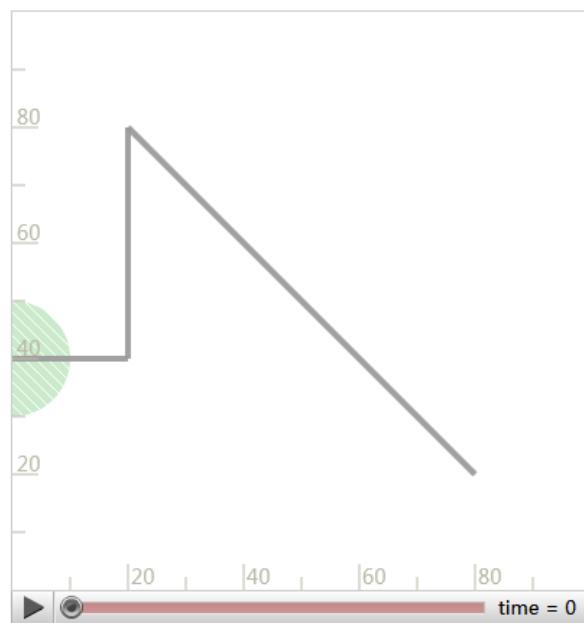
图2-35 第七关示例与答案

**第八关：**使右上角蓝色的圆和左下角红色的圆按照箭头所指方向移动，完全重合后变成绿色的圆。



图2-36 第八关示例与答案

**第九关：**观看示例，使圆按照图中给出的轨迹移动。



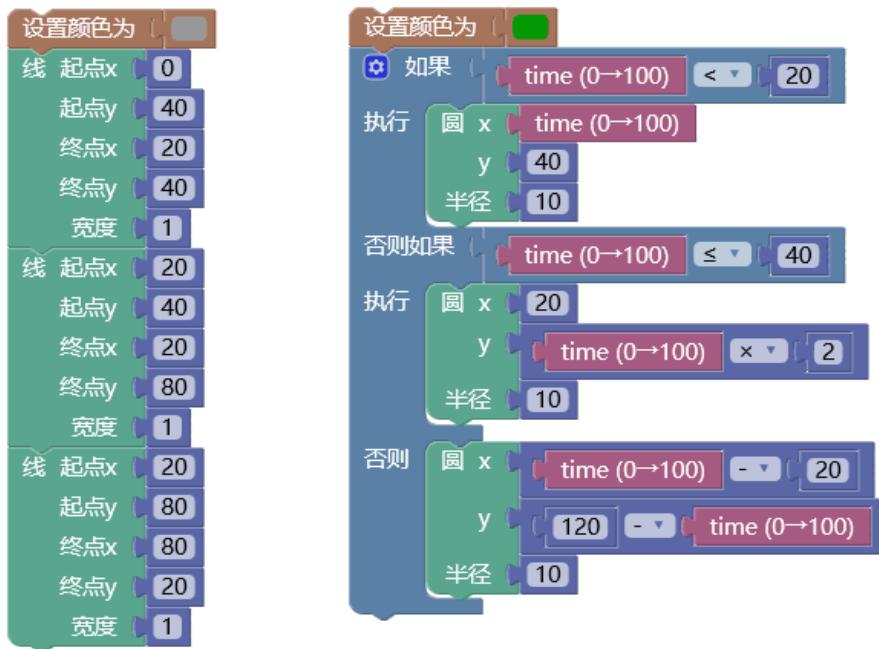


图2-37 第九关示例与答案

**第十关：**自由发挥，利用模块画出你想要的图形。

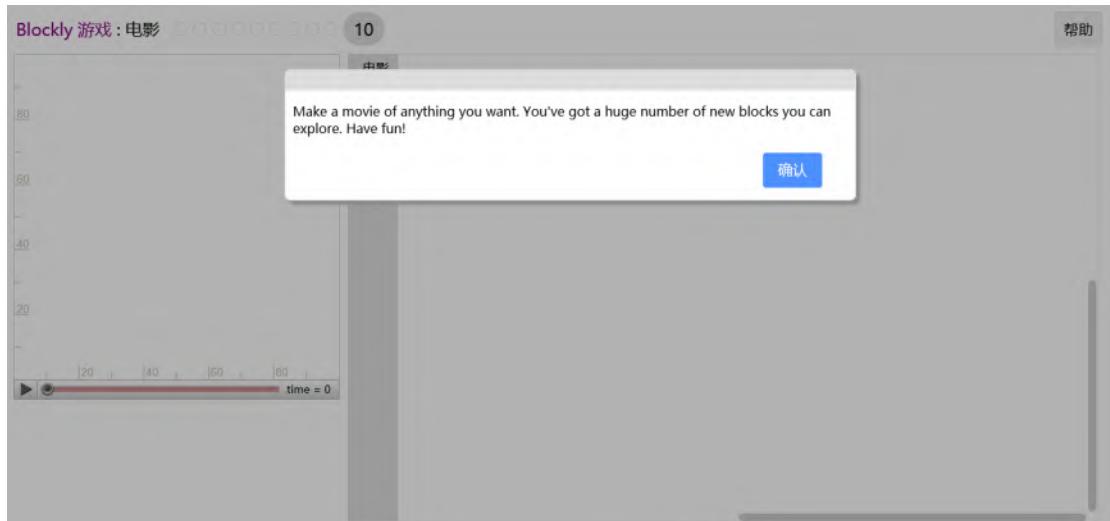


图2-38 电影游戏第十关

## 2.6 本章练习

- 对于计算机而言，无论是数字、字母、符号，在计算机中都是以0、1的形式存储和计算，但是他们在Blockly中有不同的运算规则，为什么？
- 分别求出 $a=3$ ,  $b=a+3$ ,  $b>a$ 三个表达式的值和变量a或b的值，认真思考表达式的值和变量的值有什么区别？
- 对两个整数变量的值进行互换。
- 如果是做单项选择题，请根据给定的选项，输出对应的结果。例如总共有4个字符，A、B、

C、D。你给出字符A，输出：你选择了A；你给出字符B，输出：你选择了B；你给出字符C，输出：你选择了C；你给出字符D，输出：你选择了D。

5. 根据输入的值，判断是星期几。例如，输入：1，输出：星期1。

## 2.7 课外拓展

### 二进制的由来与应用

在德国著名的郭塔王宫图书馆保存着一份弥足珍贵的手稿，其标题为：“1与0，一切数字的神奇渊源。”这是二进制的提出者戈特弗里德·威廉·莱布尼茨（Gottfried Wilhelm Leibniz，1646 – 1716）的手迹。

莱布尼茨被誉为十七世纪的亚里士多德，是历史上少见的通才，他本人是一名律师，经常往返于各大城镇，他许多的公式都是在颠簸的马车上完成的。他在数学史和哲学史上都占有重要地位。在数学上，他和牛顿先后独立发现了微积分；在哲学上，莱布尼茨的乐观主义广为流传，他和笛卡尔、巴鲁赫·斯宾诺莎被认为是十七世纪三位最伟大的理性主义哲学家。除此之外，莱布尼茨在政治学、法学、伦理学、神学、哲学、历史学、语言学诸多方向都留下了著作。

1679年戈特弗里德·威廉·莱布尼茨发明了一种计算法，用两位数代替原来的十位数，即1和0。法国汉学大师若阿基姆·布韦（Joachim Bouvet，汉名白晋，1662–1732年）曾经向莱布尼茨介绍了《周易》和八卦的系统。八卦是表示事物自身变化的阴阳系统，用“一”代表阳，用“—”代表阴，用这两种符号，按照大自然的阴阳变化平行组合，组成八种不同形式。有人说莱布尼茨发现二进制是受到了中国文化的影响，也有学者认为莱布尼兹先发现了二进制，后来才看到传教士带回的八卦系统，并发现八卦可以用他的二进制来解释，认为“阴”与“阳”基本上就是他的二进制的中国版。

现在，我们使用的计算机都是采用二进制代码来表示数字、图片、文本、视频等数据，但为什么一定要使用二进制代码来表示数据呢？原因很简单：

1. 计算机是由逻辑电路组成，逻辑电路通常只有接通与断开两个状态，这两种状态正好可以用“1”和“0”表示。
2. 二进制运算规则简单，有利于简化计算机内部结构，提高运算速度。
3. 逻辑代数是逻辑运算的理论依据，二进制只有两个数码，正好与逻辑代数中的“真”和“假”相吻合。

4. 二进制与十进制等其他进制数之间易于互相转换。
5. 用二进制表示数据具有可靠性高，抗干扰能力强的优点。

(来源：百度百科

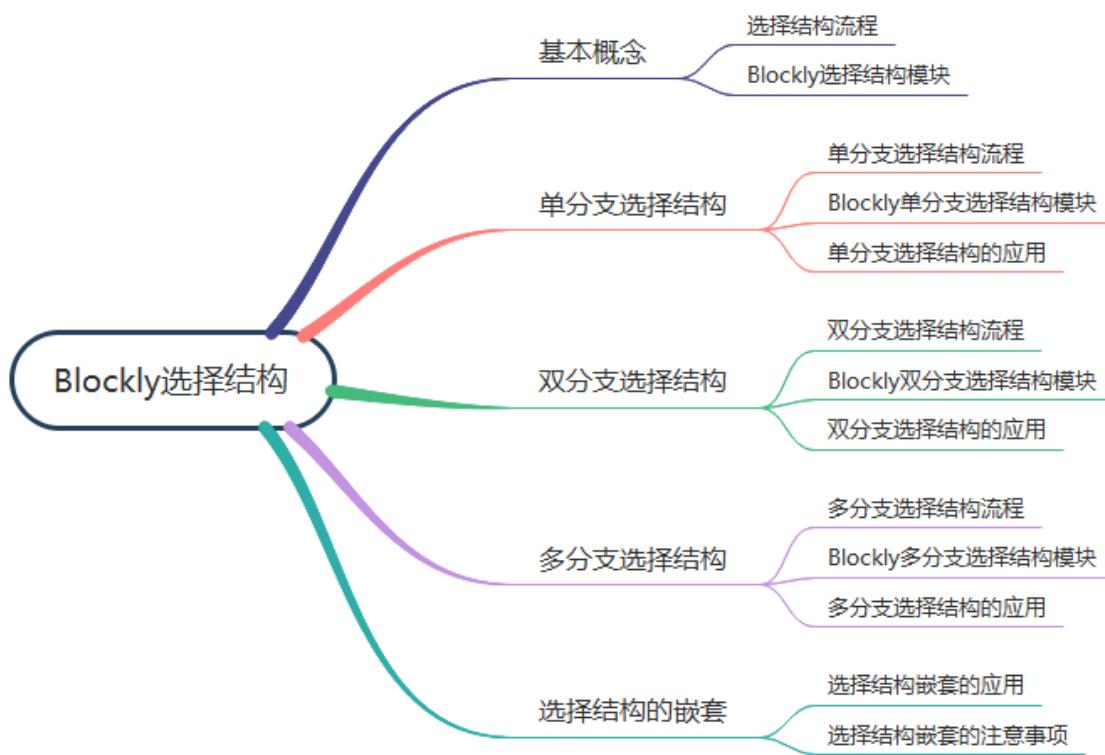
<https://baike.baidu.com/item/%E6%88%88%E7%89%B9%E5%BC%97%E9%87%8C%E5%BE%B7%C2%B7%E5%A8%81%E5%BB%89%C2%B7%E8%8E%B1%E5%B8%83%E5%B0%BC%E8%8C%A8/5028927?fromtitle=%E8%8E%B1%E5%B8%83%E5%B0%BC%E8%8C%A8&fromid=417549>)

# 第3章 Blockly选择结构

## 学习目标

- 1.理解选择结构的概念。
- 2.理解、运用单分支选择结构。
- 3.理解、运用双分支选择结构。
- 4.理解、运用多分支选择结构。
- 5.理解选择结构的嵌套。

## 知识图谱



在前面的学习中我们认识了 Blockly 中的各种基本模块的功能，并尝试了在顺序结构中进行。顺序结构在程序流程图中的体现就是用流程线将程序框自上而下地连接起来，按顺序执行算法步骤。但是我们会发现当遇到判断变量是否满足某一条件才可以执行下面的步骤时，很难只用顺序结构表达出来，这时候我们就需要运用选择结构来帮助我们对选择条件进行判

断。本章首先将带领大家认识选择结构，然后根据选择结构的构成分别介绍了单分支选择结构、双分支选择结构以及多分支选择结构，最后介绍了选择结构的嵌套。在学习完本章知识内容后，通过 Blocky Game 中 Bird 游戏对选择结构的使用进行练习。

### 3.1 基本概念

判断一个数是否为正数，并输出文字结果。流程图如图 3-1 所示，通过判断输入的数值是否大于 0，输出结果。当  $x > 0$  时，输出“正数”；否则输出结果为“非正数”。这种通过判断是否满足选择条件，来决定下一个步骤的过程就是选择结构。在 Blockly 中运用 if 和 else 语句完成选择条件，表达 if 语句的模块如图 3-2 所示。如果满足条件，则执行某一步骤。在 if 的左侧有一个设置按钮，点开后可以添加 else if 和 else 语句到右侧，从而进行多重判断。

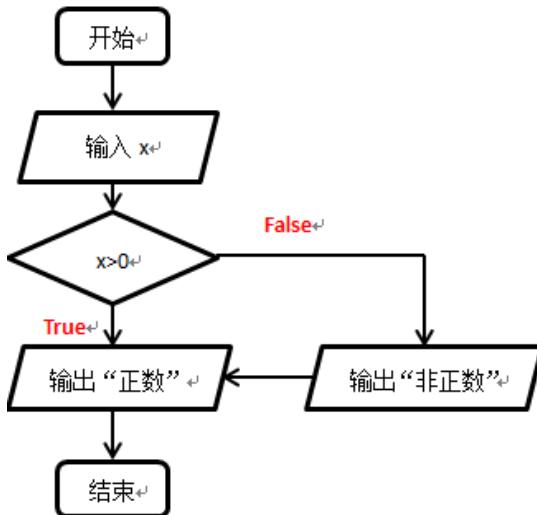


图 3-1 选择结构流程图



图 3-2 Blockly 中 if 语句模块

选择结构用于判断给定的条件，根据判断的结果判断某些条件，根据判断的结果来控制程序的流程。

### 3.2 单分支选择结构

单分支选择结构是最简单的选择结构，用 if 语句表示。当满足条件则执行某一步骤，其流程图如图 3-3 所示。而在 Blockly 中则使用图 3-4 中的模块来执行。在 if 后面输入判断语句，当逻辑判断结果为真时，则执行放在“执行”部分的语句块。

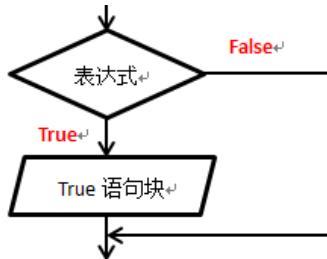


图 3-3 单分支选择结构流程图



图 3-4 Blockly 单分支选择结构模块

例 1 运用 Blockly 判断加法运算结果是否正确，若正确则提示“正确”，不正确无反馈。  
这一过程我们只需要运用单分支选择结构就可以完成，即只需要运用 if 语句，在 Blockly 中，运用“如果——执行”模块即可。假设我们判断  $1+1$  的运算结果，当我们在“=”后输入的是 2 时，可以弹出显示“正确”的提示框，当把“=”后的值改为 0 后，运行程序，则无反应，如图 3-5 所示。

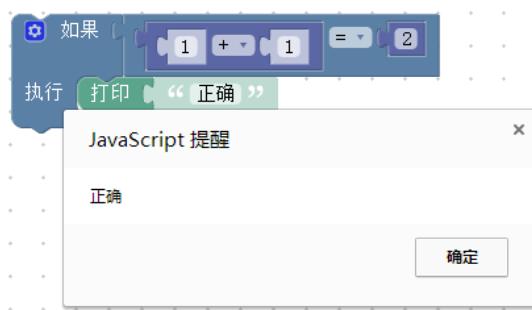


图 3-5 判断加法运算结果是否正确

## 练一练

输入一个变量，尝试运用“如果——执行”模块来判断这个变量是否为偶数，若为偶数则输出“偶数”。

### 3.3 双分支选择结构

双分支结构一般用 if else 语句表示，当满足某一条件时，执行 A 步骤；否则执行 B 步骤。其流程图如图 3-6 所示。而在 Blockly 中需要在“如果——执行”模块的基础上添加否则部分，最终显示为图 3-7。和单分支选择结构一样，在如果后面输入判断语句，当逻辑判断结果为真时，执行放在“执行”部分后面的语句块；当逻辑判断结果为假时，执行放在“否则”部分后面的语句块。

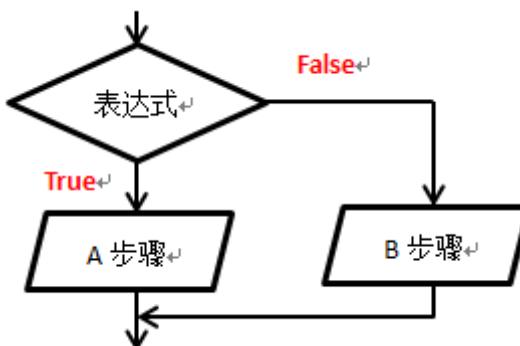


图 3-6 双分支选择结构流程图



图 3-7 Blockly 双分支选择结构模块

例 2 运用 Blockly 判断加法运算结果是否正确，若正确则提示“正确”，不正确则提示“错误”，使用双分支选择结构。

我们只需要在单分支选择结构的例子基础上，为逻辑判断错误时，即“否则”模块，后面加上输出“错误”就可以了。同样以判断  $1+1$  的运算结果为例，当我们输入的是 2 时，可

以弹出显示“正确”的提示框，当把“=”后的值改为 0 后，则弹出显示“错误”的提示框，如图 3-8 所示。

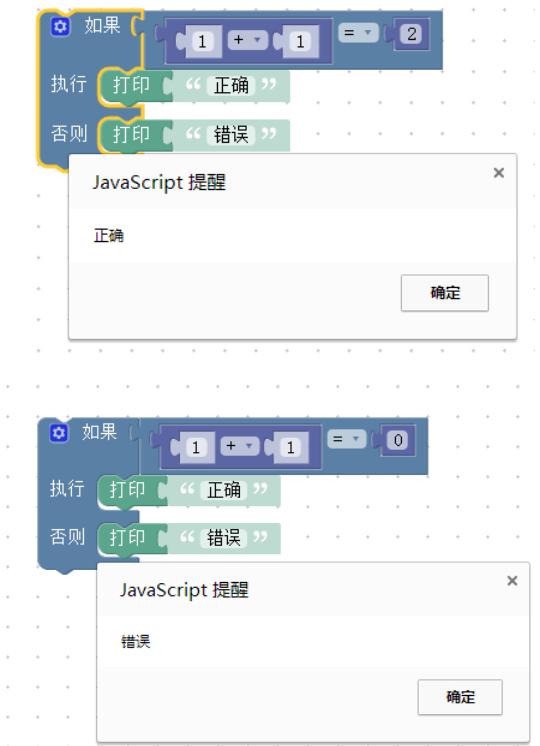


图 3-8 例 1 双分支选择结构版



输入一个变量，判断该变量是否可以被 3 整除，并输出结果“可被 3 整除”或“不可被 3 整除”。

### 3.4 多分支选择结构

多分支选择结构由 if 和 else if 语句构成，其中可以有多个 else if 结构，其流程图如图 3-9 所示。在 Blockly 中使用多分支选择结构时，需要在单分支选择结构中添加否则如果模块，否则如果模块可以不限数量。在否则如果后面可以添加否则模块，也可以不添加否则模块，最终形式如图 3-10 所示，这些都是多分支选择结构。

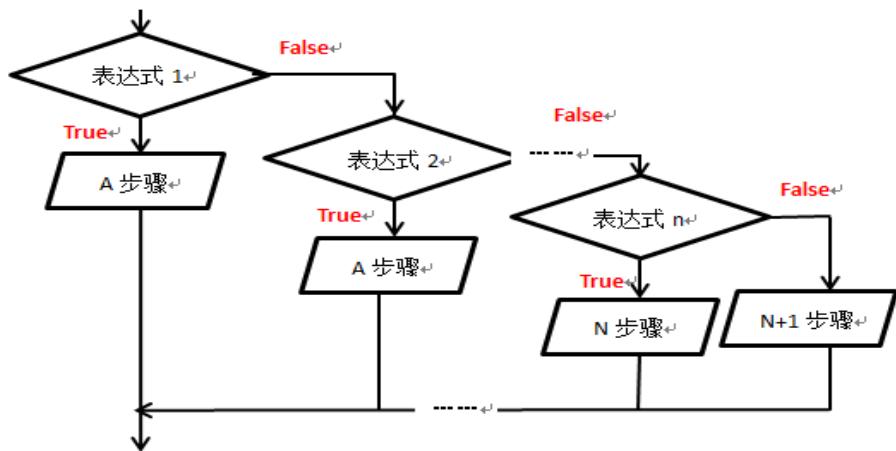


图 3-9 多分支选择结构流程图



图 3-10 Blockly 多分支选择结构模块

### 小提示

多分支选择结构中可以有多个选择判断部分，但是除第一个以外的选择判断部分都是在上一个判断的否分支上。

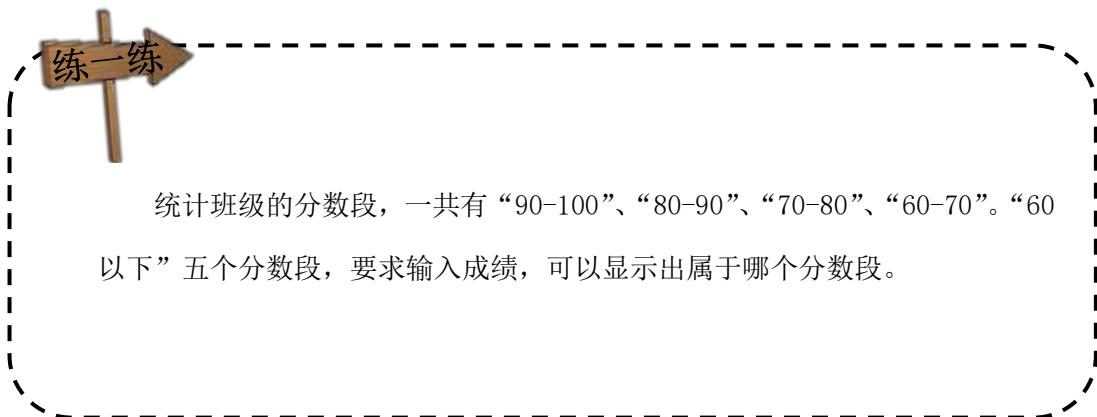


例 3 班里面需要做一个成绩统计，成绩在 90 分以上就输出 A，在 80–90 分之间输出 B，80 分以下输出 C。

这里需要将分数做三个分段，一个分段是大于 90 分的，输出“A”；剩余的里面大于 80 分的，输出“B”；否则剩下的都是“C”。根据判断条件，最终在 Blockly 中，制作出语句模块如图 3-11 所示。



图 3-11 成绩统计



### 3.5 选择结构的嵌套

选择结构的嵌套实际上就是在选择结构里面再放置一个或多个选择结构，实现了选择结构的嵌套。在 Blockly 中实现选择结构的嵌套，需要将多个“如果——执行”模块套用。可以将嵌套的“如果——执行”模块放置在执行的后面，如下图最左边的形式；也可以放置在否则的后面，比如下图中间的形式。但是将一个新的“如果——执行”模块放置在否则后面时，这种形式等同于在“如果——执行”模块的基础上添加否则如果部分。比如图 3-12 中搭建的第二个模块意义等同于第三个模块。也就是当选择结构嵌套部分放在否则内容里的时候，可以简化为多分支结构。

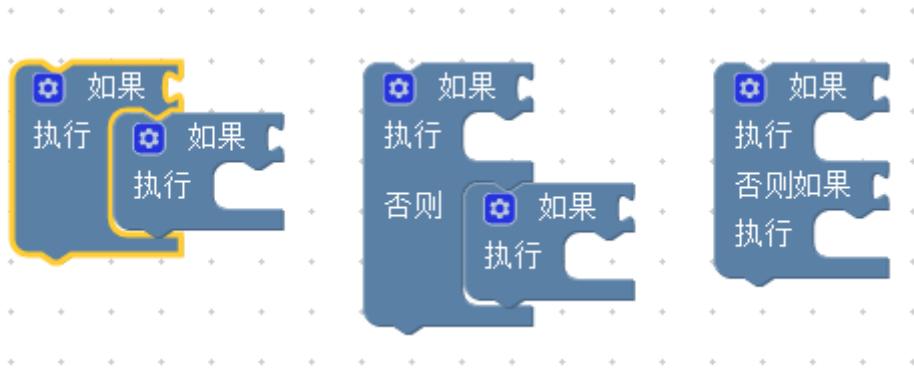


图 3-12 选择结构的嵌套

### 小提示

选择结构的嵌套没有固定的模式，规定嵌套部分需放置在哪一部分，只要选择结构内还有一个或多个选择结构，就是实现了选择结构的嵌套。



例 4 我们知道在平年中 2 月份是 28 天，闰年中 2 月份是 29 天，那么怎样利用 Blockly 编写一个判断年份是否为闰年的程序？

判断一个年份是否为闰年，首先它需要可以被 4 整除，如果不可以被 4 整除，这个年份肯定不是闰年。在可以被 4 整除后，我们还需要进一步判断这个年份的后两位是否为 0，即是否可以被 100 整除。如果不可以被 100 整除，这个年份则一定为闰年；如果可以被 100 整除，我们还需要判断这一年份是否可以被 400 整除。如果可以被 400 整除，那么这个年份是闰年；如果不可以被 400 整除，则是平年。整个判断过程如图 3-13 中所示，在 Blockly 中实现这一判断过程，需要用到三个选择结构进行嵌套完成。最终选择结构的嵌套部分在 Blockly 中实现形式如图 3-14 所示。

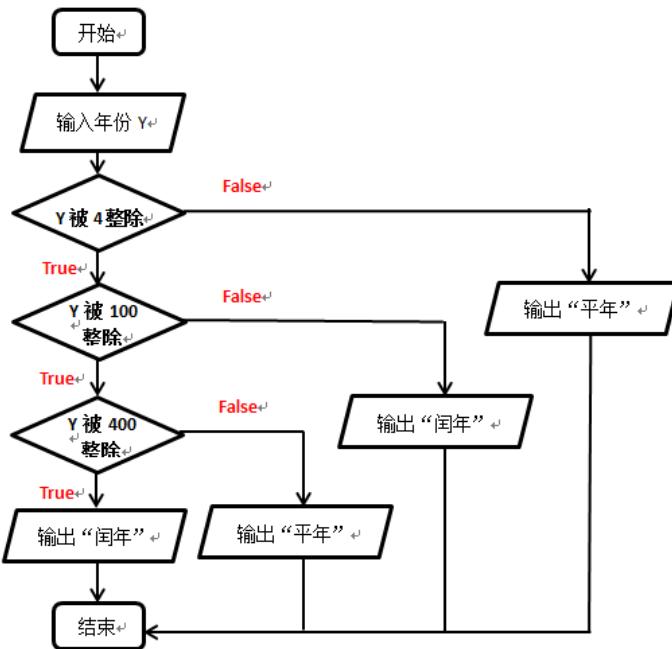


图 3-13 判断闰年程序流程图



图 3-14 判断闰年程序选择结构嵌套部分

**练一练**

创建三个变量，在 Blockly 中使用选择结构的嵌套形式，比较三个变量的大小，并将这三个数由大到小的排列。

### 3.6 小试牛刀——游戏：鸟

学习完如何在 Blockly 中使用选择结构后，我们通过一个游戏来熟练掌握这种程序结构，游 戏 的 地 址 如 下：

<http://cooc-china.github.io/pages/blockly-games/zh-hans/bird.html?lang=zh-hans>。

#### 游戏规则：

- ① 我们需要通过控制代码来让鸟叼完虫子后回到鸟窝，并保证不会撞到墙；
- ② 代码主要由选择结构和逻辑判断组成；
- ③ 点击 Run Program 按钮后程序就会执行右侧的代码。回到鸟窝后，游戏结束，顺利通关。

#### 通关详解：

**第一关：**让鸟沿着  $45^\circ$  方向向前飞行，叼完虫子回到鸟巢。



图 3-15 第一关示例与答案

**第二关：**让鸟在没叼到虫子前沿  $0^\circ$  向前飞行，叼到虫子后沿  $90^\circ$  飞行，回到鸟巢。

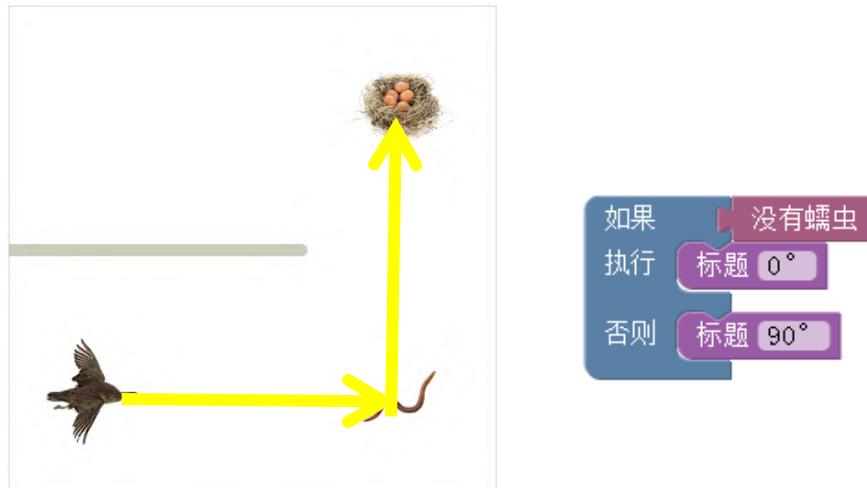


图 3-16 第二关示例与答案

**第三关：**让鸟在没有叼到虫子之前沿右下方飞行，叼到虫子之后再沿右上方飞行。



图 3-17 第三关示例与答案

**第四关：**让鸟在飞行横坐标小于 80 时，向右飞行；横坐标不小于 80 时，向下飞行到达鸟巢。



图 3-18 第四关示例与答案

**第五关：**让鸟在飞行纵坐标大 20 时，向下飞行；纵坐标不大于 20 时，向左飞行到达鸟巢。



图 3-19 第五关示例与答案

**第六关：**当鸟没有叼到虫子前沿右斜下方飞行，叼到虫子后判断鸟的纵坐标是否小于 80，当小于 80 时，向上方飞行；当不小于 80 时，向左方飞行。



图 3-20 第六关示例与答案

**第七关：**先让鸟沿左下方飞行过下面的阻拦物至纵坐标 40 处，然后令其在横坐标不超过 80 时，沿右斜下方飞行，吃到虫子；随后在纵坐标不超过 20 时，沿左方飞行回到鸟巢。



图 3-21 第七关示例与答案

**第八关：**当同时满足鸟没有吃到虫子和飞行横坐标小于 50 时，沿右上方飞行；然后设定没有吃到虫子和飞行横坐标大于 49 时，沿右下方飞行，令鸟吃到虫子。吃到虫子后判断鸟的纵坐标是否小于 50，当小于 50 时，令鸟沿着左上方飞行至界面中点；随后改为右上方的方向，回到鸟巢。



图 3-22 第八关示例与答案

**第九关：**当同时满足鸟没有吃到虫子和飞行横坐标大于 50 时，沿正左方飞行；然后设定没有吃到虫子和飞行纵坐标大于 20 时，沿正下方飞行，令鸟吃到虫子。吃到虫子后判断鸟的横坐标是否大于 40，当小于 40 时，令鸟沿着右斜上方飞行；随后改为右斜下方的方向，回到鸟巢。



图 3-23 第九关示例与答案

**第十关：**当没有吃到虫子时，鸟的飞行横坐标小于 40，则沿右斜上方飞行，大于 40 时，沿右斜下方飞行。捉到虫子后，当鸟的横坐标大于 40 时，沿左斜上方飞行，小于 40 后，沿左斜下方飞行，回到鸟巢。

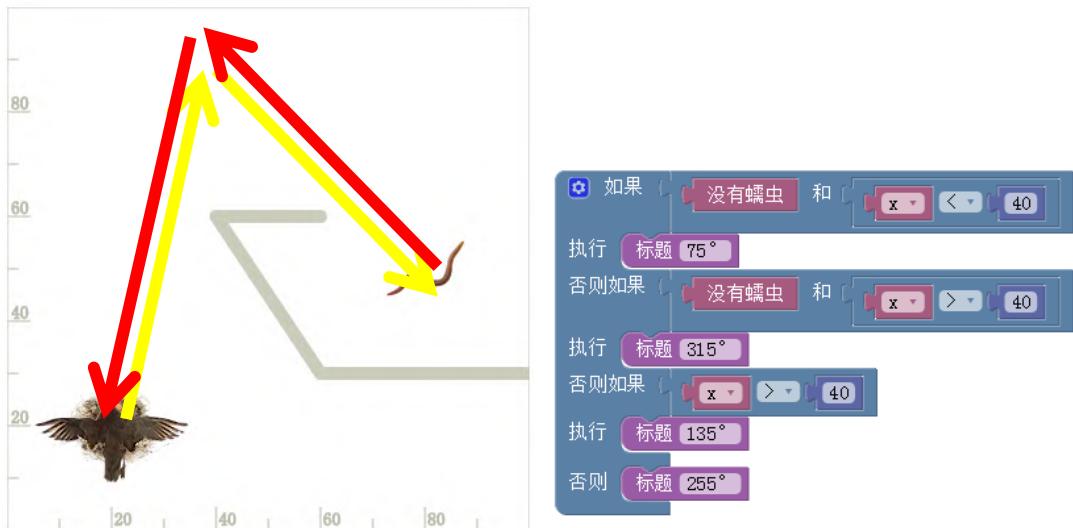


图 3-24 第十关示例与答案

### 3.7 本章练习

1. 给一个不多于五位数的正整数，请在 Blockly 中实现以下要求：

- (1) 求出它是几位数；
- (2) 分别打印出每一位数字；
- (3) 按照逆序打印出各位数字，例如原数为 321，应输出 123。

2. 函数  $y=f(x)$  表示如下，编程实现输入一个  $x$  值，输出  $y$  值。请在 Blockly 中实现这一函数。

$$y = \begin{cases} 2x+1 & (x>0) \\ 0 & (x=0) \\ 2x-1 & (x<0) \end{cases}$$

3. 自己上网查资料，搜集有关气象风级表的资料，比如说 0 级是属于无风，12 级是属于飓风等，试在 Blockly 中编写程序，输入一个风级，输出相应的概况，名称，速度等。

### 3.8 课外拓展

#### 开源软件的由来

1994年下半年，一小群分散在英国、美国内布拉斯加、旧金山以及世界其他地方的互联网软件工程师感觉自己被抛弃了，大家在网上同病相怜。在互联网进入千家万户之前，在华尔街发现网络价值之前，在浏览器战争之前，他们中的大多数都在经营网站。当时，互联网还是个要求严格的狭窄领域，以20世纪60年代末互联网发展初期的研究为指导。按照学术研究的传统，互联网软件都是免费共享的。

