Improving the Usability of App Inventor through Conversion Between Blocks and Text

Karishma Chadha ‘14
Franklyn Turbak, Advisor
Wellesley College Computer Science Department

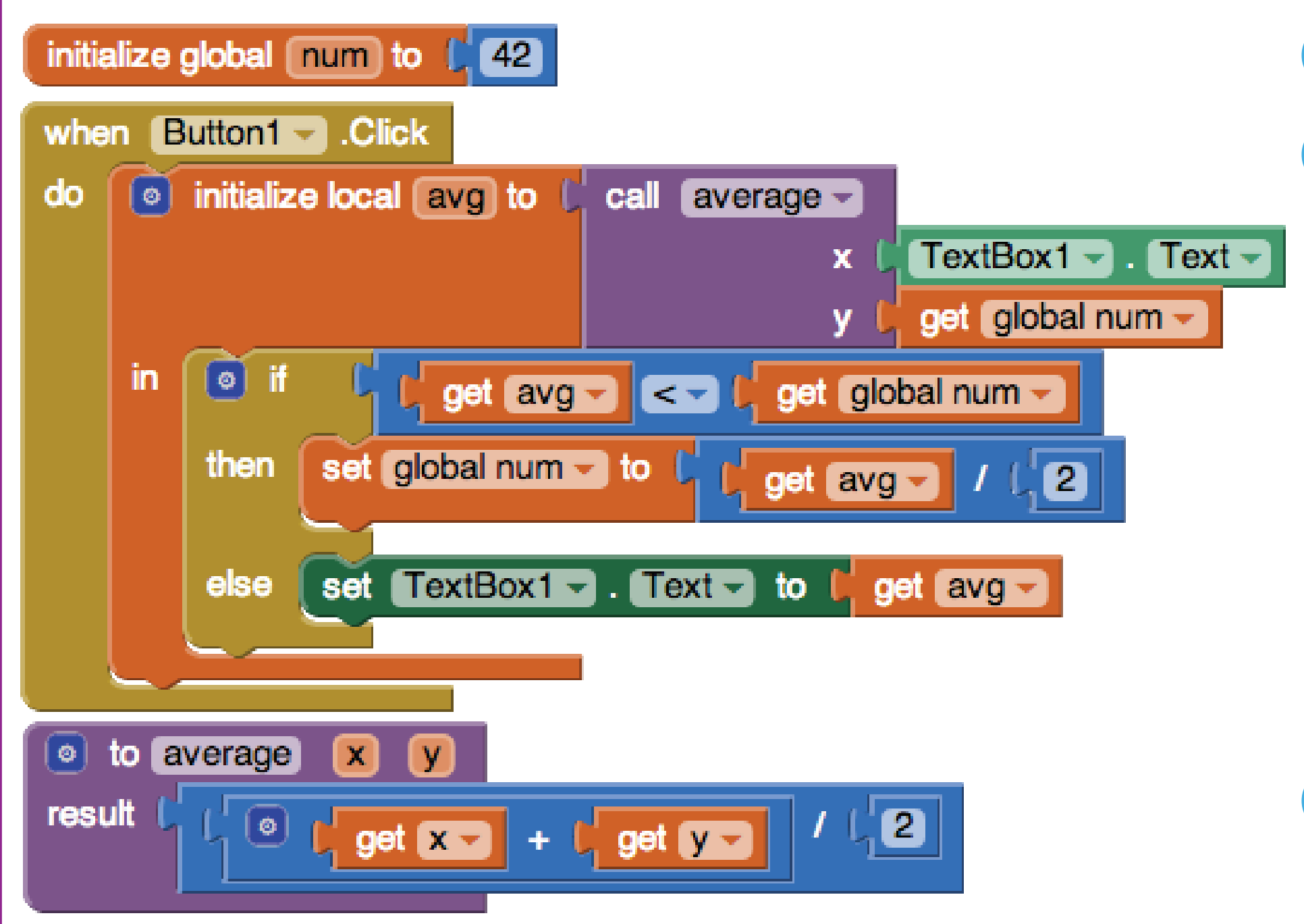
Problem

MIT App Inventor 2 (AI2), a popular online environment for Android app development, democratizes programming through its easy-to-use blocks language. While simple blocks programs are easy to read and write, complex ones become overwhelming. Creating and navigating nontrivial blocks programs is tedious, and AI2's current inability to copy blocks between projects inhibits sharing.

