SCS Project

Sbera Catalin 30434

1. Research

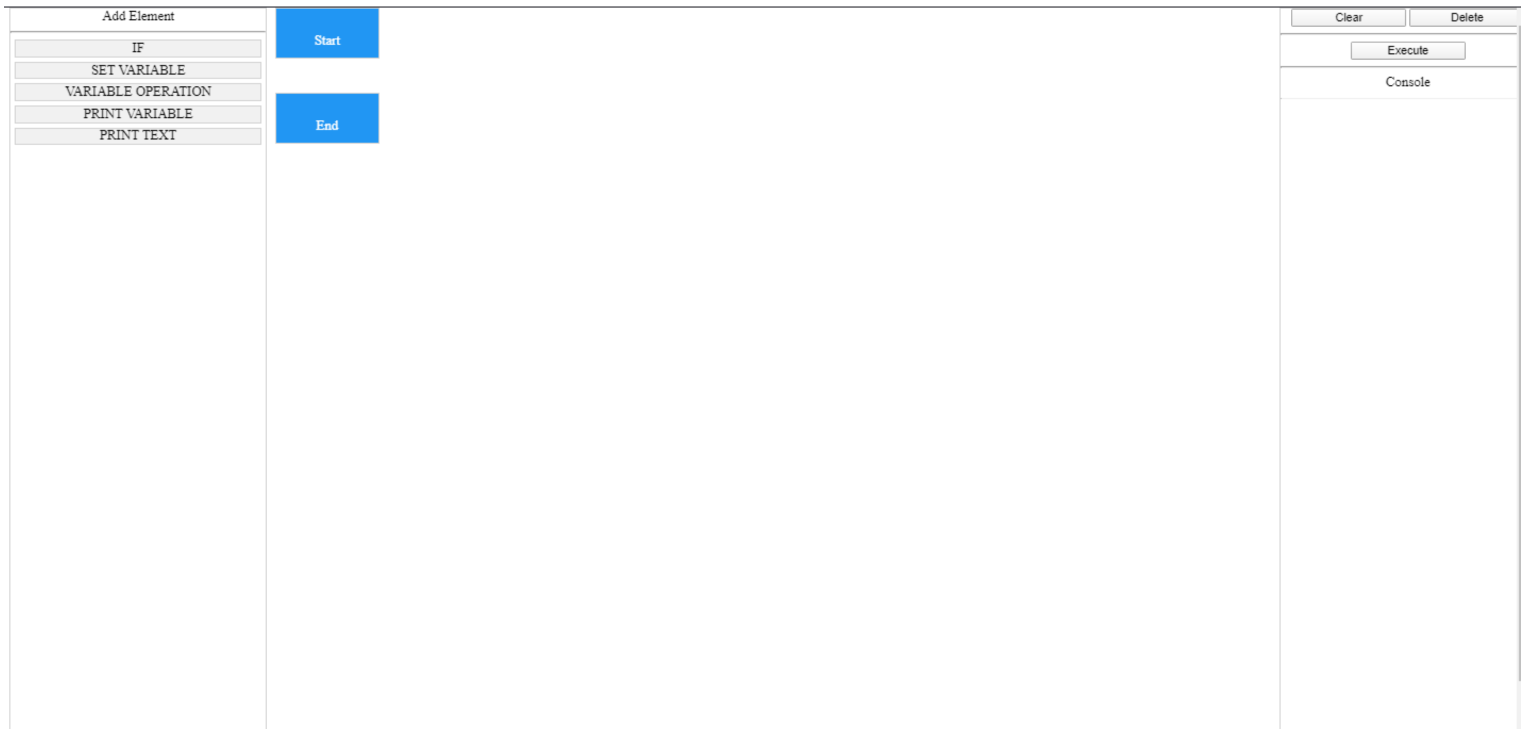
The goal of the project is to design and implement a graphical programming medium/language that allows the user to create a program using the basic programming structures (functional blocks, directional segments, decision blocks).

The example that the project was modeled after is the Scratch software, designed by a group of people from MIT. What the project tries to replicate is the easiness that one can use the basic building blocks of the project in order to display more complex programs.

