



Blockly简介

刘振坤

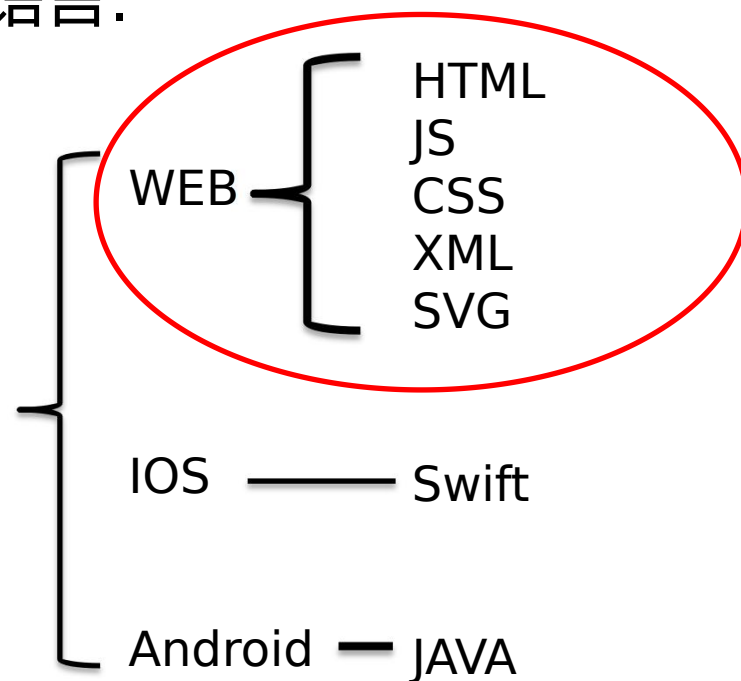
2018年1月16日

Blockly是什么

Blockly是google 2012年发布的一个完全可视化的编程语言工具，我们可以通过类似玩乐高积木的方式构建出简单的应用程序，然后根据图形生成对应的JS，Python，Dart，XML语言.



结果：打印两次"Hello World"



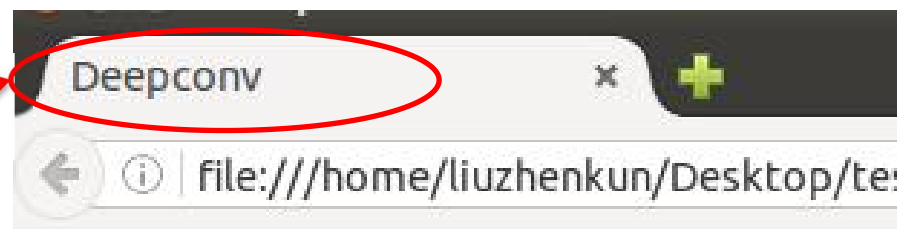
第一部分：基础篇

HTML简介

HTML(Hyper Text Markup Language)是一种超文本标记语言，使用标签来描述网页,浏览器能够自行解释html文件中标记的意义，并按照一定的格式显示。

(注：HTML中的标签都是预先定义好的)

```
2  <html>
3  <head>
4  |   <title>Deepconv</title>
5  | </head>
6  | <body>
7  | <h1>my first html</h1>
8  | <p>Hello World.</p>
9  | </body>
10 </html>
```



my first html

Hello World.

```

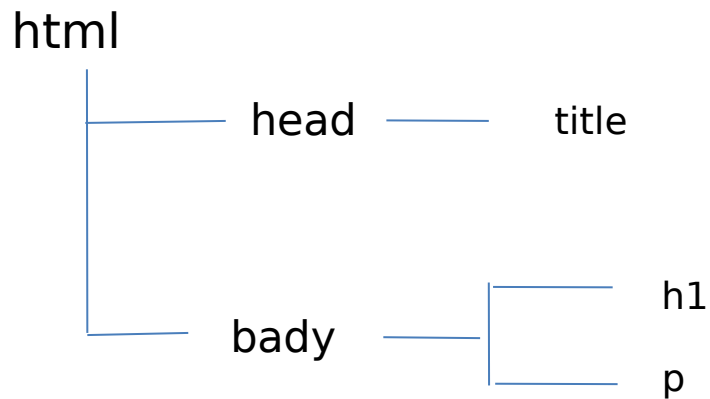
2  <html>
3  <head>
4      <title>Deepconv</title>
5  </head>
6  <body>
7  <h1>my first html</h1>
8  <p>Hello World.</p>
9  </body>
10 </html>