My Solution
TAIL (Textual App Inventor Language)

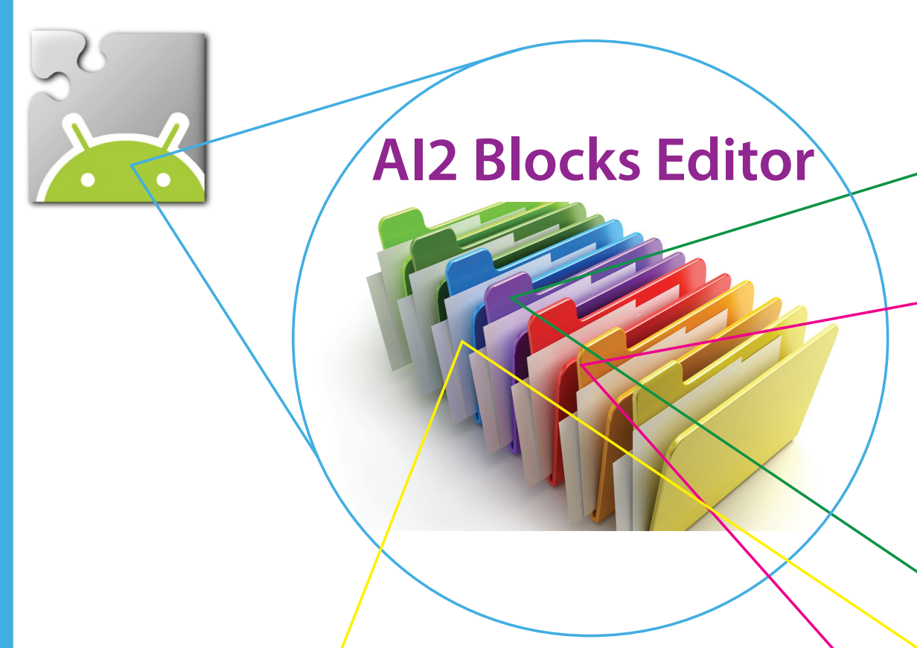
To address these issues, I have created a new textual language, TAIL, that is isomorphic to AI2's blocks language and provided a means for converting between them. This project aims to (1) increase AI2's usability by providing an efficient means for reading, constructing, and sharing programs, and (2) ease users' transitions from blocks programming to text programming.

TAIL Language Design

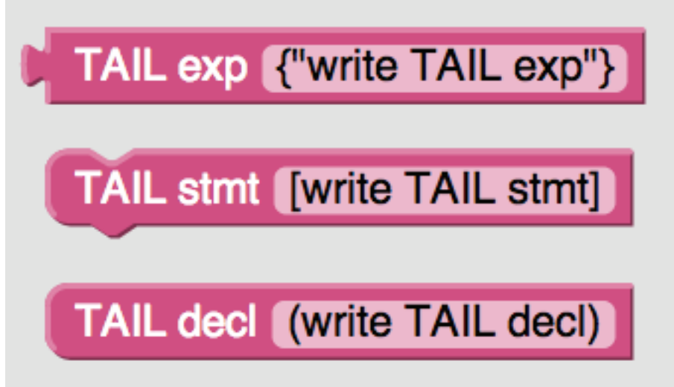


```
(initialize_global <num> to: {42})

(when Button1.Click
  do: [initialize_local <avg>
        to: {call average
               x: {TextBox1.Text}
               y: {get global num}}
        in: [if {{get avg} < {get global num}}
              then: [set global num to: {{get avg} \ {2}}]
              else: [set TextBox1.Text to: {get avg} ]]])


(to <average> <x> <y>
 result: {{{get x} + {get y}}/{2}})
```