2. Analysis and Design

The programming language chosen for this project is JavaScript. This language suits the goal of the project since a main use for it would be to be used as a introduction into the basics of computer programming and JavaScript requires only a web browser and an internet connection to run, both which are widely used.

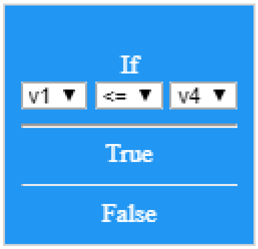
The project provides a simplistic interface that allows the user to create a program by adding and connecting programming blocks that can display data in the provided console.

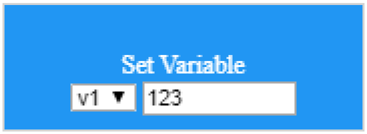


The interface is split into 3 main parts:

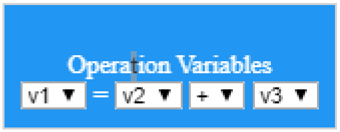
- Elements creator: This is represented by the left part and the user can click one of the 5 available buttons in order to add a new block of a certain type in the block area. The available blocks are:
    - If block: Allows the user to compare 2 variables and depending on the result of the condition, determine to what block to go.

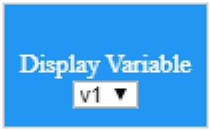
      The possible comparations are: "==" equals, ">" greater, "<" lesser, ">=" greater equal, "<=" lesser equal.



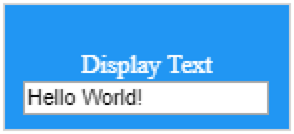


    - Set Variable Block: Allows the user to set a numeric value to one of the 5 available variables (v1-v5).
    - Variable Operation Block: Allows the user to perform a basic arithmetic operation between 2 variables and assign the result to another one. The available operations are: "+" addition, "-" subtraction, "*" multiplication, "/" division.

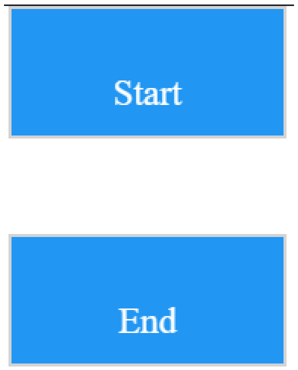


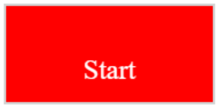


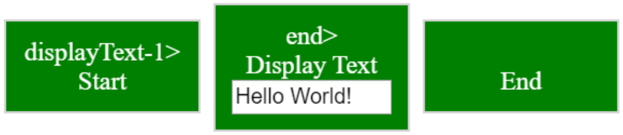
    - Print Variable Block: Displays the selected variable in the console on the console on the right side.
    - Print Text Block: Displays what the user inputs in the block's text field in the console on the right side.

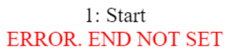


- Blocks area: Represents the middle part of the webpage. This is the place where the user can freely move the blocks by selecting them and dragging it to the desired location.
  In order to connect multiple blocks first the user must double-click the first block and then double click the block which, when the sequence is executed, will follow. Once a block is selected, it will turn red until a second block is selected, then both will turn green. If the user reselects the current block it will turn green and if an element was linked to it, it will no longer be linked to it.

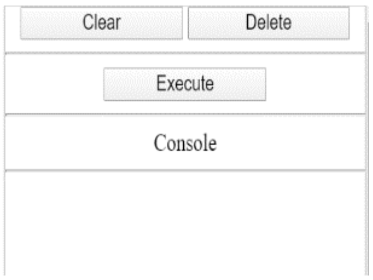







```
1: Start
ERROR. END NOT SET
```

Except the presented blocks, the 2 blocks that will always be presented are the End Block and the Start Block. The Start Block is the first one that is executed and the last block must always be the end block or an exception will be displayed in the console area. Another exception to the presented rules is the if block. In order to link another block to it, the user must double click the True or False option and then select the following block.