但是事情开始发生变化。美国国家超级计算机应用中心位于伊利诺伊大学，该中心的编程小组迁往硅谷寻找出路，加入了一家新成立的公司，也就是后来的网景公司。之前，那些同病相怜者都使用由伊利诺伊超级计算机应用中心开发的数据服务软件，将网页通过互联网传送到台式计算机上。随着伊利诺伊开发者的离去，这一大批网络专家就只能以非正式的形式在内部分享代码改进和漏洞修复。这群人中的一员布莱恩·贝伦多夫说：“我们像交换棒球卡一样交换软件补丁。”“

但是，由于缺乏领导和规范的方式，大量的快速修正很快导致程序布满混乱的补丁。因此，8名软件工程师聚集到一起制定了一套操作程序。这群人中的另一位成员兰迪·特布什说：“我们决定使用手中现有的代码开始我们自己的项目。”<sup>1</sup>他们一致同意，要在明确的软件模块上设计并开发所谓的服务器程序，以便程序员能够轻松地在一个代码块上工作，而不必担心会影响整个程序。他们建立了一个简单的控制流程，只有在需求明确且得到其他成员同意的情况下，程序员才可以为其加入其他特性。他们将这一共同努力的成果称为阿帕奇（Apache），名字来源于最初被这些自嘲的开发者戏称为布满“补丁”的服务器。

自1995年年初，阿帕奇服务器软件就被一遍遍地重写，阿帕奇程序也不断发展成为拥有40万行代码的程序。一批又一批的程序员来来去去。竞争对手的程序都是由资深程序员编写、由实力雄厚的大公司推向市场；而免费共享、由世界各地的志愿者编码并调试的阿帕奇软件，却是市场上的领头羊。2001年春天，有60%的计算机使用阿帕奇软件提供互联网网页，将微软公司和iPlanet公司（由太阳微系统公司和网景公司创建的联盟，1999年被美国在线收购）的产品远远甩在了身后。阿帕奇的开发者承认，阿帕奇并不是最快的网络服务器，也不能提供最完善的特性服务，“但我们士气高涨”，布莱恩·贝伦多夫说。

阿帕奇是“开源软件”的成功典范之一。“开源软件”既代表一种哲学思想，也代表一种软件发展模式。在像阿帕奇和Linux这样的开源项目中，软件是免费共享的，其“源代码”——经验丰富的程序员能够阅读并理解的编码指令——是公开发布的，以便其他程序员学习、分享和修改原作者的成果。

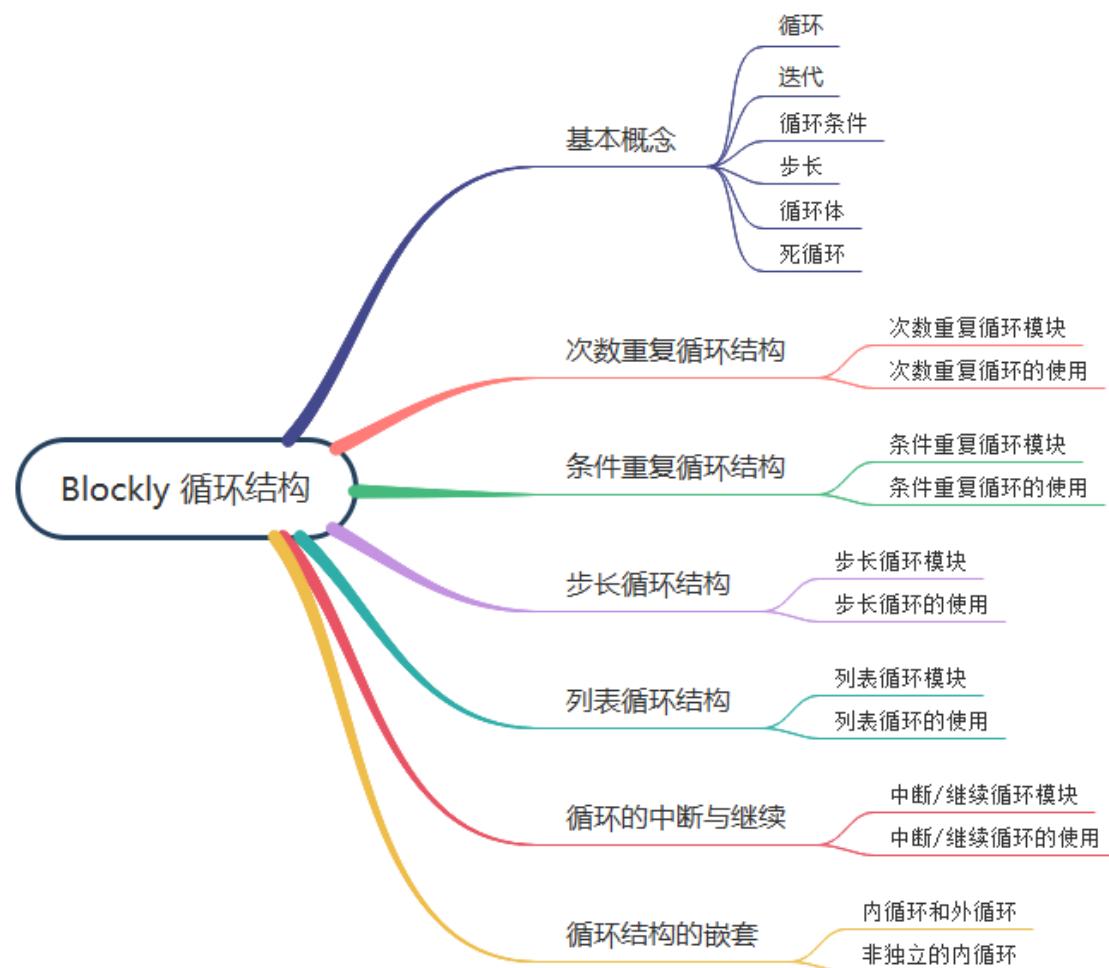
（来源：《软件故事：谁发明了哪些经典的编程语言》）

# 第4章 Blockly 循环结构

## 学习目标

1. 理解循环结构的概念。
2. 理解、运用次数重复循环结构。
3. 理解、运用条件重复循环结构。
4. 理解、运用步长循环结构。
5. 理解、运用列表循环结构。
6. 理解、运用循环的中断与继续。
7. 理解循环结构的嵌套。

## 知识图谱



结构化程序设计由三种基本结构组成，分别是顺序结构、选择结构和循环结构。在前几章中已经对顺序结构、选择结构进行了讲解，本章讲解第三种基本结构——循环结构。在本章的内容编排上，首先对循环结构中应当掌握的概念进行阐述，接下来依次讲解四种基本形式的循环和常用控制语句，然后介绍循环结构的进阶使用，最后附有本章相关的游戏和习题。在学习完本章知识内容后，我们应当能够独立分析出给定问题中一个循环的起始终止条件和重复执行次数，并能选用适当的循环语句进行实现。

## 4.1 基本概念

循环结构是指在程序中按照某种规律重复执行某一个功能的代码而设置的一种程序结构。它通常需要根据条件重复执行一段代码，直到满足某一结束条件才停止，循环条件控制是否进入一次新的循环，循环体为进入循环后进行的运算操作，如图 4-1 所示。循环结构的使用可以极大地减少重复书写的代码量，是程序设计中最能发挥计算机特长的程序结构。

在循环结构中，有几个核心的概念需要加以解释：

**迭代：**在对某一过程的多次重复中，每一次对过程的重复称为一次“迭代”，而每一次迭代得到的结果会作为下一次迭代的初始值。比如你写了一篇作文，但是你觉得有些地方需要修改，那么每一次修改的过程就可以称为一次迭代。

**循环：**程序中重复执行一段指令叫做循环。一个循环由循环条件和循环体两部分组成。

**循环条件：**用于控制循环重复次数，在能确定循环次数时可直接指定循环次数，在不能确定循环次数时可以指定循环开始和结束的条件。

**步长：**上一次循环和本次循环中循环条件的变化差值。

**循环体：**循环执行时被不断重复执行的代码段被称为循环体。

**死循环：**当指定的循环结束条件有误，循环的次数为无限次时，循环会被无休无止的重复执行，我们称一个无休止运行的程序为死循环。死循环会使得程序无法正常运行，在编写程序时，要尽量避免死循环的出现。

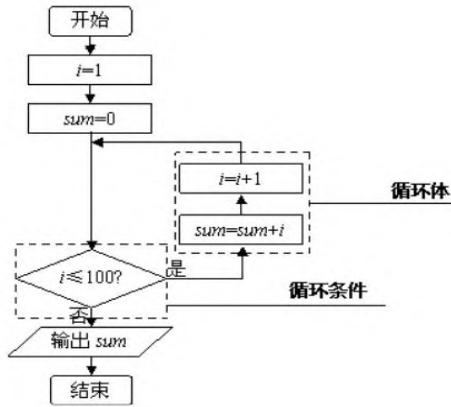


图 4-1 循环结构流程图

## 4.2 次数重复循环结构

在一个循环结构的程序中，当能够确定循环的次数时，可以使用次数重复模块用来实现计数循环。在模块中直接修改次数即可规定重复执行的次数。

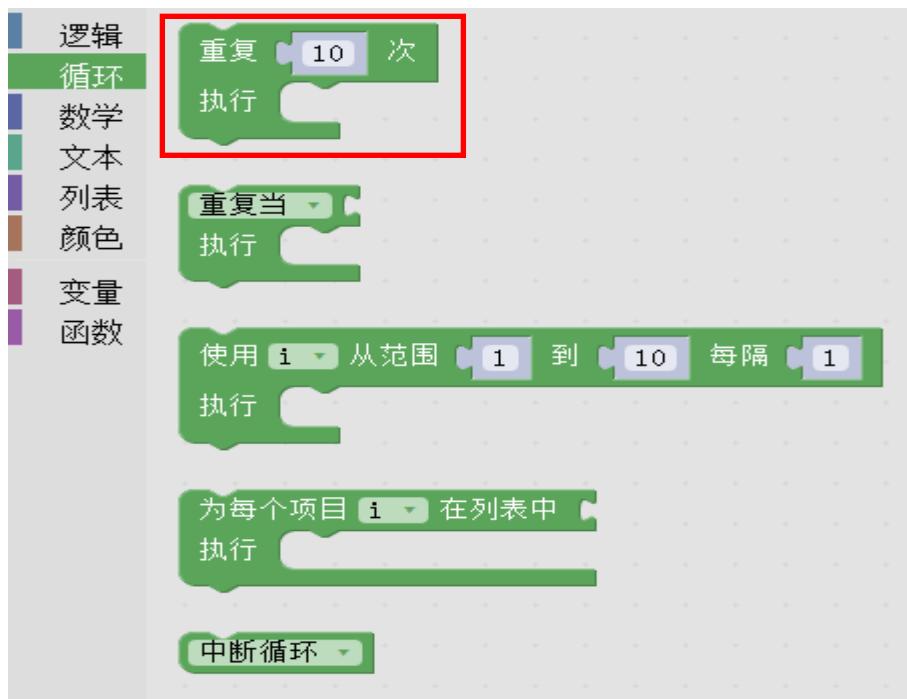


图 4-2 次数重复模块

例 1 循环三次输出“Hello World”，只要在次数重复模块的数字块中写入“3”即可。



图 4-3 输出 3 次 “Hello World”

### 4.3 条件重复循环结构

当循环的次数为未知时，可以使用条件重复模块用来实现程序的循环。条件重复模块中包含了两种不同类型的循环模式：“重复——当”和“重复——直到”。

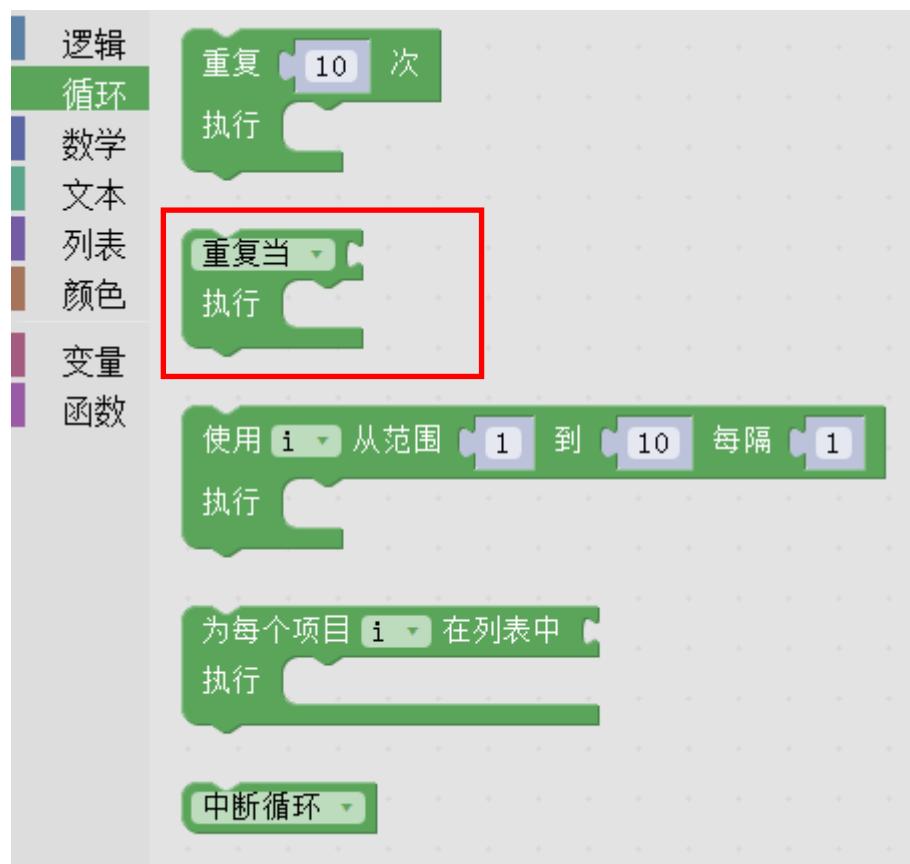


图 4-4 条件重复模块



图 4-5 条件重复循环的两种模式

条件重复模块需要与逻辑语句组合使用，可以看作是“选择+循环”的样式。在条件重复模块运行时，首先需要使用逻辑语句对循环条件进行判断，然后根据判断结果确定是否执行本次循环。

“重复——当”模块用来实现“当型”循环结构，意为“当循环条件为真时，执行循环”。只要程序的循环条件为真，就会重复执行循环体中的语句；程序的循环条件为假时，不再执行循环体中的语句，循环结束。

而“重复——直到”模块来说则与重复—当模块刚好相反，代表“直到循环条件为真时，循环才停止”。“重复——直到”是在程序的循环条件为假时，重复执行循环体中的语句；程序的循环条件为真时，停止执行循环体中的语句，循环结束。

### 小提示

对初学者来说，由于一个逻辑错误，写错了循环条件，导致相应代码段重复运行无数次，是很常见的情况。所以，在确定循环条件时，需要注意描述清楚以下几个方面：

- \*一个清晰的起始条件，作为循环的初始值。
- \*一个循环判断条件，其逻辑判断结果用来指示程序是否应该继续执行循环体。
- \*一个循环条件中数值的变化差值，最终使得循环条件能够达到让循环结束的值，不至于陷入死循环。



例 2 求从 1 一直加到 100 的和。

使用计算机语言来解决这个问题，只需要使用最容易理解的方法来解决就好：从 1 开始一个一个的加起来，一直循环加到 100。

使用“重复——当”模块时，“当循环条件为真时，执行循环”。

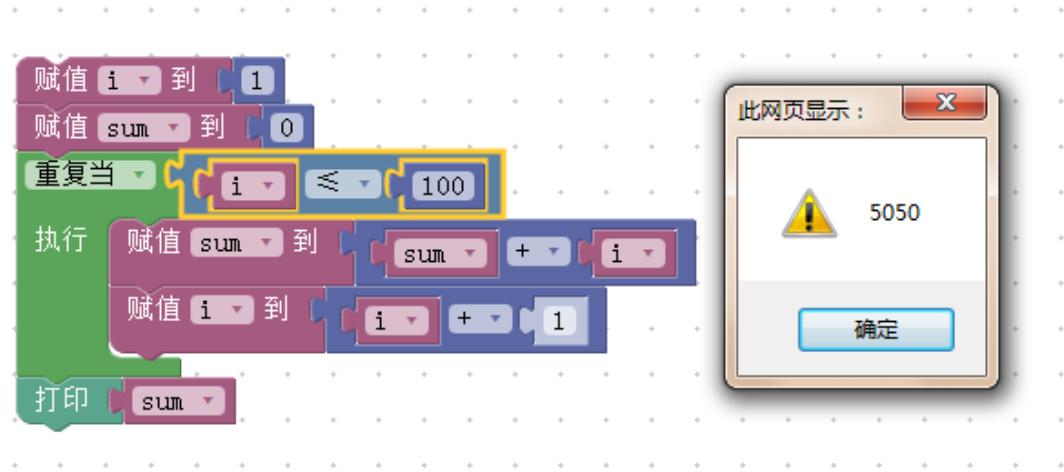


图 4-6 累加 1-100 程序方法一

使用“重复——直到”模块时，“直到循环条件为真时，循环才停止”。

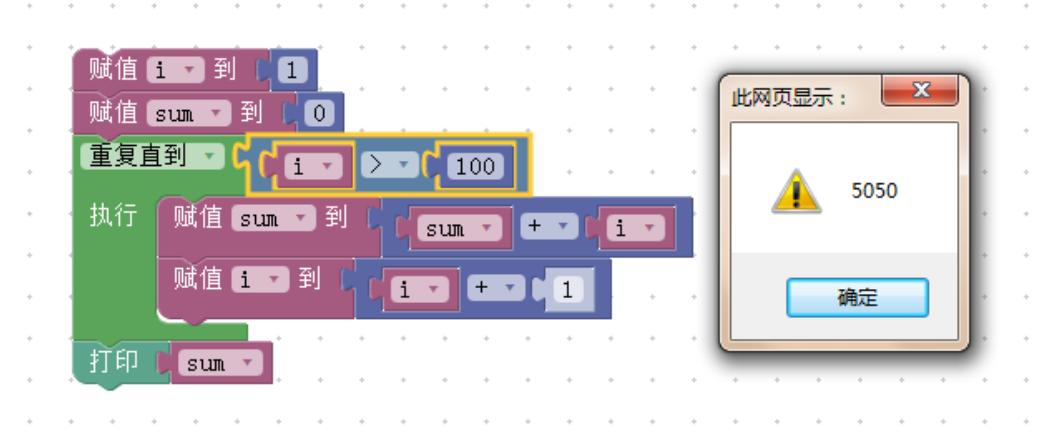


图 4-7 累加 1-100 程序方法二

练一练

使用条件重复模块的两种结构，编写程序，求 100 到 1000 相加的和的值。

## 4.4 步长循环结构

步长循环模块是在前两种模式上改进而来。在本章第一节中已经提及，步长是指上一次循环和本次循环中循环条件的变化差值。所以在步长循环模块中，最主要的特点是可以自由调节步长，创造出多样的循环条件，来解决更为复杂的问题。

步长循环模块使用最为灵活，不仅可以用于循环次数已知的情况，而且可以用于循环次数未知的情况。它完全可以替代前两种重复模块，并能够在它们的基础上自由调整循环条件的步长。

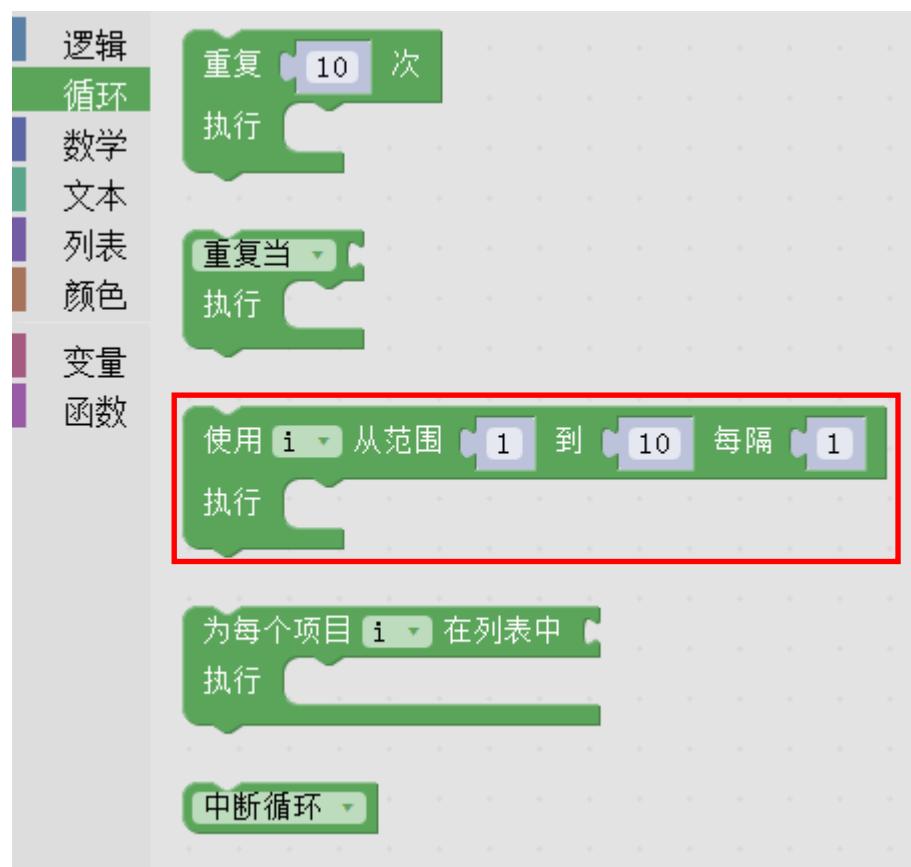


图 4-8 步长循环模块

例 3 求正整数里 1~99 中所有奇数之和。

奇数是指不能被 2 整除的数，从 1 开始，每相隔为 2 就会出现一个奇数，所有相邻奇数之间都相差 2，这个相差为 2 的差值，就是所谓的步长。



图 4-9 求正整数里 1~99 中所有奇数之和

练一练

使用步长循环模块，编写程序，求 2 到 100 中所有偶数的乘积。

## 4.5 列表循环结构

列表（list），是相同数据类型的元素按一定顺序排列的集合。列表循环模块是对列表中每一个元素进行循环迭代的模块。在术语中，这种对列表中每一个元素依次进行循环执行的过程叫做对列表元素的遍历。使用列表循环模块的循环在达到列表的最后一个值后将自动结束。所以使用列表循环模块是不太可能写成死循环的，或许唯一可以的方式是为循环使用一个含有无限元素的列表。

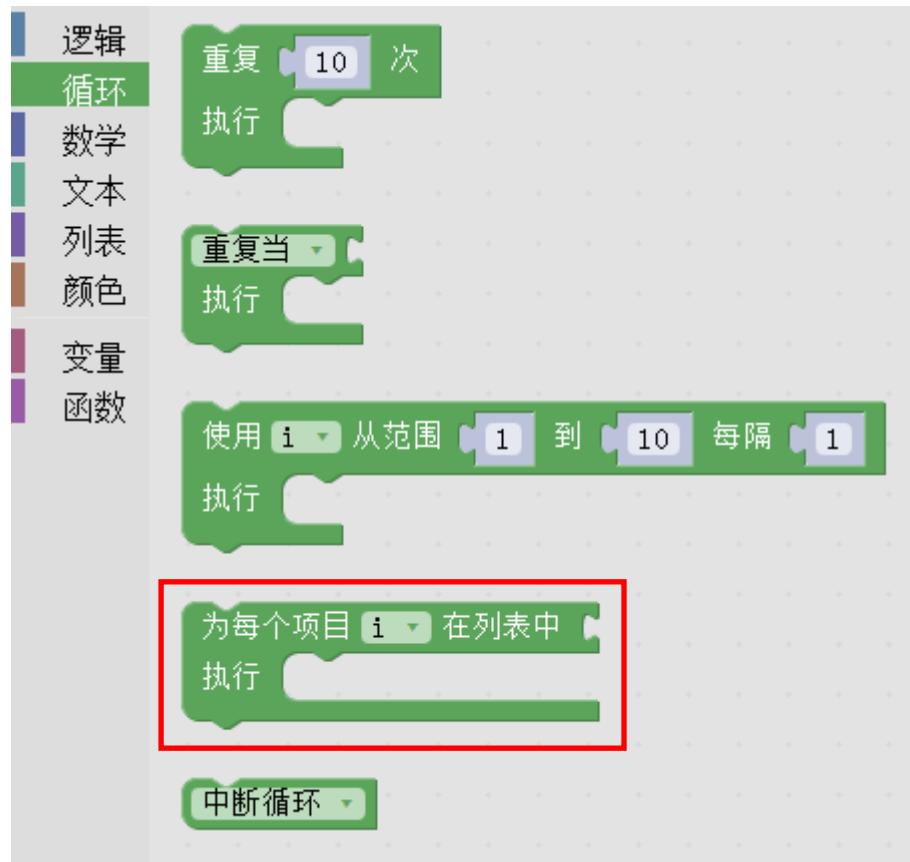


图 4-10 列表循环模块

虽然使用条件重复、步长模块也能实现循环输出每一个元素，但这种方法需要知道列表的长度，并且无法像列表循环模块这样直接使用变量定义好列表中的元素。

例 4 求一个列表中的最大元素，我们可以用列表循环模块来实现：

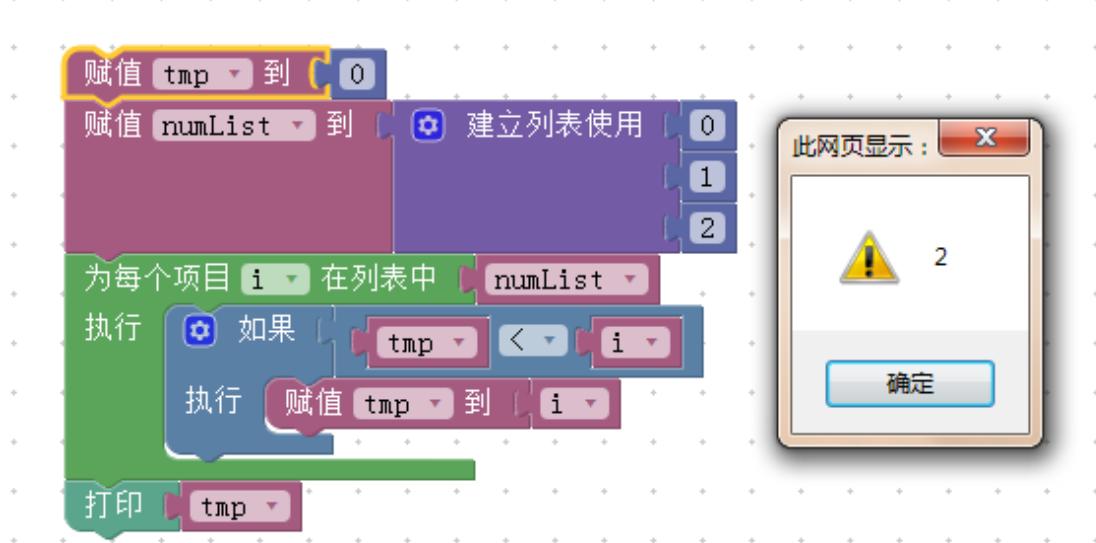


图 4-11 求一个列表中的最大元素

在这里我们定义了一个变量 tmp 去保存最大的值，将 tmp 初始化为 0 保证 tmp 刚开始

为最小，之后用列表循环模块循环拿出列表中的元素不断的与 tmp 相比较，一旦当前值大于 tmp 则修改 tmp 为这个值。最终在 tmp 中的值将是最大的值。

## 4.6 循环的中断与继续

在循环模块中，还有两种用于控制循环的特殊语句模块。

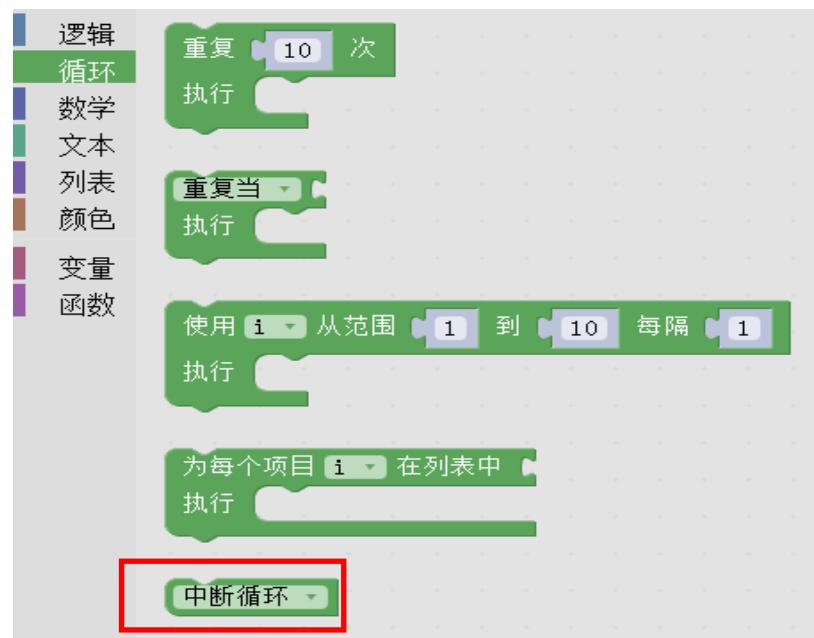


图 4-12 中断/继续模块



图 4-13 中断/继续模块选项

中断模块可以用来从循环体内跳出，即提前结束循环，接着执行循环后面的其他语句。中断模块适用于我们不知道循环次数，在程序执行过程中满足一定条件需要提前结束循环的时候。

继续模块为结束本次循环，即跳过循环体中下面尚未执行的语句，接着进行下一次循环条件中是否执行循环的判定。

继续模块与中断模块的区别是：中断模块结束整个循环过程，不再判断执行循环的条件是否成立。继续模块则只结束本次循环，接着进行下一次循环条件中是否执行循环的判定。继续模块并不终止整个循环的执行。

例 5 找出 100–200 内第一个能被 3 整除的数，把它打印出来。

在这道题中，我们只需要找到 100–200 中第一个能被 3 整除的数，而不需要找出所有能被 3 整除的数，所以在找到第一个能被 3 整除的数以后，后面的计算已经没有必要了，循环在这里就可以停止了。此时，我们需要在找到第一个数之后使用中断循环模块。



图 4-14 打印 100–200 内第一个能被 3 整除的数

在这个程序中，我们算出来 100–200 中第一个能被 3 整除的数后，因为使用了中断循环模块，整个循环过程结束了，不会再进行后续的循环。

但如果我们将题目改一下，改为找出 100–200 内所有能被 3 整除的数，并把他们打印出来。

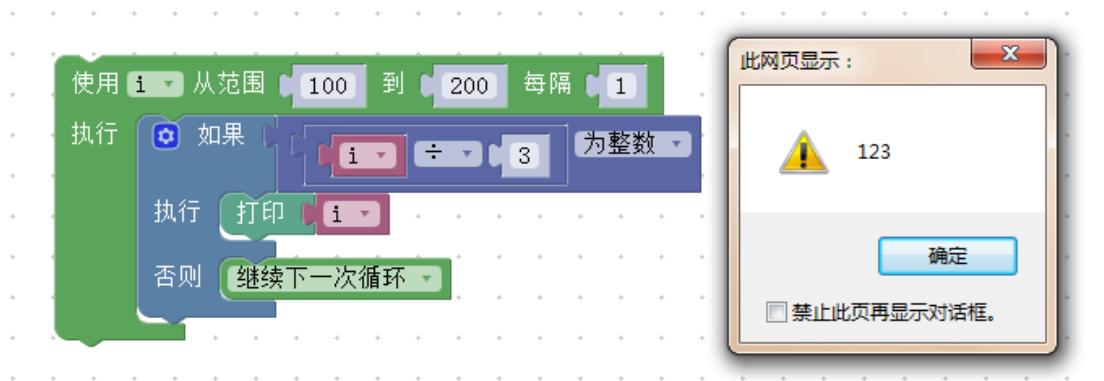


图 4-15 打印 100–200 内所有能被 3 整除的数

这道题中，我们需要找出 100–200 中所有能被 3 整除的数，在找到能被 3 整除的数时，把它打印出来，而遇到不能被 3 整除的数时，则使用继续模块跳过数字的打印直接进入下一轮循环。

## 4.7 循环结构的嵌套

和选择结构一样，循环结构中，一个循环体内又包含另一个完整的循环结构，称为循环

的嵌套。内嵌的循环中还可以再嵌套循环，这就形成了多层循环。各种语言中关于循环的嵌套的概念都是一样的。四种循环（次数重复，条件重复，步长，列表循环）可以互相嵌套，当循环嵌套在循环中，程序就变得复杂了起来。

### 4.7.1 内循环和外循环

当使用嵌套循环时，内外循环的工作原理类似于钟表的运行：一个时钟的秒针就像嵌套内层的内循环，同时分针就像对应的外循环。秒针每次转动转一圈，分针才转动一格。

同理，嵌套中的外循环是先开始后结束的运行顺序，内循环则是比外循环后开始先结束的。内循环快速运行，外循环的循环速度则慢许多。每一次外循环，都会完成一个完整的内循环。

### 4.7.2 非独立的内循环

有时，内部循环的工作方式取决于外部循环的哪一步运行。

例 6 创建一个程序打印在一年中所有的日期。

可以使用嵌套循环去打印月份的每天，但内部循环的次数取决于外部循环，这是因为每个月有不同的天数。

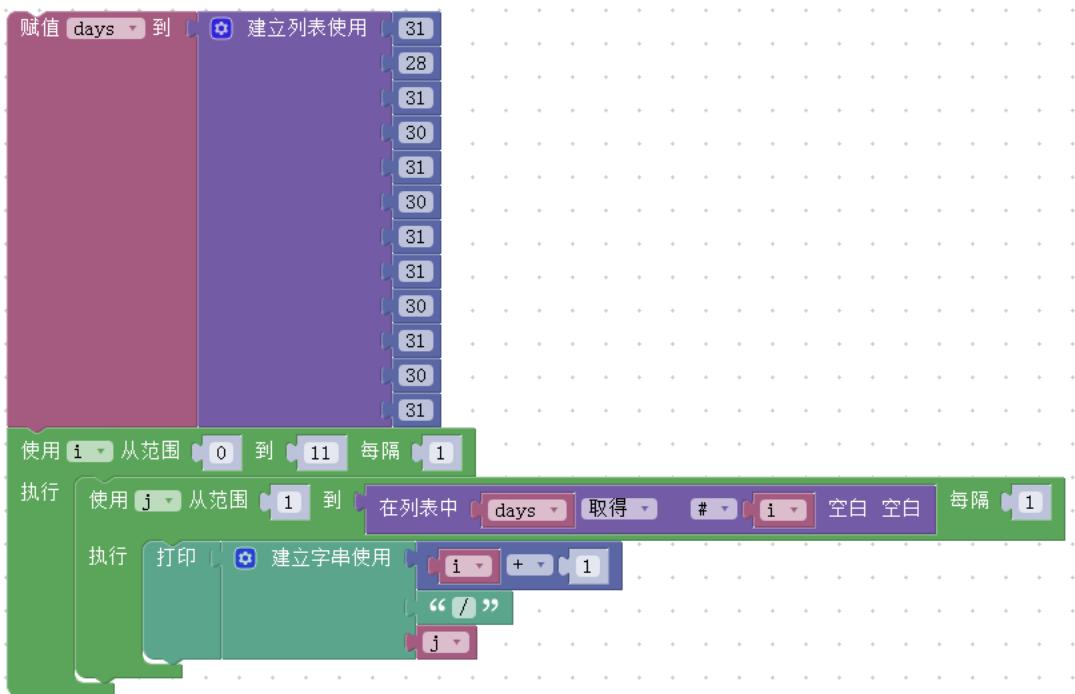


图 4-16 打印在一年中所有的日期

在这里，内部循环重复的是不同的次数，这取决于它在哪个月份上，因为它的结束条件

是  $j < \text{days}[i]$ 。对于月份，我们绘制了一个依赖于存放每一月的天数的数组的内循环，但是对于给一个非独立内循环构思一个规则需要更多思考。由于内部循环和外部循环的关系，嵌套循环可以设计出更具有挑战性的程序。

