缓存参数

从块生成代码时，经常会发现需要多次使用子块的返回值。考虑一个值块，该值块查找并返回列表的最后一个元素。该块本身将有一个输入（一个列表），并返回一个值（最后一个元素）。这是JavaScript的生成器：

```
var code = arg0 + '[' + arg0 + '.length - 1]';
```

如果arg0是变量名，则此生成器返回完全可接受的JavaScript：

```
aList[aList.length - 1]
```

但是，如果arg0是函数调用，则此生成器可能具有意外行为。考虑以下代码：

```
randomList()[randomList().length - 1]
```

返回的两个值的长度可能不同，从而导致超出范围条件。此外，如果函数调用具有副作用，则不希望两次调用它。

有两个解决方案。语句块应使用临时变量。值块应使用实用程序功能。

临时变量

最简单的解决方案是将有问题的输入分配给一个临时变量。必须注意，此变量不会偶然与现有变量发生冲突。以下代码显示了一个语句块中的临时变量示例，该变量警告列表的最后一个元素。

```
var listVar = Blockly.JavaScript.variableDB_.getDistinctName(
    'temp_list', Blockly.Variables.NAME_TYPE);
var code = 'var ' + listVar + ' = ' + arg0 + ';\n';
code += 'alert(' + listVar + '[' + listVar + '.length - 1]);\n';
```

该getDistinctName调用采用所需变量名称（“ temp_list”）的参数，并将返回要使用的非冲突名称（可能是“ temp_list2”）。

临时变量的缺点是，如果令人讨厌的输入已经是一个变量，则将生成冗余代码：

```
var temp_list = foo;
alert(temp_list[temp_list.length - 1]);
```

要生成更简洁的代码，请检查有问题的输入是否为简单文字，并相应地生成代码：

```
if (arg0.match(/^\w+$/)) {
  var code = 'alert(' + arg0 + '[' + arg0 + '.length - 1]);\n';
} else {
  var listVar = Blockly.JavaScript.variableDB_.getDistinctName(
      'temp_list', Blockly.Variables.NAME_TYPE);
  var code = 'var ' + listVar + ' = ' + arg0 + ';\n';
  code += 'alert(' + listVar + '[' + listVar + '.length - 1]);\n';
}
```

有关Blockly.JavaScript.controls_forEach临时变量的工作示例，请参见。

临时变量在语句块（在这种情况下为警报）中效果很好，在语句块中生成的代码可能跨越多行。但是，它们不能在必须位于一行中的值块中使用。对于值块，必须使用实用程序功能而不是临时变量。

实用功能

定义实用程序功能是一种创建功能块的强大方法，该功能块的工作级别高于基础语言。除非使用了实用程序功能，否则它们不会生成，并且无论使用多少次，它们只会生成一次。

```
var functionName = Blockly.JavaScript.provideFunction_(
    'list_lastElement',
    [ 'function ' + Blockly.JavaScript.FUNCTION_NAME_PLACEHOLDER_ + '(aList) {',
      '  // Return the last element of a list.',
      '  return aList[aList.length - 1];',
      '}']);
// Generate the function call for this block.
var code = functionName + '(' + arg0 + ')';
return [code, Blockly.JavaScript.ORDER_FUNCTION_CALL];
```

在上面的示例中，我们请求定义一个名为的效用函数 list_lastElement（尽管实际名称可能有所不同，以避免与用户变量冲突）。有关Blockly.JavaScript.text_endString实用程序功能的工作示例，请参见。