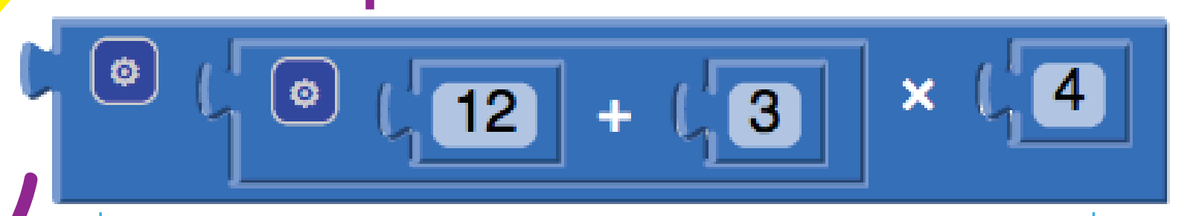
Conversion between Blocks and TAIL: The Details



TAIL Code Blocks (in JavaScript)

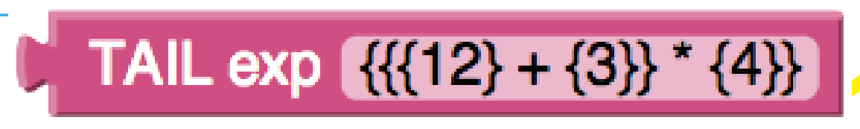

TAIL exp {"write TAIL exp"}
TAIL stmt [write TAIL stmt]
TAIL decl (write TAIL decl)

Blocks to TAIL
JavaScript Blocks to Text Converter



XML Tree

```
<block type="math_multiply" inline="true">
  <mutation items="2"></mutation>
  <value name="NUM0">
    <block type="math_add" inline="true">
      <mutation items="2"></mutation>
          <value name="NUM0">
            <block type="math_number">
              <title name="NUM">12</title>
            </block>
          </value>
          <value name="NUM1">
            <block type="math_number">
              <title name="NUM">3</title>
            </block>
          </value>
      </block>
  </value>
  <value name="NUM1">
      <block type="math_number">
          <title name="NUM">4</title>
      </block>
  </value>
</block>
```



TAIL to Blocks

ANTLR

Lexer

```
{{{12} + {3}} * {4}}
        ↓
{ { { 12 } + { 3 } } * { 4 } }
```

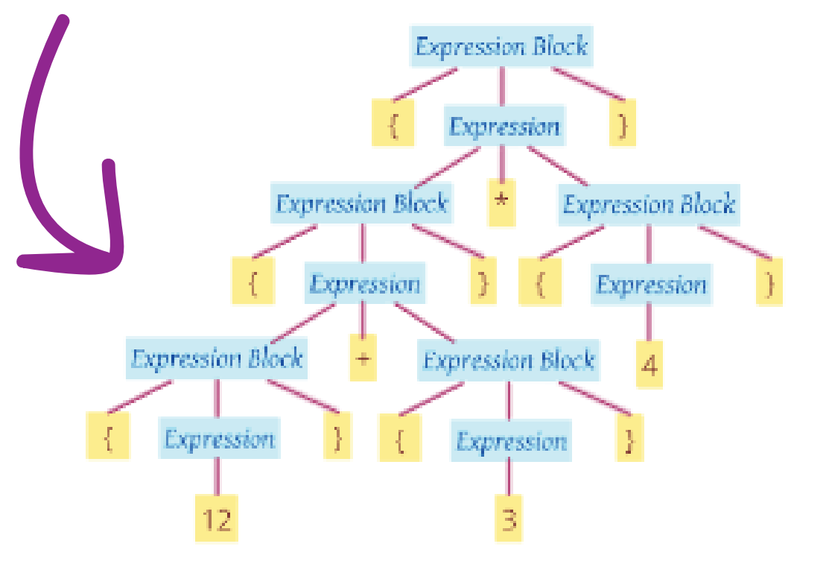
ANTLR Lexer Grammar Rules

```
NUMBER : (DIGIT* DOT DIGIT+ | DIGIT+ (DOT)?);

          fragment
          DIGIT : ('0' .. '9');

          tokens {
                  LCURLY = '{';
                  RCURLY = '}';

                  ADD = '+';

                  MULTIPLY = '*';
          }
```

