MSE Capstone Project

A Structured IDE for JavaScript using JavaScript

Overview

Write a dynamic web page that allows students to write JavaScript functions without just typing text into a text-editing window. Editing takes place by performing actions like selecting expressions from drop-down boxes; dragging and dropping statements into place and typing names into text boxes.

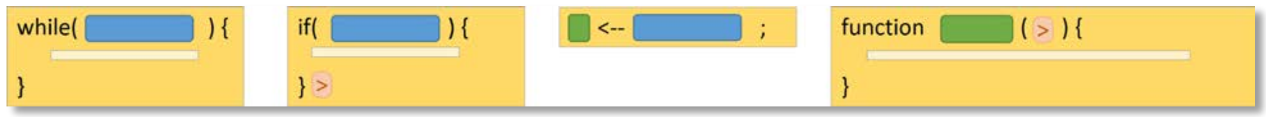
Details

This is an HTML, CSS, and JavaScript project. The entire app should use one HTML file with a CSS style sheet and JavaScript libraries as needed. No server-side code is required. Our structured editor will support a language similar to what is described below:

```
<app> ::= <stmts>
<stmts> ::= <stmt> <stmts> | <stmt>
<stmt> ::= <assignment> ; | <while> | <if> | <function>
<function> ::= function <variable> ( <arglist> ) { <stmts> }
<arglist> ::= <variable> | <variable> , <arglist> | <empty>
<assignment> ::= <variable> ← <expr>
<while> ::= while ( <expr> ) { <stmts>  }
<if> ::= if( <expr> ) { <stmts> } |
         if( <expr> ) { <stmts> } else { <stmts> } |
         if( <expr> ) { <stmts> } else <if>
<expr> ::= <factor> | <expr> * <factor> | <expr> / <factor>
<factor> ::= <logical> | <factor> + <logical> | <factor> - <logical>
<logical> ::= <comparison> | <logical> AND <comparison> | <logical> OR
<comparison> | NOT <logical>
<comparison> ::= <term> | <comparison> <COP> <term>
<COP> ::= < | > | <= | >= | = | <>
<term> ::= <variable> | <number> | ( <expr> )
```

Each statement (<stmt>) should be a draggable object that can be pulled from a “statement menu” and dropped into place. These objects are empty shells that have place-holders and interactive elements that allow users to fill in the necessary remaining parts.

The following figure gives a rough idea of what a “statement menu” might look like. Each of these elements is draggable and represents a statement (assignment, while, if, function) that can be placed into a program. The application must also have an “expression menu” that contains draggable elements for each of the expression (<expr>) elements.



Once a student has ‘filled in’ all of the blanks, they will have written a valid JavaScript program. The application should also support a ‘run’ button with some type of debugging utility that allows students to

1. Step through the program
2. Show the computational state of each executed element
3. Handle exceptions (rather than crashing the application) if they arise.