验证器

验证器是一种函数，它接受字段的新值，然后对其进行操作。 它们是自定义字段的简单方法。 它们允许您在字段的值更改时触发功能，修改输入或限制可接受的值。

一些常见的例子：

- 将文本字段限制为仅接受字母。
- 要求文本字段为非空。
- 要求日期在将来。
- 根据下拉菜单修改块的形状。

注意：验证程序修改字段的值，而不是其文本。

验证器的类型

验证器在不同的时间执行，具体取决于它们是哪种类型的验证器。

类验证器是字段类型的类定义的一部分，通常用于限制字段允许的值的类型（例如，数字字段仅接受数字字符）。 类验证器在传递给该字段的所有值（包括传递给构造函数的值）上运行。

有关类验证器的更多信息，请参见创建自定义字段中的实现类验证器部分。

本地验证器是在构造字段时定义的。 本地验证器对传递给该字段的所有值（传递给构造函数的值除外）运行。 这意味着它们继续运行：

- XML中包含的值。
- 传递给setValue的值。
- 传递给setFieldValue的值。
- 值由用户更改。

类验证器在本地验证器之前运行，因为它们的作用类似于网守。 他们在传递值之前确保该值是正确的类型。

有关值验证顺序和值的一般信息，请参阅值。

注册本地验证器

可以通过两种方式注册本地验证器：

1. 直接添加到字段的构造函数中。
   注意：字段构造函数的签名可能会有所不同，具体取决于字段类型。

```
Blockly.Blocks['validator_example'] = {
  init: function() {
    // Remove all 'a' characters from the text input's value.
    var validator = function(newValue) {
      return newValue.replace(/\a/g, '');
    };
```

```
    this.appendDummyInput()
        .appendField(new Blockly.FieldTextInput('default', validator));
  }
};
```

2. 使用setValidator

```
Blockly.Blocks['validator_example'] = {
  init: function() {
    // Remove all 'a' characters from the text input's value.
    var validator = function(newValue) {
      return newValue.replace(/\a/g, '');
    };

    var field = new Blockly.FieldTextInput('default');
    field.setValidator(validator);

    this.appendDummyInput().appendField(field);
  }
};
```

以上两种方法都可以包装在扩展名中，以支持JSON格式。

该字段的值可能会有所不同，具体取决于要验证的字段的类型（例如，数字字段将存储数字，而文本输入字段将存储字符串），因此最好在创建表单之前阅读特定字段的文档。

注意：仅可编辑字段接受验证器，因此请务必检查特定字段的文档。

返回值

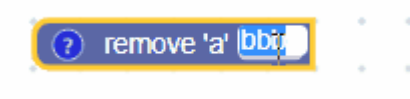
验证器的返回值确定该字段下一步做什么。 有三种可能性：

1. 修改后的返回值

修改或不同的值，然后成为字段的新值。 这通常用于清除值，例如通过删除结尾的空格。

修改验证器的示例：

```
// Remove all 'a' characters from the text input's value.
var validator = function(newValue) {
  return newValue.replace(/\a/g, '');
};
```



2. 空返回值

Null，表示给定值无效。 在大多数情况下，该字段将忽略输入值。 确切的行为由字段的doValueInvalid_函数指定。

空验证器的示例：

```
// Any value containing a 'b' character is invalid.  Other values are valid.
var validator = function(newValue) {
  if (newValue.indexOf('b') != -1) {
    return null;
  }
  return newValue;
};
```

3. 未定义的返回值

未定义（或没有return语句）或输入值，这意味着输入值应成为字段的新值。 这些类型的验证器通常充当更改侦听器。

侦听器验证器的示例：

```
// Log the new value to console.
var validator = function(newValue) {
  console.log(newValue);
};
```

注意: 显示文本如何不一定反映字段的值。

this 的值

在验证器内部，这是指字段，而不是块。 如果需要访问验证器中的块，请使用getSourceBlock函数。 您还可以使用bind函数设置在其中调用验证器的上下文。

使用getSourceBlock的示例代码：

```
Blockly.Blocks['colour_match'] = {
  init: function() {
    this.appendDummyInput()
        .appendField(new Blockly.FieldColour(
            null, this.validate
        ), 'COLOUR');
    this.setColour(this.getFieldValue('COLOUR'));
  },

  validate: function(colourHex) {
```

```
    this.getSourceBlock().setColour(colourHex);
  }
};
```

绑定open_in_new的示例代码：

```
Blockly.Blocks['colour_match'] = {
  init: function() {
    this.appendDummyInput()
      .appendField(new Blockly.FieldColour(
          null, this.validate.bind(this)
      ), 'COLOUR');
    this.validate(this.getFieldValue('COLOUR'));
  },

  validate: function(colourHex) {
    this.setColour(colourHex);
  }
};
```