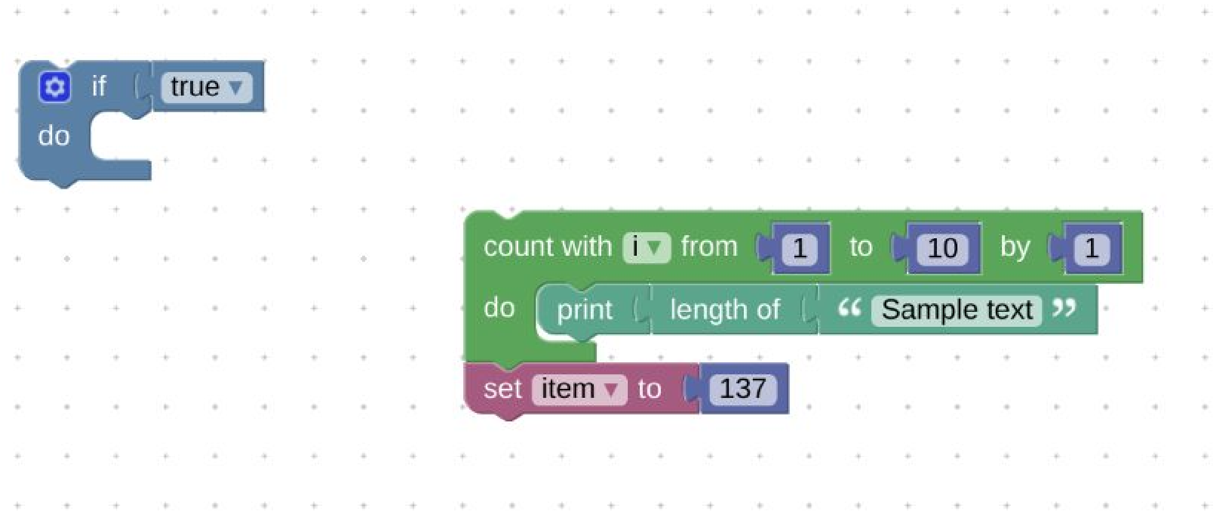
Blockly XML Structure

Author: fenichel@
Last revised September 2015

Examples

Note: The blocks are described using JSON and loaded separately. XML is only used for storing user's code and only includes the values that the user can modify (including the locations of blocks in the workspace). More info on the JSON format can be found here.

Sample blocks in workspace:


if
true
do
count with i from 1 to 10 by 1
do print length of “Sample text”
set item to 137

Corresponding XML:

```
<xml xmlns="http://www.w3.org/1999/xhtml">
  <block type="controls_if" x="38" y="62">
    <value name="IF0">
      <block type="logic_boolean">
        <field name="BOOL">TRUE</field>
      </block>
    </value>
  </block>
  <block type="controls_for" x="263" y="138">
    <field name="VAR">i</field>
    <value name="FROM">
      <block type="math_number">
        <field name="NUM">1</field>
      </block>
    </value>
    <value name="TO">
      <block type="math_number">
        <field name="NUM">10</field>
      </block>
    </value>
    <value name="BY">
      <block type="math_number">
        <field name="NUM">1</field>
      </block>
    </value>
    <statement name="DO">
      <block type="text_print">
        <value name="TEXT">
          <block type="text_length">
            <value name="VALUE">
              <block type="text">
                <field name="TEXT">Sample text</field>
              </block>
            </value>
          </block>
        </value>
      </block>
    </statement>
    <next>
      <block type="variables_set">
        <field name="VAR">item</field>
        <value name="VALUE">
          <block type="math_number">
            <field name="NUM">137</field>
          </block>
        </value>
      </block>
    </next>
  </block>
</xml>
```

Tags

In general, unknown tags or attributes (such as the legacy `id` attribute on `blocks`) are silently ignored and do not need to be included in the serialized output of any newer version of Blockly.

XML

The top-level tag is an `xml` tag with namespace `"http://www.w3.org/1999/xhtml"`.

```
<xml xmlns="http://www.w3.org/1999/xhtml">
  ...
</xml>
```

Block

All tags directly nested in the top-level `xml` tag are `blocks`. These are called top-level blocks (TLB), in contrast to blocks that exist on the inputs and connections of a top-level block. There are two top-level blocks in this example.

```
<xml xmlns="http://www.w3.org/1999/xhtml">
  <block type="controls_if" x="38" y="62">
    ...
  </block>
  <block type="controls_for" x="263" y="138">
    ...
  </block>
</xml>
```

`Block` attributes

All `blocks` have an attribute named `type`. This gives the name of the prototype block to load from the `BlockFactory` and populate according to the XML. This is variously called name, id, and type in code and documentation.

All top-level `blocks` have `x` and `y` attributes. These denote the location of the `block` in the workspace. Non-TLBs calculate their locations depending on the `blocks` that they are connected to, and do not have `x` and `y` attributes.

Tags that can be found inside a <`block`/> tag are: `value`, `field`, `next`, and `statement`.

Field

```
<block type="logic_boolean">
        <field name="BOOL">TRUE</field>
</block>
```

`Field` attributes

A `field` has one attribute, the `name`. If the `name` does not correspond with a `field` on the `block` being constructed, the `field` is skipped.

A `field` is always nested directly inside a `block`. No other tags may be nested inside a `field`.

The text between the start and end tags of the `field` holds the value of the `field`, and must be parsed by a `Field` object of the appropriate type.

A `field` tag does not contain any information about the type of data it contains (angle, date, variable, input, etc). That information is stored in the JSON and by extension in the `block` obtained from a `BlockFactory`.

Value

```
    <value name="IF0">
      <block type="logic_boolean">
        ...
      </block>
    </value>
```

`Value` attributes

A `value` has a `name` attribute. This is the same as the `name` of the corresponding `InputValue` on the `Block` being generated. If the `name` does not correspond with an `input` the `value` is skipped.

A single `block` and its descendents can be nested inside a `value` tag.

`Next`

```
<next>
      <block type="variables_set">
        ...
      </block>
    </next>
```

`Next` attributes

A `next` tag has no attributes.

A single `block` and its descendents can be nested inside a `next` tag.

`Statement`

```
<statement name="DO">
      <block type="text_print">
        ...
      </block>
    </statement>
```

`Statement` attributes

A `statement` has a `name` attribute. This is the same as the `name` of the corresponding `InputStatement` on the `Block` being generated. If the `name` does not correspond with an `input` the `statement` is skipped.

A single `block` and its descendents can be nested inside a `statement` tag.