- Buttons area: Represents the right side of the website and consist of 3 buttons and the console. The 3 buttons are:
  - Clear: Clears the Blocks area by refreshing the webpage.
  - Delete: Deletes the selected block.
  - Execute: Start the execution of the program and displays the results in the console part
- Console Area: Represents the area below the “Console” text and displays what the program executes. It will always start with the Start instruction and end with End block. The only blocks that will display in the console are the Start, End, Print Variable and Print Text blocks. Before each display line there will be a number indicating the order of the block.

3. Implementation

As before mentioned, the chosen language is JavaScript. Along this language, some HTML and CSS code is necessary. As for external library, I used only JQuery in order to manipulate the execution order faster.

Main.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script src="jquery-3.4.1.min.js"></script>
    <link rel="stylesheet" type="text/css" href="main.css">
    <title>Scuffed Scratch</title>
</head>
<body>
<div id="controls">
    <button id="clearButton" type="button">Clear</button>
    <button id="deleteButton" type="button">Delete</button>
    <hr>
    <button id="executeButton" type="button">Execute</button>
    <hr>
    <p>Console</p>
    <hr>
    <div id="console"></div>
</div>

<div id="menu">
    <p>Add Element</p>
    <hr>
    <div id='ifButton' class="new">IF</div>
    <div id='setVariableButton' class="new">SET VARIABLE</div>
    <div id='operationVariablesButton' class="new">VARIABLE OPERATION</div>
    <div id='displayVariableButton' class="new">PRINT VARIABLE</div>
    <div id='displayTextButton' class="new">PRINT TEXT</div>
</div>

<div class="movable" id="start" style="top: 0; left: 320px">
```

```html
    <p>
        <span class="prev"></span>
        <span class="next"></span>
        <br>
        Start
    </p>
</div>

<div class="movable" id="end" style="top: 100px; left: 320px">
    <p>
        <span class="prev"></span>
        <span class="next"></span>
        <br>
        End
    </p>
</div>

<script src="main.js"></script>

<script src="buttons/execute.js"></script>
<script src="buttons/delete.js"></script>
<script src="buttons/clear.js"></script>

<script src="blocks/displayText.js"></script>
<script src="blocks/setVariable.js"></script>
<script src="blocks/displayVariable.js"></script>
<script src="blocks/operationVariable.js"></script>
<script src="blocks/if.js"></script>
</body>
</html>
```

Main.css

```css
p,.zeroSpace {
    margin: 0;
    padding: 0;
    border: 0;
}

.new {
    border: 1px solid #d3d3d3;
    text-align: center;
    margin: 5px;
    background-color: #f1f1f1;
}

.new:hover {
    background-color: #fafafa;
    cursor: pointer;
}

.movable {
    padding: 10px;
    width: auto;
    min-width: 100px;
    cursor: move;
    z-index: 10;
    color: #fff;
    position: absolute;
    border: 1px solid #d3d3d3;
    text-align: center;
    background-color: #2196F3;
}

#menu {
    border: 1px solid #d3d3d3;
    position: absolute;
    width: 300px;
    height: 1000px;
    overflow: auto;
    top: 0px;
    text-align: center;
}

#controls {
    position: absolute;
    top: 0px;
    left: 1000px;
    width: 300px;
    height: 1000px;
    border: 1px solid #d3d3d3;
    text-align: center;
}

button {
    width: 45%;
}

.prev {
    text-align: left;
}

.next {
    text-align: right;
}
```

These 2 files represent the visual part of the project.

Main.js

