Testing Framework for Blockly

Team TBD
Sarah Nicholson, Peyton Hartzell, Mykal Burris, and Kelly Ding

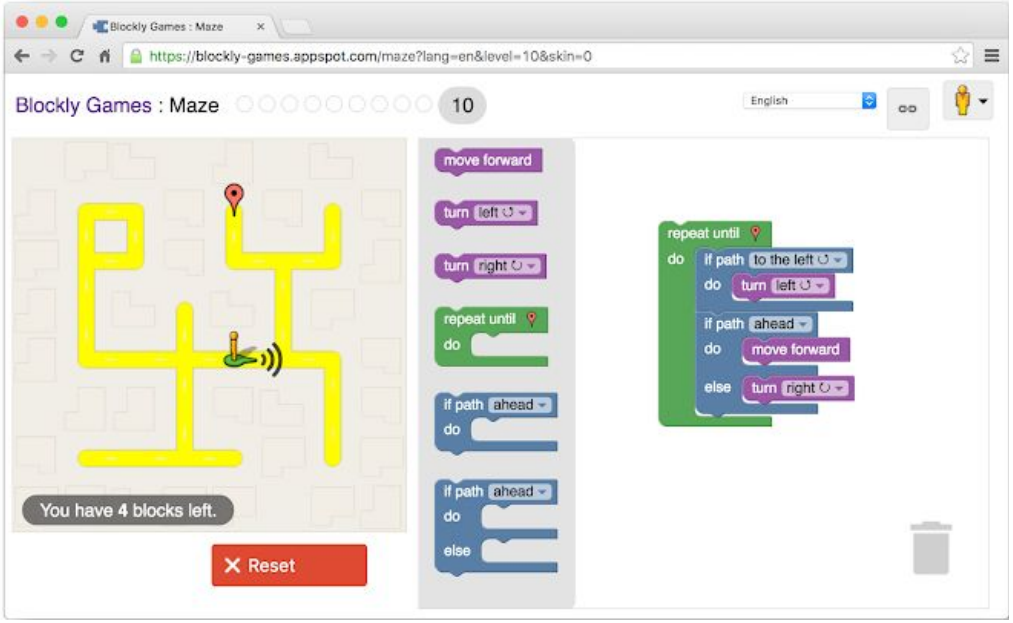
CSCI 362 - Software Engineering

Outline

- Introduction
- Testing Framework
- Test Cases
- Injecting Faults
- Demo!

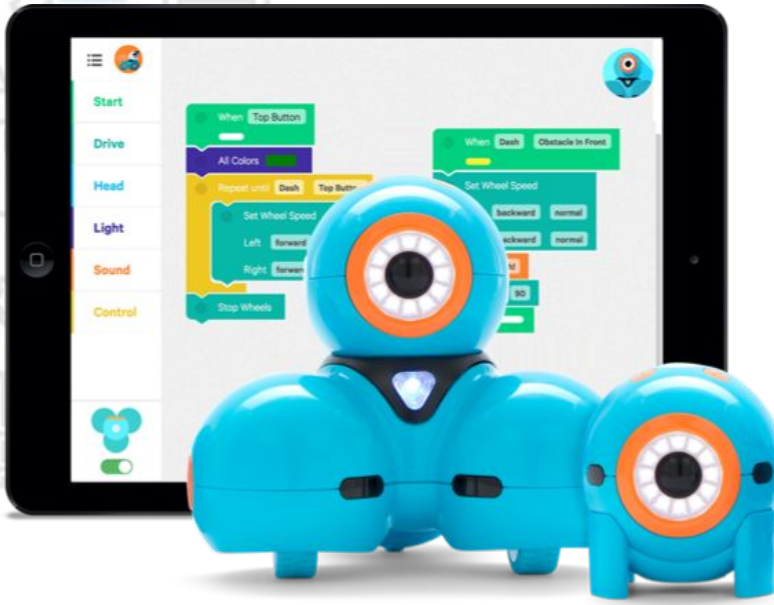
What is Blockly?

- Blockly is an intuitive, visual way to build code created by Google
- It is 100% client side, requiring no support from the server
- It emits syntactically correct user-generated code which can export to:
  - JavaScript
  - Python
  - PHP
  - Lua
  - Dart



Why did we choose Blockly?