## 4.8 小试牛刀——游戏 1：迷宫

学习完如何在 Blockly 中使用循环结构后，我们通过一个游戏来熟练掌握这种程序结构，游 戏 的 地 址 如 下：

<http://cooc-china.github.io/pages/blockly-games/zh-hans/maze.html?lang=zh-hans>。

### 游戏规则：

- ① 在本游戏中，玩家扮演图中的人物，要从给定的位置到达红色指示符标注的目的地，其中人物下方绿色的箭头指示人物目前面对的方向，黄色的宽线代表可通行的路；
- ② 玩家需要通过所学的 Blockly 知识，从右侧给定的代码模块中选取适当的模块，构建 Blockly 模块组来控制人物的移动，使得人物移动到红色指示符标注的目的地即可通关。

### 通关详解：

**第一关：**直接前进即可到达，注意，街道的长度需要向前移动两步。



图 4-17 第一关示例与答案

**第二关：**需要经历两个拐弯才能到达目的地。



图 4-18 第二关示例与答案

**第三关:** 加入了循环结构, 循环使用“向前移动”即可到达。



图 4-19 第三关示例与答案

**第四关:** 开始进入带转向的循环结构, 根据路线拆分出循环体, 然后进行循环即可到达。

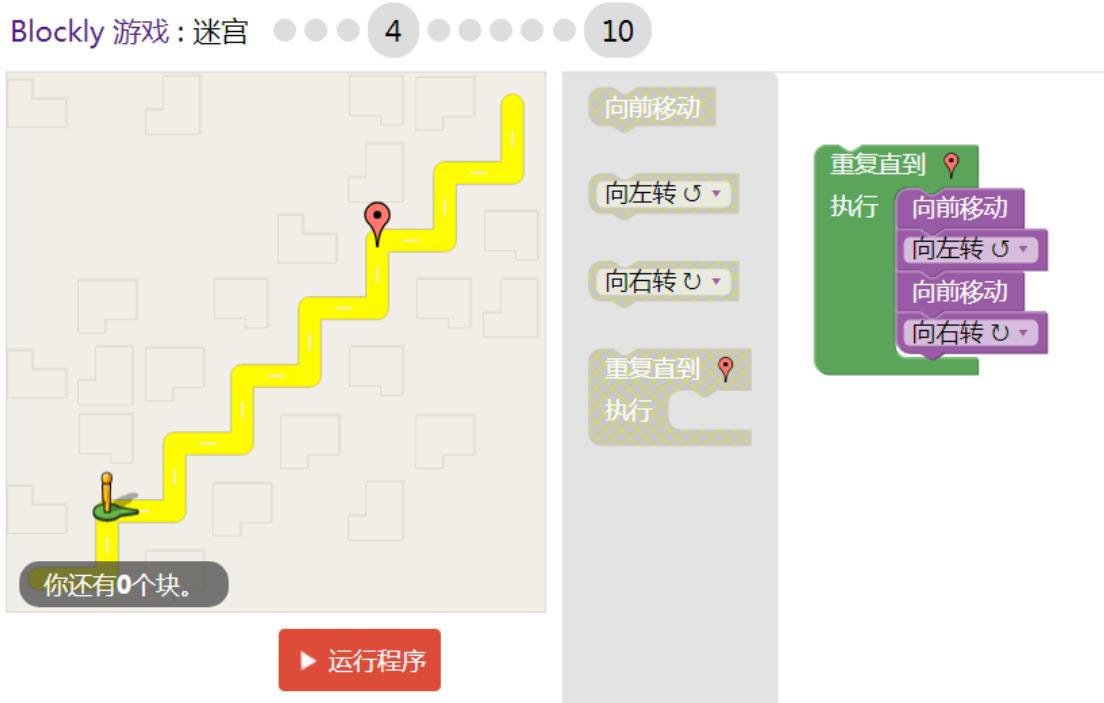


图 4-20 第四关示例与答案

**第五关：**先进行一个转向，之后再进行循环。



图 4-21 第五关示例与答案

**第六关：**每次拐弯的方向都是一致的，根据这一点进行循环条件的设置。



图 4-22 第六关示例与答案

**第七关：**在遇到岔路口时，需要考虑不同转向的先后顺序。沿着最短的路径走即可到达。



图 4-23 第七关示例与答案

**第八关：**根据路线不同方向的拐弯，设置不同的循环分支。

### Blockly 游戏：迷宫

8 10

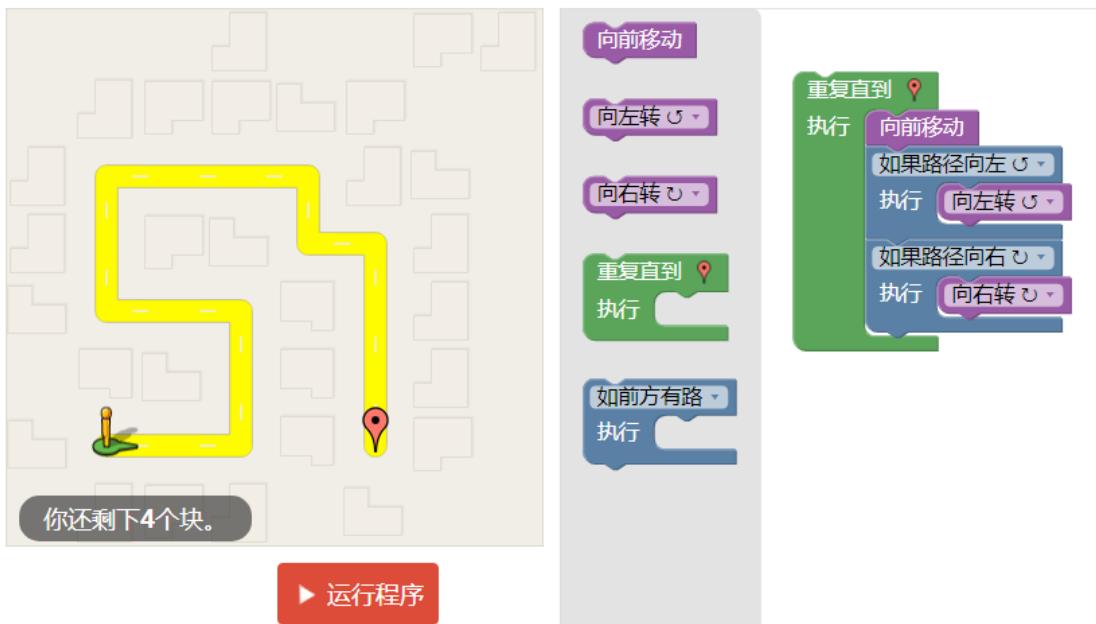


图 4-24 第八关示例与答案

**第九关：**注意循环和选择中的嵌套关系，沿最短路径走即可到达，警惕途中的方框回路，不要陷入死循环。

### Blockly 游戏：迷宫

9 10



图 4-25 第九关示例与答案

**第十关：**迷宫设置了很多容易进入的误区陷阱，正确的路线需要沿着左边的墙走。

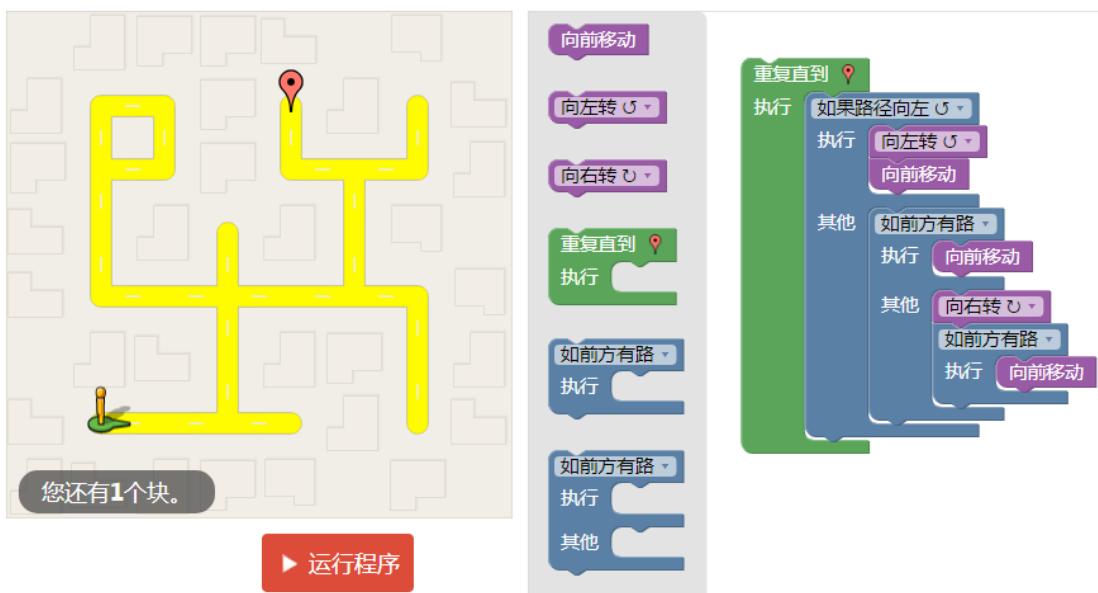


图 4-26 第十关示例与答案

## 4.9 小试牛刀——游戏 2：乌龟

乌龟这款 Blockly 游戏分为 10 个关卡，主要是为了让大家能够更好的掌握循环语句的使用，在游戏中体会 Blockly 编程的乐趣，进而掌握基础程序语言的运用。游戏地址如下：  
<http://cooc-china.github.io/pages/blockly-games/zh-hans/turtle.html?lang=zh-hans>

### 游戏规则：

- ① 我们需要通过控制代码来让乌龟沿着特定的线路爬行；
- ② 代码主要由选择结构、循环结构和逻辑判断组成；
- ③ 点击“运行程序”按钮后程序就会执行右侧的代码。乌龟走完特定线路后游戏结束，顺利通关。

### 通关详解：

**第一关：**让乌龟向前爬行 100 步，然后右转 90° 接着爬行 100 步，直到爬完一个正方形线路为止，如图 4-27 所示。

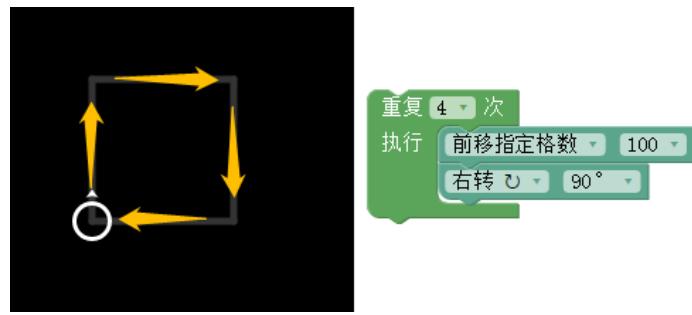


图 4-27 第一关示例与答案

**第二关：**让乌龟向前爬行 100 步，然后右转  $72^\circ$  接着爬行 100 步，直到爬完一个正五边形线路为止，如图 4-28 所示。

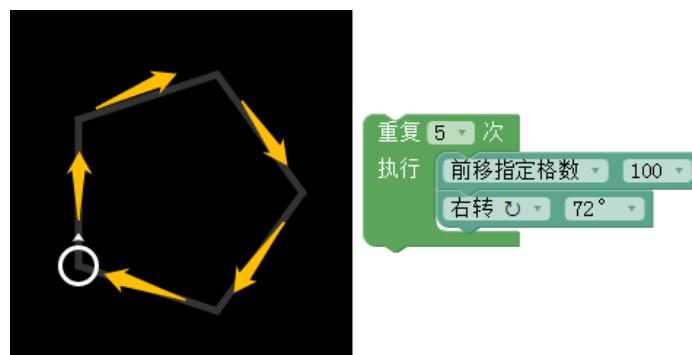


图 4-28 第二关示例与答案

**第三关：**让乌龟向前爬行 100 步，然后右转  $144^\circ$  接着爬行 100 步，直到爬完一个正五角星线路为止，如图 4-29 所示。

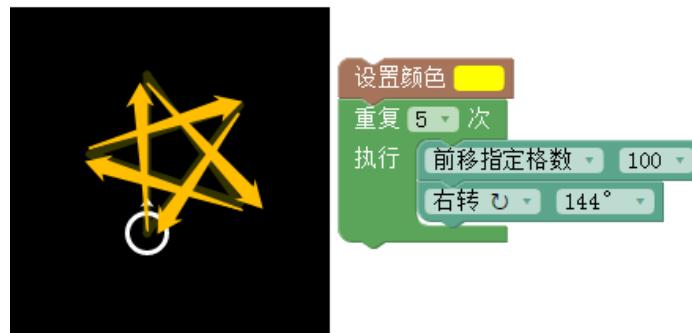


图 4-29 第三关示例与答案

**第四关：**先将画笔颜色设置为黄色，再让乌龟向前爬行 50 步，然后右转  $144^\circ$ 接着爬行 100 步，直到爬完一个正五角星形线路后拿起笔，爬行 150 步后放下笔，继续爬行 20 步，如图 4-30 所示。

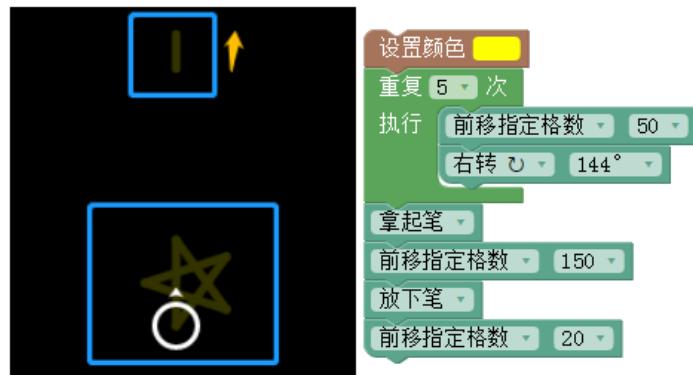


图 4-30 第四关示例与答案

**第五关：**先将画笔颜色设置为黄色，让乌龟向前爬行 50 步，然后右转  $144^\circ$  接着爬行 50 步，直到爬完一个正五角星线路后，拿起笔往前爬行 150 步，右转  $90^\circ$  后继续完成一个五角星的线路，直到爬完四个正五角星线路为止，如图 4-31 所示。

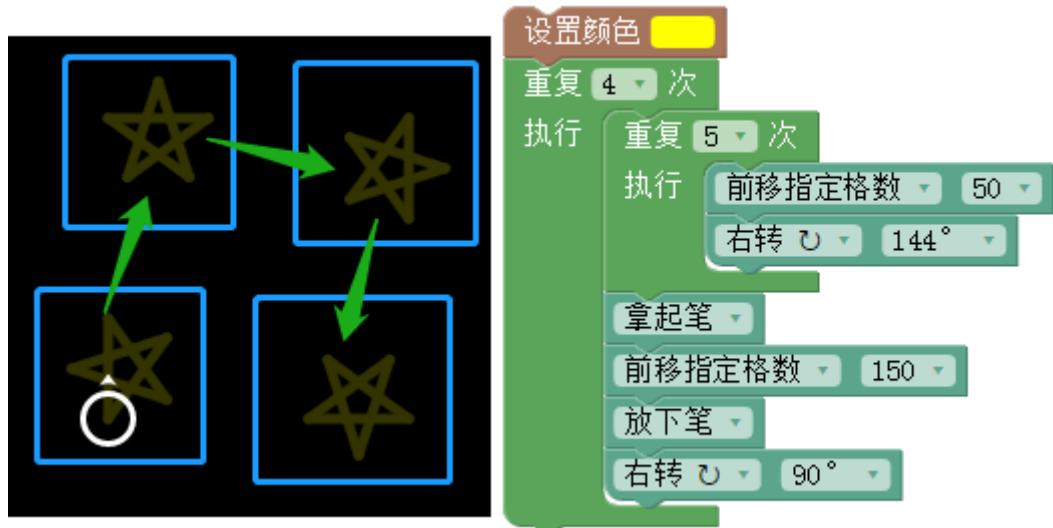


图 4-31 第五关示例与答案

**第六关：**如图 4-32 所示，先将画笔颜色设置为黄色，让乌龟向前爬行 50 步，然后右转  $144^\circ$  接着爬行 50 步，直到爬完一个正五角星线路后拿起笔，向前爬行 150 步后，向右转  $120^\circ$ ，接着爬完一个正五角星线路，直到完成三个五角星线路后，将颜色设置为白色，再次拿起笔，向前爬行 100 步后发下笔，继续爬行 50 步为止，其模块组合如图 4-33 所示。

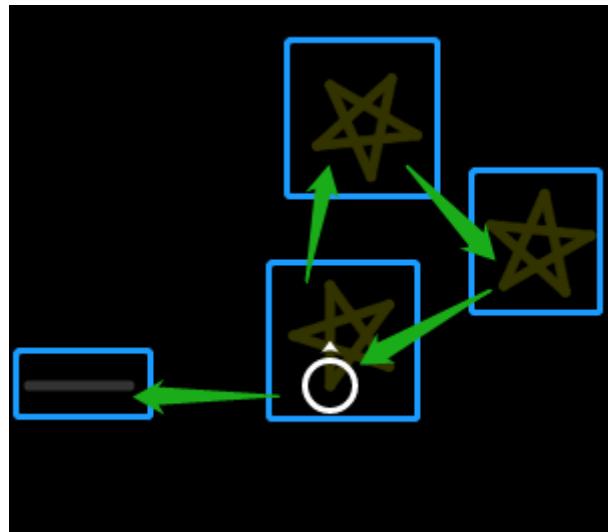


图 4-32 第六关乌龟爬行路线

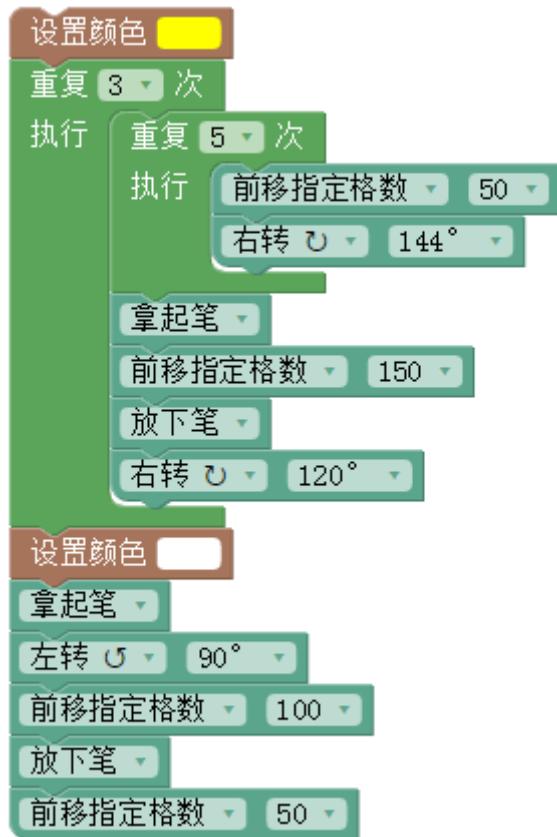


图 4-33 第六关答案

**第七关:** 如图 4-34 所示, 前面部分和第六关相同, 可参照第六关内容。当乌龟再次爬回原点时, 将颜色设置为白色, 再次拿起笔, 向前爬行 100 步后发下笔, 继续爬行 50 步, 后退 50 步, 右转 45° 后继续爬行 50 步, 重复 4 次为止, 其模块组合如图 4-35 所示。

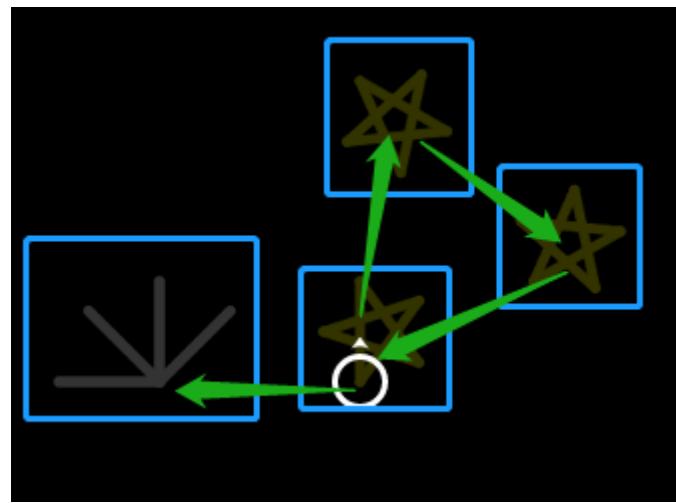


图 4-34 第七关乌龟爬行路线



图 4-35 第七关答案

**第八关：**如图 4-36 所示，前面部分和第六关相同，可参照第六关内容。当乌龟再次爬回原点时，将颜色设置为白色，再次拿起笔，向前爬行 100 步后发下笔，继续爬行 50 步，后退 50 步，右转 1° 后继续爬行 50 步，重复 360 次画出满月为止，其模块组合如图 4-37 所示。

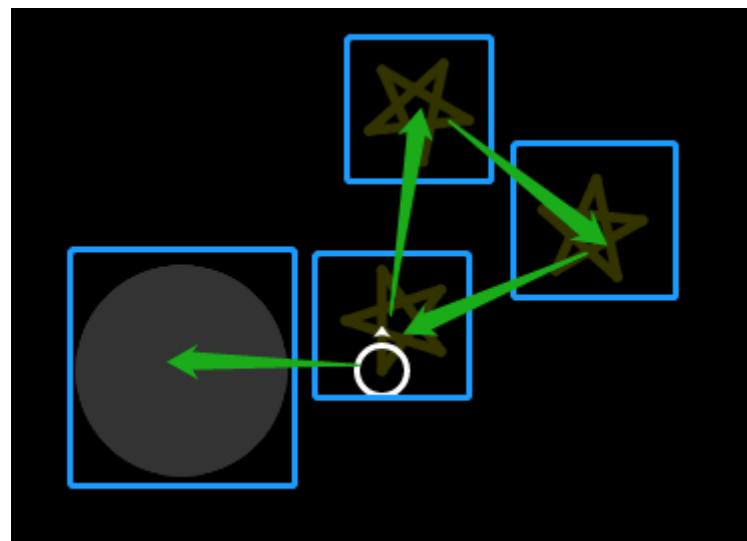


图 4-36 第八关乌龟爬行路线

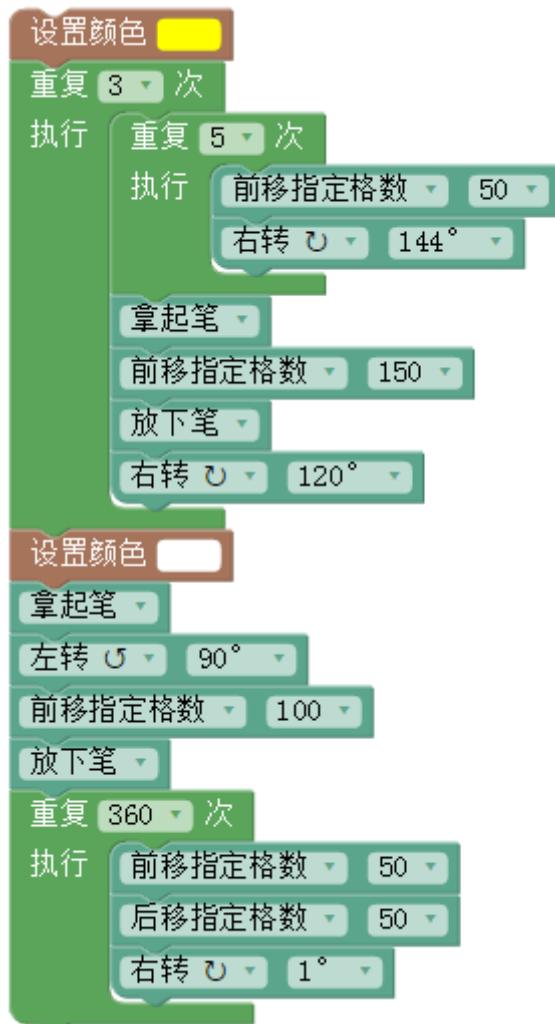


图 4-37 第八关答案

**第九关:** 如图 4-38 所示, 前面部分和第八关相同, 可参照第六关内容。当乌龟再次爬行出满月的圆形图案后, 右转  $120^\circ$  拿起笔, 向前爬行 20 步, 将颜色设置为黑色, 放下笔。左转  $1^\circ$  向前爬行 50 步, 后退 50 步, 重复 360 次画出月牙状为止, 其模块组合如图 4-39

所示。

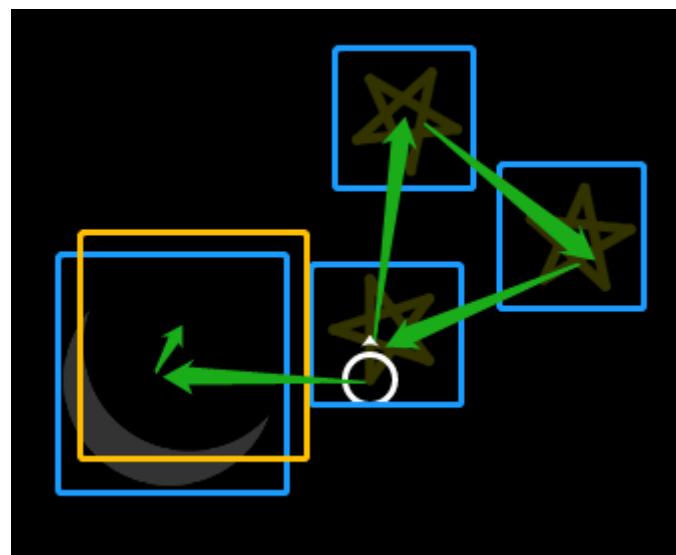


图 4-38 第九关乌龟飞行路线

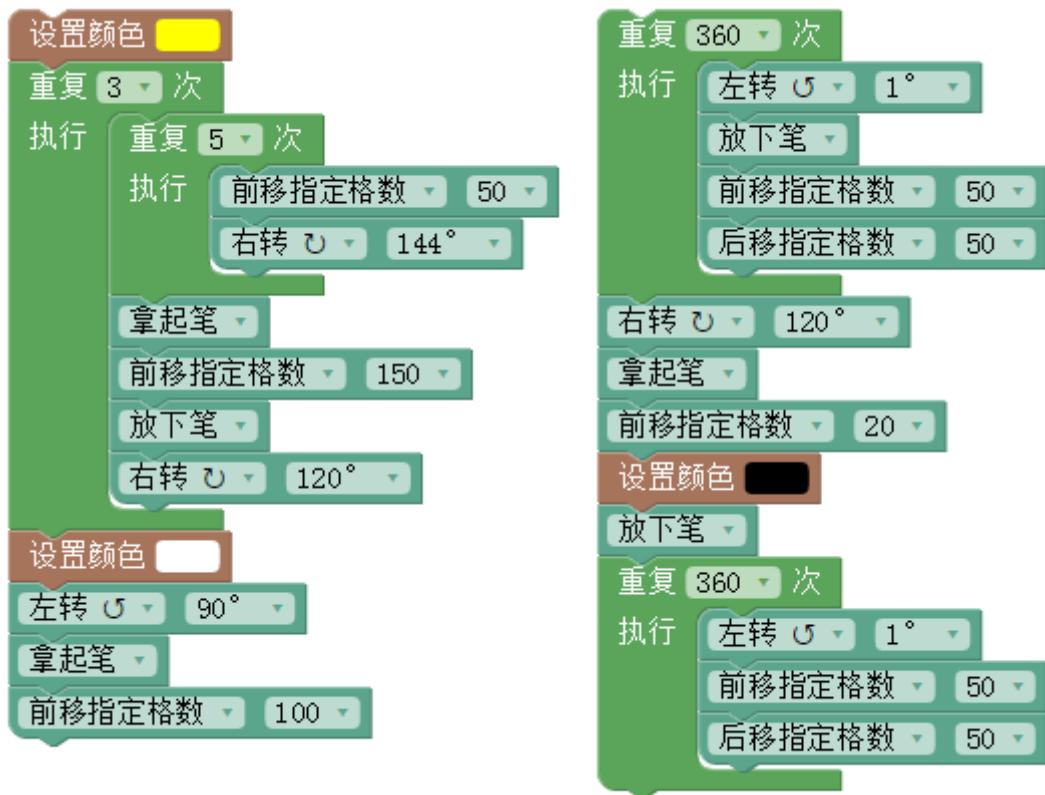


图 4-39 第九关答案

**第十关：**进入这一关后就弹出如图 4-40 的提示，没有给出明确的通关要求，但是需要大家自由发挥，画出自己任何想要的形状，大胆进行探索，看看由模块组成的画笔可以构造出来什么样的神奇画面吧。

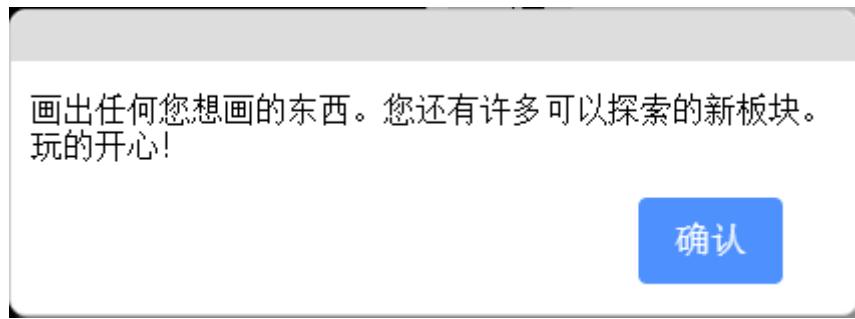


图 4-40 第十关关卡提示

## 4.10 本章练习

1. 一球从 100 米高度自由落下，每次落地后反跳回原高度的一半，再落下。求它在第 10 次落地时，共经过多少米？第 10 次反弹多高？
2. 输入一个数，判断这个数是否为素数。（注：素数为仅能被 1 和自身整除的数。）
3. 编写程序，给出年、月、日，计算出该日是该年的第 n 天。

## 4.11 课外拓展

### 高级语言的循环结构

Blockly 循环结构的各个模块，其实与实际的程序代码中的基本语句是相互对应的。下面就来一起看一看，在 C 语言或者 JAVA 语言等高级语言中，Blockly 循环结构中各模块的代码模样吧。

条件重复中的重复-当模块，对应的是 While 语句，适合判断次数不明确的操作，它的执行条件是“只要控制表达式为 true，while 循环就会反复地执行语句”。

While 语句基本格式为：

```
While(条件表达式){
```

    语句块

```
}
```

条件重中的重复-直到模块，对应的类似 Do-while 语句，它先执行循环体，然后根据判断条件决定是否再次执行循环，即 Do-while 语句至少执行一次循环。

Do-while 语句基本格式为：

```
Do{
```

    语句块

```
}while(条件表达式);
```

步长重复模块，对应的是 For 语句，适合针对一个已知范围判断进行循环操作。

For 语句基本格式为：

```
For(初始表达式; 条件表达式; 步进表达式){
```

语句块

```
}
```

列表重复模块，对应 JAVA 中的 For-each 语句，在 C 语言中，并没有这个语句。For-each 语句是 for 语句的特殊简化版本，主要用于遍历数组和集合，但是 for-each 语句并不能完全取代 for 语句。

For-each 语句基本格式为：

```
For(类型变量: 数组或集合){
```

语句块

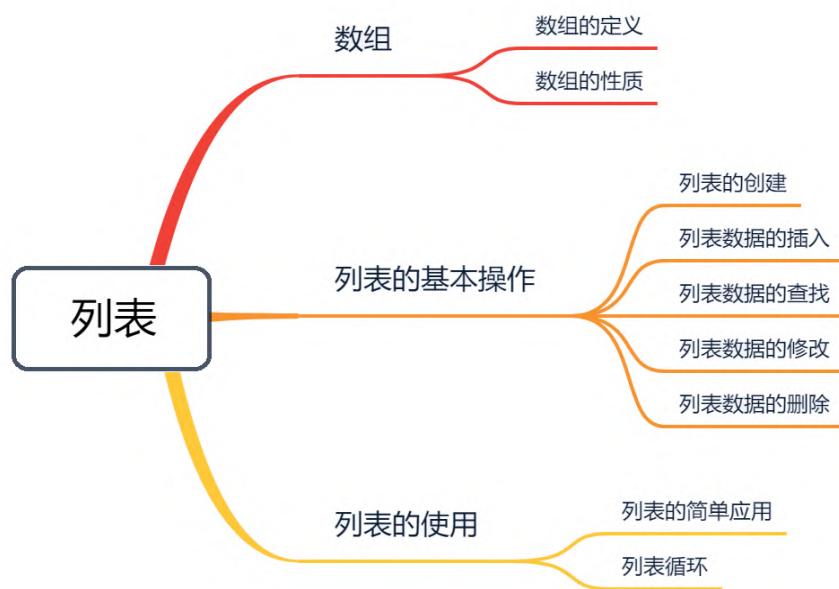
```
}
```

中断模块对应的为 break 语句，用来中断跳出循环；继续模块对应 continue 语句，为结束本次循环，接着开始下一次循环条件的判断。

# 第 5 章 Blockly 列表

## 学习目标

1. 了解数组的定义。
2. 了解列表的创建，插入，修改，查找，删除操作。
3. 掌握列表的使用。



## 知识图谱

在本章中，我们将学习 Blockly 列表的相关知识，包括什么是数组、列表的创建、查找、插入、删除操作。此外，我们还将学习列表的一些简单应用，同时列表也能帮助我们使用循环语句，是非常重要的一部分。

### 5.1 数组

#### 5.1.1 数组的定义

数组是用于储存多个相同类型数据的集合。

在 Blockly 中，一个列表即为一个数组，同时也是一个变量，若将有限个类型相同的变

量所组成的集合命名，那么这个名称为数组名，也是列表名。数组是在程序设计中，为了处理方便，把具有相同类型的若干元素按无序的形式组织起来的一种形式。这些无序排列的同类数据元素的集合称为数组。

## 5.1.2 数组的性质

数组作为一种能够被使用的变量，有以下特点：

1. 数组是相同数据类型的元素的集合。
2. 数组中的各元素的存储是有先后顺序的，它们在内存中按照这个先后顺序连续存放在一起。
3. 数组元素用整个数组的名字和它自己在数组中的顺序位置来表示。例如， $a[0]$  表示名字为  $a$  的数组中的第一个元素， $a[1]$  代表数组  $a$  的第二个元素，以此类推，在 blockly 中则具体用“#”+数字来表示元素的位置。