```
const variables = new Array(6);
for (let el of document.getElementsByClassName("movable")) {
    dragElement(el);
}

let selected = null;

function dragElement(elmnt) {
    let pos1 = 0, pos2 = 0, pos3 = 0, pos4 = 0;
    if (elmnt.id.startsWith('if')) {
        // if present, the header is where you move the DIV from:
        $(elmnt).find('.tr')[0].ondblclick = bindElements;
        $(elmnt).find('.fa')[0].ondblclick = bindElements;
    }
    elmnt.onmousedown = dragMouseDown;
    elmnt.ondblclick = bindElements;

    function dragMouseDown(e) {
        e = e || window.event;
        e.stopPropagation();
        // get the mouse cursor position at startup:
        pos3 = e.clientX;
        pos4 = e.clientY;
        document.onmouseup = closeDragElement;
        // call a function whenever the cursor moves:
        document.onmousemove = elementDrag;
    }

    function elementDrag(e) {
        e = e || window.event;
        e.stopPropagation();
        // calculate the new cursor position:
        pos1 = pos3 - e.clientX;
        pos2 = pos4 - e.clientY;
        pos3 = e.clientX;
        pos4 = e.clientY;
        // set the element's new position:
        elmnt.style.top = Math.max(elmnt.offsetTop - pos2, 0) + "px";
        elmnt.style.left = Math.min(Math.max(elmnt.offsetLeft - pos1, 320), 800) +
"px";
    }

    function closeDragElement() {
        // stop moving when mouse button is released:
        document.onmouseup = null;
        document.onmousemove = null;
        // snap element to postition
        elmnt.style.top = parseInt((elmnt.offsetTop) / 10) * 10 + "px";
        elmnt.style.left = parseInt((elmnt.offsetLeft) / 10) * 10 + "px";
    }

    function bindElements() {
        if(selected && $(selected).parent()[0].id === this.id) return;
        if(selected && this.id === selected.id){
            $(selected).find('.next').text('');
            selected = null;
            this.style.backgroundColor = "green";
            return;
        }

        if (selected === null && this.id !== 'end' && !this.id.startsWith('if')) {
            selected = this;
            this.style.backgroundColor = 'red';
        } else if (this.id !== 'start' && this.id !== 'tr' && this.id !== 'fa'){
            if($(selected).hasClass('tr') || $(selected).hasClass('fa')){
                $(selected).find('.next').text(this.id + '>');
            } else {
                selected.childNodes[1].childNodes[3].innerHTML = (this.id + '&gt;');

            }
            selected.style.backgroundColor = "green";
            this.style.backgroundColor = "green";
            selected = null;
        }
    }
}
```
This class is responsible for handling the moving of the blocks and linking them together.

If.js

```
const $ifTemplate = $(
    '<div class="movable if" id="if-1" style="top: 0; left: 320px">\n' +
    '    <p>\n' +
    '        <span class="prev"></span>\n' +
    '        <br>\n' +
    '        If\n' +
    '    </p>\n' +
    '    <select name="varNr1">\n' +
    '        <option value="0">v1</option>\n' +
    '        <option value="1">v2</option>\n' +
    '        <option value="2">v3</option>\n' +
    '        <option value="3">v4</option>\n' +
    '        <option value="4">v5</option>\n' +
    '    </select>\n' +
    '    <select name="operation">\n' +
    '        <option value="==">==</option>\n' +
    '        <option value="<">&lt;</option>\n' +
    '        <option value=">">&gt;</option>\n' +
    '        <option value="<=">&lt;=</option>\n' +
    '        <option value=">=">&gt;=</option>\n' +
    '    </select>\n' +
    '    <select name="varNr2">\n' +
    '        <option value="0">v1</option>\n' +
    '        <option value="1">v2</option>\n' +
    '        <option value="2">v3</option>\n' +
    '        <option value="3">v4</option>\n' +
    '        <option value="4">v5</option>\n' +
    '    </select>\n' +
    '    <hr>\n' +
    '    <div class="tr">True <span class="next"></span></div>\n' +
    '    <hr>\n' +
    '    <div class="fa">False <span class="next"></span></div>\n' +
    '</div>'),
    $ifButton = $('#ifButton');
let ifNr = 0;

$ifButton.click(function () {

    let $newElement = $ifTemplate.clone();
    $newElement.attr('id', 'if-' + (++ifNr));
    dragElement($newElement[0]);
    $('body').append($newElement);
});
```
setVariable.js