- We chose Blockly because:
  - It’s written in Javascript
  - We are interested in the early education of children in CS
  - It is documented very well
  - It already has many tests including unit tests and testing for the different languages



Testing Framework

- Focused on testing Blockly's core
- Framework structure:
  - TestAutomation
  - docs
  - oracles
  - project
    - src
      - index.js
      - test.js
  - scripts
  - style
  - temp
  - testCaseExecutables
  - testCases
- runAllTests.sh is what handles running all of the tests

Testing Script

```
cat project/src/index.js > project/src/test.js
cat testCaseExecutables/*.js >> project/src/test.js

# create the html file
echo "<html>
  <head>
    <link rel='stylesheet' href='../style/style.css'>
    <meta charset='utf-8'>
    <title>Unit Tests for Blockly - Team TBD</title>
    <script src='../../../blockly-master/blockly_uncompressed.js'></script>
    <script>goog.require('goog.testing.jsunit');</script>
  </head>" >> t utput.html
```

This is what concatenates the test files together

This is what runs the tests

This is what outputs the table

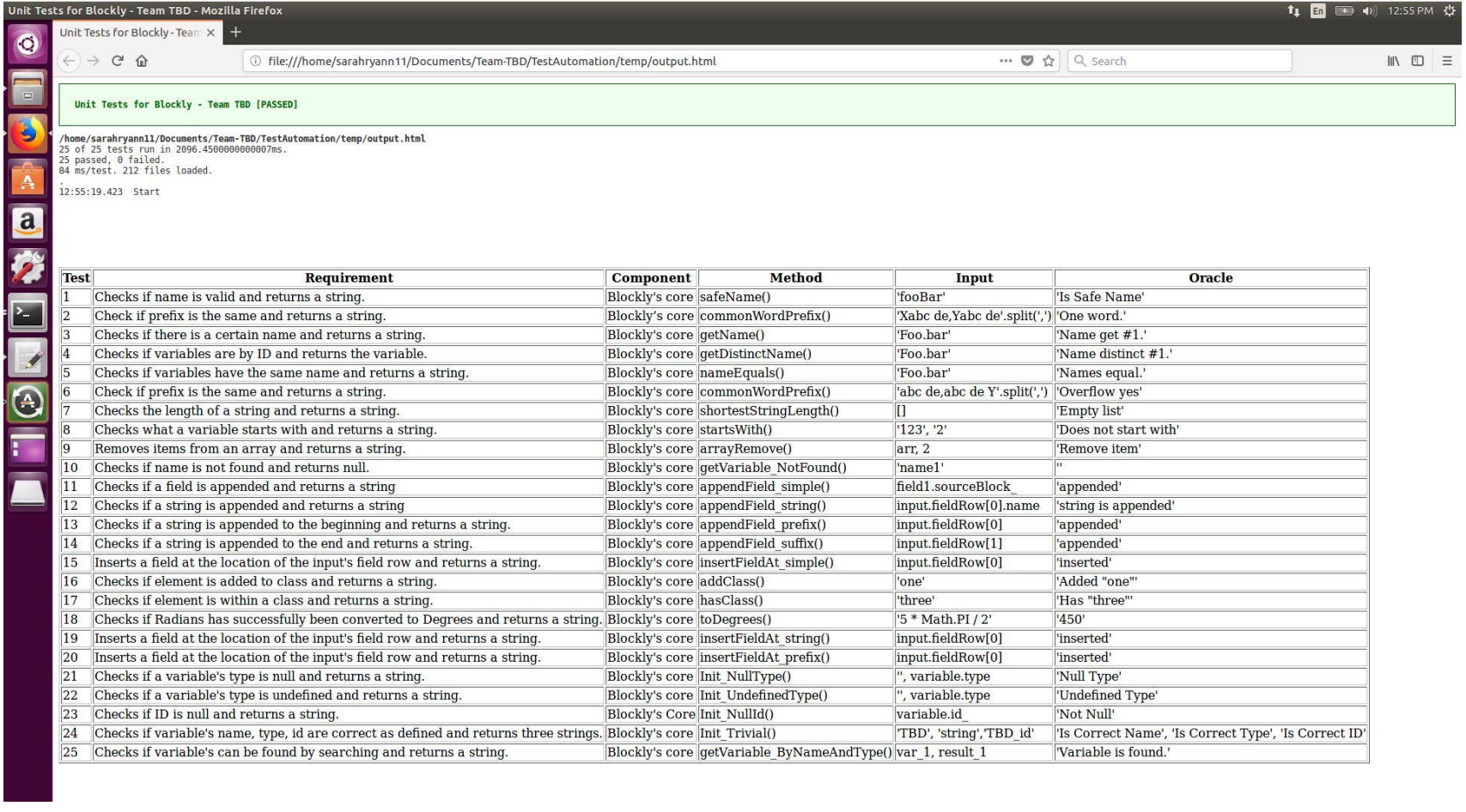
```
# loops through all test cases in the directory.
for filename in testCases/*.txt; do
        i="${filename//[^0-9]/}"
        echo "Performing test $i."
        # create array and populate with contents of te  files
        ARRAY=()
        while read LINE; do
                ARRAY+=("$LINE")
        done < "$filename"

# creating table
        echo "
        <tr><td>${ARRAY[0]}</td>
        <td>${ARRAY[1]}</td>
        <td>${ARRAY[2]}</td>
        <td>${ARRAY[3]}</td>
        <td>${ARRAY[4]}</td>
        <td>${ARRAY[5]}</td></tr>" >> temp/output.html
done
```