数组在计算机程序中可以分为一维数组和多维数组，在 blockly 中我们只用到一维数组。

## 5.2 列表的基本操作

### 5.2.1 列表

列表是一种数据项构成的有限序列，即按照一定的线性顺序，排列而成的数据项的集合，在这种数据结构上进行的基本操作包括对元素的查找，插入，和删除。

通过上面的学习，我们知道了列表是 blockly 中数组的表现形式，我们可以将创建的列表看作是一个数组变量，以此来完成我们的操作。

### 5.2.2 列表的创建

在 blockly 中，列表的创建是通过“创建列表”块来实现的，将“创建列表”块拖动到代码区，手动输入需要添加的元素，元素可以为字符，数字等。



如图所示，我们可以创建一个列表为 { “1”, “2”, “3” }，同样也可以创建其他数据类型的数组，这里要注意的是，同一列表中的元素数据类型必须相同，即必须同为字符或数字。

另外，当我们需要创建同一元素组成的列表时，可以使用另一代码块：



在这个代码块中，我们是建立了一个有五个元素的列表，这五个元素都为“123”，即 {123, 123, 123, 123, 123}。

到这里我们可以看到，当我们创建一个列表后，变量模块会自动出现一个新的名为“列表”的变量，这就是我们目前所创建的列表，我们可以通过变量模块对所创建的列表名进行修改，只有当我们将所创建的列表赋值给相应的变量时，列表才能够被显示。

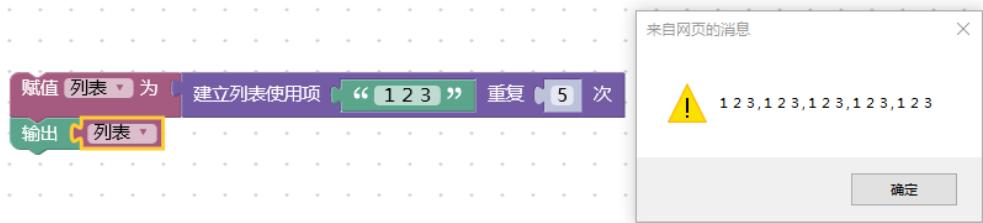
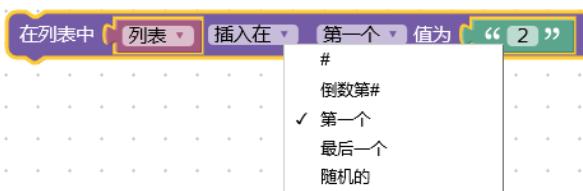


图 5-1 列表

显示结果如上图所示。

### 5.2.3 列表数据的插入

列表在创建好之后，可以被修改，我们可以根据我们的要求加入在规定位置插入元素，这里需要使用到的是插入代码块：



我们可以看到 blockly 提供了插入元素的功能，如左图所示，我们将需要的元素插入到需要的位置。blockly 为用户提供了“第一个”“最后一个”“随机的”这三个默认值，同时也支持用户自己填写列表元素

的位置，通过“#”+的方式，并且不仅可以从前往后数，也能从后向前。

最终显示结果如图所示。



图 5-2 列表 2

### 5.2.4 列表的查找和修改

当一个列表较长或是较为复杂的时候，我们如何从列表中获取数据呢？这就需要列表模块中的查找代码块：



通过查找代码块，我们可以获取到列表中第#项的元素，以此满足我们的需求。具体显示结果如下图所示：



图 5-3 列表 3

除此之外，我们还能通过元素来判断该元素的位置：



例如寻找第一次出现某个元素是在列表的第几项，显示结果如图所示：



图 5-4 列表 4

同样，当我们创建好一个列表时，因为各种不同的需要，我们有时会对列表中的元素进行修改，查找到该元素之后，blockly 同样为用户提供了修改的功能，在这里我们使用列表模块中的修改代码块：



我们可以看到在这个代码块中我们可以随意修改列表中的元素，通过元素在列表中的位置，来确定我们需要修改的元素，通过该代码块修改第#项元素。



图 5-5 列表 5

显示结果如图 5-5 所示。

## 5.2.5 列表数据的删除

在学习完以上几个操作后，相信大家都能想到我们接下来要了解到的就是 blockly 列表数据的删除功能，同样， blockly 为我们提供了删除的代码块：



如图所示，该代码块与查找代码块为同一个，我们可以在代码块上进行操作，选择我们需要的功能，并且，在这个过程中，我们不光可以删除该元素，还能获取到该元素。

具体显示结果如下图所示：



图 5-6 列表 6

以上，我们为大家介绍了 blockly 中列表的创建，列表数据的插入，查找修改和删除几项基本操作，接下来我们将学习到列表的一些简单应用。

## 5.3 列表的使用

### 5.3.1 列表的简单应用

#### 1. 统计字符串长度

首先我们要提到的是列表的性质一列表的长度，列表的长度具体是指该列表拥有多少项数据元素，统计一个列表的长度能够帮助我们更好地使用数据，blockly 中有专门为统计列表长度所提供的代码块：

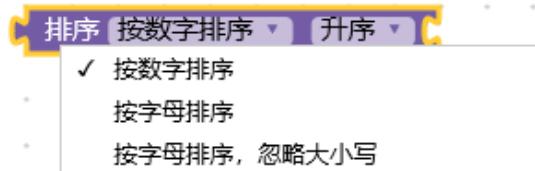
这个代码块不仅能够统计 的长度，同时也能对字符串的长度进行计算，通过该代码块可以快速地知道所输入的字符串长度。

这里要提到另一个需要注意的概念：空列表——当一个列表的长度为 0 时，我们称该列表为空列表。blockly 同样有能够与空列表相关的代码块，如创建空列表：



## 2. 排序功能

在其他程序语言中，有许多的排序方式，例如冒泡排序等，blockly列表也有排序的功能：



在该代码块中有三种默认的排序方式：按数字排序，按字母排序，按字母排序并忽略大小写，通过此功能可对一组数据，例如学生的期末成绩进行排序。

## 5.3.2 列表循环

列表循环模块是对列表（list）中每一个元素进行循环迭代的模块，也可以称为是对列表元素的遍历。所谓列表，是相同数据类型的元素按一定顺序排列的集合。使用列表循环模块的循环在达到列表的最后一个值后将自动结束。所以使用列表循环模块是不太可能错误地写成死循环的，或许唯一可以的方式是为循环使用一个含有无限元素的列表！

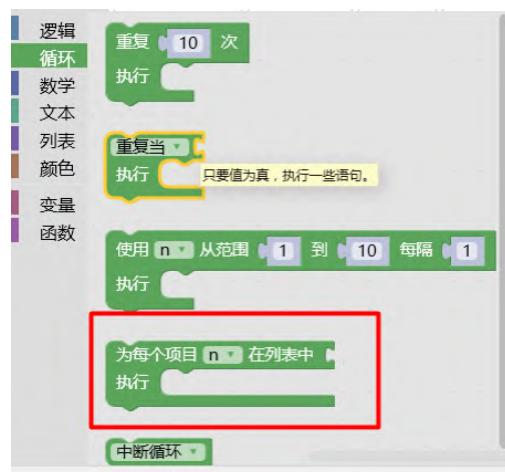


图 5-7 为项目执行

如下图所示，通过使用列表循环模块可以输出列表 numList 中的每一个元素。



图 5-8 列表

虽然使用重复、步长模块也能实现循环输出每一个元素，但用这种方法不但需要知道列表的长度，而且无法像列表循环模块一样直接就定义好了变量去代表列表的元素。



图 5-9 列表

例如求一个列表中的最大元素，我们可以用列表循环模块来实现：在这里我们定义了一个变量 tmp 去保存最大的值，将 tmp 初始化为 0 保证 tmp 刚开始为最小，之后用列表循环模块循环拿出列表中的元素不断的与 tmp 相比较，一旦当前值大于 tmp 则修改 tmp 为这个值。最终在 tmp 中的值将是最大的值。

## 5.4 小试牛刀——制作一个自动售货机

通过上面几节的学习，我们对列表的使用有了大致的了解，那么接下来我们就来利用列表制作一个简易的自动售货机吧。

需要实现的功能包括：

1. 能够看见售货机中有的物品名称。
2. 能够让用户自己选择并输入需要的物品。
3. 能够在用户输入后自动输出物品的价格。

代码实现：

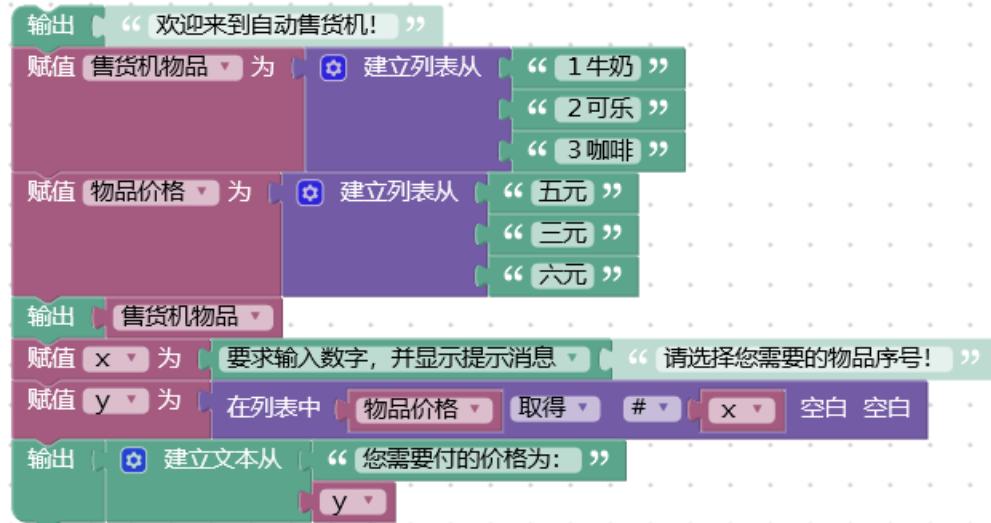
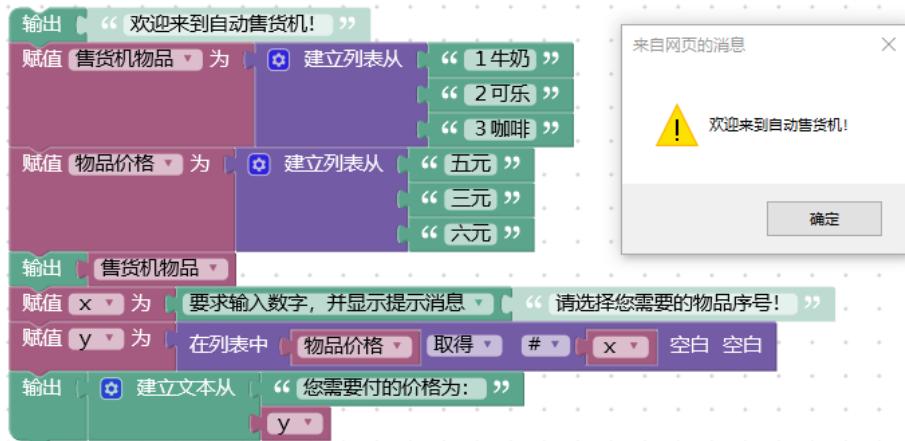


图 5-10 自动售货机

在这部分代码中，我们创建了两个列表，并利用列表的查找功能来输入我们需要的信息。具体实现如下：



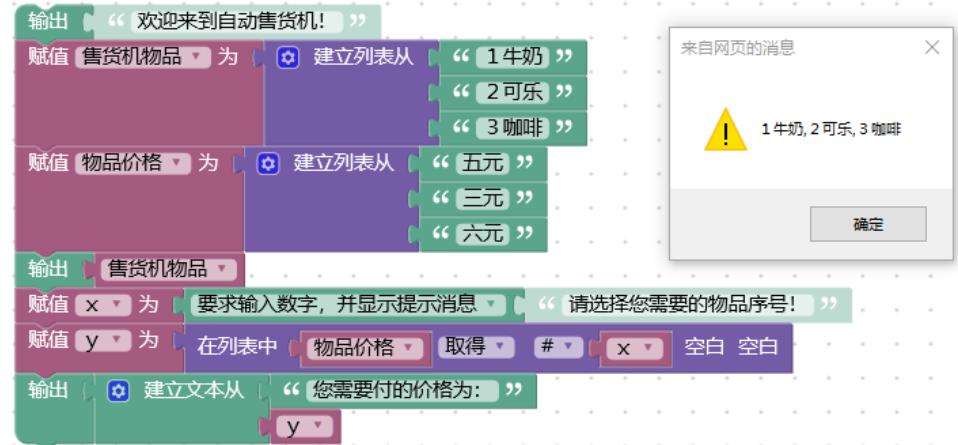


图 5-11 自动售货机 2

## 5.5 本章练习

1. 创建列表，并对该列表进行插入，查找，修改和删除操作。
2. 创建“期末成绩”列表，数据包括：85, 95, 84, 83, 78, 91, 96, 94, 93, 89. 并对它们进行排序。

## 5.6 课外拓展

### 列表

列表是一种数据项构成的有限序列，即按照一定的线性顺序，排列而成的数据项的集合，在这种数据结构上进行的基本操作包括对元素的查找，插入，和删除。在 **blockly** 的列表中我们可以只把列表看作是一个数组，但实际上，列表的两种主要表现是数组和链表，栈和队列是两种特殊类型的列表。

栈（stack）又名堆栈，它是一种运算受限的线性表。其限制是仅允许在表的一端进行插入和删除运算。这一端被称为栈顶，相对地，把另一端称为栈底。向一个栈插入新元素又称作进栈、入栈或压栈，它是把新元素放到栈顶元素的上面，使之成为新的栈顶元素；从一个栈删除元素又称作出栈或退栈，它是把栈顶元素删除掉，使其相邻的元素成为新的栈顶元素。

队列是一种特殊的线性表，特殊之处在于它只允许在表的前端（front）进行删除操作，而在表的后端（rear）进行插入操作，和栈一样，队列是一种操作受限制的线性表。进行插入操作的端称为队尾，进行删除操作的端称为队头。

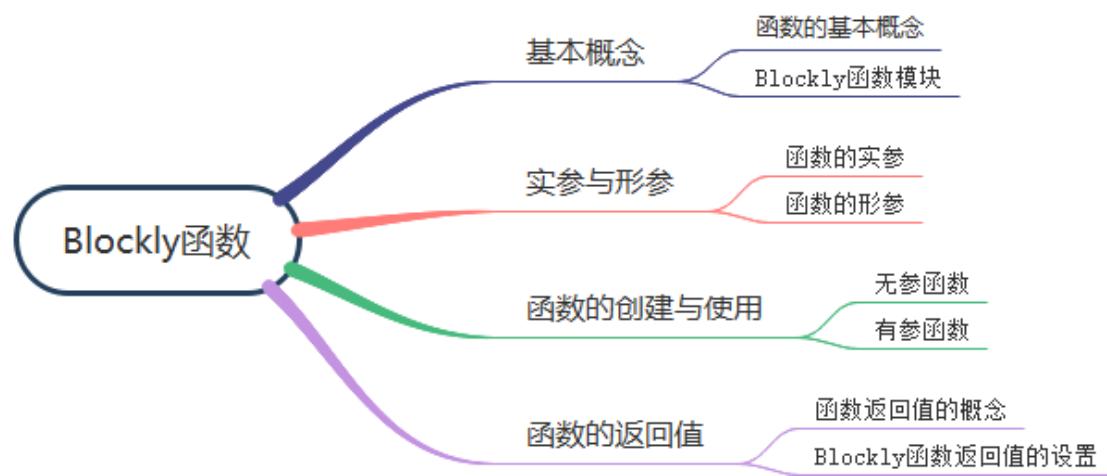
列表在计算机编程中应用的地方很多，大家有机会可以更多的去了解其他编程语言中的列表。

# 第6章 Blockly 函数

## 学习目标

1. 理解函数的概念。
2. 理解函数的实参与形参。
3. 掌握函数的创建与使用。
4. 理解函数的返回值。

## 知识图谱



之前的课程中，我们学习了程序设计的三种基本结构：顺序结构、选择结构、循环结构，但这还远远不够，要想实现一个完善的程序，我们需要模块化程序设计。在编写程序的时候，为了程序更简洁，我们往往喜欢把程序模块化。而模块化，就免不了要自己编写函数。在本章当中我们会对编写程序时使用的函数进行介绍，包括函数的基本概念、实参与形参、函数的创建与使用以及函数的返回值等内容。

### 6.1 基本概念

一个较大的程序一般应分为若干个程序模块，每一个模块用来实现一个特定的功能。所有的高级语言中都有子程序这个概念，用子程序实现模块的功能。比如在 C 语言中，子程序的作用是由函数完成的，一个 C 程序可由一个主函数和若干个函数构成，由主函数调用其他

函数，其他函数也可以相互调用，同一个函数可以被一个或多个函数调用任意多次。在 Blockly 中，也支持函数的定义和使用。



图 6-1 Blockly 函数模块

在程序设计中，常将一些常用的功能模块编写成函数，放在函数库中供公共选用，所以要善于利用函数，以减少重复编写代码的工程量，如图 6-1 即 Blockly 函数模块。

## 6.2 实参与形参

在 C 语言和别的语言中，函数的一个明显的特征就是使用时带括号 ()，必要的话，括号中还要包含数据和变量，我们称之为参数(Parameter)，参数是函数需要处理的数据。

函数的参数分为形参和实参两种。形参出现在函数定义中，在整个函数体内都可以使用，离开该函数则不能使用；实参出现在主调函数中，进入被调函数后，实参变量也不能使用。形参和实参的功能是作数据传送。发生函数调用时，主调函数把实参的值传送给被调函数的形参，从而实现主调函数向被调函数的数据传送。

形式参数（简称形参）是在用户自定义函数过程、子过程名后圆括号中出现的变量名，多个形参用逗号分隔。实际参数（简称形参）是在调用上述过程时，在过程名后的参数，其作用是将他们的数据（值或地址）传送给被调过程对应的形参变量。

形参可以是变量或者带有一对括号的数组名；实参可以是同类型的常数、变量、数组元素、表达式、数组名（带有一对圆括号）。

## 6.3 函数的创建与使用

根据参数的有无，可将函数简单的分为无参函数和有参函数。这对于没有接触过 C 语

言或其他编程语言的同学可能比较抽象，不过不必担心，接下来我们会通过 Blockly 向您详细的解释。

### 6.3.1 无参函数



图 6-2 Blockly 函数模块

图 6-2 是我们从 Blockly 工具箱中拖出的一个函数块，其中：对函数进行参数的设置，无参函数不需要用到此选项；在 这里输入函数的名称；在 在这里添加函数所实现的功能语句；而 则对函数的功能进行了描述，如图 6-3 所示。

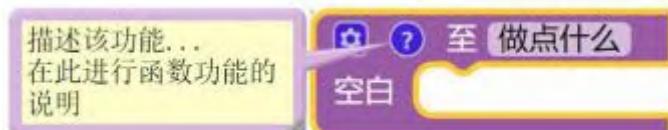


图 6-3 Blockly 函数模块功能描述

对工具箱中的 Blockly 有了简单的了解之后，尝试动手设计自己的函数。



图 6-4 无参函数

图 6-4 是一个简单的无参函数，它的函数名就叫“打印 ‘Hello World!’”，当你从工具箱拖动一个块在编辑区的同时，在工具箱中的函数选项卡中会生成一个对应的函数块，当再用到此函数时，就可以像使用其他工具箱中的块一样直接使用，如图 6-5。

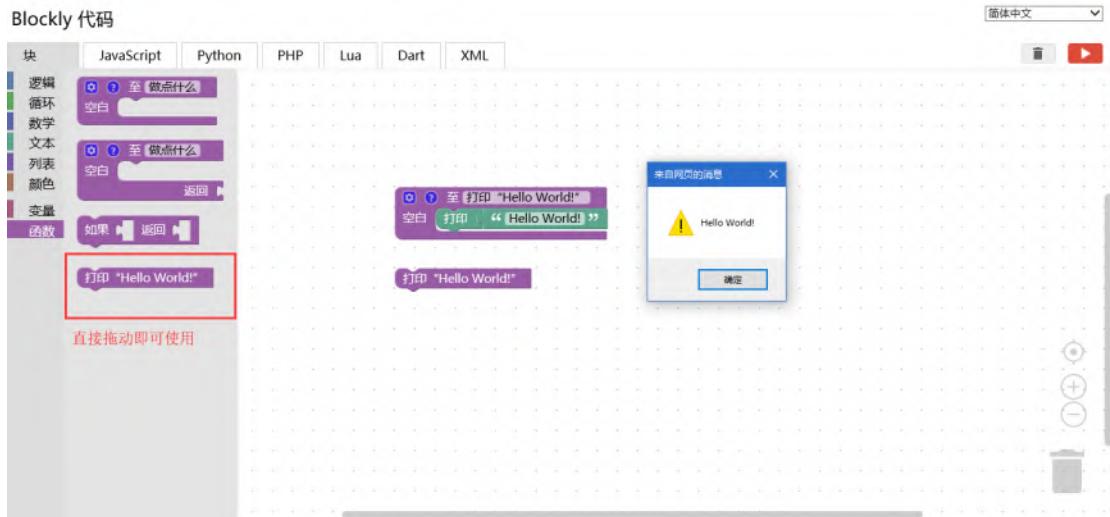


图 6-5 无参函数的使用

### 6.3.2 有参函数

与无参函数不同，有参函数需要在 中对参数进行设置，拖动 至右侧“输入”中，并对参数进行命名即可，如图 6-6 所示。

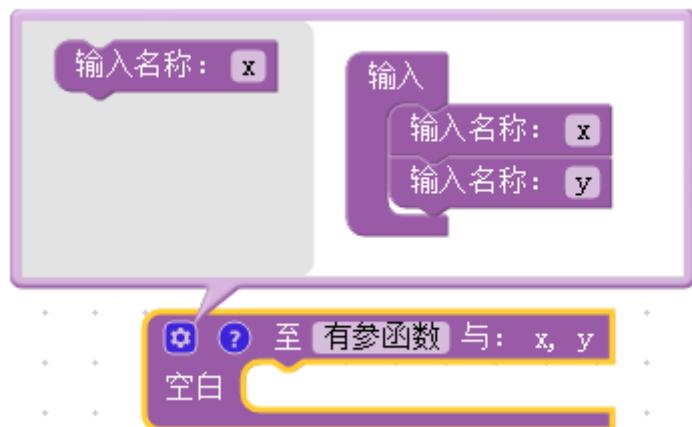


图 6-6 有参函数模块

在创建完成后，我们可以赋予该函数一个功能，在这里，我们令此函数的功能为打印参数 x 的值，如图 6-7 所示。该函数的使用方法同无参函数类似，区别在于使用时需要为参数赋值，如图 6-8 所示。



图 6-7 打印参数 x 函数模块

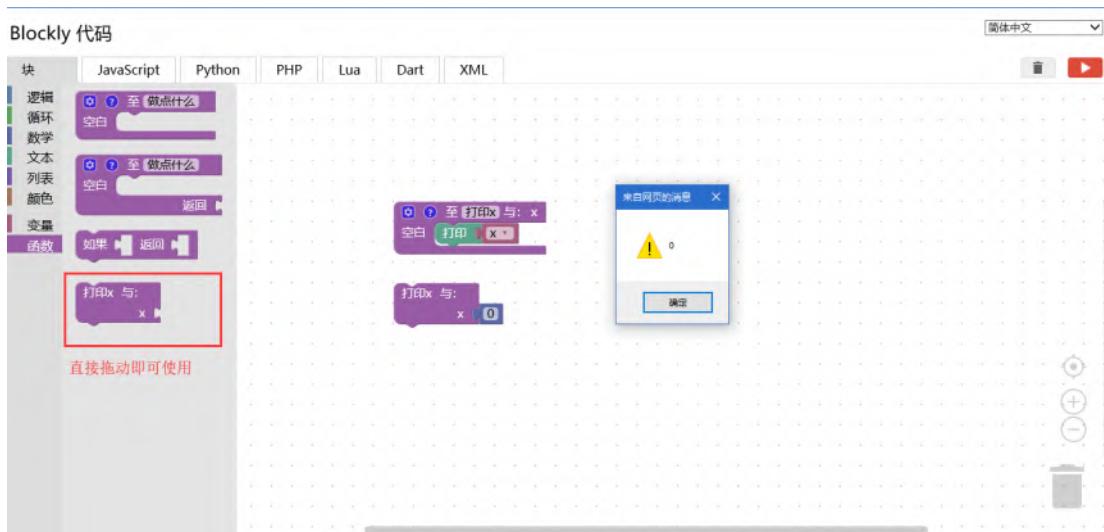


图 6-8 有参函数的使用

## 6.4 函数的返回值

函数的另外一个明显的特征就是返回值，既然函数可以进行数据处理，那就有必要将处理结果告诉我们，所以很多函数都有返回值，所谓的返回值就是函数的执行结果，如图 6-9 所示。

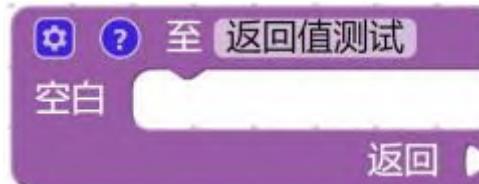


图 6-9 带返回值函数模块

当创建的函数需要返回值时，可直接从工具箱中拖动自带返回值的函数块，可见工具箱中生成的函数块左侧带有凸起的连接，如图 6-10 所示。



图 6-10 Blockly 返回值模块

这种方式生成的函数，只有当函数执行完成后才会返回值，如果在函数执行过程中就已

经产生了想要的结果，也可以拖动  结束正在执行的函数，并返回执行结果。

### 小提示

在使用时，只有当函数最初设计有返回值时才有返回值，否则只是简单的结束正在执行的函数，如图 6-11 所示。



图 6-11 注意有无“返回值”

例 1 设计一个求  $x, y$  中最大者的函数，名为  $\text{Max}(x, y)$ 。

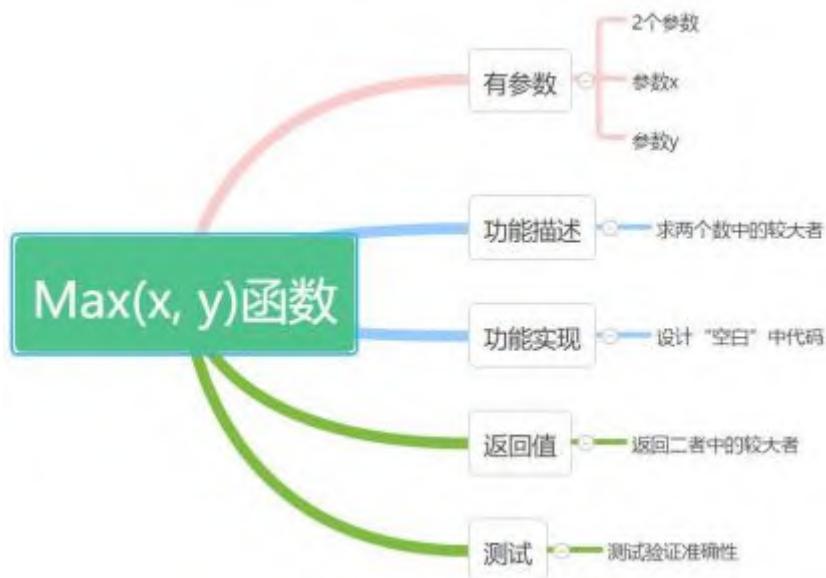


图 6-12  $\text{Max}(x, y)$  函数思维导图

可以按照思维导图，一步一步进行  $\text{Max}(x, y)$  函数的设计，如图 6-12 所示。

在进行程序设计时，无论你是简单的学习，还是进行复杂的开发，在动手之前，一定要对你所设计的程序有一个良好的规划。磨刀不误砍柴工，好的习惯很重要，它可以帮你提升编程水平，提高编程效率。



图 6-13 完成的  $\text{Max}(x, y)$  语言模块

我们设计好的模块语言如图 6-13 所示。当你设计完成后，剩下的就是测试验证程序结果了，测试时不需要很复杂，如果可以，最简单的就是使用 ，如图 6-14 所示。

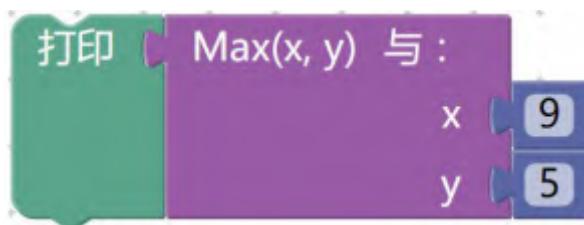
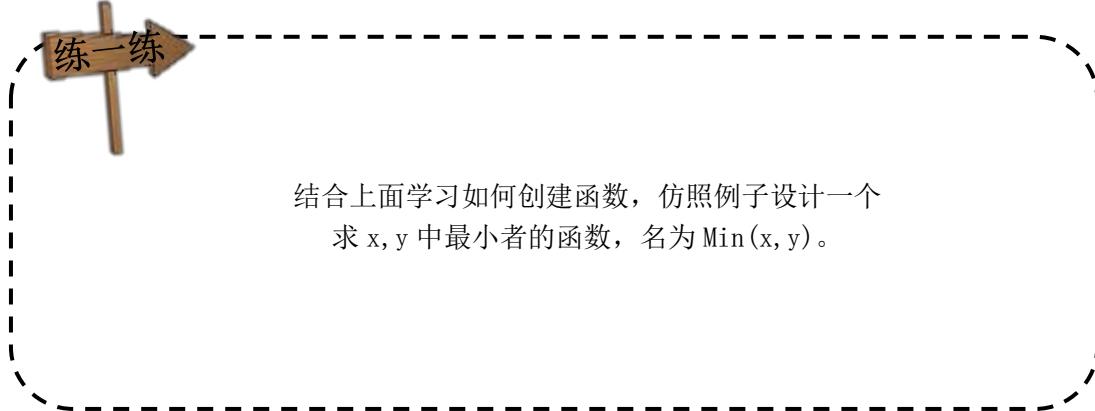


图 6-14 打印  $\text{Max}(x, y)$  结果



## 6.5 小试牛刀——游戏：池塘导师

池塘导师这款游戏分为 10 个关卡，为了让大家更好的理解 Blockly 游戏，我将在这里带领大家一一突破每一关，在游戏中体会 Blockly 编程的乐趣，进而掌握基础程序语言的运用。

游 戏 地 址 如 下：

<http://cooc-china.github.io/pages/blockly-games/zh-hans/pond-tutor.html?lang=zh-hans>

### 游戏规则：

- ① 我们需要通过控制代码来让玩家发射加农炮击败对手；
- ② 代码主要由循环结构和逻辑判断组成；

③ 点击“运行程序”按钮后程序就会执行右侧的代码。当玩家将对手的血槽攻击为零后即可击败对手，并完成每个关卡。

#### 通关详解：

**第1关：**调整距离和角度，击败红色角色。



图 6-15 第一关示例与答案

**第2关：**编写代码，调整距离和角度参数，击败红色角色。

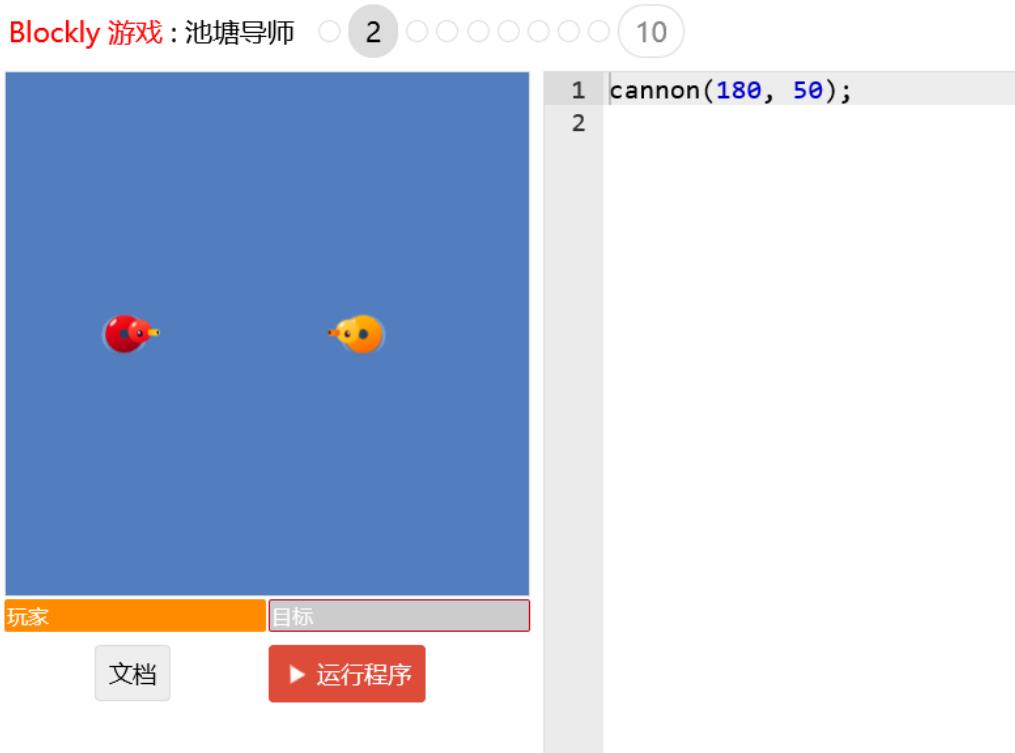


图 6-16 第二关示例与答案

**第 3 关:** 加入了循环结构，循环使用“cannon 模块”即可。

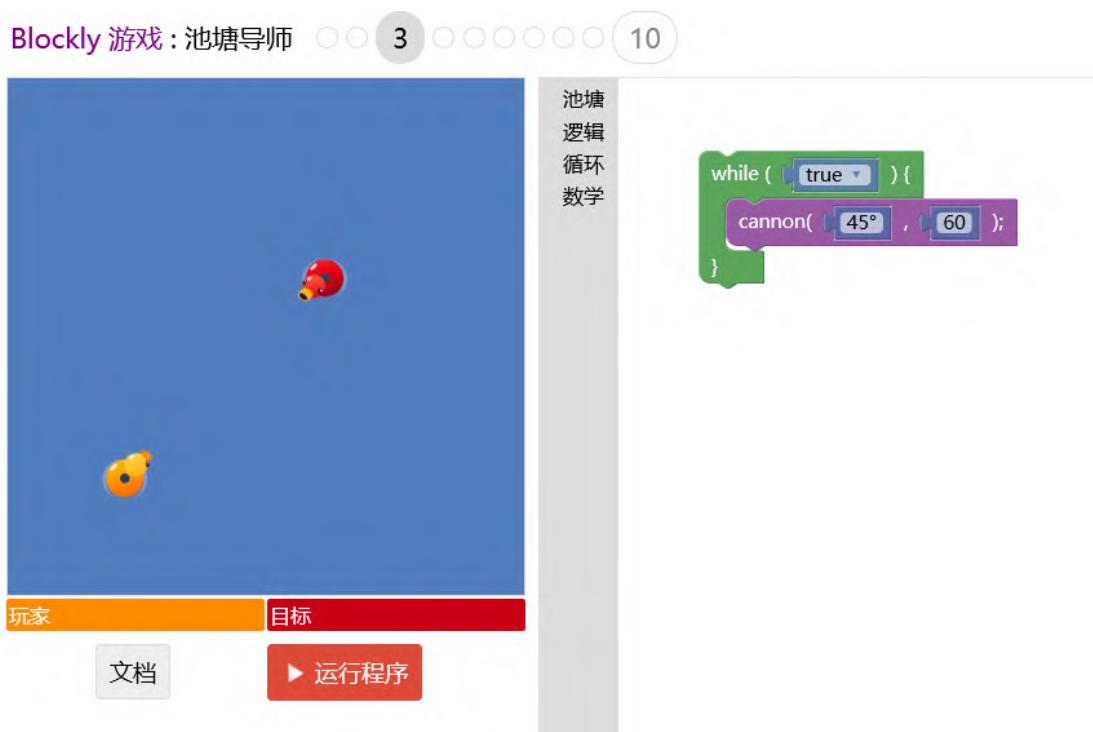


图 6-17 第三关示例与答案

**第 4 关:** 编写代码，加入循环代码，调整距离和角度参数，击败红色角色。



图 6-18 第四关示例与答案

**第 5 关:** 红色角色来回移动，因此很难被攻击。“scan”表达式返回特定位置具体范围内的对手。



图 6-19 第五关示例与答案

**第 6 关:** 编写代码，加入“scan”，调整参数，击败红色角色。



图 6-20 第六关示例与答案

**第 7 关:** 新增“swim”模块，用来调整角色与目标间的距离，调整参数，使角色游向红色角色。



图 6-21 第七关示例与答案

**第 8 关:** 编写代码, 控制角色游向红色目标。



图 6-22 第八关示例与答案

**第 9 关:** 角色与红色目标间距离过远, 加入选择结构, 缩短角色与目标之间的距离, 然后击败目标。



图 6-23 第九关示例与答案

第 10 关： 编写代码，利用所学所有模块，击败红色角色。



图 6-24 第十关示例与答案

## 6.6 本章练习

1. 写一个判断素数的函数，在主函数输入一个整数，输出是否是素数。
2. 设计定义一个自己的工具块。

## 6.7 课外拓展

### 可视化编程

可视化编程，亦即可视化程序设计：以“所见即所得”的编程思想为原则，力图实现编程工作的可视化，即随时可以看到结果，程序与结果的调整同步。

可视化编程是与传统的编程方式相比而言的，这里的“可视”，指的是无须编程，仅通过直观的操作方式即可完成界面的设计工作，是目前最好的 Windows 应用程序开发工具。

可视化编程语言的特点主要表现在两个方面：一是基于面向对象的思想，引入了类的概念和事件驱动；二是基于面向过程的思想，程序开发过程一般遵循以下步骤，即先进行界面的绘制工作，再基于事件编写程序代码，以响应鼠标、键盘的各种动作。

编程十问：

(1) 什么是可视化程序设计?