```
const $setVariableTemplate = $(
    '<div class="movable setVariable" id="setVariable-1" style="top: 0; left:
320px">\n' +
    '    <p>\n' +
    '        <span class="prev"></span>\n' +
    '        <span class="next"></span>\n' +
    '        <br>\n' +
    '        Set Variable\n' +
    '    </p>\n' +
    '    <select name="varNr">\n' +
    '        <option value="0">v1</option>\n' +
    '        <option value="1">v2</option>\n' +
    '        <option value="2">v3</option>\n' +
    '        <option value="3">v4</option>\n' +
    '        <option value="4">v5</option>\n' +
    '    </select>\n' +
    '    <label>\n' +
    '        <input type="number" style="width: 45%;color: black; alignment:
right">\n' +
    '    </label>\n' +
    '</div>'),
    $setVariableButton = $('#setVariableButton');
let setVariableNr = 0;

$setVariableButton.click(function () {

    let $newElement = $setVariableTemplate.clone();
    $newElement.attr('id', 'setVariable-' + (++setVariableNr));
    dragElement($newElement[0]);
    $('body').append($newElement);
});
```
operationVariable.js

```
const $operationVariablesTemplate = $(
    '<div class="movable operationVariables" id="operationVariables-1" style="top: 0;
left: 320px">\n' +
    '    <p>\n' +
    '        <span class="prev"></span>\n' +
    '        <span class="next"></span>\n' +
    '        <br>\n' +
    '        Operation Variables\n' +
    '    </p>\n' +
    '    <select name="varNr0">\n' +
    '        <option value="0">v1</option>\n' +
    '        <option value="1">v2</option>\n' +
    '        <option value="2">v3</option>\n' +
    '        <option value="3">v4</option>\n' +
    '        <option value="4">v5</option>\n' +
    '    </select>\n' +
    '    <span>=</span>\n' +
    '    <select name="varNr1">\n' +
    '        <option value="0">v1</option>\n' +
    '        <option value="1">v2</option>\n' +
    '        <option value="2">v3</option>\n' +
    '        <option value="3">v4</option>\n' +
    '        <option value="4">v5</option>\n' +
    '    </select>\n' +
    '    <select name="operation">\n' +
    '        <option value="+">+</option>\n' +
    '        <option value="-">-</option>\n' +
    '        <option value="/">/</option>\n' +
    '        <option value="*">*</option>\n' +
    '    </select>\n' +
    '    <select name="varNr2">\n' +
    '        <option value="0">v1</option>\n' +
    '        <option value="1">v2</option>\n' +
    '        <option value="2">v3</option>\n' +
    '        <option value="3">v4</option>\n' +
    '        <option value="4">v5</option>\n' +
    '    </select>\n' +
    '</div>'),
    $operationVariablesButton = $('#operationVariablesButton');
let operationVariablesNr = 0;

$operationVariablesButton.click(function () {

    let $newElement = $operationVariablesTemplate.clone();
    $newElement.attr('id', 'operationVariables-' + (++operationVariablesNr));
    dragElement($newElement[0]);
    $('body').append($newElement);
});
```

displayVariable.js

```
const $displayVariableTemplate = $(
    '<div class="movable setVariable" id="displayVariable-1" style="top: 0; left: 
320px">\n' +
    '    <p>\n' +
    '        <span class="prev"></span>\n' +
    '        <span class="next"></span>\n' +
    '        <br>\n' +
    '        Display Variable\n' +
    '    </p>\n' +
    '    <select name="varNr">\n' +
    '        <option value="0">v1</option>\n' +
    '        <option value="1">v2</option>\n' +
    '        <option value="2">v3</option>\n' +
    '        <option value="3">v4</option>\n' +
    '        <option value="4">v5</option>\n' +
    '    </select>\n' +
    '</div>'),
    $displayVariableButton = $('#displayVariableButton');
let displayVariableNr = 0;

$displayVariableButton.click(function () {

    let $newElement = $displayVariableTemplate.clone();
    $newElement.attr('id', 'displayVariable-' + (++displayVariableNr));
    dragElement($newElement[0]);
    $('body').append($newElement);
});
```
displayText.js