Test Cases

- All test cases are focused on functions of Blockly’s core
- The items tested are outlined below:
  - Test Number
  - Requirement Tested
  - Component Tested
  - Method Tested
  - Test Input
  - Expected Outcome
- Sample Test Case:
  - function test_safeName () {
  - var varDB = new Blockly.Names('window,door');
  - assertEquals('Is Safe Name', 'fooBar', varDB.safeName_('fooBar'));
  - }
  -

Test Output

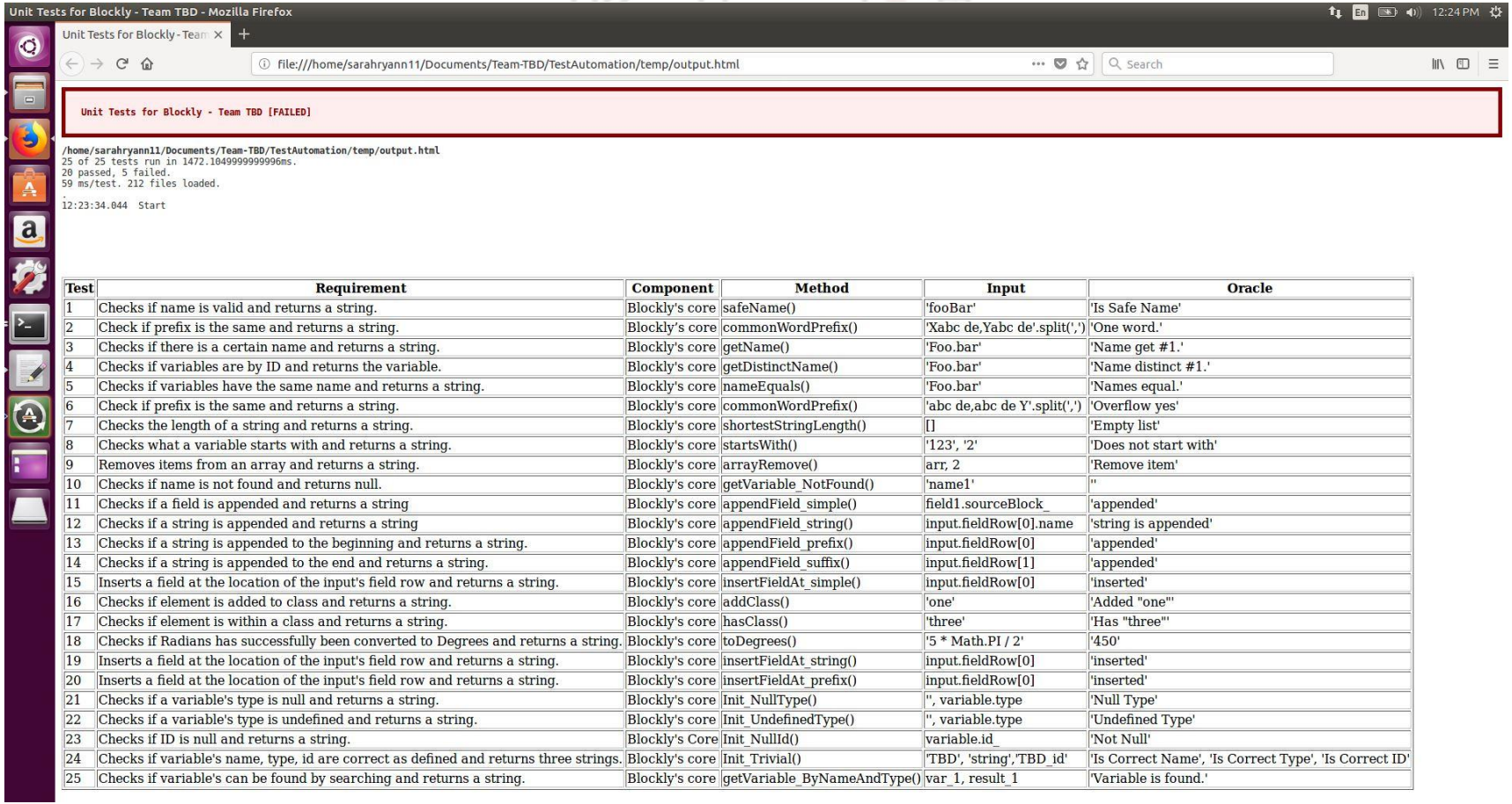


Test	Requirement	Component	Method	Input	Oracle
1	Checks if name is valid and returns a string.	Blockly's core	safeName()	'fooBar'	'Is Safe Name'
2	Check if prefix is the same and returns a string.	Blockly's core	commonWordPrefix()	'Xabc de,Yabc de'.split(',')	'One word.'
3	Checks if there is a certain name and returns a string.	Blockly's core	getName()	'Foo.bar'	'Name get #1.'
4	Checks if variables are by ID and returns the variable.	Blockly's core	getDistinctName()	'Foo.bar'	'Name distinct #1.'
5	Checks if variables have the same name and returns a string.	Blockly's core	nameEquals()	'Foo.bar'	'Names equal.'
6	Check if prefix is the same and returns a string.	Blockly's core	commonWordPrefix()	'abc de,abc de Y'.split(',')	'Overflow yes'
7	Checks the length of a string and returns a string.	Blockly's core	shortestStringLength()	[]	'Empty list'
8	Checks what a variable starts with and returns a string.	Blockly's core	startsWith()	'123', '2'	'Does not start with'
9	Removes items from an array and returns a string.	Blockly's core	arrayRemove()	arr, 2	'Remove item'
10	Checks if name is not found and returns null.	Blockly's core	getVariable_NotFound()	'name1'	''
11	Checks if a field is appended and returns a string	Blockly's core	appendField_simple()	field1.sourceBlock_	'appended'
12	Checks if a string is appended and returns a string	Blockly's core	appendField_string()	input.fieldRow[0].name	'string is appended'
