生成代码

大多数Blockly应用程序需要将块转换为代码以执行。本页描述如何将代码生成器添加到自定义块。

首先，转到generators/目录并选择与您要生成的语言（JavaScript，Python，PHP，Lua，Dart等）相对应的子目录。假设您的代码块不适合现有类别，请创建一个新的JavaScript文件。这个新的JavaScript文件需要包含在<script ...>编辑器的HTML文件中的标记列表中。

一个典型的块的代码生成器如下所示：

```
Blockly.JavaScript['text_indexOf'] = function(block) {
  // Search the text for a substring.
  var operator = block.getFieldValue('END') == 'FIRST' ? 'indexOf' :
'lastIndexOf';
  var subString = Blockly.JavaScript.valueToCode(block, 'FIND',
      Blockly.JavaScript.ORDER_NONE) || '\'\'';
  var text = Blockly.JavaScript.valueToCode(block, 'VALUE',
      Blockly.JavaScript.ORDER_MEMBER) || '\'\'';
  var code = text + '.' + operator + '(' + subString + ')';
  return [code, Blockly.JavaScript.ORDER_MEMBER];
};
```

采集参数

任何块的代码生成器的首要任务是收集所有参数和字段数据。此任务通常使用几种功能：

- getFieldValue
- valueToCode
- statementToCode

getFieldValue

```
block.getFieldValue('END')
```

此函数从指定名称的字段返回值。

- 对于文本字段，此函数返回输入的文本。例如“ Hello World”。
- 如果是下拉菜单，此函数将返回与所选选项关联的语言无关的文本。英文块可能有一个下拉菜单，其中选择了“ first”一词，而德语的相同下拉菜单将显示“ erste”。代码生成器不必知道所有可能的人类语言，因此该getFieldValue函数将返回创建下拉菜单时指定的与语言无关的文本（Blockly的核心块通常使用大写英文单词，例如“ FIRST”）。
- 对于变量下拉列表，此函数返回变量下拉列表的面向用户的名称。重要的是要注意，此名称不必与生成的代码中使用的变量名称相同。例如，变量名“ for”在Blockly中是合法的，但在大多数语言中会与保留字冲突，因此将重命名为“ for2”。同样，阿拉伯语变量名“ متغير”在Blockly中是合法的，但在大多数语言中

都是非法的，因此将其重命名为“ _D9_85_D8_AA_D8_BA_D9_8A_D8_B1”。要获取生成的代码中可能使用的Blockly变量名称，请使用以下调用：

```
Blockly.JavaScript.variableDB_.getName(block.getFieldValue('VAR'),
Blockly.Variables.NAME_TYPE);
```

请注意，JavaScript应改为相应的语言（Python，PHP，Lua，Dart因为每一种语言，等等）拥有的保留字不同的列表。

valueToCode

```
Blockly.JavaScript.valueToCode(block, 'FROM', Blockly.JavaScript.ORDER_ADDITION)
|| '0'
```

此函数查找连接到命名值输入（'FROM'）的块，生成该块的代码，然后将代码作为字符串返回。在未连接输入的情况下，此函数返回null，这就是为什么通常在函数后加上布尔值“或”和默认值的原因。因此，在上面的示例中，如果没有块附加到名为“ FROM”的输入，则此输入的默认代码将为字符串“ 0”。

第三个参数指定嵌入所需的操作信息的顺序。每个语言生成器都有一个优先顺序列表。该 valueToCode功能需要传递对应于将要施加到所述返回代码的最大的力的顺序值。valueToCode如果需要，可以将代码包装在括号中。有关详细信息，请参阅运算符优先级页面。

请注意，JavaScript应改为相应的语言（Python，PHP，Lua，Dart，等）。

statementToCode

```
Blockly.JavaScript.statementToCode(block, 'DO')
```

此函数查找连接到指定语句输入的嵌套块堆栈，为该堆栈生成代码，缩进代码，然后将代码作为字符串返回。如果未连接输入，此函数将返回一个空字符串。

请注意，JavaScript应改为相应的语言（Python，PHP，Lua，Dart，等）。

汇总代码

一旦收集了所有参数，就可以汇编最终代码。对于大多数块而言，这很简单。这是一个while循环的示例：

```
var code ='while ('+ argument0 +') {\n'+ branch0 +'}\n';
return code;
```

值块（那些确实返回值的块）要复杂一些。这是基本算术运算符（加号，减号等）的示例：

```
var code = argument0 +' '+operator+' '+ argument1;
```

本示例说明了操作顺序问题。考虑两个相连的算术块构成表达式的情况(2 * (3 + 4))。使用上面的代码片段，加法块将返回字符串，"3 + 4" 而乘法块将其用作return的输入"2 * 3 + 4"。这个结果是不正确的，因为在执行时，3它将与乘法紧密结合。

为了解决这个问题，值块必须返回一个包含两个值的列表：代码和相应的订单值：

```
return [ code , Blockly.JavaScript.ORDER_ADDITION ];
```

每个语言生成器都有一个优先顺序列表。返回的订单值指定将代码绑定在一起的最小作用力。有关详细信息，请参阅运算符优先级页面。

如果生成的代码要求将子块的代码包含两次，则应缓存参数以提高效率并防止产生副作用。