Parser
ANTLR Parser Grammar Rules

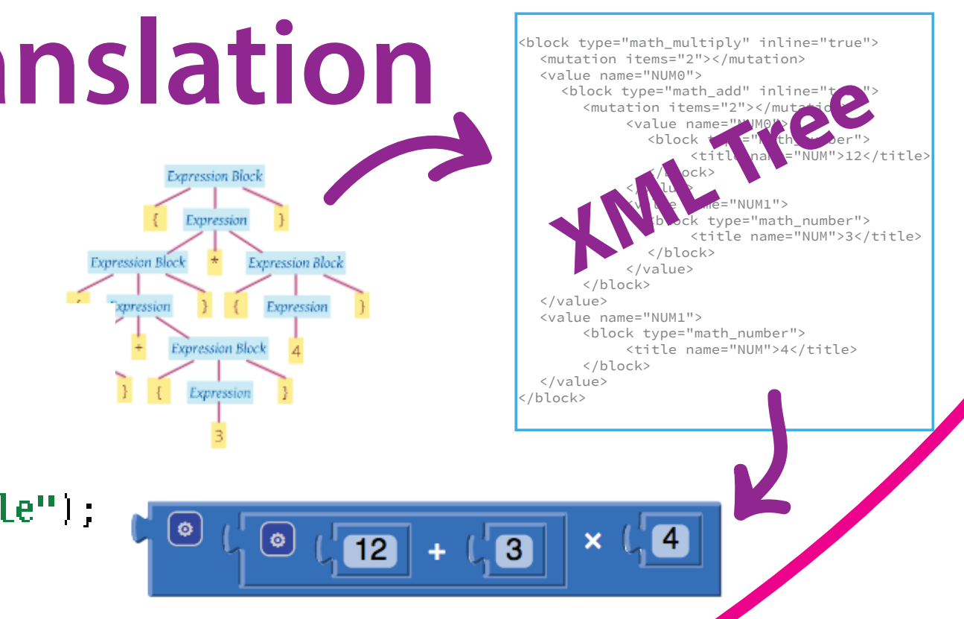
```
expression_block
        : LCURLY expression RCURLY
        ;

expression returns [var elt]
        : math_expr {$elt = $math_expr.elt;}
        | atom{$elt = $atom.elt;}
        ;

math_expr
        : mutable_arith_expr
        | non_mutable_arith_expr
        | special_math_expr
        | unary_math_expr
        | math_trig_expr
        ;

mutable_arith_expr

        : expression_block
          ( (ADD)+  | (MULTIPLY)+ )
          expression_block
        ;

atom

            : NUMBER
            | STRING
            | TRUE
            | FALSE
            ;
```

```
{ { { 12 } + { 3 } } * { 4 } }
```