13	Checks if a string is appended to the beginning and returns a string.	Blockly's core	appendField_prefix()	input.fieldRow[0]	'appended'
14	Checks if a string is appended to the end and returns a string.	Blockly's core	appendField_suffix()	input.fieldRow[1]	'appended'
15	Inserts a field at the location of the input's field row and returns a string.	Blockly's core	insertFieldAt_simple()	input.fieldRow[0]	'inserted'
16	Checks if element is added to class and returns a string.	Blockly's core	addClass()	'one'	'Added "one"'
17	Checks if element is within a class and returns a string.	Blockly's core	hasClass()	'three'	'Has "three"'
18	Checks if Radians has successfully been converted to Degrees and returns a string.	Blockly's core	toDegrees()	'5 * Math.PI / 2'	'450'
19	Inserts a field at the location of the input's field row and returns a string.	Blockly's core	insertFieldAt_string()	input.fieldRow[0]	'inserted'
20	Inserts a field at the location of the input's field row and returns a string.	Blockly's core	insertFieldAt_prefix()	input.fieldRow[0]	'inserted'
21	Checks if a variable's type is null and returns a string.	Blockly's core	Init_NullType()	'', variable.type	'Null Type'
22	Checks if a variable's type is undefined and returns a string.	Blockly's core	Init_UndefinedType()	'', variable.type	'Undefined Type'
23	Checks if ID is null and returns a string.	Blockly's Core	Init_NullId()	variable.id_	'Not Null'
24	Checks if variable's name, type, id are correct as defined and returns three strings.	Blockly's core	Init_Trivial()	'TBD', 'string','TBD_id'	'Is Correct Name', 'Is Correct Type', 'Is Correct ID'
25	Checks if variable's can be found by searching and returns a string.	Blockly's core	getVariable_ByNameAndType()	var_1, result_1	'Variable is found.'