```

<title> <p> ...
为开始标签

</title> </p>...
为结束标签

<title>...</title>
<p>....</p>
...
这是代表一个元素



元素之间的包含关系

HTML只关心元素之间的嵌套关系，不在乎对其，大小写等

HTML元素三个重要的属性：

class	类别，可以通过它获取一类的元素，主要用于样式表
id	元素的唯一标示，可以通过id找到唯一的元素
name	与id作用类似，但是不唯一

JS简介

JavaScript 是一种基于对象和事件驱动的具有安全性的脚本语言，被设计为向 HTML 页面增加交互性。

JavaScript : 改变 HTML 内容

使用 JavaScript 来处理 HTML 内容是非常强大的功能。

实例

```
x=document.getElementById("demo") //查找元素  
x.innerHTML="Hello JavaScript"; //改变内容
```

作用：当用户点击该元素时，会弹出一个对话框

JavaScript : 写入 HTML 输出

实例

```
document.write("<h1>This is a heading</h1>");  
document.write("<p>This is a paragraph</p>");
```

document是指当前的html文件
作用：相当于手动向html文件中加两行标签

demo是html中的一个元素的id
作用：相当于修改id为'dome'的元素的文本

JavaScript : 对事件作出反应

实例

```
<button type="button" onclick="alert('Welcome!')">点击这里</button>
```

如何在HTML中使用js

在html中使用js需要通过<script></script>标签来产生一个script元素，浏览器会自行解释

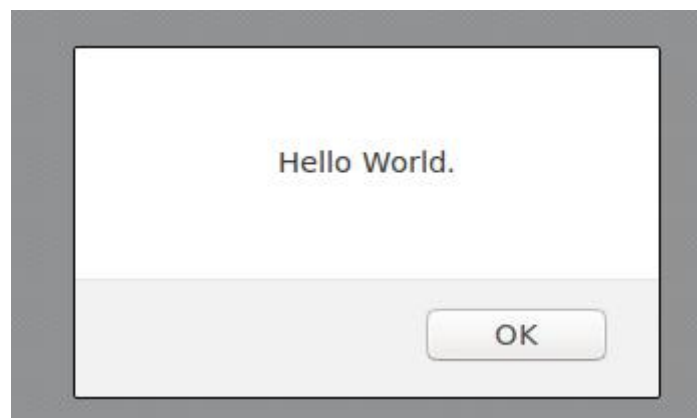
例1：在本地使用js

```
4 //test.html
5 <html>
6 <body>
7 <script type="text/javascript">
8     alert("Hello World.");
9 </script>
10 </body>
11 </html>
```

例2：使用外部js

```
3
4 //test.html
5 <html>
6 <body>
7 <script type="text/javascript" src="test.js"> </script>
8 // 将 */alert("Hello Wolrd ");/* 移到test.js文件中
9 </body>
10 </html>
```

以上两段代码运行结果相同：



JS事件处理

JS中的事件分类：

- 鼠标事件：click,mousemove,
- 键盘事件：keypress,keydown
- HTML事件：load,resize

js中的事件处理：

方法1：HTML中处理事件

```
2
3  <input value="按钮" type="button" onclick="showmsg();">
4  <script>
5      Function showmsg(){
6          Alert("HTML添加事件处理");
7      }
8  </script>
```

本方法将事件处理直接嵌套到html元素中.

优点：使用方便，容易理解.

缺点：不灵活,后续修改起来会很费劲.