可视化(Visual)程序设计是一种全新的程序设计方法,它主要是让程序设计人员利用软件本身所提供的各种控件,像搭积木式地构造应用程序的各种界面。

(2) 可视化程序设计有哪些优点?

可视化程序设计最大的优点是设计人员可以不用编写或只需编写很少的程序代码,就能完成应用程序的设计,这样就能极大地提高设计人员的工作效率。

(3) 能够进行可视化程序设计的集成开发环境有哪些?

能进行可视化程序设计的集成开发环境很多,比较常用的有微软的 Visual Basic、Visual C++、中文 Visual Foxpro、Borland 公司的 Delphi 等。

(4) 可视化程序设计中有哪些基本概念?

主要的几个基本概念有表单、组件、属性、事件、方法等。

(5) 什么是表单(Form)?

表单是指进行程序设计时的窗口,我们主要是通过在表单中放置各种部件(如命令按钮、复选框、单选框、滚动条等)来布置应用程序的运行界面。

(6) 什么是组件?

所谓组件,就是组成程序运行界面的各种部件,如:命令按钮、复选框、单选框、滚动条等。

(7) 什么是属性?

属性就是组件的性质。它说明组件在程序运行的过程中是如何显示的、组件的大小是多少、显示在何处、是否可见、是否有效……

(8) 属性可以分成哪几类?

属性可分成三类,设计属性:是在进行设计时就可发挥作用的属性;运行属性:这是在程序运行过程中才发挥作用的属性;只读属性:是一种只能查看而不能改变的属性。

(9) 什么是事件?

事件就是对一个组件的操作。如用鼠标点击一个命令按钮,在这里,点击鼠标就称为一个事件(Click 事件)。

(10) 什么是方法?

方法就是某个事件发生后要执行的具体操作,类似以前的程序。例如当我们用鼠标单击“退出”命令按钮时,程序就会通过执行一条命令而结束运行,命令的执行过程就叫方法。

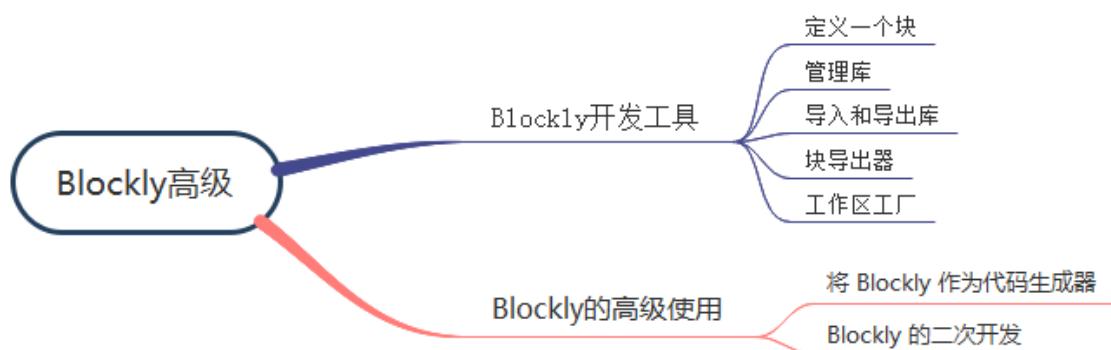
(来源: 360 百科  
<https://baike.so.com/doc/6125971-6339130.html>)

# 第7章 Blockly高级

## 学习目标

1. 了解 Blockly 开发工具的使用。
2. 了解如何自定义 Blockly 模块。
3. 了解 Blockly 作为代码生成器的功能。
4. 了解 Blockly 的二次开发。

## 知识图谱



在本章中，我们需要使用 Blockly 工厂、开发工具，自定义一个模块，并对其进行管理。

我们将学习到 Blockly 的高级使用方法，如何利用 Blockly 导出自己所需要的代码块。在学习完本章知识内容后，我们将可以利用 Blockly 开发工具设计自己所需的模块，更灵活地进行编程。

### 7.1 Blockly 开发工具

在前几章的学习中，每章结尾我们都有小游戏环节。每个小游戏虽然是可视化编程，和我们学习的 Blockly 很像，但是又有所不同，而这些不同由何而来？这就是我们本章所讲的重点，Blockly 开发工具(Blockly Developer Tools)，通过它用户可以自定义新块，这使得 Blockly 可扩展性大大提升，同时也是 Blockly 的灵活和强大之处。



图 7-1 游戏：迷宫游戏模块

本节面向希望在 **Blockly** 中创建新块的开发人员。它的基本要求是，有一个可以编辑的 **Blockly** 的本地副本，大体上熟悉了 **Blockly** 的使用，并且对 **JavaScript** 有一个基本的理解。

**Blockly** 带有大量的预定义块，从数学函数到循环结构的一切，然而为了与外部应用程序接口，必须创建自定义块以形成 API。例如，当创建绘图程序时，可能需要创建“绘制半径 R 的圆”块。而在大多数情况下，最简单的方法是找到一个已经存在的真正相似的块，复制它，并根据需要修改它。

第一步是创建一个块：指定其形状，字段和连接点。使用 **Blockly DeveloperTools** 是编写此代码的最简单的方法，或者，可以在学习 API（应用程序编程接口）之后手动编写该代码，高级块可以响应于用户或其他因素而动态地改变它们的形状。

第二步是创建生成器代码以将新块导出为编程语言（例如 **JavaScript**, **Python**, **PHP**, **Lua** 或 **Dart**）。为了生成既干净又正确的代码，必须注意给定语言的操作列表顺序，创建更复杂的块需要使用临时变量和调用函数，当输入使用两次并需要缓存时，这是尤为重要的。**Blockly** 开发工具是一个基于 **Web** 的开发工具，可自动完成 **Blockly** 配置过程的各个部分，包括创建自定义块，构建工具箱和配置 **Web Blockly** 工作区。

使用该工具的 **Blockly** 开发者进程包括三个部分：

1. 使用块工厂和块导出器创建自定义块。
2. 使用 **Workspace Factory** 构建工具箱和默认工作空间。
3. 使用 **Workspace Factory** 配置工作空间（当前是仅限 **Web** 的功能）。

### 7.1.1 定义一个块



图 7-2 Block Factory

定义一个块需要使用到 Blockly 开发工具中的块工厂（Block Factory），块工厂主要分为三个区域：

- (1) 编辑区：对新增块进行设计和编辑
- (2) 预览区：对编辑区编辑的块进行实时预览
- (3) 代码区：代码区分为两个部分 Language code 和 Generator stub，其中 Language code 区指定和控制新增块所呈现的形状，Generator stub 区负责新增块所要生成的代码。

在编辑区的左侧，可以看到 4 个基本块，Input、Field、Type 和 Colour，它们是四个原料库，使用者可以从这些库中获取所需要的任意“原料”，来合成自己的新块。



图 7-3 Block Factory 基本块

先从最简单的介绍，颜色（Colour）块，它默认定义了九种基本颜色，直接将你想要的颜色拖到右侧，拼接到对应的 colour 的凹槽，便可立即在预览区看到新块的颜色。

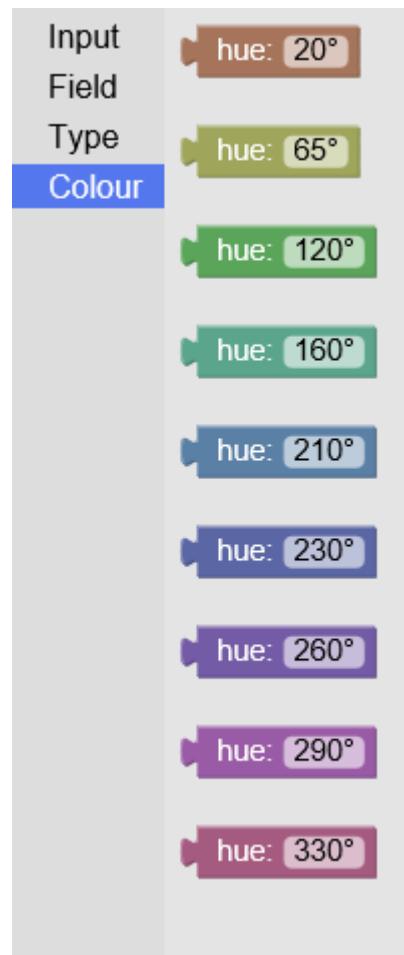


图 7-4 颜色块



图 7-5 预览区效果

如果默认色彩中没有你想要的颜色，可以拖动任意色彩块到编辑区拼接完成后，点击色块中的数字，在色块的下方出现一个圆形的调色盘，转动调色盘，选择你喜欢的颜色。

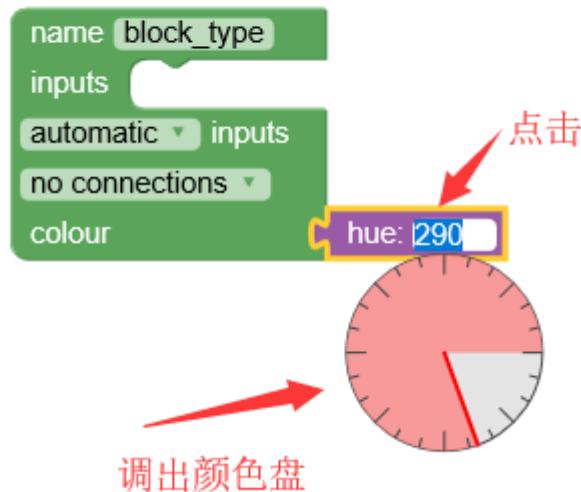


图 7-6 调色盘

在 Blockly 中，同一类型的块通常采用相同颜色，所以新块的颜色选择不能仅凭喜欢，还需要前后兼顾。

一个新块不仅需要有颜色，还需要与其他块进行衔接，这就需要设计新增块的输入和输出，它们将决定新增块的功能、属性和类别。

接下来看一看输入 (Input)，这是新增块与其他块连接的接口之一。

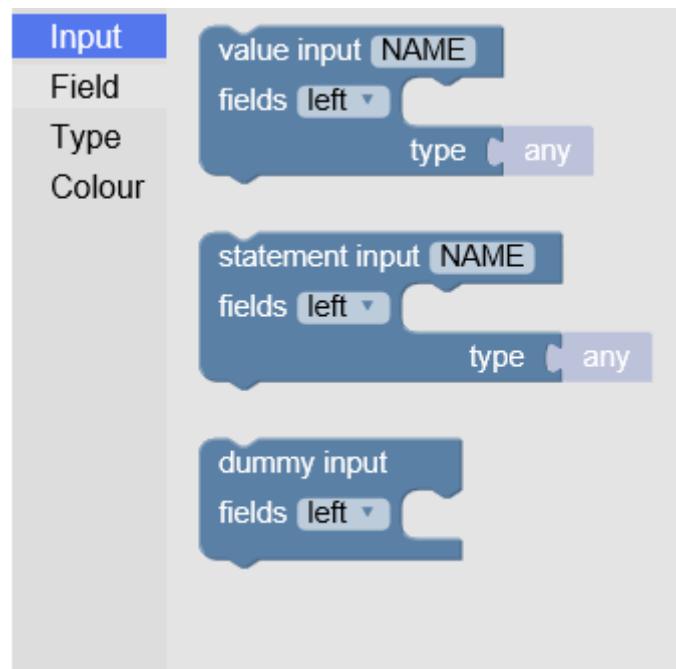


图 7-7 Input 模块

输入可以分为三种类型：值输入 (value input)，声明输入 (statement input)，模拟输

入(dummy input)。首先以值输入为例，拖动值输入至右侧与 Inputs 连接，可看到预览区新增块多了一个凹槽：

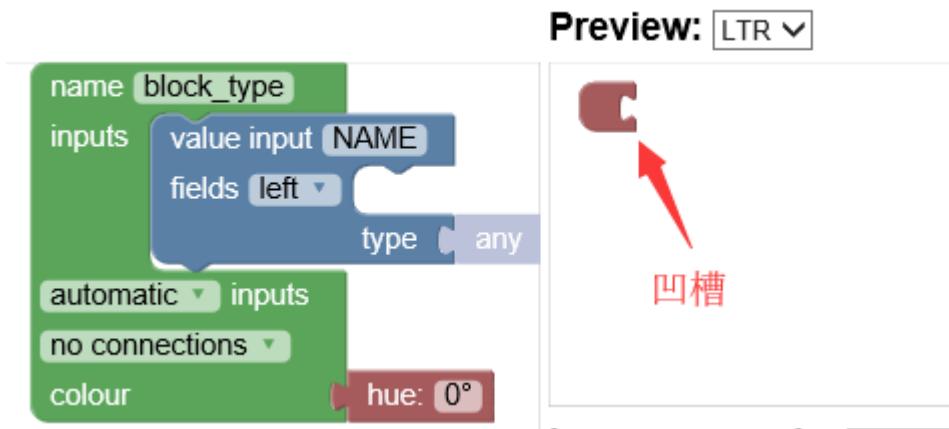


图 7-8 单个 Input 模块效果图

根据需要，使用者还可以添加多个输入值，但要注意多个输入值的名字不能相同，否则会出现警告，而且在后续调用的时候，也会冲突报错，新块名字也是如此，不能与其他块同名，就好比如果班里有两个学生名字一样，那老师点名提问的时候就有可能出现两个同学同时起立的尴尬。

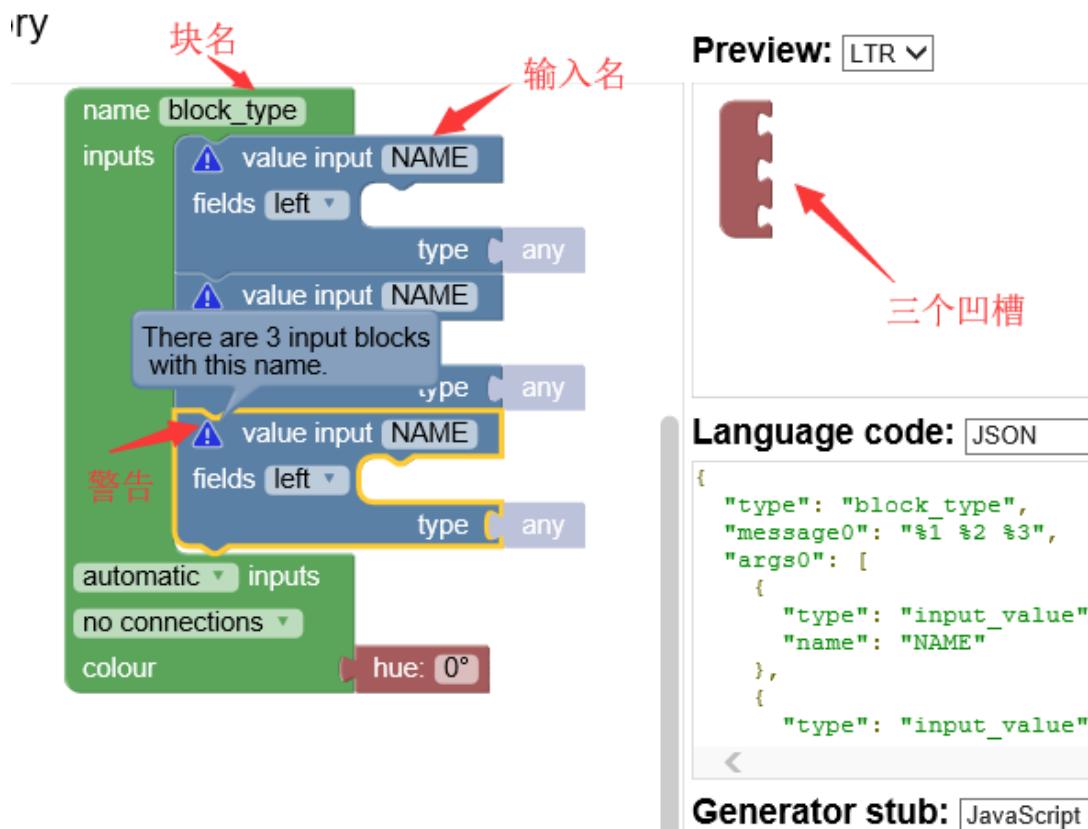


图 7-9 多个 Input 模块效果图

在值输入中还可以添加域(field)，比如加入最简单的文本域，即可在输入中提示相对应的文本，域中的下拉选择框可设置文本的对齐方式。

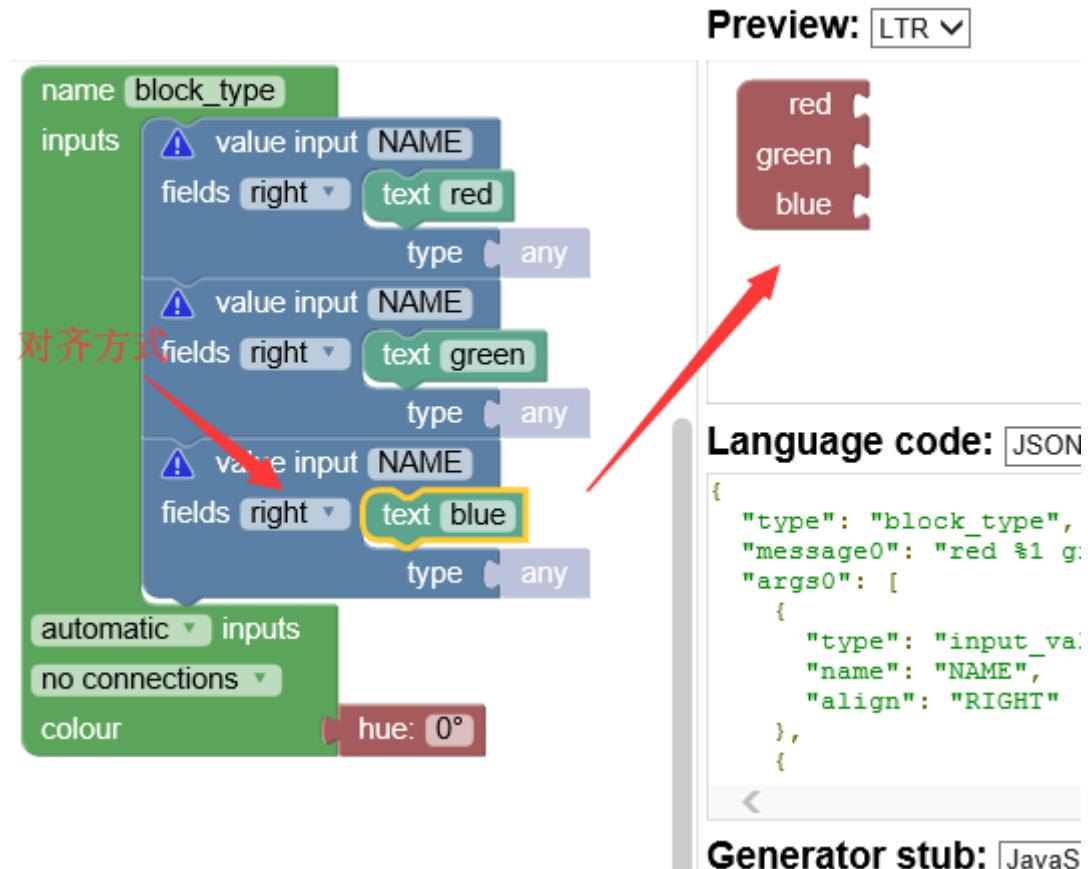


图 7-10 Input 模块对齐方式

这些设置完毕，选择新块的输入方式，外接式和内嵌式：

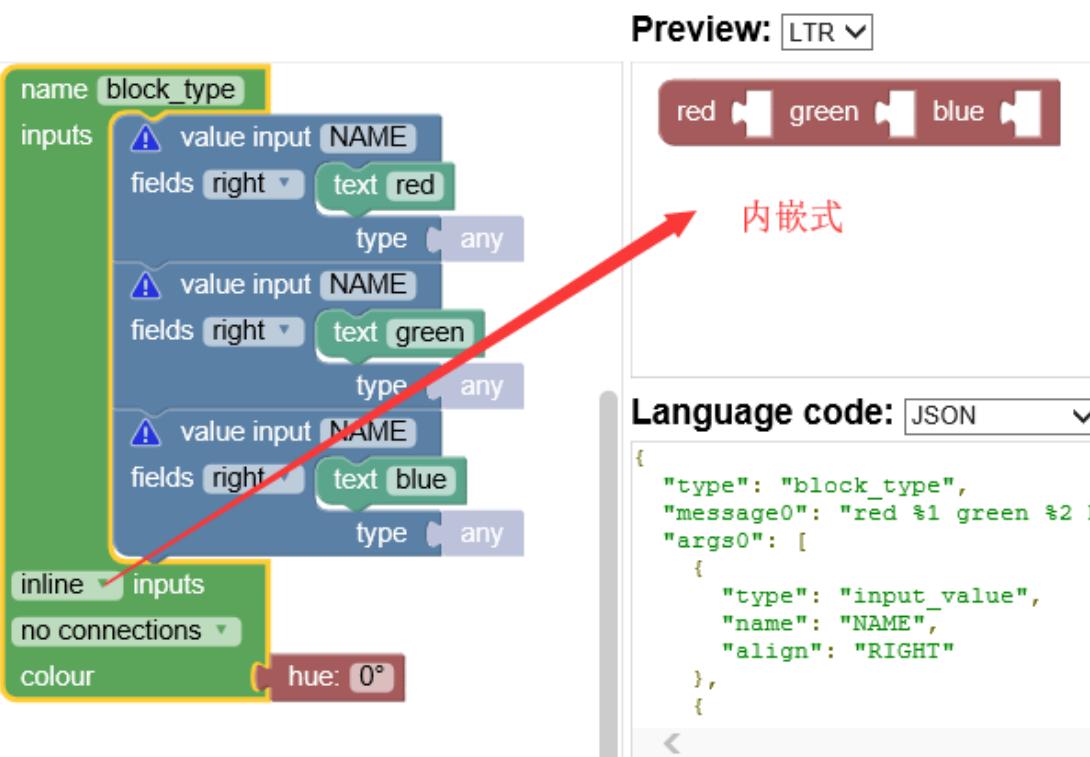


图 7-11 外接式与内嵌式效果图

有了输入之后，别的块就可以很容易的通过凹槽加入到新块了，但是，这时另外一个值

得考虑的问题又出现了，怎样将新增块加入到其他的块之中呢？我们有五种选择：

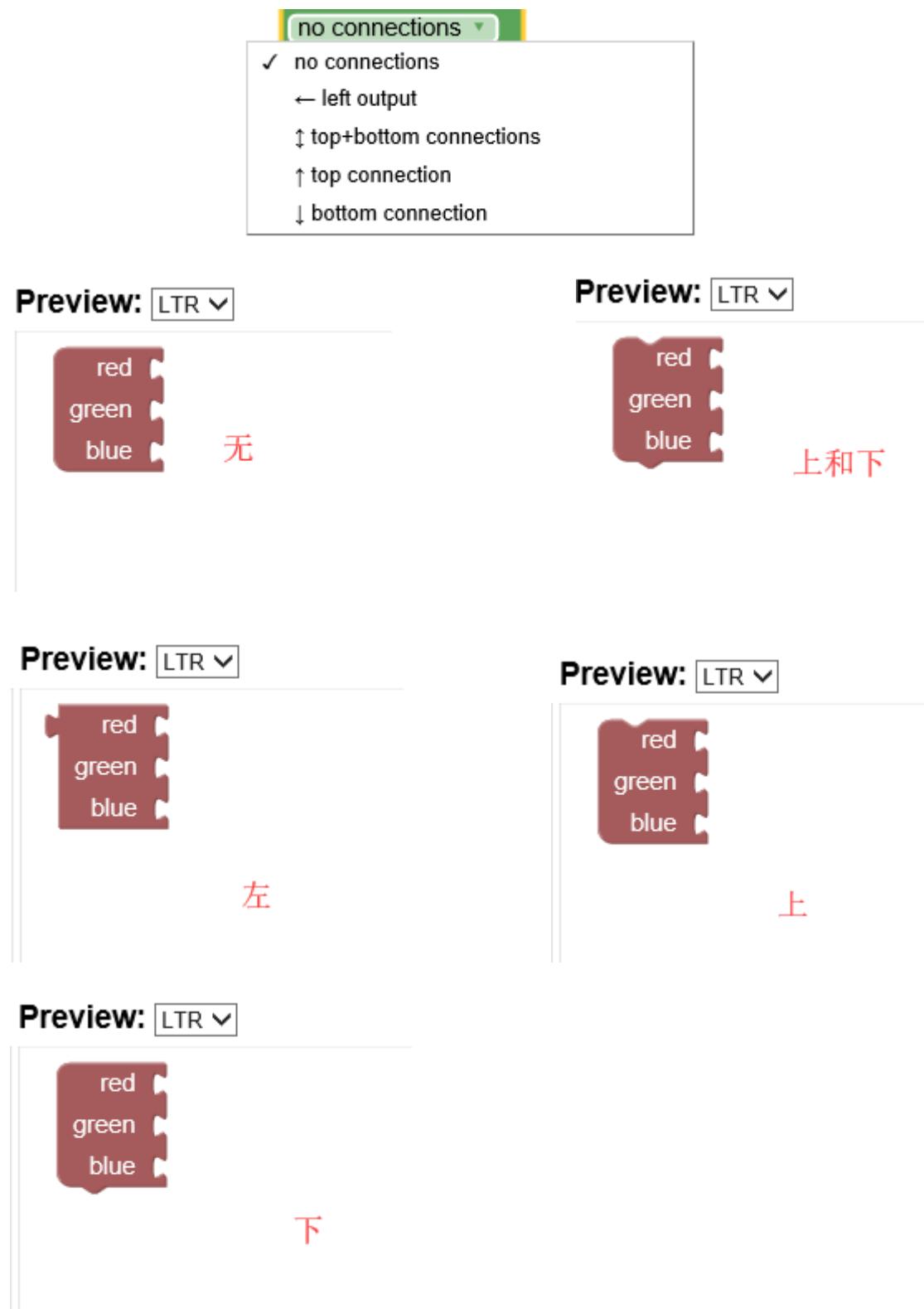


图 7-12 接口效果图

看完值输入之后，再一起来看一下另一个常用的输入类型，声明输入(statement input)，它通常用作条件控制和循环控制。

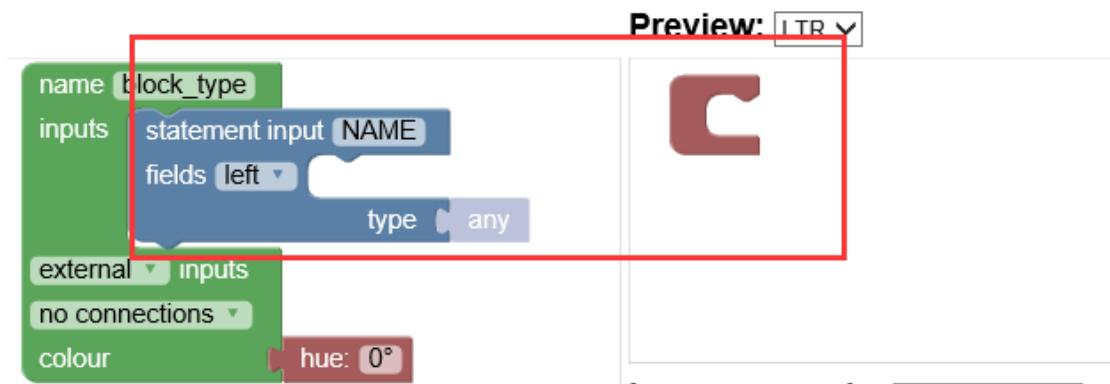


图 7-13 声明输入

使用值输入和声明输入，可以很容易的设计出编程中常用的条件语句和循环语句：

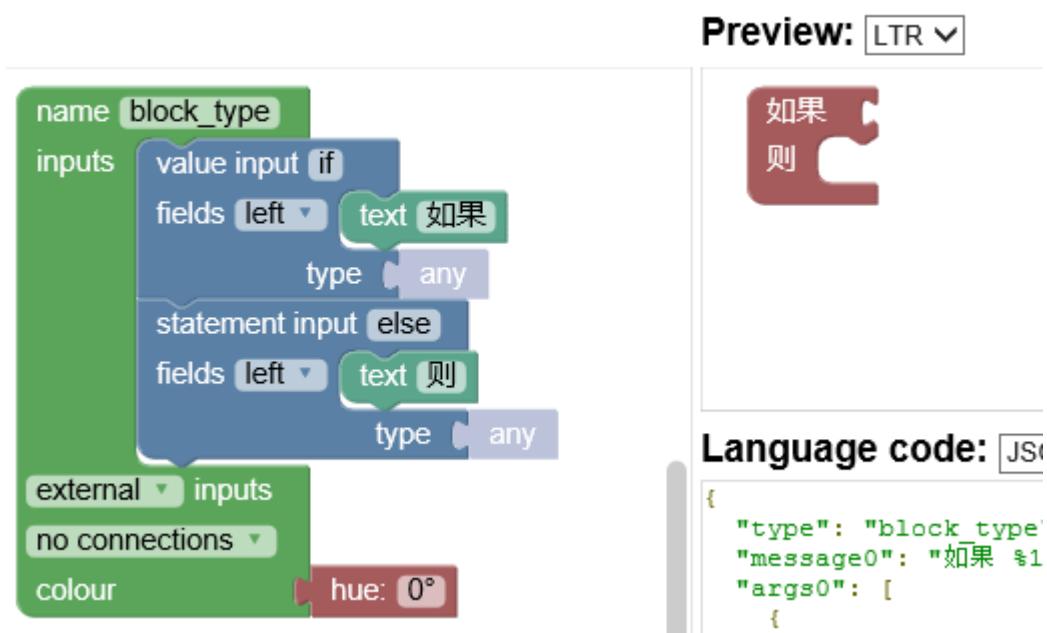
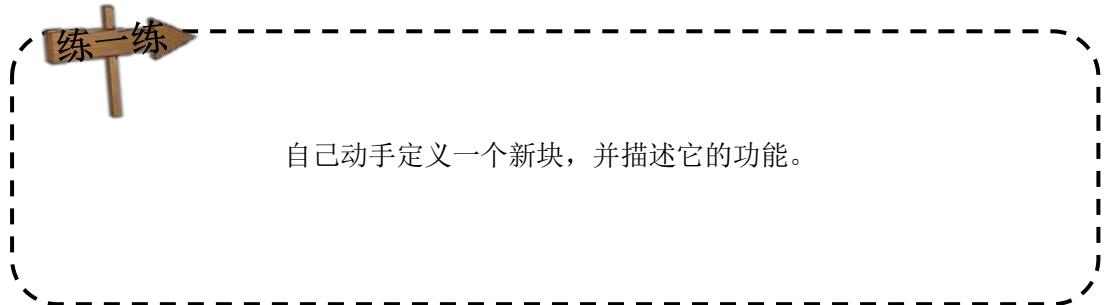


图 7-14 设置条件语句



图 7-15 设置循环语句



### 7.1.2 管理库

块由其名称引用，因此要创建的每个块都必须具有唯一的名称。 用户界面强制执行此操作，并在您“保存”新块或“更新”现有块时清除。

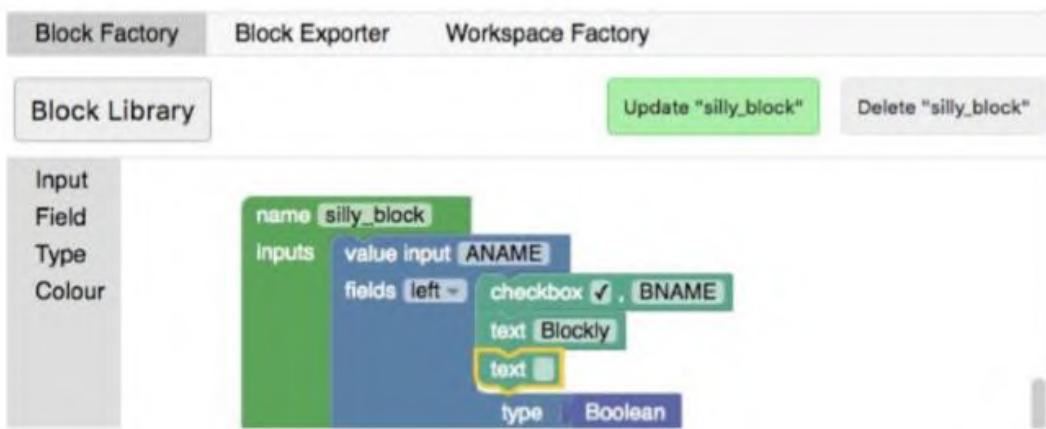
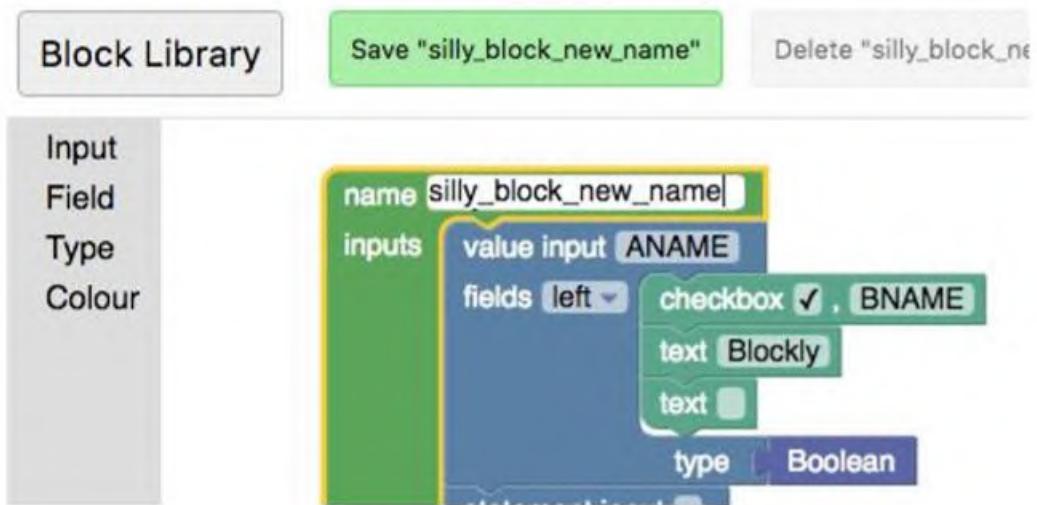