```
const $displayTextTemplate = $(
    '<div class="movable displayText" id="displayText-1" style="top: 0; left: 
320px">\n' +
    '    <p>\n' +
    '        <span class="prev"></span>\n' +
    '        <span class="next"></span>\n' +
    '        <br>\n' +
    '        Display Text\n' +
    '    </p>\n' +
    '    <label>\n' +
    '        <input type="text" style="width: 95%;color: black">\n' +
    '    </label>\n' +
    '</div>'),
    $displayTextButton = $('#displayTextButton');
let displayTextNr = 0;

$displayTextButton.click(function () {

    let $newElement = $displayTextTemplate.clone();
    $newElement.attr('id', 'displayText-' + (++displayTextNr));
    dragElement($newElement[0]);
    $('body').append($newElement);
});
```
These scripts have the templates for creating new blocks with a unique id.

Clear.js

```
const $clearButton = $('#clearButton');

$clearButton.click(function () {
    location.reload();
})
```
Delete.js

```
const $deleteButton = $('#deleteButton');

$deleteButton.click(function () {
    if(selected !== null){
        if(selected.id !== 'start' && selected.id !== 'end' && 
!($(selected).hasClass('tr') || $(selected).hasClass('fa')))
            $(selected).remove();
        selected.style.backgroundColor = '#2196F3';
        selected = null;
    }
});
```
Execute.js

```
const $executeButton = $('#executeButton'),
    operations = {
        '+': function (x, y) {
            return x + y;
        },
        '-': function (x, y) {
            return x - y;
        },
        '*': function (x, y) {
            return x * y;
        },
        '/': function (x, y) {
            return x / y;
        },
        '==': function (x, y) {
            return x === y;
        },
        '>=': function (x, y) {
            return x >= y;
        },
        '<=': function (x, y) {
            return x <= y;
        },
        '>': function (x, y) {
            return x > y;
        },
        '<': function (x, y) {
            return x < y;
        },
    };

$executeButton.click(function () {
    let $current = $('#start'),
        $console = $('#console');
    let index, value, nr = 1,index1,index2,op,sel;

    $console.empty();
    $console.append('<p style="color: black">1: Start</p>');

    while ($current.attr('id') !== 'end') {

        let next = $current.find('.next').first();
        if (next[0] && next[0].innerText !== '') {
            next = next.text().slice(0, -1);
            $current = $(('#' + next));
            console.log($current);
            next = next.split('-')[0];
            switch (next) {
                case 'displayText':
                    $console.append('<p style="color: black">' + (++nr) + ': ' +
$current.find('input').val() + '</p>');
                    break;
                case 'setVariable':
                    index = parseInt($current.find('select').val());
                    value = parseInt($current.find('input').val());
                    variables[index] = value;
                    break;
                case 'displayVariable':
                    index = parseInt($current.find('select').val());
                    $console.append('<p style="color: black">' + (++nr) + ': v' +
(index + 1) + ' = ' + variables[index] + '</p>');
                    break;
                case 'operationVariables':
                    sel = $current.find('select');
                        index = parseInt($(sel[0]).val());
                        index1 = parseInt($(sel[1]).val());
                        index2 = parseInt($(sel[3]).val());
                        op = ($(sel[2]).val());
                    variables[index] = operations[op](variables[index1] || 0,
variables[index2] || 0);
                    break;
                case 'if':
                    sel = $current.find('select');
                        index1 = parseInt($(sel[0]).val());
                        index2 = parseInt($(sel[2]).val());
                        op = ($(sel[1]).val());
                    if(operations[op](variables[index1] || 0, variables[index2] ||
0)){

                        $current = $current.find('.tr');

                    } else {
                        $current = $current.find('.fa');
                    }
                    break;
                case 'end':
                    $console.append('<p style="color: black">' + (++nr) + ':
End</p>');

                    break;
                default:
                    $console.append('<p style="color: black">WIP</p>');
            }
        } else {
            $console.append('<p style="color: red">ERROR. END NOT SET</p>');
            break;
        }
    }

});
```

4. Testing and Validation

The testing was done mostly by me and by trying edge cases for the code, like infinite loops, which are still possible.