JS事件处理

方法2：DOM级事件处理1

注：DOM--document object model 一切皆是节点

```
1
2 <input id="btn" value="按钮" type="button">
3 <script>
4   var btn= document.getElementById("btn");//获取按钮元素
5   btn.onclick=function(){
6     alert("DOM级添加事件处理");
7   }
8   btn.onclick=null;//如果想要删除btn的点击事件，将其置为null即可
9 </script>
```

该方法只能绑定一个事件处理函数

方法3：DOM级事件处理2

```
4 <input id="btn" value="按钮" type="button">
5 <script>
6   var btn=document.getElementById("btn");
7   btn.addEventListener("click",showmsg,false);//
8   function showmsg(
9     alert("DOM级添加事件处理程序");
10  }
11   btn.removeEventListener("click",showmsg,false);//把这个事件删除
12 </script>
```

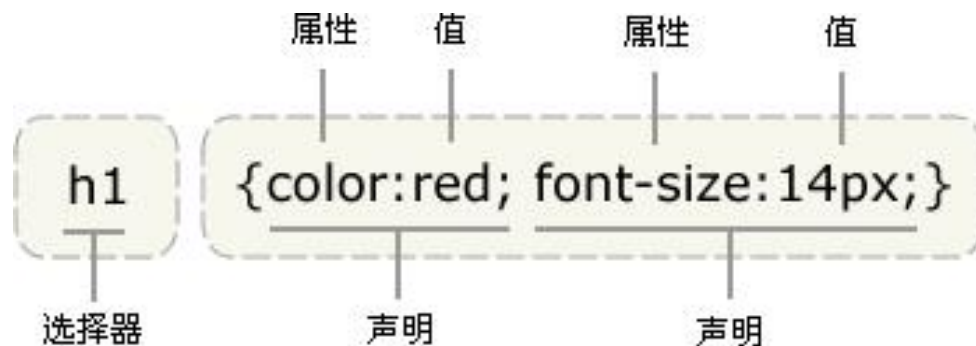
该方法可以绑定多个事件处理函数，先绑定先处理。

优点：灵活.

缺点：使用起来麻烦.

CSS简介

作用： CSS(样式表)定义html元素如何显示的,CSS可以精准的控制每一个元素，是对html语言的处理样式的最好的补充，可以把内容和格式处理相分离，减少工作量。



CSS主要由选择器和声明构成.

如以上样式表定义了元素h1的颜色为red，字体大小为14px.

CSS的选择器类型为：class，id，元素

XML简介

XML一种可扩展标记语言，与html相比，XML的标签没有被预定义，html的标签是之前预定义好的，设计宗旨是传输数据。

例：

```
<note>
  <to> George </to>    //收件人
  <from> John </from>  //发件人
  <heading> Reminder </heading> //标题
  <body> Don't forget the meeting! </body> //内容
</note>
```

JS解释XML：

```
var oParser = new DOMParser();
var dom = oParser.parseFromString(text, 'text/xml');
```

通过以上可以将XML转化为一个DOM对象。

SVG简介

SVG(Scalable Vector Graphics)是使用XML来描述二维图形和绘制程序的语言.不单独保存成文件，嵌入到html或者js文件中.

优点：

- 基于纯文本，方便编辑;
- 与jpeg，gif等相比，尺寸更小，可压缩性更强，不同分辨率下不失真;
- 图像中的文本可搜索;

例：画一个300x100的矩形，填充色为rgb(0,0,255),边宽1,边颜色为rgb(0,0,0)