图 7-16 块的保存与更新

您可以在先前保存的块之间切换，或通过单击库按钮创建新的空块。更改现有块的名称是快速创建具有类似定义的多个块的另一种方法。

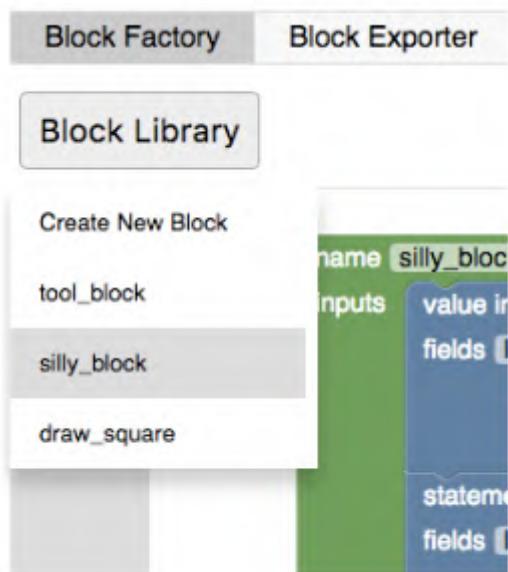


图 7-17 块的创建

### 7.1.3 导入和导出库

块被保存到浏览器的本地存储，清除浏览器的本地存储将删除您的块。要无限期保存块，您必须下载库。您的块库将下载为可导入的 XML 文件，以将您的块库设置为下载文件时的状态。请注意，导入块库将替换当前的库，因此您可能需要先备份导出。

导入和导出功能也是维护和共享不同组自定义块的推荐方式。

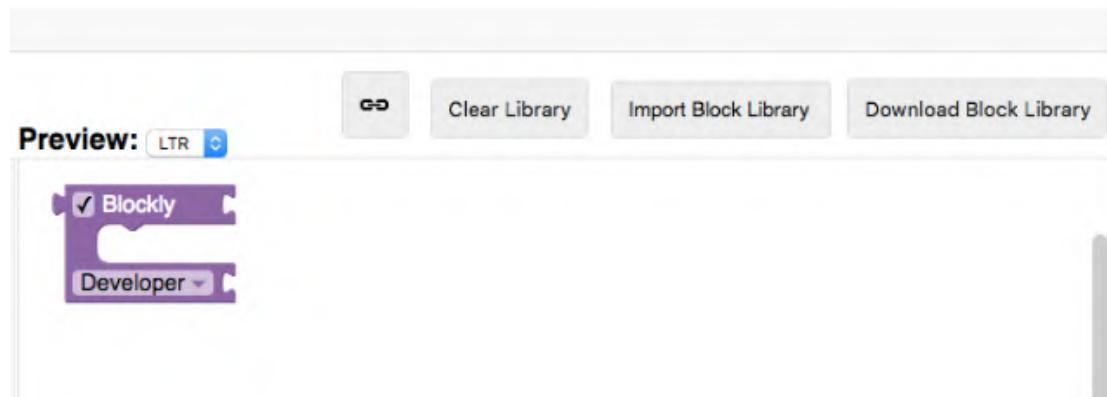


图 7-18 块的导入导出

### 7.1.4 块导出器

如果你设计了块，并且想要在应用程序中使用它们的时候，可以在块导出器重完成块定

义和生成器的导出。

存储在块库中的每个块都将显示在块选择器中。单击块以选择或取消选择要导出的块。如果要选择库中的所有块，请使用“选择”→“所有存储在块库”选项。如果使用“工作区出厂”选项卡构建了工具箱或配置了工作区，则还可以通过单击“选择”→“在工作区工厂中使用”选择所有使用的块。

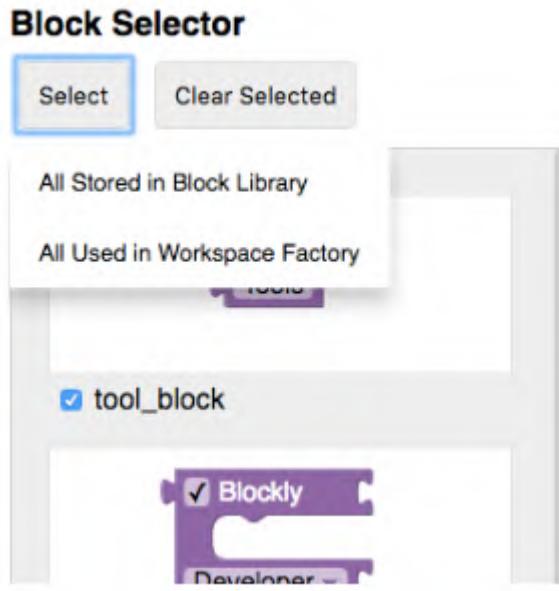


图 7-19 块选择器

导出设置允许您选择要定位的生成语言，以及是否需要所选块的定义。选择这些文件后，点击“导出”即可下载文件。

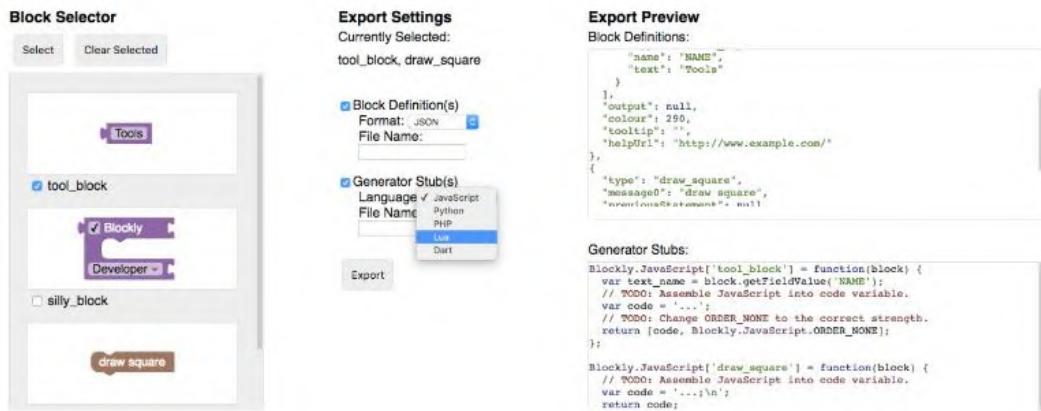


图 7-20 块导出相关设置

## 7.1.5 工作区工厂

工作区工厂可以方便地配置工具箱和工作区中的默认块组。您可以使用“工具箱”和

“工作区”按钮在编辑工具箱和起始工作区之间切换。



图 7-21 工作区工厂

### (1) 构建工具箱

此选项卡有助于构建工具箱的 XML，该材料假定使用者熟悉工具箱的功能，如果您在此处要编辑工具箱的 XML 时，可以通过单击“加载到编辑”加载它。

### (2) 没有类别的工具箱

如果您有几个块，它们没有任何类别，想要显示它们的时候，只需将它们拖动到工作区中，您将看到您的块出现在工具箱的预览中。

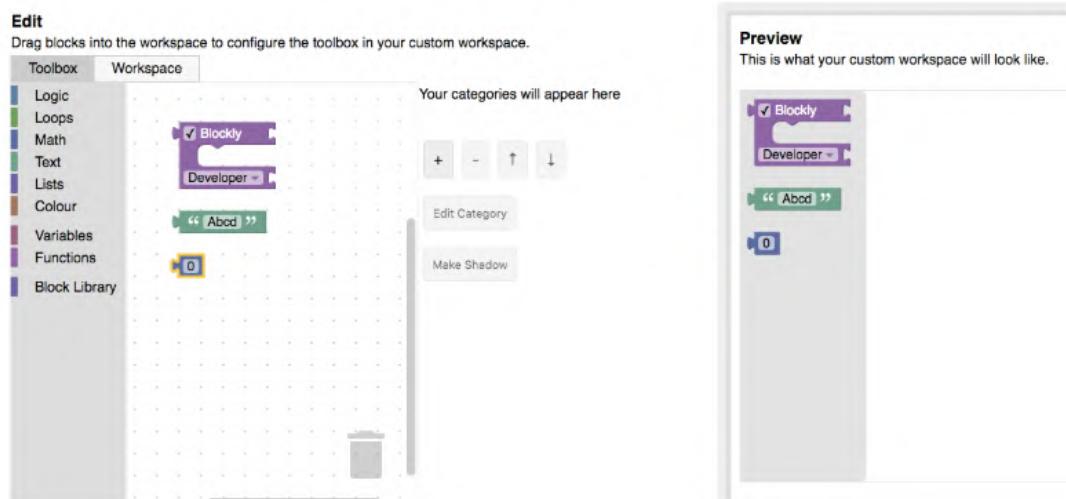


图 7-22 工具箱预览

### (3) 有类别工具箱

如果你想要显示块类别，点击“+”按钮，并选择下拉项目为新类别。这将向您的类别列表中添加一个类别，您可以选择和编辑。选择“标准类别”以添加单个标准块类别（逻辑，循环等）或“标准工具箱”以添加所有标准块类别。使用箭头按钮重新排序类别。

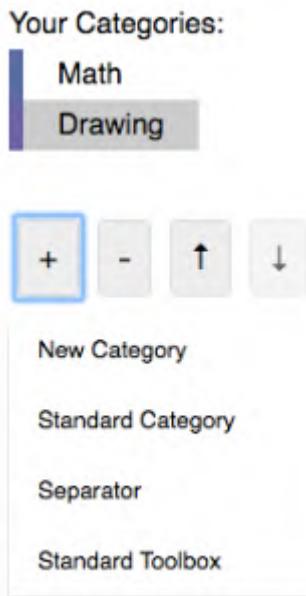


图 7-23 显示块类别

要更改所选类别名称或颜色，请使用“编辑类别”下拉菜单。将块拖动到工作区中将将其添加到所选类别。

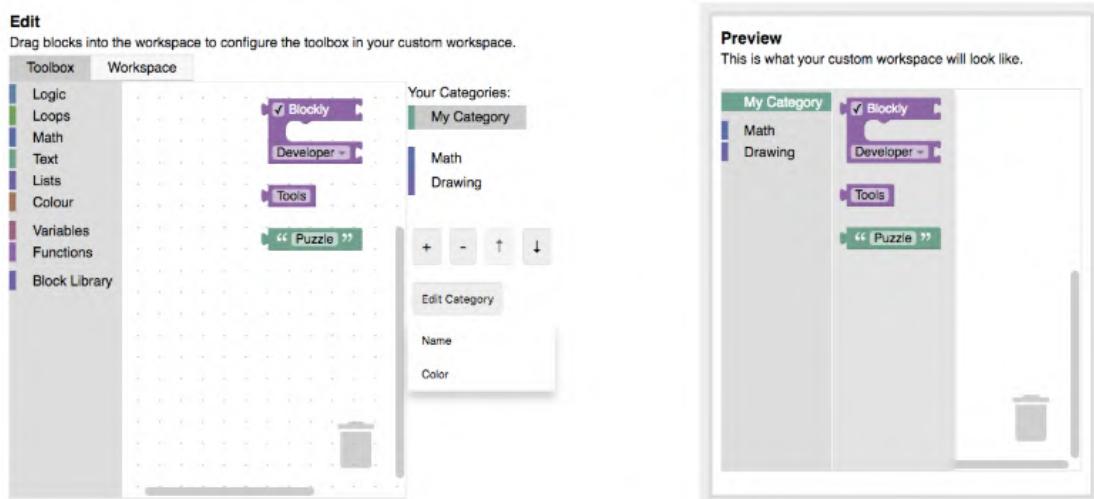


图 7-24 块的编辑

#### (4) 选择工作区选项

为配置选项设置不同的值，并在预览区域中查看结果。启用网格或缩放会显示更多配置选项。此外，切换到使用类别通常需要更复杂的工作空间：当您添加第一个类别时，会自动添加垃圾桶和滚动条。

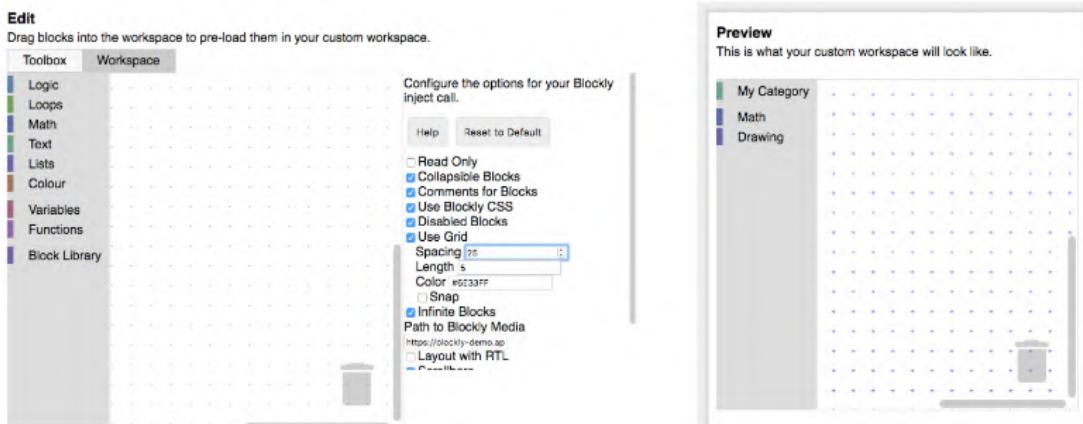


图 7-24 工作区选项

#### (5) 将预加载块添加到工作区

这是可选的，但如果要在工作空间中显示一组块，则可能需要：

- 当应用程序加载时显示
- 当触发事件（提高级别，单击帮助按钮等）时显示

将块拖动到编辑空间中，可以在预览区中查看它们。您可以创建块组，禁用块，并在选择某些块时创建阴影块。

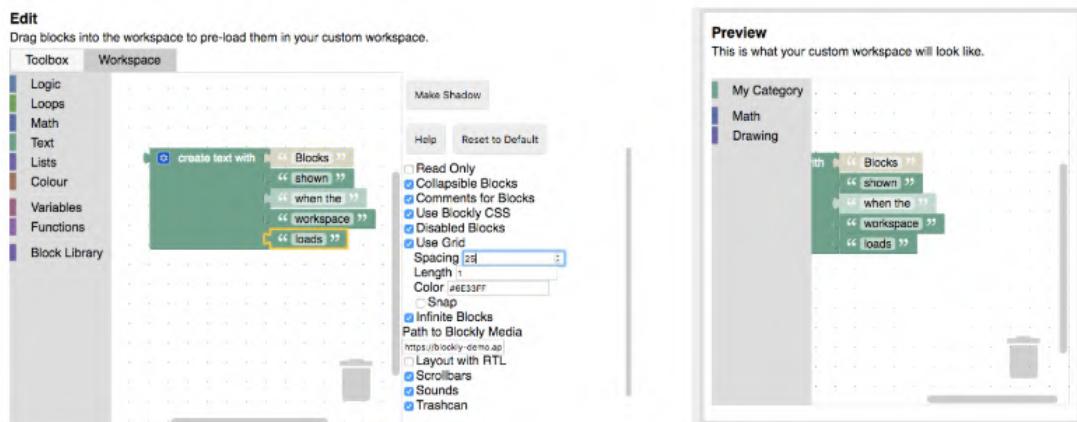


图 7-25 工作区选项

#### (6) 导出

工作区工厂提供以下导出选项：

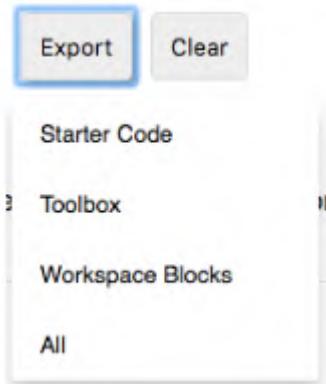


图 7-26 导出选项

- ✧ Starter Code: 生成 html 和 JavaScript 以注入您的自定义 Blockly 工作区。
  - ✧ 工具箱生成 XML 以指定您的工具箱。
  - ✧ 工作区块生成可以加载到工作区中的 XML。
- ( 7 ) 更多创建自定义块的信息，可参考 Google Blockly:<https://developers.google.com/blockly/guides/create-custom-blocks/overview>

## 7.2 二次开发案例——Simple Blockly

其实从 Github 上下载下来的 `blockly-master` 中很多文件并不是必不可少的，通过 Simple Blockly 案例，梳理 Blockly 的文件目录，实现精简版的 Blockly。

### 7.2.1. 准备工作

#### (1) HTML 知识准备

首先需要了解一点 HTML 的基础知识，比如最小的 HTML 文档：

```
1. <!DOCTYPE> // 定义文档的类型
2. <html> // 定义一个 HTML 文档
3. <head> // 定义关于文档的信息
4.   <meta charset="utf-8"> // 对于中文网页，声明编码，否则会出现乱码
5.   <title>文档标题</title> // 定义文档的标题
6. </head>
7.
8. <body> // 定义文档的主体
9.   文档内容.....
10. </body>
11. </html>
```

除了上述最小 HTML 文档中包含的内容，我们会用到还有：

Script 标签，定义客户端脚本

```
1. <script src="helloworld.js"></script>
2.     Link 标签，链接样式表
3. <link rel="stylesheet" type="text/css" href="theme.css">
```

根据上述内容，完成下面的小案例，要求：

- \* 在最小 HTML 文档的基础上，显示 HelloWorld；
- \* 通过内部 js 脚本控制弹框输出 HelloWorld；
- \* 通过外部 js 脚本控制弹框输出 HelloWorld；

```
1. /**示例代码*/
2. <!DOCTYPE>
3. <html>
4. <head>
5.   <meta charset="UTF-8">
6.   <title>HelloWorld</title>
7.   <!--<script>
8.     alert("Hello World!");
9.   </script>-->
10.  <script src="alert_helloworld.js"></script>
11. </head>
12. <body>
13. Hello World!
14. </body>
15. </html>
```

(2) 下载 Blockly-master 文件，留作备用

### 7.2.2. 动手实践

(1) 新建文件目录

新建一个文件夹，重命名为 SimpleBlockly，并在文件夹中新建 css，js 两个子文件夹，将 Blockly-master/Demos/Code 中的 index.html 复制到 SimpleBlockly 根目录下，如下图：

名称	修改日期	类型	大小
css	2018/5/23 14:41	文件夹	
js	2018/5/23 14:42	文件夹	
index.html	2018/3/10 5:55	Chrome HTML D...	12 KB

图 7-27 index 文件

(2) 导入、调整文件

使用 notepad++ 打开 index.html 文件，修改 head 部分的内容，并将 head 部分的涉及的内容，从 blockly-master 中复制添加到 SimpleBlockly 中的 css 和 js 文件夹中。

- \* 将 title 修改为 Simple Blockly;
- \* 将 blockly-master/demos/code 文件中的 style.css 文件移至 css 文件夹中，并将 href 引号中对应的字符串改为"css/style.css";
- \* storage.js 直接删除即可，它和 Google 的 Cloud Storage 有关，案例中用不会涉及这部分功能；
- \* 将 blockly-mater 文件中的 blockly\_compressed.js 文件移至 js 文件夹中，并将 src 引号中对应的字符串改为"js/blockly\_compressed.js";
- \* 将 blockly-mater 文件中的 blockls\_compressed.js 文件移至 js 文件夹中，并将 src 引号中对应的字符串改为"js/blockls\_compressed.js";
- \* 将 blockly-mater 文件中的 javascript\_compressed.js 文件移至 js 文件夹中，并将 src 引号中对应的字符串改为"js/javascript\_compressed.js";
- \* 将 blockly-mater 文件中的 python\_compressed.js 文件移至 js 文件夹中，并将 src 引号中对应的字符串改为"js/python\_compressed.js";
- \* 将 blockly-mater 文件中的 php\_compressed.js 文件移至 js 文件夹中，并将 src 引号中对应的字符串改为"js/php\_compressed.js";
- \* 将 blockly-mater 文件中的 lua\_compressed.js 文件移至 js 文件夹中，并将 src 引号中对应的字符串改为"js/lua\_compressed.js";
- \* 将 blockly-mater 文件中的 dart\_compressed.js 文件移至 js 文件夹中，并将 src 引号中对应的字符串改为"js/dart\_compressed.js";
- \* 将 blockly-master/demos/code 文件中的 code.js 文件移至 js 文件夹中，并将 src 引号中对应的字符串改为"js/code.js";

【注】此步的目的是将 Blockly 相关文件的位置进行调整，并且修改 html 部分代码，让其能指向对应文件；找文件的一个技巧：“..”代表返回父文件目录，文件前没有内容的话，就表明和 html 文件在同一文件夹下。

上述操作完成之后，保存 html 文件，并在浏览器中打开：

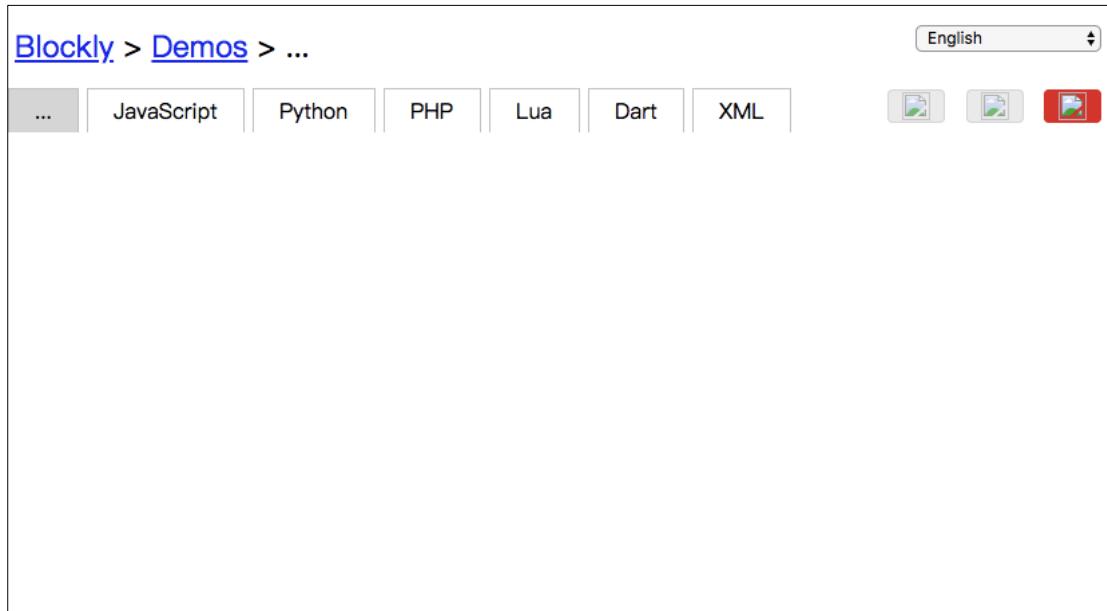


图 7-28toolbox 未显示

如上图，比较直观的可以看出 Toolbox 并未显示出来。文件明明已经导入并调整好了，

为什么还会出现这种情况？我们可以借助 Chrome 浏览器的“检查”功能，查看出现问题的原因：

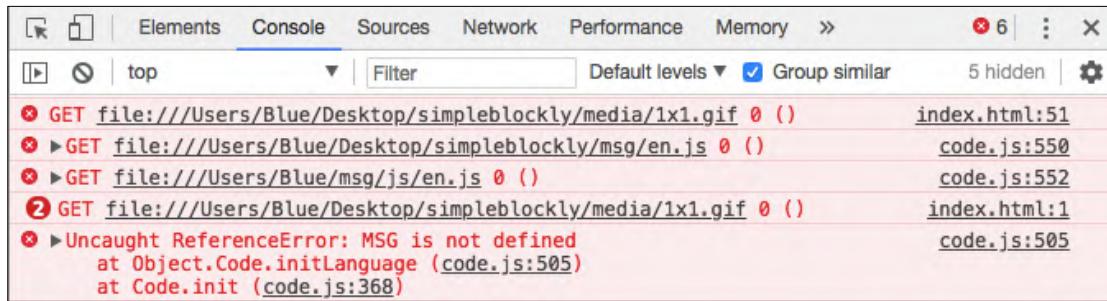


图 7-29 报错

在上图中，发现有几个错误，仔细看这些错误，发现除了 media 文件相关的错误之外，出现频率较高的就是 msg 相关的错误。

【注】对比之前的“删除文件”案例，在删除 media 或 msg 文件夹之后的现象。

msg 文件这么重要吗？它有什么功能？

对于 msg 文件，可分为 2 部分，一部分 blocks 的 msg 文件，负责 blocks 语言的转换；一部分是 category 的 msg 文件，负责 category 语言的转换；

如何解决这种情况？

a. 方案

分别导入 blocks 和 category 的 en.js 文件；（可分别添加，然后刷新两个文件对界面的影响）；界面转换成中文（两个 zh-hans.js 文件导入）。

b. 方案

通过 a. 方案可以实现基本功能，达到预期的效果，不过语言转换功能就失效了；如果需要完整的功能，就要通过 code.js 文件修改，直接进行调整 msg 文件指向的路径；

```
1. /**在修改 code 之前，需要了解一个基础知识*/
2. document.write("test");
3. /**可以在 demo2-1 中进行测试，不过要区分与 code.js 中代码的细微区别*/
```

先将根目录下的 msg 文件和 code 目录下的 msg 文件，分别拷贝到 SimpleBlockly/msg 文件中，为方便区分，分别重命名为 blocks-msg 和 category-msg，然后修改 code.js 中的代码指向这些文件。刷新网页，即可正常工作。

```
1. // Load the Code demo's language strings.
2. document.write('<script src="msg/category-msg/' + Code.LANG + '.js"></script>
3. \n');
4. // Load Blockly's language strings.
5. document.write('<script src="msg/blocks-msg/' + Code.LANG + '.js"></script>\n');
```

### (3) 媒体文件的导入

通过查看 Chrome 的 Console，发现 media 文件缺失，导致界面有裂图。所以将 media 文件拷贝到 SimpleBlockly 文件的目录中，然后修改 html 和 code.js 文件中的 media 对应路径即可。

【注】code.js 文件中的 media 文件对应第 420 行

至此，Simple Blockly 就大功告成了！

## 7.3Blocks 二次开发中的代码

在开始本案例之前，大家先通过下面的小例子，巩固一下 Blockly 开发者工具的使用。实现如图 7-30 所示代码块的设计：

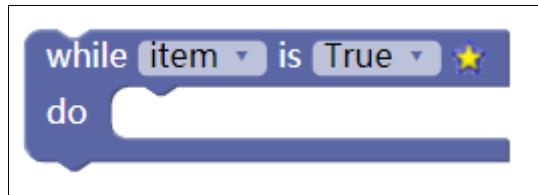


图 7-30 while 模块

参考答案如图 7-31 所示：

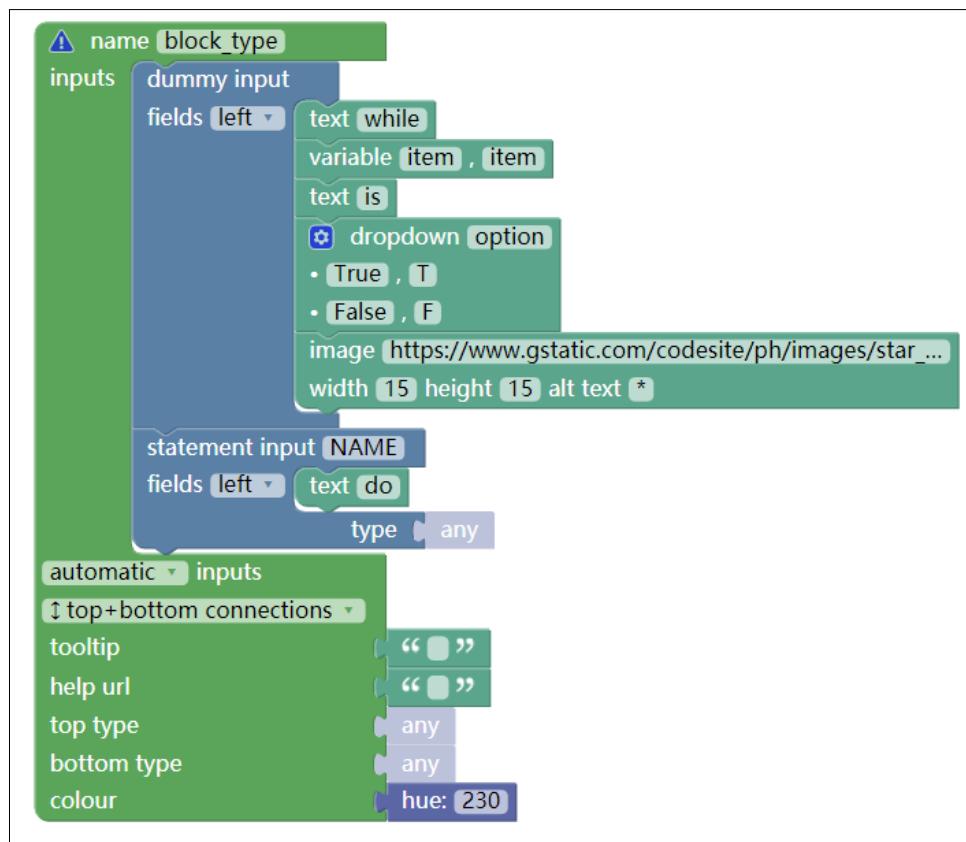


图 7-31 参考答案

### 7.3.1Plane 中 raw 块的模拟

#### (1) 块的设计

在 Block Factory 中设计 raw 代码块的外观，完成设计后点击“Save raw\_block”按钮，将新建块保存到 Blocks Library 中。

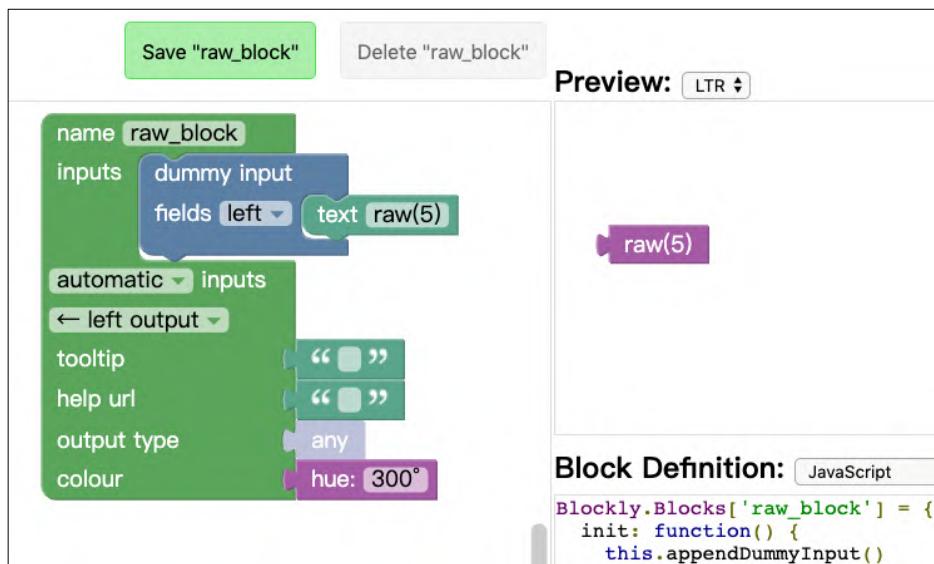


图 7-32 设计界面

## (2) 块的代码导出

点击 Block Exporter 选项卡，选择 raw\_block 代码：



图 7-33 raw\_block 代码块

修改界面中间部分的 Export Setting 属性：

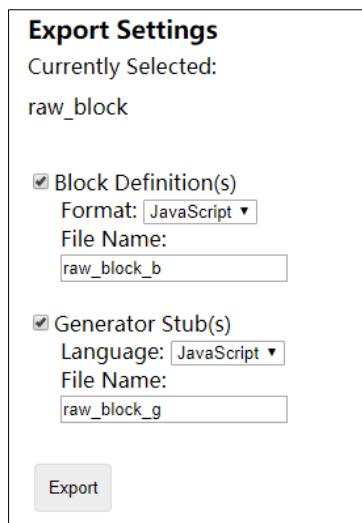


图 7-34 Export Setting

- \* 勾中 Block Definition(s)和 Generator Stub(s)选项；
- \* 将二者的 Format 属性（语言格式），都选为 JavaScript；
- \* 文件的命名，为方便辨认和引用，文件命名时建议采用块名（raw\_block）+ b/g 的形

式，如命名为 raw\_block\_b 和 raw\_block\_g。

属性修改完成之后，点击 Export 按钮即可对部分代码进行下载。导出的代码分两部分，一部分是 Block Definition 代码，一部分是 Generator Stub 代码。

### 【注】

#### (1) Block Definition 代码

此部分代码，负责控制新建块 (raw\_blocks) 的外观，包括样式、颜色等，最终放在 blocks 文件夹中。

#### (2) Generator Stub 代码

此部分代码，负责控制新建块 (raw\_blocks) 的代码转换，最终放在 generators 的对应子文件夹中。

#### (3) html 文件的修改

在完成块的代码设计与导出之后，需要将其导入至 html 文件中，以呈现给用户。

##### A. 导入 raw\_block\_b 文件

```
1. <script src="../../blocks/raw_block_b.js"></script>
```

##### B. 导入 raw\_block\_g 文件

```
1. <script src="../../generators/javascript/raw_block_g.js"></script>
```

##### C. 导入 raw\_block\_g 块

```
1. <block type="raw_block"></block>
```

完成之后保存 html 文件，并在浏览器中打开，即可看到你所新建的块 raw(5)了。



图 7-35 raw 的使用

### 【提示】

\* 直接导出的文件放到指定的文件夹下并导入 html 文件中，可以直接使用；不过保险起见，最好类比其他 blocks 和 generators 中的文件，在文件顶部加上相应的“头”文件。如下是在 raw\_block\_b 中添加的代码：

```
1. 'use strict';
2. goog.provide('Blockly.Blocks.repeat_times'); // Deprecated
3. goog.provide('Blockly.Constants.repeat_times');
4. goog.require('Blockly.Blocks');
5. goog.require('Blockly');
```