Injecting Faults

- We have chosen to induce failures in the following ways:
  - Change the signatures of the tests
  - Change the assert methods within the tests
  - Inject syntax errors
- Example:
  - function test_safeName_ () {
  - var varDB = new Blockly.Names('window;door');
  - assertNull('Is Safe Name', 'fooBar', varDB.safeName_('fooBar'));
  - }

Fault Output



Test	Requirement	Component	Method	Input	Oracle
1	Checks if name is valid and returns a string.	Blockly's core	safeName()	'fooBar'	'Is Safe Name'
2	Check if prefix is the same and returns a string.	Blockly's core	commonWordPrefix()	'Xabc de,Yabc de'.split(',')	'One word.'
3	Checks if there is a certain name and returns a string.	Blockly's core	getName()	'Foo.bar'	'Name get #1.'
4	Checks if variables are by ID and returns the variable.	Blockly's core	getDistinctName()	'Foo.bar'	'Name distinct #1.'
5	Checks if variables have the same name and returns a string.	Blockly's core	nameEquals()	'Foo.bar'	'Names equal.'
6	Check if prefix is the same and returns a string.	Blockly's core	commonWordPrefix()	'abc de,abc de Y'.split(',')	'Overflow yes'
7	Checks the length of a string and returns a string.	Blockly's core	shortestStringLength()	[]	'Empty list'
8	Checks what a variable starts with and returns a string.	Blockly's core	startsWith()	'123', '2'	'Does not start with'
9	Removes items from an array and returns a string.	Blockly's core	arrayRemove()	arr, 2	'Remove item'
10	Checks if name is not found and returns null.	Blockly's core	getVariable_NotFound()	'name1'	''
11	Checks if a field is appended and returns a string	Blockly's core	appendField_simple()	field1.sourceBlock_	'appended'
12	Checks if a string is appended and returns a string	Blockly's core	appendField_string()	input.fieldRow[0].name	'string is appended'
13	Checks if a string is appended to the beginning and returns a string.	Blockly's core	appendField_prefix()	input.fieldRow[0]	'appended'
14	Checks if a string is appended to the end and returns a string.	Blockly's core	appendField_suffix()	input.fieldRow[1]	'appended'
15	Inserts a field at the location of the input's field row and returns a string.	Blockly's core	insertFieldAt_simple()	input.fieldRow[0]	'inserted'
16	Checks if element is added to class and returns a string.	Blockly's core	addClass()	'one'	'Added "one"'
17	Checks if element is within a class and returns a string.	Blockly's core	hasClass()	'three'	'Has "three"'
18	Checks if Radians has successfully been converted to Degrees and returns a string.	Blockly's core	toDegrees()	'5 * Math.PI / 2'	'450'
19	Inserts a field at the location of the input's field row and returns a string.	Blockly's core	insertFieldAt_string()	input.fieldRow[0]	'inserted'
20	Inserts a field at the location of the input's field row and returns a string.	Blockly's core	insertFieldAt_prefix()	input.fieldRow[0]	'inserted'
21	Checks if a variable's type is null and returns a string.	Blockly's core	Init_NullType()	'', variable.type	'Null Type'
22	Checks if a variable's type is undefined and returns a string.	Blockly's core	Init_UndefinedType()	'', variable.type	'Undefined Type'
23	Checks if ID is null and returns a string.	Blockly's Core	Init_NullId()	variable.id_	'Not Null'
24	Checks if variable's name, type, id are correct as defined and returns three strings.	Blockly's core	Init_Trivial()	'TBD', 'string','TBD_id'	'Is Correct Name', 'Is Correct Type', 'Is Correct ID'
25	Checks if variable's can be found by searching and returns a string.	Blockly's core	getVariable_ByNameAndType()	var_1, result_1	'Variable is found.'

Lessons Learned

- Time Management
- Git and Github for team collaboration
- Command Line/Linux
- Adapt to Challenges


Demo!