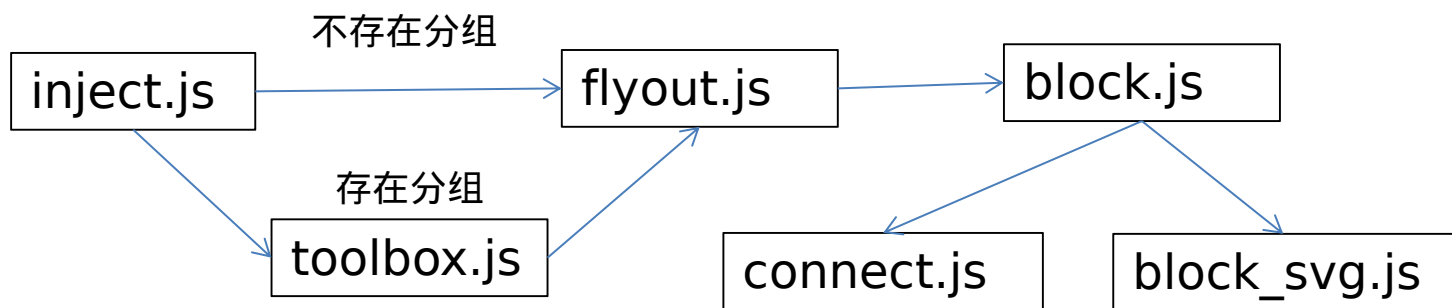
```
3 <svg width="100%" height="100%" version="1.1" xmlns="http://www.w3.org/2000/svg">
4 <rect width="300" height="100" style="fill:rgb(0,0,255); stroke-width:1;
5     stroke:rgb(0,0,0)" />
6 </svg>
```

第二部分：Blockly篇

blockly的目录结构

core	核心代码目录，包括事件处理， block ， flyout 定义等
blocks	组件定义目录，包括print，text，list等具体的组件
generators	代码生成目录，包括每个组件对应生成代码的代码
msg	语言目录，包括一些语言支持
domes	演示目录，一些例子