\* 在 html 文件修改的过程中，应注意 raw\_block\_b、raw\_block\_g 和 raw\_block 插入的位置和顺序，会影响显示的效果，可自行探索尝试。

### 7.3.2print-py 块的设计

#### (1) 块的设计

在 Block Factory 中设计 print\_py 代码块的外观，完成设计后点击“Save print\_py”按钮，将新建块保存到 Blocks Library 中。

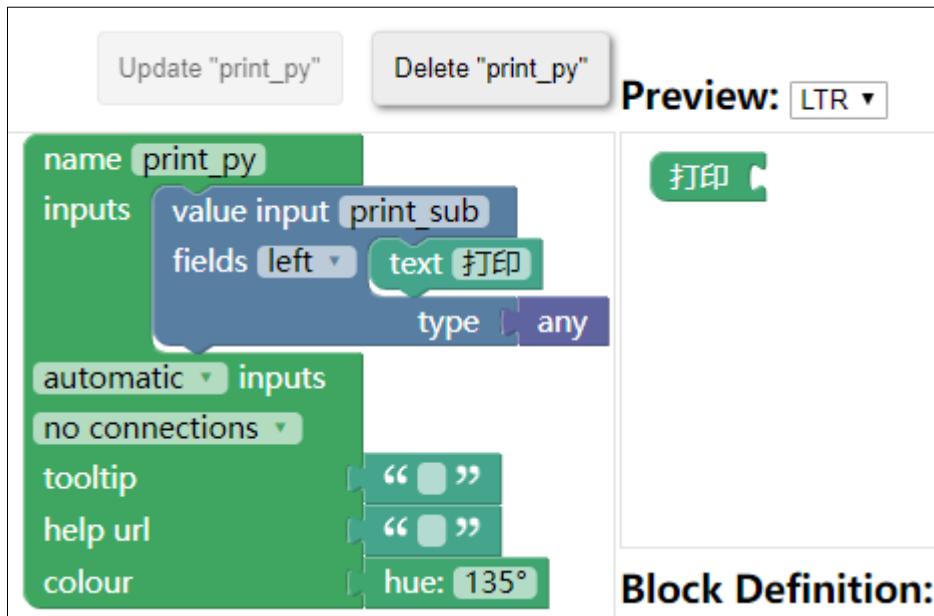


图 7-36 打印模块设计

#### (2) 块的代码导出

点击 Block Exporter 选项卡，选择 print\_py 代码：



图 7-37 打印模块

修改界面中间部分的 Export Setting 属性：

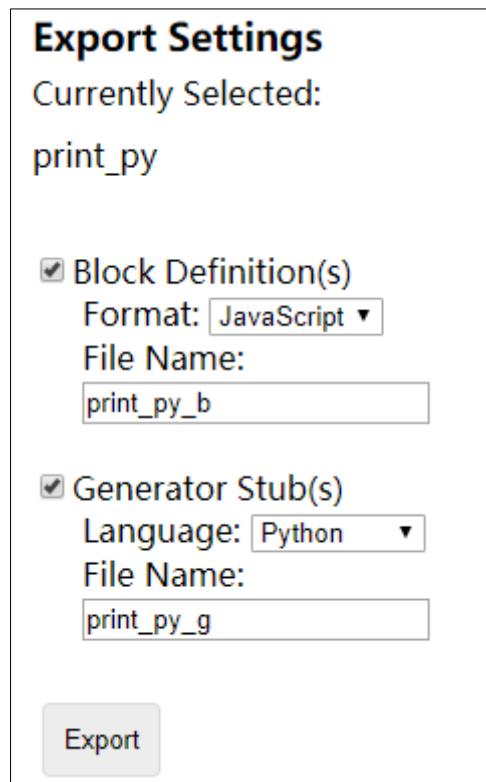


图 7-38 Export Setting

- \* 勾中 Block Definition(s) 和 Generator Stub(s) 选项；
- \* 选择二者的 Format 属性（语言格式），前者选为 JavaScript，后者选为 Python；
- \* 文件的命名，为方便辨认和引用，文件命名时建议采用块名（print\_py）+ b/g 的形式，如命名为 print\_py\_b 和 print\_py\_g。

属性修改完成之后，点击 Export 按钮即可对部分代码进行下载。导出的代码分两部分，一部分是 Block Denifition 代码，一部分是 Generator Stub 代码。

#### 【注】

##### (1) Block Denifition 代码

此部分代码，负责控制新建块（print\_py）的外观，包括样式、颜色等，最终放在 blocks 文件夹中。

##### (2) Generator Stub 代码

此部分代码，负责控制新建块（print\_py）的代码转换，最终放在 generators 的对应子文件夹中。

##### (3) html 文件的修改

在完成块的代码设计与导出之后，需要将其导入至 html 文件中，以呈现给用户。

##### A. 导入 print\_py\_b 文件

```
1. <script src="../../blocks/print_py_b.js"></script>
```

##### B. 导入 print\_py\_g 文件

```
1. <script src="../../generators/python/print_py_g.js"></script>
```

### C. 导入 print\_py 块

```
1.  <block type="print_py"></block>
```

完成之后保存 html 文件，并在浏览器中打开，即可看到你所新建的块“打印”了。



图 7-39 打印

#### (4) 优化改进

完成前 3 步之后，功能已基本实现，现在有 3 个小任务，对 print\_py 块进行优化。

\* print\_py 代码是否可以与 raw 代码整合到一个文件中，如果可以，为什么？

\* print\_py 块如何类比 print 块加上默认阴影？

```
1. <block type="print_py">
2. <value name="print_sub"> // value 的 name 为块设计中的 value input 的 name
3. <shadow type="text">
4.   <field name="TEXT">Test</field>
5. </shadow>
6. </value>
7. </block>
```

\* 完善 print\_py 块的其他几种语言的转换功能

### 7.3.3 Repeat-Do 块的复现

#### (1) 代码块设计

在 Block Factory 中设计 repeat\_do 代码块的外观，完成设计后点击“Save repeat\_do”按钮，将新建块保存到 Blocks Library 中。

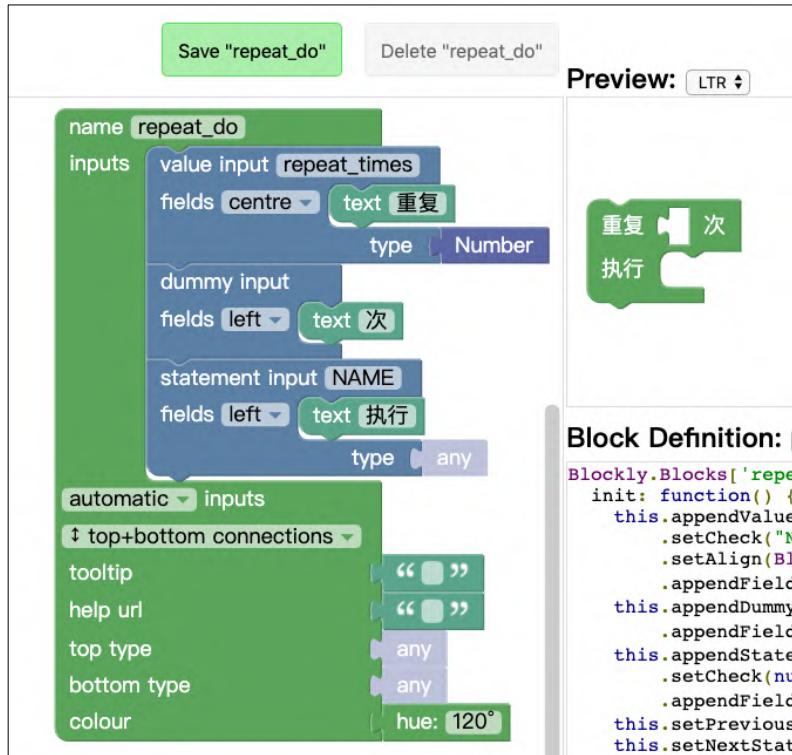


图 7-40 save

### (2) 块的代码导出

可以选择与前两个 demo 中一样的方法，直接 Export 代码块对应的代码，不过这里为了方便，直接用 raw\_block 的 js 文件，复制 repeat\_do 的两部分代码，分别粘贴到 raw\_block\_b.js 和 raw\_block\_g.js 文件中，保存即可。

### (3) html 文件的修改

在完成块的代码设计与导出之后，需要将其导入至 html 文件中，以呈现给用户。

A. raw\_block\_b 和 raw\_block\_g 文件已导入，无须重复导入

```

1. <script src="../../blocks/raw_block_b.js"></script>
2. <script src="../../generators/javascript/raw_block_g.js"></script>

```

B. 导入 repeat-do 代码块，并加上默认阴影

```

1. <block type="repeat_do">
2. <value name="repeat_times">
3. <shadow type="math_number">
4.   <field name="NUM">10</field>
5. </shadow>
6. </value>
7. </block>

```

完成之后保存 html 文件，并在浏览器中打开，即可看到你所新建的块 repeat\_do 代码块了。

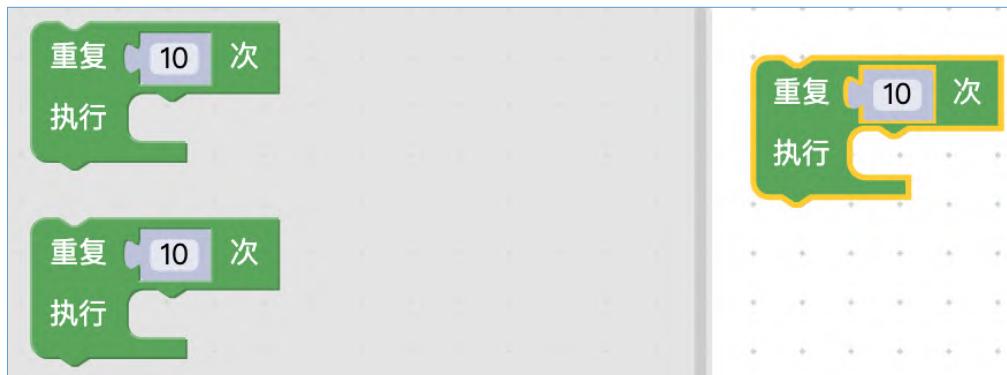


图 7-41 重复执行

是不是已经分辨不出哪个块是你自己创建的了？

#### (4) 代码转换功能完善

虽然从外观已经很难分辨出哪个是我们自定义的 repeat\_do 代码块了，但是我们并没有给它添加实际的代码转换功能，如图 7-42 所示：



图 7-42 重复打印

一个 repeat\_do 代码块会有两个“参数”的输入，一个是次数，一个是需要 do 的语句，它们转换后的 JavaScript 代码应该为：

```

1. for (var count = 0; count < 10; count++) {
2.     window.alert('abc');
3. }
```

所以我们参照这个样式，来进行 repeat\_do 自定义块代码的设计：

```

1. Blockly.JavaScript['repeat_do'] = function(block) {
2.     var value_repeat_times = Blockly.JavaScript.valueToCode(block, 'repeat_t
imes', Blockly.JavaScript.ORDER_ATOMIC);
3.     var statements_repeat_statement = Blockly.JavaScript.statementToCode(blo
ck, 'repeat_statement');
4.     // TODO: Assemble JavaScript into code variable.
5.     var code = 'for(var count=0;count<' + value_repeat_times + ';count++){\n
    ' + statements_repeat_statement + '}\n';
6.     return code;
7. };
```

至此，repeat\_do 代码块大功告成！

## 7.4 二次开发案例—puzzle 游戏的制作

### 7.4.1 Simple Blockly

在正式开始之前，先做一个小案例热热身，这也是谷歌官网的有提到的一个小案例，可能因为是英文的，可能大家没有留意，这里就带大家动手操作一下。

The screenshot shows the Blockly documentation page. On the left, there's a sidebar with links like 'Overview', 'Get Started', 'Configure Blockly' (which is expanded to show 'Web', 'Toolbox', etc.), and 'Android'. The main content area is titled 'Toolbox' and contains text explaining how to construct a toolbox using XML. It includes a code example:

```
<xml id="toolbox" style="display: none">
  <block type="controls_if"></block>
  <block type="controls_whileUntil"></block>
</xml>
<script>
  var workspace = Blockly.inject('blocklyDiv',
    {toolbox: document.getElementById('toolbox')});
</script>
```

On the right, there's a sidebar with links for 'Categories', 'Dynamic categories', 'Tree of Categories', 'Block Groups', 'Shadow blocks', 'Separators', 'Buttons and Labels', 'Disabled', and 'Changing the Toolbox'. At the top right, there are five star rating icons.

图 7-43SimpleBlockly 案例

#### 1. 超精简 Blockly

##### (1) 新建 html 文件

新建文件夹，并在文件夹中新建 index.html 文件，并在 html 文件中，编写之前提到的“最简”html 代码：

```
1. 1. <!DOCTYPE>
2. 2. <html>
3. 3. <head>
4. 4.   <meta charset="UTF-8">
5. 5.   <title>SSimple Blockly</title>
6. 6. </head>
7. 7. <body>
8. 8. </body>
9. 9. </html>
```

##### (2) 修改 html 文件

向上一步 html 代码中，添加新的“控制”代码：

```
1. 1. /*添加到 head 中*/
2. 2. <script src="blockly_compressed.js"></script>
3. 3. <script src="blocks_compressed.js"></script>
4. 4. <script src="messages.js"></script>
```

```
5.  
6. /*添加到 body 中*/  
7. <div id="blocklyDiv"></div>  
8. <xml id="toolbox" style="display: none">  
9.     <block type="controls_if"></block>  
10.    <block type="controls_whileUntil"></block>  
11. </xml>  
12. <script>  
13.     var workspace = Blockly.inject('blocklyDiv',  
14.         {toolbox: document.getElementById('toolbox')});  
15. </script>
```

### (3) 复制对应 js 文件

将 blockly 文件夹中的 blockly\_compressed.js、blocks\_compressed.js、messages.js，放到此文件目录下，用浏览器打开 html 文件，即可看到你的超精简 blockly。（与 Blockly Demo 中的 Fixed Blockly 很像）

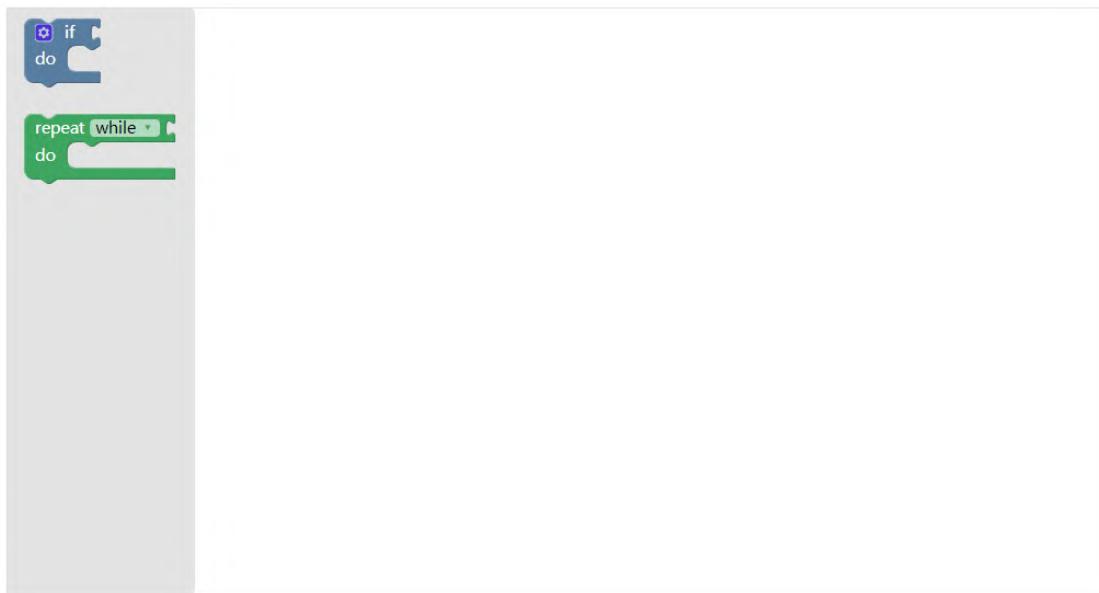


图 7-44 超精简 blockly

当然，这里的 blockly 只添加了 blocks，没有添加 category，大家可以自己动手添加。

【思考】为什么这里面没有导入 blocks 的 msg 文件可以正常显示？

## 2. 较完整的 Blockly

前面我们用到了 Blockly 的开发者工具，但只是用它来进行块的设计，它的 Workspace Factory 我们并没有深入接触，这个较完整的 Blockly，就是教大家用 Workspace Factory，定义你的 Blockly。

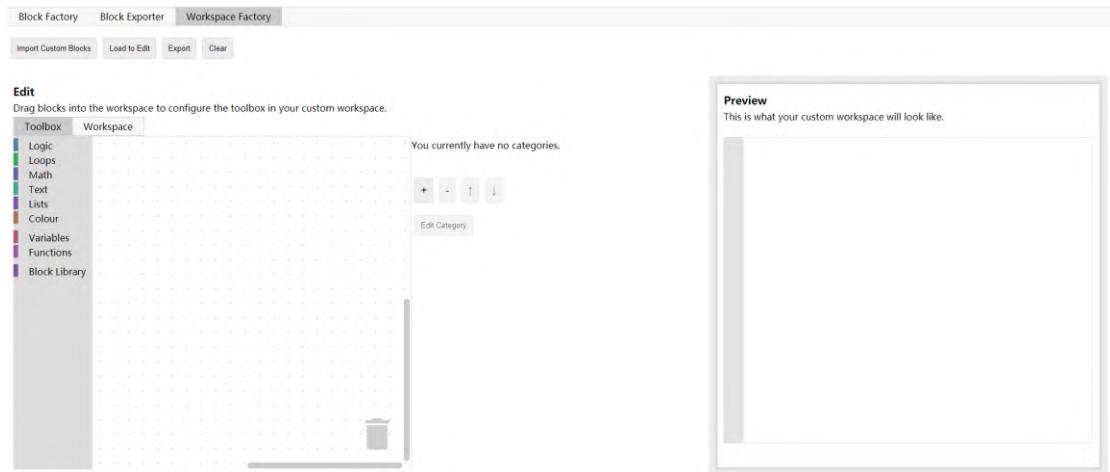


图 7-45 Workspace Factory 界面

### (1) 定义 Toolbox

当选项卡选中 Toolbox 时，可以通过中部的“+”和“-”，添加或删除 Toolbox 中的类或块。

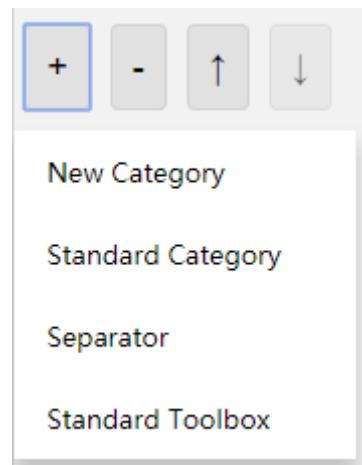


图 7-46 “+”选项

- New Category，输入类名，新建全新的类；

[此网页显示](#)

Enter the name of your new category:

确定
取消

图 7-47 New Category

- Standard Category，输入标准类的类名，将直接添加整个类；

此网页显示

Enter the name of the category you would like to import (Logic, Loops, Math, Text, Lists, Colour, Variables, or Functions)

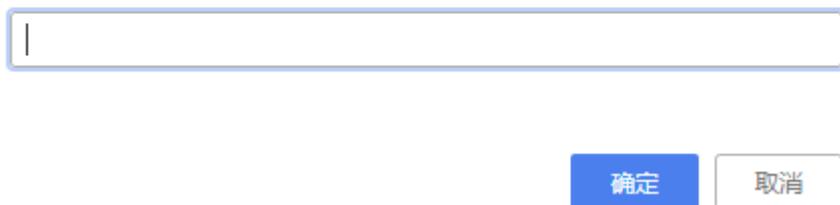


图 7-48 Standard Category

- Separator, 在类与类之间, 新建一个分界线;
- Standard Toolbox, 添加完整的 Toolbox 到工作区;

### (2) 定义 Workspace

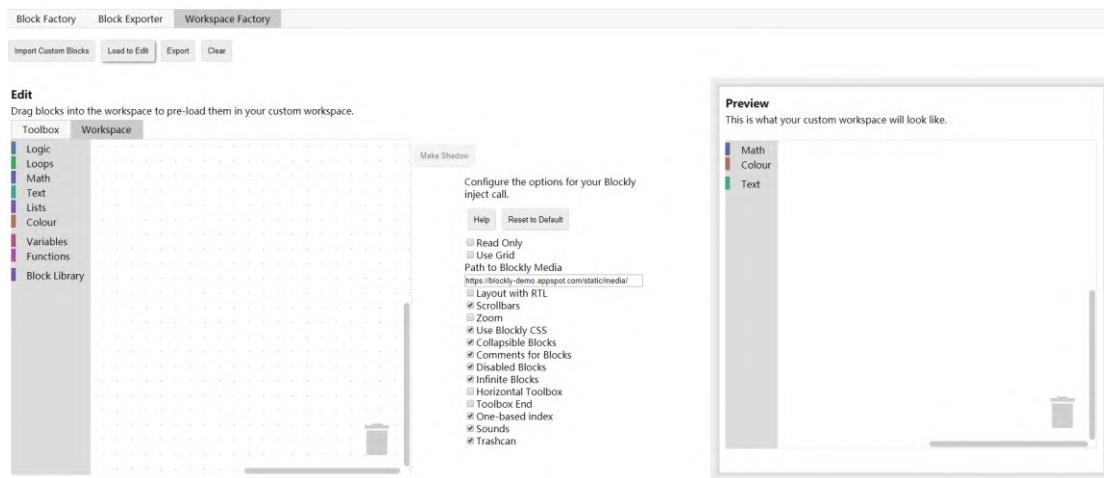


图 7-49 Workspace 界面

当选项卡选中 Workspace 时, 界面看似和 Toolbox 没什么区别, 不过这里从左侧 Toolbox 中拖拽到工作区的代码块, 都将默认显示在你的工作区, 像 Plane 游戏中的 seat 块一样, 而且可以通过中部的选项, 定义工作区的样式, 是否添加 Zoom, 是否添加 Grid, 是否添加 Scrollbars 等。

### (3) 导出文件

完成前两步之后, 导出点击“Export”按钮, 选择你要导出的文件。

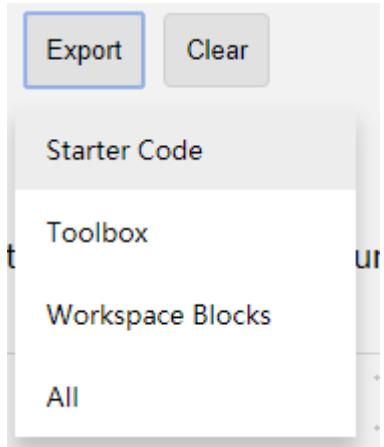


图 7-50 Export 选项

这里，我们选择导出 ALL。

#### (4) 新建 html

如“超精简 Blockly”案例一样，先新建“最简”html 代码：

```

1.  <!DOCTYPE>
2.  <html>
3.  <head>
4.      <meta charset="UTF-8">
5.      <title>SSimple Blockly2</title>
6.  </head>
7.  <body>
8.  </body>
9. </html>
```

打开（3）中导出的 workspace.xml 代码，将其复制添加到 body 中：

```

1.  <xml xmlns="http://www.w3.org/1999/xhtml" id="workspaceBlocks" style="display:none">
2.      <variables></variables>
3.  </xml>
```

打开（3）中导出的 toolbox.xml 代码，将其复制添加到 body 中；

将 workspace.js 文件复制到该文件目录下，并用编辑器打开：

```

1. /* TODO: Change toolbox XML ID if necessary. Can export toolbox XML from Workspace Factory. */
2. var toolbox = document.getElementById("toolbox");
3.
4. var options = {
5.     toolbox : toolbox,
6.     collapse : true,
7.     comments : true,
```

```

8.      disable : true,
9.      maxBlocks : Infinity,
10.     trashcan : true,
11.     horizontalLayout : false,
12.     toolboxPosition : 'start',
13.     css : true,
14.     media : 'https://blockly-demo.appspot.com/static/media/',
15.     rtl : false,
16.     scrollbars : true,
17.     sounds : true,
18.     oneBasedIndex : true
19. };
20.
21. /* Inject your workspace */
22. var workspace = Blockly.inject(/* TODO: Add ID of div to inject Blockly into
   * /, options);
23.
24. /* Load Workspace Blocks from XML to workspace. Remove all code below if no
   blocks to load */
25.
26. /* TODO: Change workspace blocks XML ID if necessary. Can export workspace b
   locks XML from
27. Workspace Factory. */
28. var workspaceBlocks = document.getElementById("workspaceBlocks");
29.
30. /* Load blocks to workspace. */
31. Blockly.Xml.domToWorkspace(workspaceBlocks, workspace);

```

在 `workspace.js` 代码中，我们只需要修改如下代码：

```

1. -----修改前-----
2. /* Inject your workspace */
3. var workspace = Blockly.inject(/* TODO: Add ID of div to inject Blockly into
   * /, options);
4. -----修改为-----
5. /* Inject your workspace */
6. var workspace = Blockly.inject("blocklyDiv", options);

```

修改完成之后，在 `body` 底部添加：

```
1. <script src="workspace.js"></script>
```

注意：`workspace.js` 文件插入的位置，如果插入到 `head` 中，可能无法显示。

以上步骤都完成之后，用浏览器打开 `html` 文件，即可看到你设计的 `Blockly` 界面。

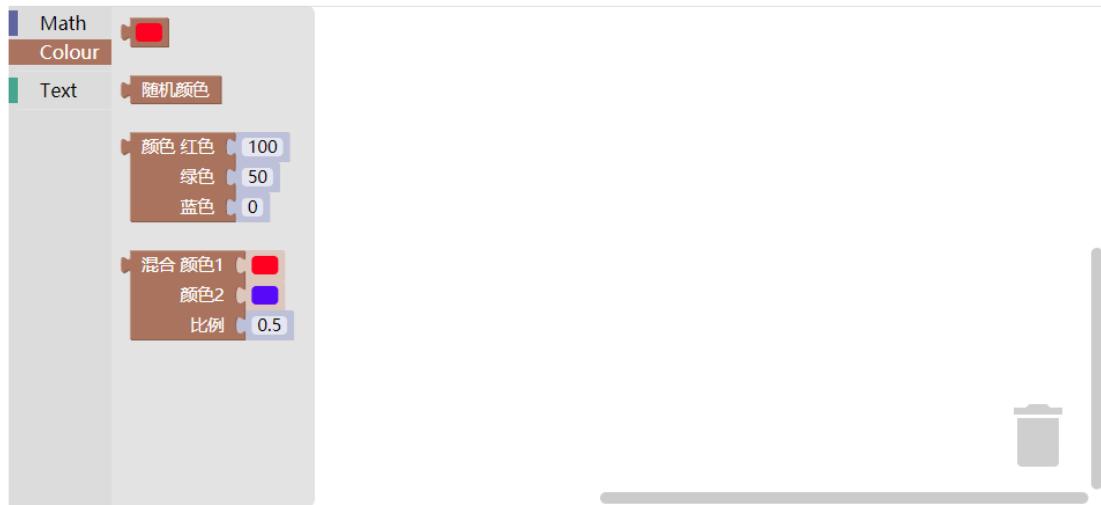


图 7-51 我定义的 Blockly 界面

### 7.4.2 制作 puzzle 游戏

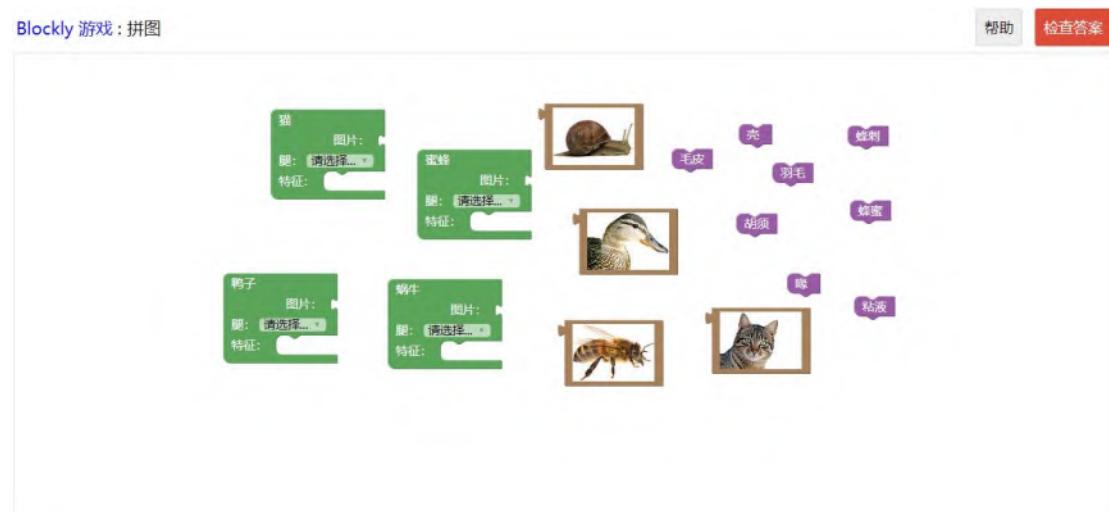


图 7-52puzzle 游戏

#### 1. 新建代码块

使用 Blockly Developer Tools 新建拼图所涉及的代码块。

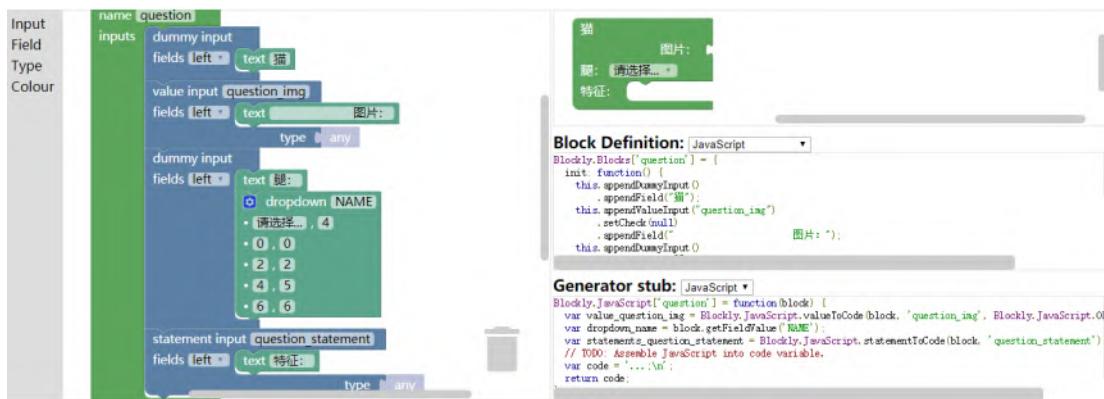


图 7-53 拼图模块 1

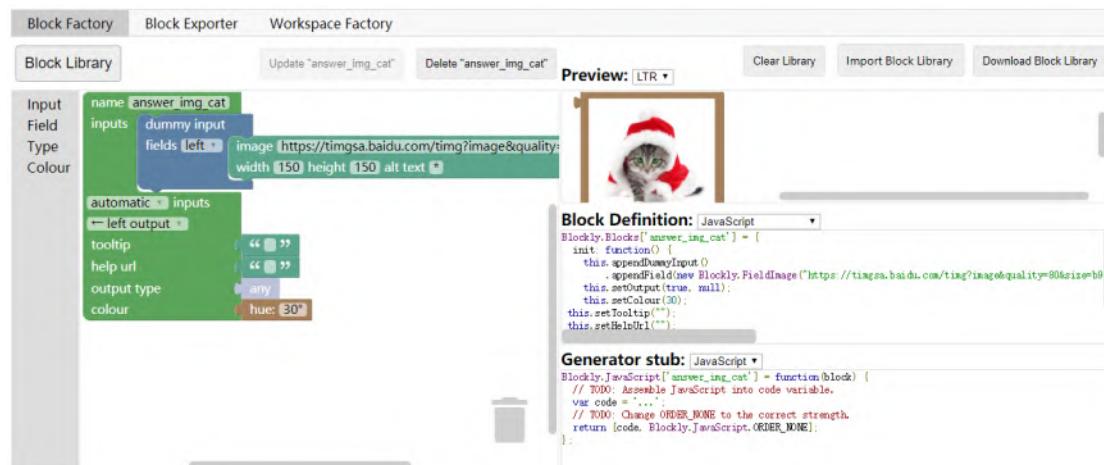


图 7-54 拼图模块 2

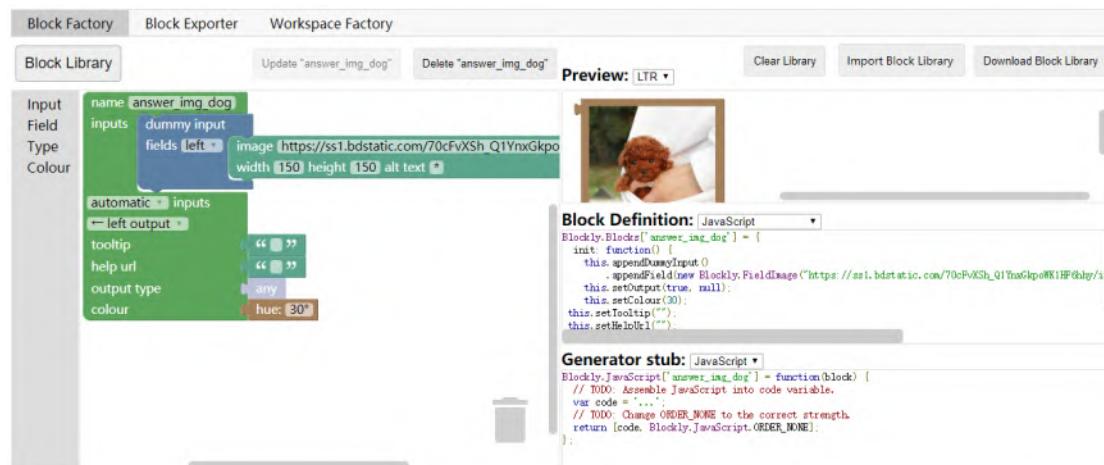


图 7-55 拼图模块 3

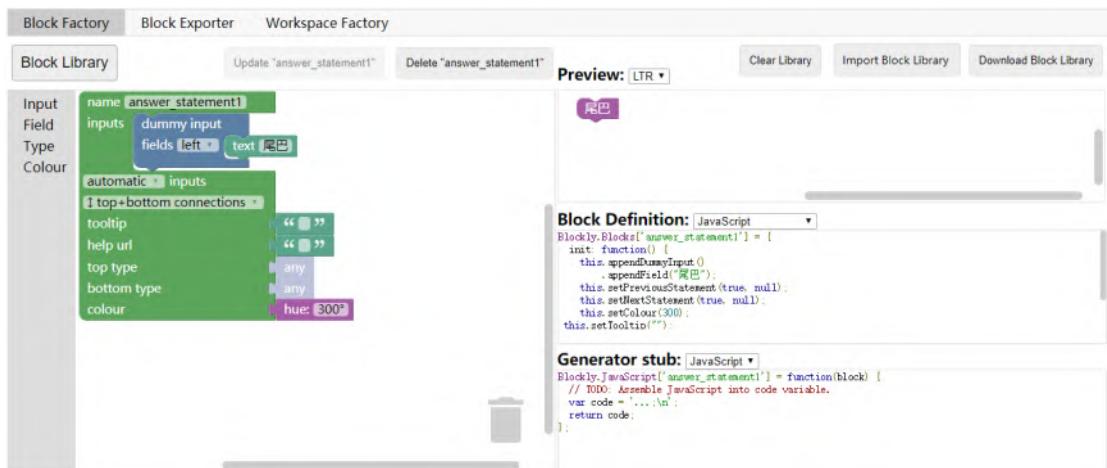


图 7-56 拼图模块 4

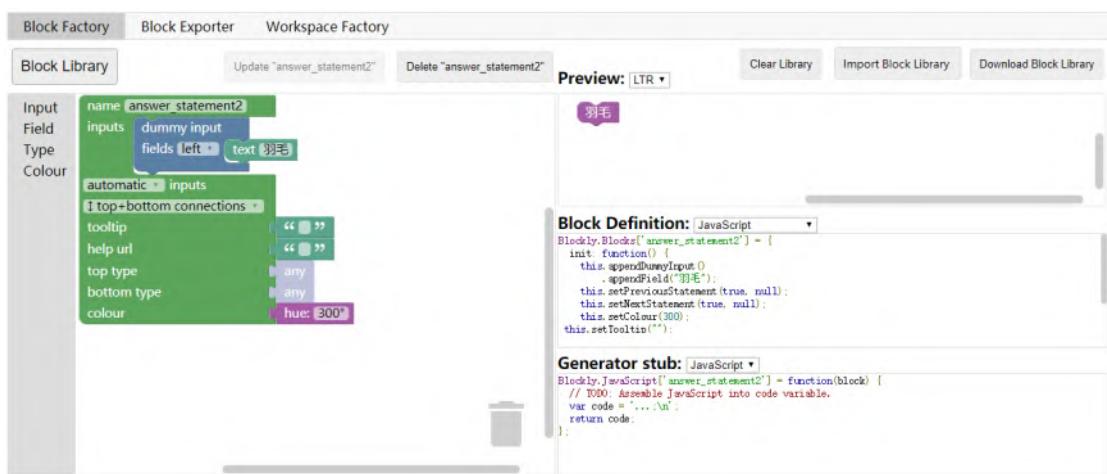


图 7-57 拼图模块 5

## 2. 导出代码块

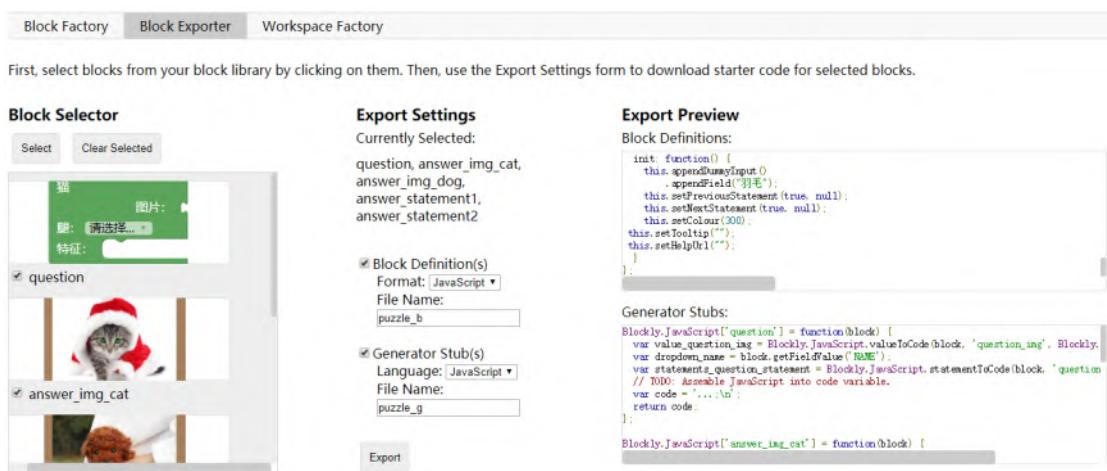


图 7-58 导出代码块

点击 Block Exporter 选项卡，全选所有代码块，选择 Block Definition(s) 和 Generator Stub(s) 的文件类型，并分别定义文件名，点击 Export，即可将所有的块的定义代码导入到 puzzle\_b.js 文件夹中，将所有的块的生成代码导入到 puzzle\_g.js 文件夹中。

