Testing Plan

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

Introduction

Blockly is a library created by Google that adds a visual code editor to both web and mobile apps. The Blockly editor uses interlocking, graphical blocks, similar to Scratch, to represent code concepts like loops, logical expressions, variables, and more. It allows users to apply programming principles without having to worry about minute details such as syntax.

Some of the reasons why we found Blockly attractive is how well it is documented and how it already has many unit tests. As part of our testing framework, we will be testing five methods in the following files:

- *names_test.js*
- *utils_test.js*
- *variable_map_test.js*
- *workspace_test.js*

Requirements Traceability

The requirements for each of the methods we are testing are different for the most part, but all have to deal with Blockly's core functions. We will use assert methods to check if the input is valid or invalid.

Tested Items

The items tested are outlined below:

1. Test Number
2. Requirement Tested
3. Method Tested
4. Test Input
5. Expected Outcome

Proposed Testable Methods

The methods we will be testing are all in Blockly's core functioning. They all accept some sort of input and will assert if it is valid or invalid. The following methods will be tested:

1. function safeName(name)
    a. Checks if name has special characters and returns the name.
2. function commonWordPrefix(Array)
    a. Check if prefix is the same and returns the count.
3. function getVariableTypes(name, type, id)
    a. Checks the variable type and returns the type.
4. function getVariableById(id)
    a. Checks if variables are by ID and returns the variable.
5. function compareByName(var1, var2)
    a. Checks if variables have same name and returns a number.

Five Potential Test Cases

1

Requirement: Checks if name has special characters and returns the name.
Method: function safeName(name)
Test Input: “%$@)<.*”
Expected Outcome: “%$@)<.*”

2

Requirement: Check if prefix is the same and returns the count.
Method: function commonWordPrefix(Array)
Test Input: [‘aa’, ‘abc’, ‘de’, ‘gd’, ‘ax’]
Expected Outcome: 3

3

Requirement: Checks the variable type and returns the type.
Method: function getVariableTypes(name, type, id)
Test Input: [‘type0’, ‘type1’, ‘type2’]
Expected Outcome: [‘type0’, ‘type1’, ‘type2’]

4

Requirement: Checks if variables are by ID and returns the variable.
Method: function getVariableById(id)
Test Input: ‘id0’
Expected Outcome: variable ‘id0’

5

Requirement: Checks if variables have same name and returns a number.
Method: function compareByName(var1, var2)
Test Input: var1, var2
Expected Outcome: -1

Test Recording Procedures

We will send the output to a web page and to files. The test outputs will also be compared to the expected outcomes along with whether the test succeeded or failed.

Software Requirements

- Ubuntu 14.04
- The following Google Libraries:
    - Blockly Library
    - Closure Library

Testing Schedule

Our testing schedule will span about four weeks in two phases. The first phase will end with the completion of Deliverable 2 on October 15. In this phase, we will:

- Identify and select 5 methods we will test to be our 5 of the eventual 25 test cases that we will develop
- Elaborate on the specific test elements and cases within the methods we select
- Once we finish elaborating on the 5 test cases, and there is sufficient time remaining, we will then begin writing the actual test cases of the 5 we identified

Our second phase will begin once our Deliverable 2 is due and end when our Deliverable 3 is due on November 9. During this phase, we will:

- Refactor test elements that need to be changed from the first phase
- Design and build an automated testing framework that we will use to implement our test plan

Constraints

Most of the constraints we will face will be due to our busy schedules. Two of us are able to work together three times a week in between two of our classes, but the other two of us have class during that time. If needed, we might be able to meet on some weekends. We do communicate regularly through Discord and work on all deliverables

together through using a Google Doc. Another constraint can be a lack of knowledge in JavaScript, but one of us is well versed in it and can help the others if any issues arise. A final constraint that has arisen is the possibility of a hurricane hitting and Fall Break being close to when Deliverable 3 is due.