core目录下文件之间的主要关系：

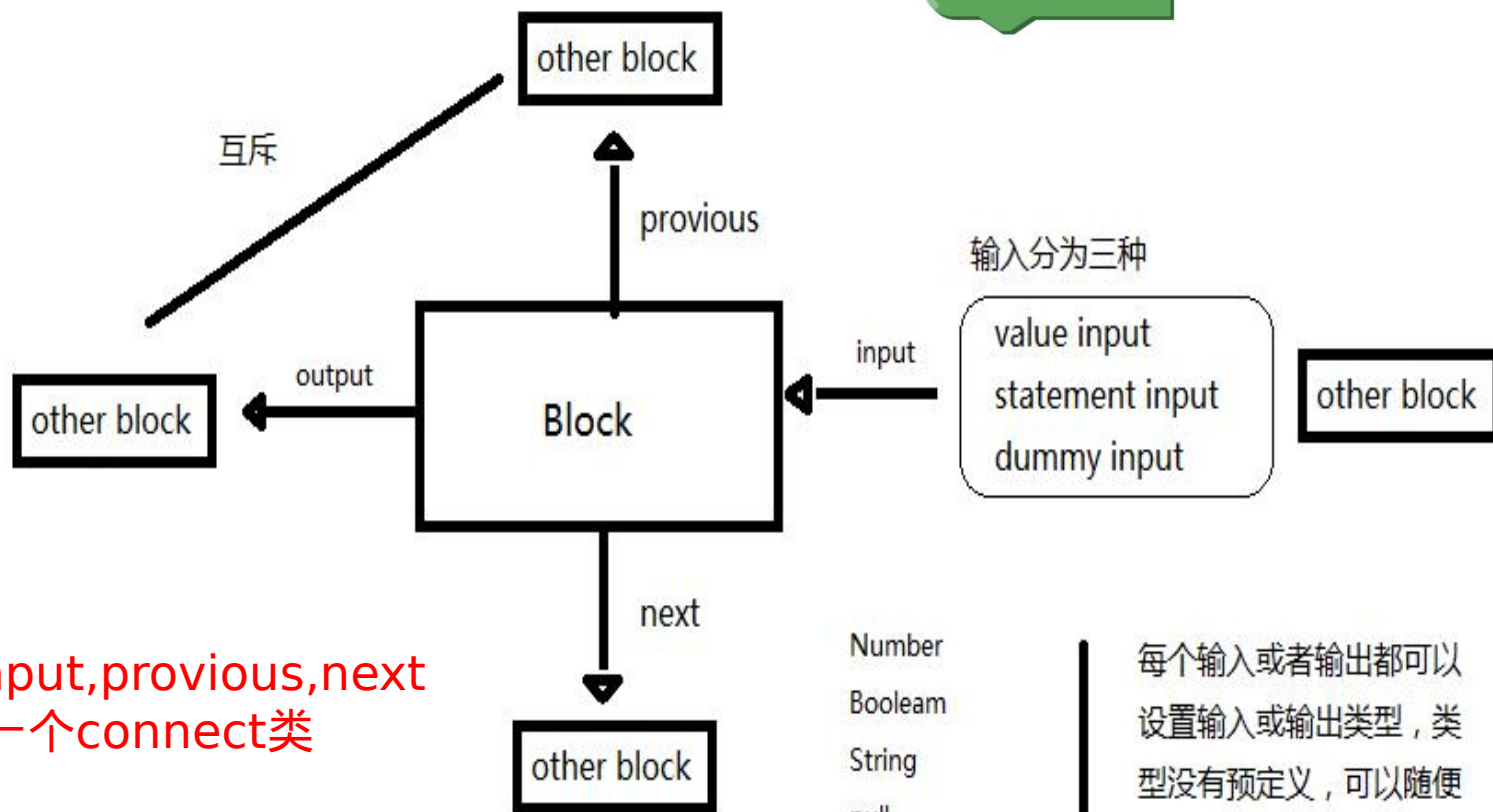


inject.js: 入口
toolbox.js: 分组显示工具
flyout.js: 一个弹窗，显示block

block.js: 定义块的文件
connect.js: 块的向四周链接的属性
block_svg.js: 显示块(svg画出来的)

block类

blockly的所有操作都是基于block.



output,input,previous,next
分别对应一个connect类

Number
Boolean
String
null
other(没有预定义)

每个输入或者输出都可以
设置输入或输出类型，类
型没有预定义，可以随便
写，但是要完全对应(区分
大小写)

connect类

每一个connect都有以下两个重要属性：

connect	
type	connect的类型，决定了与之相连接的connect的类型
x,y	connect的位置

type类型

PROVIOUS_STATEMENT
NEXT_STATEMENT
OUTPUT_VALUE
INPUT_VALUE

connect的类型与block外接属性的关系

previous	PROVIOUS_STATEMENT
next	NEXT_STATEMENT
output	OUTPUT_VALUE
input::statement	NEXT_STATEMENT
input::value/dummy	INPUT_VALUE