### 3. 导出工作区

拼图案案例不涉及 Toolbox 工具箱，所以这里直接定义工作区即可，将需要用到的块添加到工作区，然后导出 Starter Code 和 Workspace Blocks 文件。

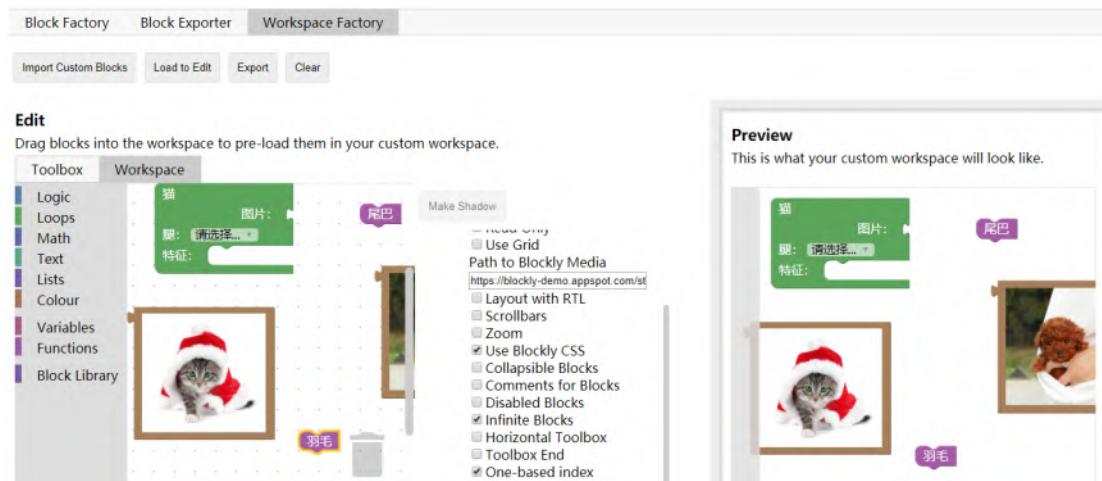


图 7-59 设置并导出工作区

### 4. 文件的整理

新建文件夹，并将 2 和 3 步中导出的文件，都复制粘贴到新建的文件夹中；将 blockly\_compressed.js 和 javascript.js 文件复制粘贴到新建的文件夹中；

blockly_compressed.js	2018/3/10 5:55	JavaScript 文件	717 KB
javascript_compressed.js	2018/3/10 5:55	JavaScript 文件	46 KB
puzzle.html	2018/5/25 21:22	Chrome HTML D...	1 KB
puzzle_b.js	2018/5/25 20:31	JavaScript 文件	2 KB
puzzle_g.js	2018/5/25 20:31	JavaScript 文件	2 KB
workspace.js	2018/5/25 20:33	JavaScript 文件	1 KB

图 7-60 js 文件整理

js 文件导入：

```
1. <script src="blockly_compressed.js"></script>
2. <script src="puzzle_b.js"></script>
3. <script src="javascript_compressed.js"></script>
```

```
4. <script src="puzzle_g.js"></script>
```

并将 workspace 的 xml 文件，插入到 body 标签中，修改导出的 workspace.js 中的代码：

```
1. /* Inject your workspace */
2. var workspace = Blockly.inject('blockDiv', options);
```

完成以上步骤之后，保存所有的文件，并用浏览器打开 html 文件，在浏览器中查看是否创建成功。



图 7-61 puzzle 界面成功加载

## 5. 对“检查答案”功能进行设计

### (1) 添加按钮

在 body 中添加“检查答案”的按钮

```
1. <button id="check_button">检查答案</button>
```

### (2) 设计生成代码

给每个块设置返回代码，这里为了方便匹配和判断，选取简单的数字作为返回值：

```
1. Blockly.JavaScript['question'] = function(block) {
2.   var value_question_img = Blockly.JavaScript.valueToCode(block, 'question_i
mg', Blockly.JavaScript.ORDER_ATOMIC);
3.   var dropdown_leg_number = block.getFieldValue('leg_number');
4.   var statements_question_statement = Blockly.JavaScript.statementToCode(blo
ck, 'question_statement');
5.   // TODO: Assemble JavaScript into code variable.
```

```

6.     var code = (value_question_img=="(#cat)" && dropdown_leg_number=="4" && statements_question_statement==21 );
7.     if(code){
8.         return "yes";
9.     }else{
10.        return "no";
11.    }
12. };
13.
14. Blockly.JavaScript['answer_img_cat'] = function(block) {
15.   // TODO: Assemble JavaScript into code variable.
16.   var code = '#cat';
17.   // TODO: Change ORDER_NONE to the correct strength.
18.   return [code, Blockly.JavaScript.ORDER_NONE];
19. };
20.
21. Blockly.JavaScript['answer_img_dog'] = function(block) {
22.   // TODO: Assemble JavaScript into code variable.
23.   var code = '#dog';
24.   // TODO: Change ORDER_NONE to the correct strength.
25.   return [code, Blockly.JavaScript.ORDER_NONE];
26. };
27.
28. Blockly.JavaScript['answer_statement1'] = function(block) {
29.   // TODO: Assemble JavaScript into code variable.
30.   var code = '';
31.   code = '21';
32.   return code;
33. };
34.
35. Blockly.JavaScript['answer_statement2'] = function(block) {
36.   // TODO: Assemble JavaScript into code variable.
37.   var code = '22';
38.   return code;
39. };

```

并且在 workspace.js 文件中添加对应的 button 控制代码：

```

1.  /* 创建点击函数 */
2.  function button_click(){
3.      var code = Blockly.JavaScript.workspaceToCode(workspace);
4.      console.log(code);
5.      console.log("----" + code.match("no"))

```

```
6.     if(code.match("no")== null){  
7.         alert("恭喜你，全对！");  
8.     }else{  
9.         alert("别灰心，继续加油！");  
10.    }  
11. }  
12.  
13. /* 添加点击事件监听 */  
14. document.getElementById("check_button").addEventListener("click", button_ck);
```

修改完成之后，保存所有代码，用浏览器打开 html 文件进行测试：



图 7-62 恭喜你，全对！



图 7-63 别灰心，继续加油！

## 7.5 Blockly 的高级使用

在之前的学习中，我们通过使用 Blockly，学习一些基础程序设计中的经典的例子，并通过 Blockly 的可视化代码编辑器，进行了编程的实践练习，我们所接触和使用的这些，并不是 Blockly 设计的初衷。Blockly 是一个库，它为 Web 和 Android 应用程序添加了一个可视化代码编辑器，Blockly 编辑器使用互锁的图形块来表示代码概念，如变量、逻辑表达式、循环等，它允许用户应用编程原则，而不必担心语法或命令行上闪烁的光标。

### 7.5.1 将 Blockly 作为代码生成器

每个人不可能精通甚至熟悉每一种语言，但有时候，在学习、工作中又可能会用到所未接触过的语言，如果我们没有额外的时间且精力，尤其当这种语言再极少使用时，我们可能不乐意去花时间和精力去学习，但又不得不用，于是经常陷入两难。针对这一常见现象，我们就可以使用 Blockly 作为代码生成工具。

(1) 假如现在我们需要一个判断平年闰年的 Python 代码的小例子，但我们之前又没接触过 Python，我们又不想学习 Python，那么就可以打开 Blockly，在编辑区拖动块来编写：



图 7-64 Blockly 判断闰年代码

拖动完成，验证无误，点击 Python 选项卡，复制代码至你的 Python 环境中，即可直接运行。

The screenshot shows a Python code editor window with the following code:

```
File Edit Format Run Options Window Help
year = None
leap = None
_E9_A1_B9_E7_9B_AE = None

def text_prompt(msg):
    try:
        return raw_input(msg)
    except NameError:
        return input(msg)

year = int(text_prompt('请输入年份: '))
if year % 4 == 0:
    if year % 100 == 0:
        if year % 400 == 0:
            leap = 1
        else:
            leap = 0
    else:
        leap = 1
else:
    leap = 0
if leap == 1:
    year = str(year) + str('是闰年!')
    print(year)
else:
    year = str(year) + str('是平年!')
    print(year)
```

Ln: 10 Col: 0

图 7-65 Python 判断闰年代码

Python 环境中的运行结果：

```
=====
RESTART: E:/Document/Code/python/year.py =====
请输入年份: 2018
2018是平年!
>>> |
```

图 7-66 Python 运行结果

(2) 假如现在需要一个 JavaScript 的执行脚本，而且我们对 JavaScript 也有所了解，我们也可以尝试在 Blockly 中进行编程开发，比如这个猜数字的小游戏：



图 7-67 Blockly 猜数字游戏程序主题部分

上面是程序的主体部分，包括循环、提示和中断。



图 7-68 Blockly 猜数字游戏程序变量赋值部分

在程序之前，创建了三个变量并进行了初始化：

步长：用于计数，针对猜的次数进行不同的提示。

Target：存放随机生成的目标数，与所猜数 Number 进行比较。

Flag：开关变量，用于标记是否猜对，从而决定是否提示下方内容。



图 7-69 Blockly 猜数字游戏程序验证部分

同样，验证无误后，点击 JavaScript 选项卡，复制 js 代码并保存。

```
var number, count, target, flag;

function mathRandomInt(a, b) {
    if (a > b) {
        // Swap a and b to ensure a is smaller.
        var c = a;
        a = b;
        b = c;
    }
    return Math.floor(Math.random() * (b - a + 1) + a);
}

count = 0;
flag = true;
target = mathRandomInt(1, 30);
for (var count2 = 0; count2 < 5; count2++) {
    number = window.prompt('请输入你猜的数字');
    count = count + 1;
    if (number == target) {
        if (count == 1) {
            window.alert('真神了！猜对了！');
            flag = false;
            break;
        }
        if (count == 2) {
            window.alert('厉害，对啦！');
            flag = false;
            break;
        }
        if (count == 3) {
            window.alert('不错，对啦！');
            flag = false;
            break;
        }
        if (count == 4) {
            window.alert('有点慢，对啦！');
            flag = false;
            break;
        }
        if (count == 5) {
            window.alert('下次快点，对啦！');
            flag = false;
            break;
        }
    } else {
```

```
if (number > target) {  
    window.alert('大了，继续！');  
} else {  
    window.alert('小了，继续！');  
}  
}  
}  
if (flag) {  
    window.alert('游戏失败，重新开始吧！');  
}
```

图 7-70 Blockly 猜数字游戏 js 代码

将保存后的 js 代码导入到 html 文件中测试。

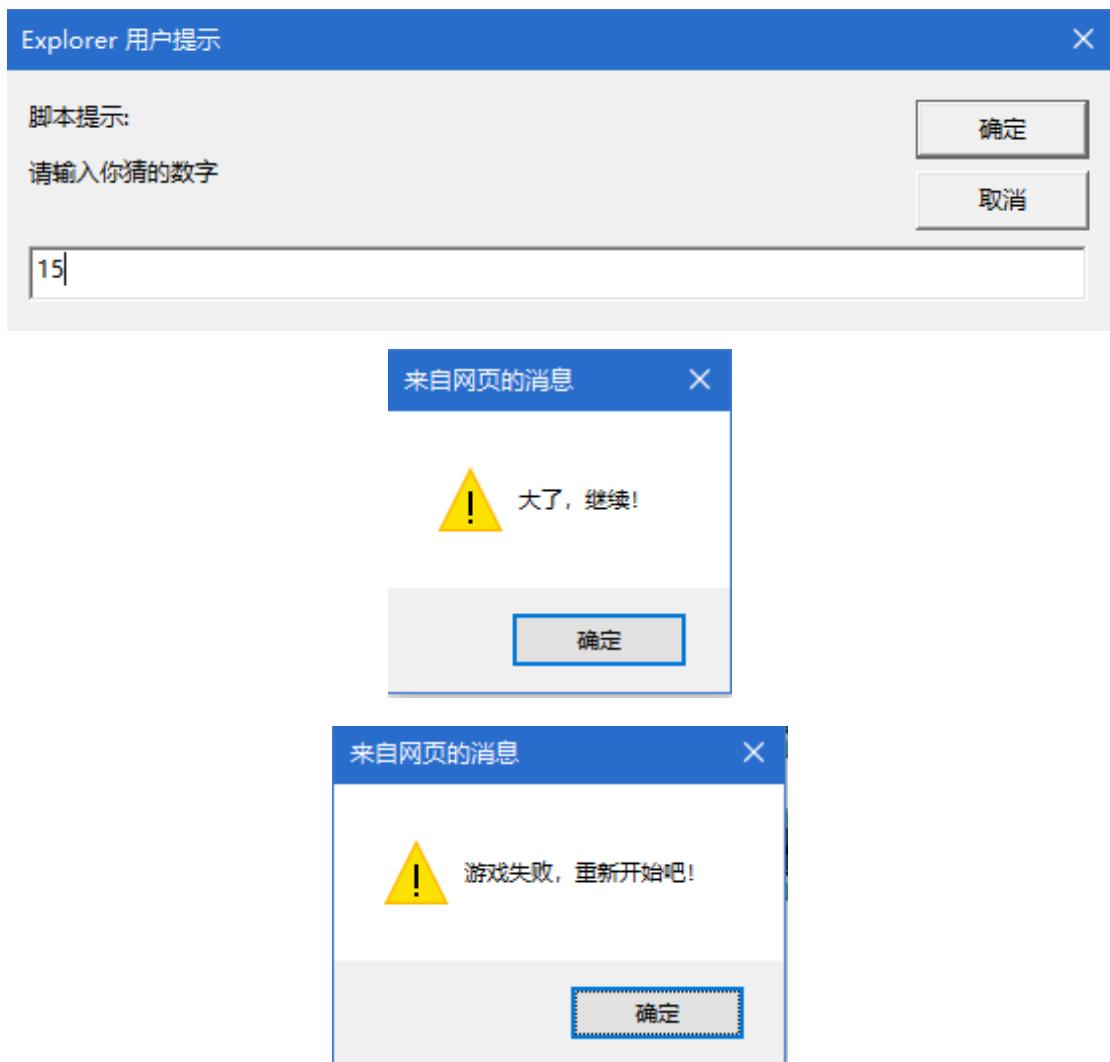


图 7-71 Blockly 猜数字游戏 js 代码运行结果

当然，也可以把猜数字小游戏的 Python 代码导出，同样可执行。

```
请输入你猜的数字12  
大了，继续！  
请输入你猜的数字23  
大了，继续！  
请输入你猜的数字123  
大了，继续！  
请输入你猜的数字1  
有点慢，对啦！  
>>> |
```

图 7-72 Blockly 猜数字游戏 Python 代码运行结果

### 小提示

Python 代码的导出执行，当程序涉及输入且输入的是数字时，需要使用 `int()`，将输入的字符串型“数字”强制类型转换成整型。

```
#Blockly生成  
number = text_prompt('请输入你猜的数字')  
#修改加入强制类型转换后  
number = int(text_prompt('请输入你猜的数字'))
```

如果不进行强制类型转换，执行脚本时可能会报错，即使不报错，结果也可能不正确。但这一问题在 JavaScript 导出代码中不存在，兼容性良好。

```
TypeError: unorderable types: str() > int()
```



### 小提示

对于 JavaScript 代码的测试，可以导入到 html 中，在浏览器中执行，测试效果与 Blockly 中效果相同。html 代码如下：

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>猜数字</title>  
</head>  
<body>  
    <script src="GuessNumberGame.js">  
    </script>  
</body>  
</html>
```



本条小提示专门针对没有 JavaScript 基础，但是对 Blockly 代码生成工具又感兴趣，想亲手测试验证的读者。

关于 Blockly 作为代码生成工具的使用，我们这里只举了两个基础的、有代表性的例子，当然如果你学有余力或者对所生成的目标代码十分熟悉，可以自行尝试更加有趣、更加复杂的例子，如果你觉得这并不能满足你的需求，那么可以尝试自己动手定义你想要的块，生成代码的格式、类型和种类。

## 7.5.2 Blockly 的二次开发

随着 Blockly 逐渐的完善，它被越来越多的人所熟知，同时，凭借它可视化编程，良好的可扩展性等特点，很多的开发者利用 Blockly 进行二次开发，因此衍生出许多优秀的产品和工具。

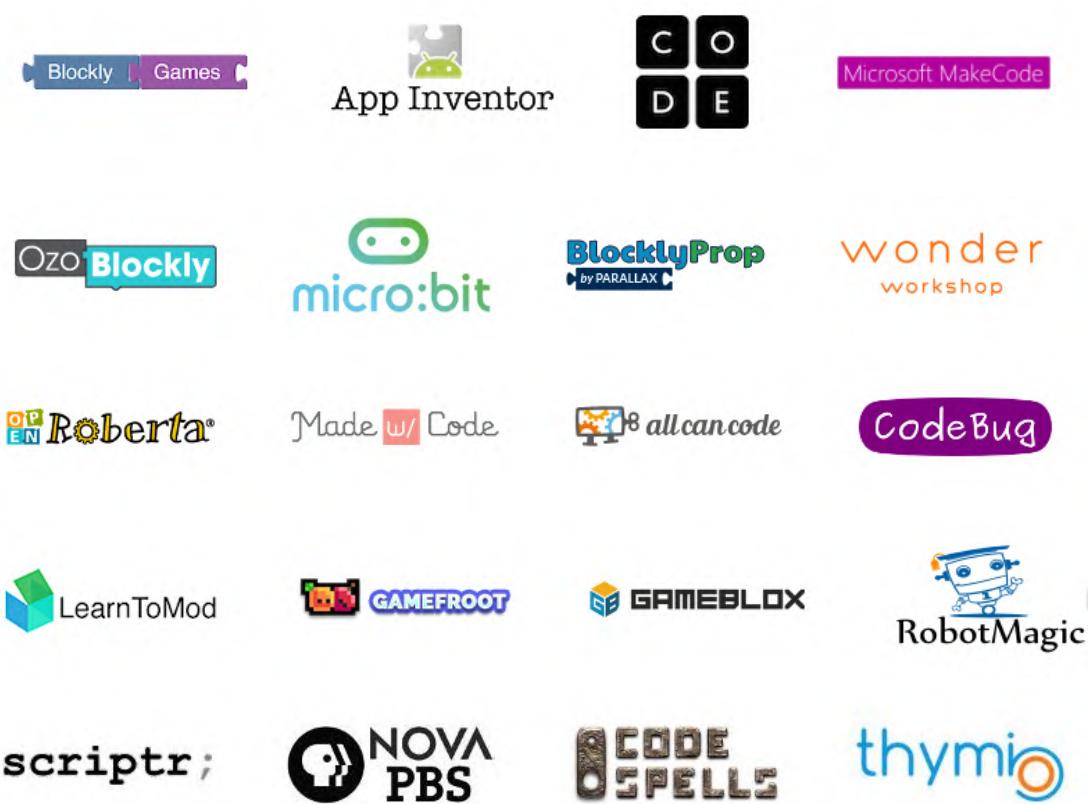


图 7-73 Blockly 衍生产品和工具

前面曾提到过，Blockly 的初衷是针对开发人员设计的，它是一个针对有经验的开发人员的复杂库。从用户的角度来看，Blockly 是一种直观，可视化的构建代码的方法。从开发人员的角度来看，Blockly 本质上是一个包含正确语法、生成代码的文本框，Blockly 可以

导出多种语言，例如 JavaScript, Python, PHP, Lua, Dart 等，下面是对 Blockly 进行二次开发的步骤：

a.集成块编辑器。 Blockly 编辑器包括用于存储块类型的工具箱和用于排列块的工作空间。

b.创建应用程序的块。一旦你的应用程序中有 Blockly，你就需要创建块供用户编码，然后将它们添加到您的 Blockly 工具箱。

c.构建应用程序的其余部分。本身，Blockly 只是一种生成代码的方法，你的应用程序的核心在于如何处理该代码。可能单纯的文字描述比较抽象，难以理解，以 FreDuino 为例，它是基于 Blockly 二次开发而成的一个远程硬件控制平台。

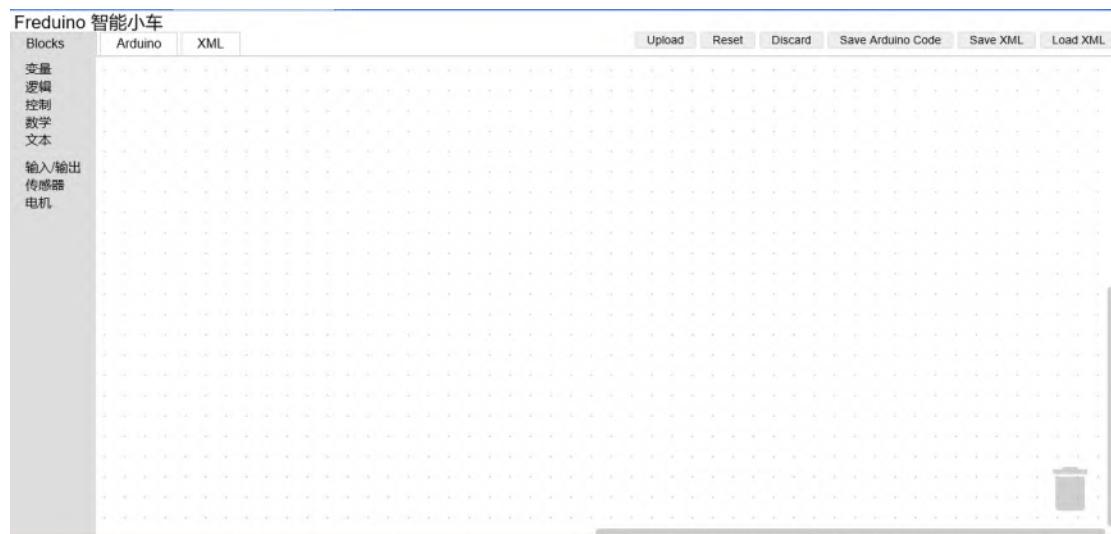


图 7-74 FreDuino 平台

a.集成块编辑器，在工具箱里增添了与硬件外设进行交互的编码块。



图 7-75 FreDuino 工具箱

b. 创建应用程序块，实现了 Blockly 块与硬件控制代码之间的转换。



图 7-76 FreDuino 创建应用程序块

c. 在构建应用程序部分，通过与硬件外设建立通信，实现代码的上传，进而完成与硬件的交互。

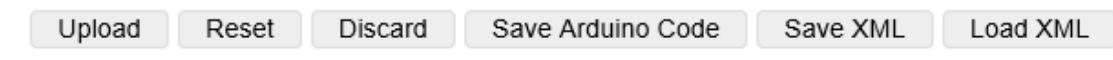


图 7-77 FreDuino 构建应用程序

FreDuino 就是通过上述三个步骤而诞生的，实现了对硬件外设的远程可视化控制。

## 7.6 小试牛刀——游戏：池塘

通过池塘导师的游戏，您已经知道如何操纵角色击败一个敌人，池塘游戏是一个开放式的比赛，您将继续扮演黄色小鸭子的角色，需要击败其余三种颜色的角色，游戏地址如下：  
<http://cooc-china.github.io/pages/blockly-games/zh-hans/pond-duck.html?lang=zh-hans>。

### 游戏规则：

- ① 综合您学到的关于 Blockly 的知识以及您在池塘导师中学习到的新模块，拼接出正确的代码块；
- ② 击败其他三个角色，游戏结束，顺利通关。

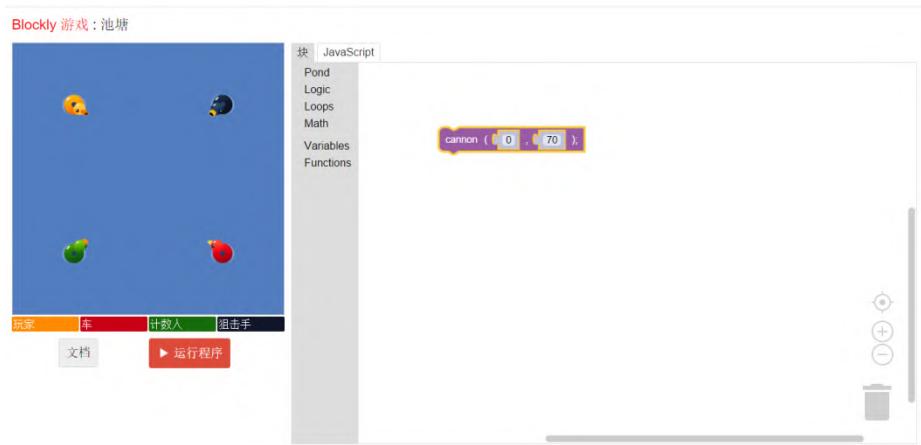


图 7-78 池塘游戏

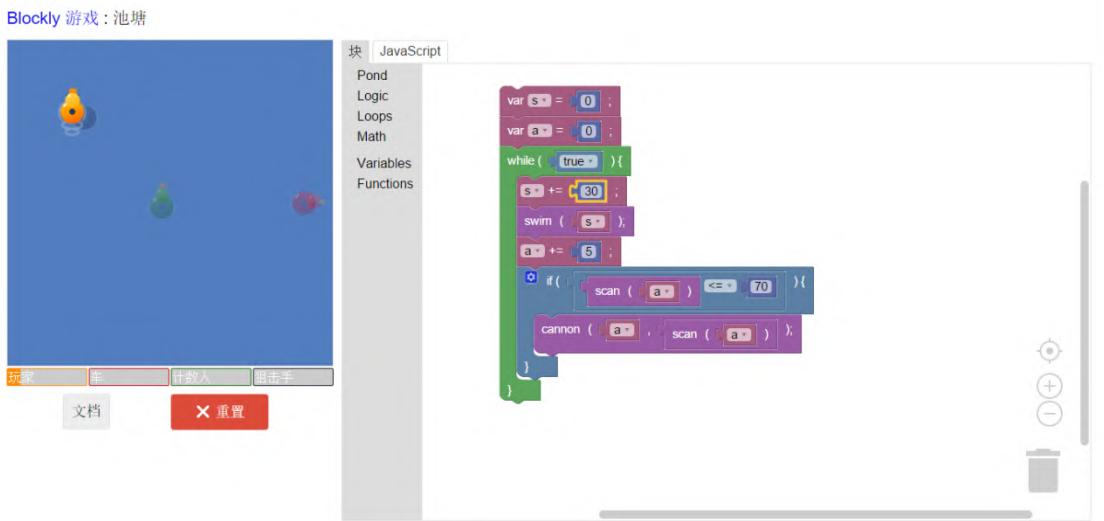


图 7-79 池塘游戏答案之一

## 7.7 本章练习

- 设计定义一个自己的工具块

## 7.8 课外拓展

### 集成开发环境

集成开发环境（IDE， Integrated Development Environment）是用于提供程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面等工具。集成了代码编写功能、分析功能、编译功能、调试功能等一体化的开发软件服务套。所有具备这一特性的软件或者软件套（组）都可以叫集成开发环境。如微软的 Visual Studio 系列， Borland 的 C++ Builder、 Delphi 系列等。该程序可以独立运行，也可以和其他程序并用。IDE 多被用于开发 HTML 应用软件。例如，许多人在设计网站时使用 IDE（如 HomeSite、 DreamWeaver 等），因为很多项任务会自动生成。

现在有大量的免费开源的和商用的 IDE。这里为大家列出最常用、最著名、最好用的几款 IDE。

#### 1. 微软 Visual Studio (VS)

VS 支持创建各种类型的程序，包括从桌面应用、Web 应用、移动 APP、到视频游戏。对于初学者到高级专业开发人员来说都是最棒的开发工具。VS 有可定制仪表板和可停靠的窗口。它支持多达 36 种不同的编程语言，如：ASP.NET、DHTML、JavaScript、Jscript、Visual Basic、Visual C#、Visual C ++、Visual F#， XAML 及更多。这个列表还在每天实时的增

长着。

## 2. Pycharm

PyCharm 是著名的 PythonIDE，由知名的 IDE 开发商 JetBrains 出品。除了最常用的 IDE 功能支持外，PyCharm 特别对 Python Web 开发进行优化设计（Django、Flask、Pyramid、Web2Py）。PyCharm 还支持 Google App Engine 和 IronPython/Jupyter。除了 Python 之外，它还支持其他 Web 开发语言：JavaScript、Node.js、CoffeeScript、TypeScript、Dart、CSS、HTML。它可以很容易地与 Git，Mercurial 和 SVN 等版本管理（VCS）工具集成。

## 3. Eclipse

被广泛应用的免费开源的 Java 编辑器和 IDE。可以灵活的适用于初学者和专业人。有很好的插件机制，支持各种各样的扩展和插件。最初是一个 Java IDE。现在扩展到支 C/C ++，Java、Perl、PHP、Python、Ruby 以及更多的语言。Eclipse 也是一个跨平台的 IDE，支持 Windows、Linux 和 Mac OS X，目前最新版本为 Eclipse Oxygen 3 (4.7.3) 版本。

## 4.Komodo

Komodo 是一个开源的跨平台多语言支持的 IDE。对于使用 Mozilla 和 Scintilla 代码库的动态编程语言来说非常有用。它广泛支持各种语言，看他的标语就是体现出来了。但是主要用于 PHP 开发，也用于 Perl、Python、Ruby、Tcl 以及 JavaScript、CSS、HTML、XML。

（来源：百家号

<https://baijiahao.baidu.com/s?id=1596921669367023584&wfr=spider&for=pc;>

百度百科

<https://baike.baidu.com/item/集成开发环境/298524?fr=aladdin>)

B  
L  
O  
C  
K  
L  
Y

### 亚利桑那州立大学物联网及机器人教育实验室主任 陈以农

Blockly is an excellent tool for understanding computational thinking and learning the first programming language. The book gives the beginners an easy entrance to the world of computer science and engineering.

### 北京景山学校 毛澄洁

本书不仅对 Blockly 的基础知识和程序结构进行了详细介绍，还结合拼图游戏，对 Blockly 的二次开发、高级应用进行深入浅出的讲解，非常实用。

### 华南师范大学附属中学 黄秉刚

本书配有丰富的案例、精选的插图、有效的教学设计，全面展示 Blockly 的编程基础知识和高级应用。本书还注意融入计算机科学文化的知识，关注信息素养的培养。

### 深圳市第三高级中学 陈向群

本书以学生最感兴趣的游戏作为切入点，通过生动有趣的案例让学生学习原本枯燥乏味的计算机编程原理，掌握计算机编程知识，培养学生的计算思维和创新思维，是一本通俗易懂、可操作性强的编程入门读物。

### 西安交通大学附属小学 向金

通过本书，无论你是学生还是老师，抑或青少年开发者，都可以从中获益，更是适合家长送给孩子的编程入门书籍。

### 上海世界外国语中学 王丽丽

本书由兰州大学周庆国教授团队结合多年的 Blockly 教师培训和教学实践经验倾力撰写。书中内容由浅入深，不仅适合初学者，也适合有一定编程基础的学习者。