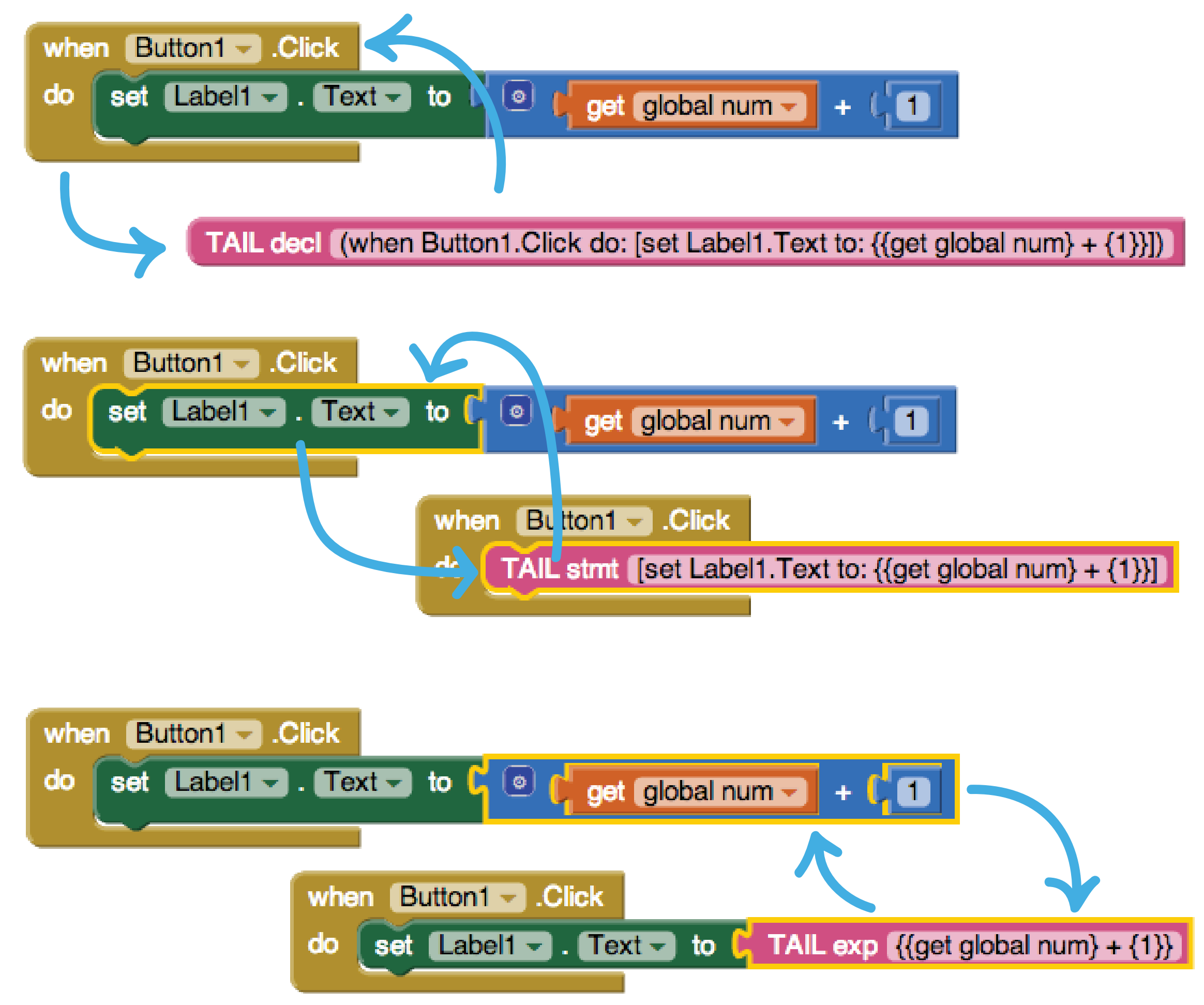


Tree Translation
ANTLR Parser Actions in Javascript Target

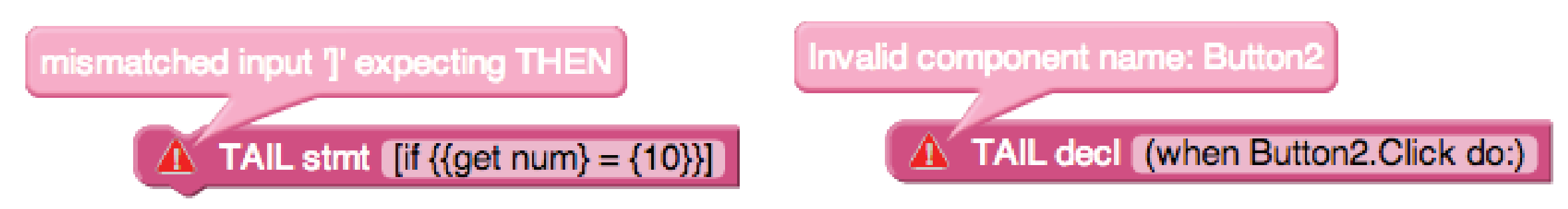
```
atom returns [var elt]
@init{
  $elt = document.createElement("block");

  var title = document.createElement("title");
}
  : NUMBER {
    $elt.setAttribute("type","math_number");

    title.setAttribute("name","NUM");
    title.innerHTML = $NUMBER.text;
    $elt.appendChild(title);
  }
```



Blocks <—> Text Conversion


when Button1 .Click
do set Label1 . Text to
get global num + 1
TAIL decl (when Button1.Click do: [set Label1.Text to: {{get global num} + {1}}])
TAIL stmt [set Label1.Text to: {{get global num} + {1}}]
TAIL exp {{get global num} + {1}}

Error Detection


mismatched input ']' expecting THEN
TAIL stmt [if {{get num} = {10}}]
Invalid component name: Button2
TAIL decl (when Button2.Click do:)