不同类型之间的链接对应关系

PROVIOUS_STATEMENT	NEXT_STATEMENT
OUTPUT_VALUE	INPUT_VALUE

blockly的加载过程

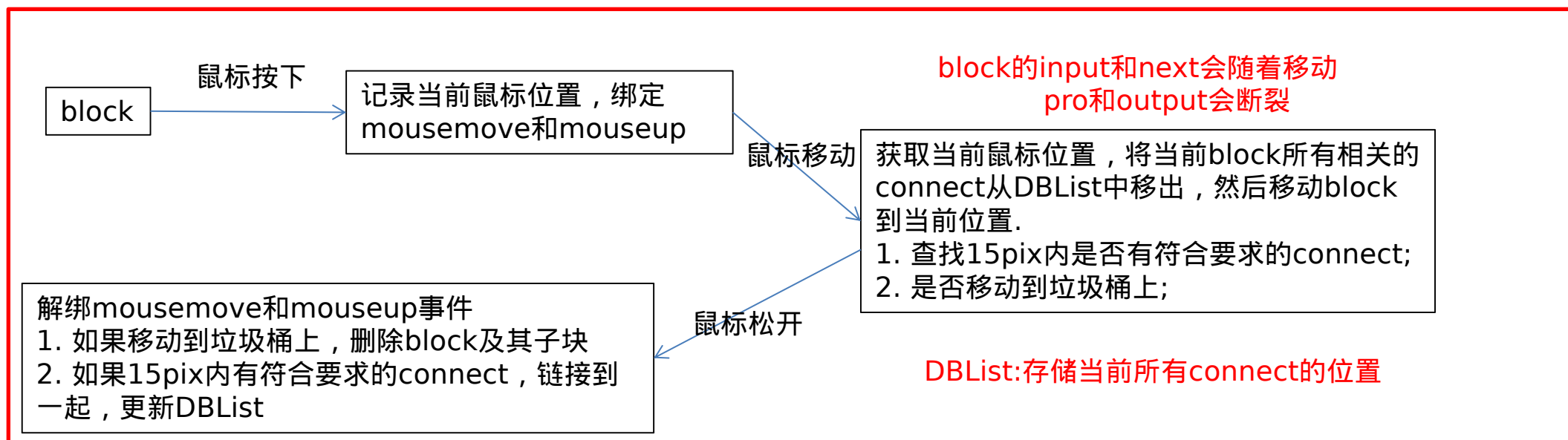
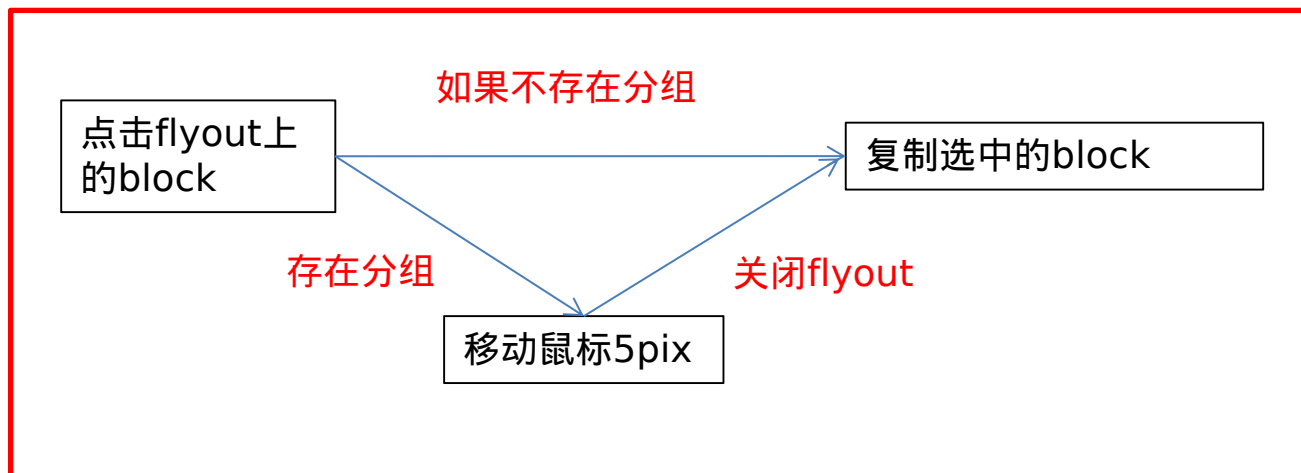
1. 每一个blockly的使用都会以一个html文件作为入口,参考domes/generators/index.html讲解.
 2. toolbox中每个block在对应的blocks目录下的分类中都可以找到对应的定义文件.
- 例：text 在blocks/text.js->Blockly.Blocks['text']...

```
31  Blockly.Blocks['text'] = {
32    // Text value.
33    init: function() {
34      this.setHelpUrl(Blockly.Msg.TEXT_TEXT_HELPURL);
35      this.setColour(160);
36      this.appendDummyInput()
37        .appendTitle(new Blockly.FieldImage(Blockly.pathToBlockly +
38        'media/quote0.png', 12, 12))
39        .appendTitle(new Blockly.FieldTextInput(''), 'TEXT')
40        .appendTitle(new Blockly.FieldImage(Blockly.pathToBlockly +
41        'media/quote1.png', 12, 12));
42      this.setOutput(true, 'String');
43      this.setTooltip(Blockly.Msg.TEXT_TEXT_TOOLTIP);
44    }
45  };
46
```



block鼠标事件处理

block都是通过flyout(弹窗)显示出来的



代码生成过程

每个block都有一个自己的代码生成函数，保存在generators/对应分组的js文件中。
每个block生成代码时只关心两件事：1. 自己的输入语句； 2. 自身；

