升级自定义字段

在2019年7月（版本2.20190722）中，添加了更编纂的字段API。 它旨在尽可能地向后兼容。 这意味着，如果您在2019年7月之前创建了自定义字段，则很有可能会继续工作。 在决定是否需要升级自定义字段之前，应通读“危险区域”部分，并对您的字段进行全面测试。

由于在2019年7月之前字段之间缺乏标准化，因此很难涵盖开发人员可能需要进行的所有更改。 本文档尝试涵盖所有可能的更改，但是如果本文档未涵盖您感兴趣的内容，请阅读有关获得升级帮助的部分。

注意：如果您的自定义字段数量很少，则使用自定义字段创建说明简单地重写它们可能会比尝试逐步升级它们更容易。

危险区域

危险区域是已知的API发生更改的地方，您的字段可能会损坏。

setText

Blockly核心不再调用setText函数，因此，如果setText函数包含逻辑，则需要将其移至值处理函数集，getText函数和呈现函数（取决于setText函数的确切含义） 在做）。

```
CustomFields.UpgradeField.prototype.setText = function(newText) {
  // Do validation.
  if (typeof newText != 'string' || newText === this.text_) {
    return;
  }

  // Fire event.
  if (this.sourceBlock_ && Blockly.Events.isEnabled()) {
    Blockly.events.fire(new Blockly.Events.BlockChange(
        this.sourceBlock_, 'field', this.name, this.text_, newText
    ));
  }

  // Update text value.
  this.text_ = 'prefix' + newText;

  // Rerender.
  this.size_.width = 0;
};
```

变成

```
CustomFields.UpgradeField.prototype.doClassValidation_ = function(newValue) {
  if (typeof newValue != 'string') {
    return null;
  }
```

```
  return newValue;
};

CustomFields.UpgradeField.prototype.getText = function() {
  return  'prefix' + this.value_;
}
```

Blockly自动处理：

- 检查新值是否与旧值不同
- 更新值
- 引发变更事件
- 渲染字段

您将需要处理：

- 值验证（doClassValidation_）
- 字段文本（getText）
- 字段的UI

注意：我们建议您将任何setText调用更改为setValue调用，因为setValue支持验证，而setText不支持。

推荐升级

建议的升级是已更改字段API的地方，但是如果您选择不进行更改，则您的字段很可能仍会工作。

可序列化

有关EDITABLE和SERIALIZABLE属性的更多信息，请参见Editable和Serializable属性。

```
CustomFields.UpgradeField.prototype.SERIALIZABLE = true;
```

下面的警告是可忽略的，但是您可以通过定义SERIALIZABLE属性来解决它：

```
Detected an editable field that was not serializable. Please define SERIALIZABLE
property as true on all editable custom fields. Proceeding with serialization.
```

上面的警告意味着Blockly认为您希望将该字段序列化为XML（因为EDITABLE属性为true），但是直到您定义SERIALIZABLE属性才能确定。 如果您选择不使用此功能，那么一切将正常运行，并且您的字段将被序列化，但是您会收到控制台警告。

size_.width

this.size_.width = 0; 变成 this.isDirty_ = true;

下面的警告是可忽略的，但是您可以通过设置isDirty_属性而不是size_.width属性来解决此警告：

```
Deprecated use of setting size_.width to 0 to rerender a field. Set field.isDirty_
to true instead.
```

上面的警告表示Blockly已检测到您正在使用旧的方法重新渲染字段，并且希望您使用新的方法。

init

已将init函数制成模板函数，以减少子类中的重复代码。

```
CustomFields.UpgradeField.prototype.init = function() {
  if (this.fieldGroup_) {
    // Already initialized once.
    return;
  }

  // Call superclass.
  CustomFields.UpgradeField.superClass_.init.call(this);

  // Create DOM elements.
  this.extraDom_ = Blockly.utils.createSvgElement('image',
      {
        'height': '10px',
        'width': '10px'
      });
  this.extraDom_.setAttributeNS('http://www.w3.org/1999/xlink',
      'xlink:href', 'image.svg');
  this.extraDom_.style.cursor = 'pointer';
  this.fieldGroup_.appendChild(this.extraDom_);

  // Bind events.
  this.mouseOverWrapper_ =
    Blockly.bindEvent_(
        this.getClickTarget_(), 'mouseover', this, this.onMouseOver_);
  this.mouseOutWrapper_ =
    Blockly.bindEvent_(
        this.getClickTarget_(), 'mouseout', this, this.onMouseOut_);

  // Render.
  this.setValue(this.getValue());
};
```

变成

```
CustomFields.UpgradeField.prototype.initView = function() {
  CustomFields.UpgradeField.superClass_.initView.call(this);

  this.extraDom_ = Blockly.utils.dom.createSvgElement('image',
      {
```

```
          'height': '10px',
          'width': '10px'
        });
    this.extraDom_.setAttributeNS('http://www.w3.org/1999/xlink',
        'xlink:href', 'image.svg');
    this.extraDom_.style.cursor = 'pointer';
    this.fieldGroup_.appendChild(this.extraDom_);
};

CustomFields.UpgradeField.prototype.bindEvents_ = function() {
    CustomFields.UpgradeField.superClass_.bindEvents_.call(this);

    this.mouseOverWrapper_ =
      Blockly.bindEvent_(
          this.getClickTarget_(), 'mouseover', this, this.onMouseOver_);
    this.mouseOutWrapper_ =
      Blockly.bindEvent_(
          this.getClickTarget_(), 'mouseout', this, this.onMouseOut_);
};
```

这意味着现在块式自动处理：

- 检查该字段是否已经初始化
- 创建fieldGroup_
- 渲染领域
- 绑定工具提示并显示编辑器事件

您将需要处理：

- 添加其他DOM元素（initView）
- 添加其他事件绑定（bindEvents_）
- 处理事件绑定（dispose）

重要提示：所有DOM创建都应在initView内完成，因为这是最有效的。 如果在构造函数setText，setValue或render_中发生DOM创建，请考虑重构。

onMouseDown_

```
CustomFields.UpgradeField.prototype.onMouseDown_ = function(e) {
    // ...
};
```

变成

```
CustomFields.UpgradeField.prototype.showEditor_ = function() {
    // ...
}
```

我们建议您覆盖showEditor_函数来处理鼠标单击，而不要覆盖onMouseDown_函数，因为它允许输入通过手势系统。

setValue

setValue函数现在是一个模板函数，用于减少子类中的重复代码。 如果setValue函数包含逻辑，请考虑对其进行重构以适合“值处理”中所述的值处理路径。

text_

我们建议您不要直接访问或更新字段的text_属性。 而是使用getText函数访问字段的用户可读文本，并使用setValue函数更新字段的存储值。

获得升级帮助

注意：在寻求升级帮助之前，请尽力使用所提供的信息独立升级您的领域。

提供什么

当寻求帮助时，最好问一些具体问题：

不建议：“此字段出了什么问题？”

也不建议：“帮我升级此字段”。

推荐：“字段文本未正确更新。”

也有必要向协助您的人员提供资源。 这些文件应该易于他人使用。

不建议：

- 图片的代码
- 代码不完整

建议：

- 在整齐的文本中包含所有问题代码
- 问题相关的GIF图片
- 能重现错误字段问题的步骤
- 您要从中升级的blockly版本

在哪里发布

在blockly的开发者论坛上发布升级问题。

如果您确定问题出在区块核心，则您也可以在区块GitHub上发布问题。 如果您决定发布问题，请填写所有要求的信息。