例：text和print组件的代码生成函数保存在generators/text.js中

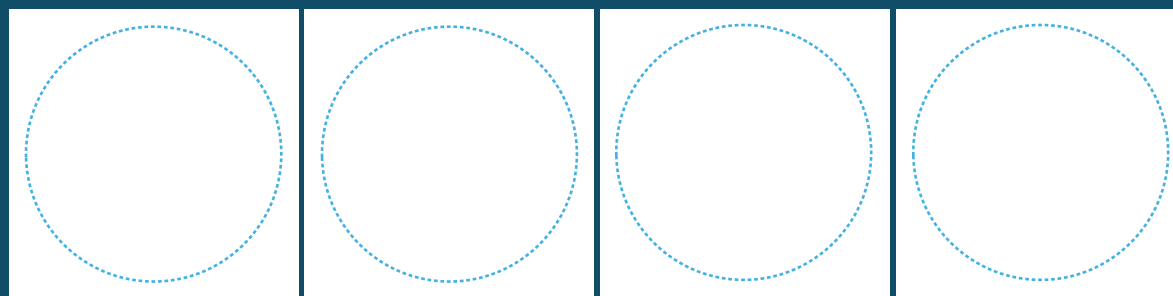
```
241 Blockly.JavaScript['text'] = function(block) {
242   // Text value.
243   var code = Blockly.JavaScript.quote_(block.getTitleValue('TEXT'));
244   return [code, Blockly.JavaScript.ORDER_ATOMIC];
245 };
246
247 Blockly.JavaScript['text_print'] = function(block) {
248   // Print statement.
249   var argument0 = Blockly.JavaScript.valueToCode(block, 'TEXT',
250     Blockly.JavaScript.ORDER_NONE) || '\'\';
251   return 'window.alert(' + argument0 + ');\\n';
252 };
253
```


for语句代码生成

```
for(int i=0; i < [??1]; i++)  
{  
    ??2  
}
```

??1 对应的block中名字为'TIMES'的值输入
??2 对应的block中名字为'DO'的语句输入

```
Blockly.JavaScript['controls_repeat_ext'] = function(block) {  
    // Repeat n times (external number).  
    var repeats = Blockly.JavaScript.valueToCode(block, 'TIMES',  
        Blockly.JavaScript.ORDER_ASSIGNMENT) || '0';  
    var branch = Blockly.JavaScript.statementToCode(block, 'DO');  
    if (Blockly.JavaScript.INFINITE_LOOP_TRAP) {  
        branch = Blockly.JavaScript.INFINITE_LOOP_TRAP.replace(/%1/g,  
            '\'' + block.id + '\'') + branch;  
    }  
    var code = '';  
    var loopVar = Blockly.JavaScript.variableDB_.getDistinctName(  
        'count', Blockly.Variables.NAME_TYPE);  
    var endVar = repeats;  
    if (!repeats.match(/^\\w+$/) && !Blockly.isNumber(repeats)) {  
        var endVar = Blockly.JavaScript.variableDB_.getDistinctName(  
            'repeat_end', Blockly.Variables.NAME_TYPE);  
        code += 'var ' + endVar + ' = ' + repeats + ';\n';  
    }  
    code += 'for (var ' + loopVar + ' = 0; ' +  
        loopVar + ' < ' + endVar + '; ' +  
        loopVar + '++) {\n' +  
        branch + '}\n';  
    return code;  
};
```

THANK YOU FOR YOUR ATTENTION!
