CSE 3302 Lab Assignment 2

Due July 16, 2014

Goals:

Understanding of JavaScript and elementary compiler/interpreter concepts.

Requirements:

1. Add the following as built-in ("intrinsic") functions to PL/0:

   a. `cvclear()` simply clears the new canvas using `clearRect()`. The canvas should have a width of 500 and a height of 300.
   b. `cvball()` has three arguments: x, y, and radius for calling `arc()` to draw a filled red ball centered at x and y.
   c. `cvdraw()` has five arguments: picture number, x, y, width, and height for calling `drawImage()` to render an image. The picture number will be in the range 1 .. `imagemax`, where `imagemax` will be a built-in constant.

2. To allow PL/0 users to perform animation, add a `wait()` function whose single argument gives the number of milliseconds to delay. This feature will be much more convenient than stall loops. It will be implemented using JavaScript's `setTimeout()`.

3. Email your *zipped* files to `sourabh.bose@mavs.uta.edu` by 12:45 p.m. on July 16. MavMail will block a number of file types/extensions, including .js. The body of your message should indicate the browser(s) you tested with.

Getting Started:

1. Useful files are at: `http://ranger.uta.edu/~weems/NOTES3302/LAB2SUM14/`

2. Minimal changes will be needed for the compiler portion of PL/0 (e.g for generating code to call built-in functions). You will need a new "kind" for built-in functions. Like the implementation of `in` and `out`, the necessary names will be loaded into `table` before the initial call to `block()`.

3. The interpreter will need code for each of the new built-in functions in the `cal` instruction processing. The needed parameter values should be available on the RTS when the `cal` is encountered.

4. The pictures for `cvdraw()` should be pre-loaded from the same directory as your `.html` and `.js` files, e.g. such as the `start_canvas()` from the Notes 02 examples. You should have at least three pictures.

5. Implementing `wait()` completely is especially challenging. `setTimeout()` will be called with a function to wake up the interpreter along with the duration of the wait, but you will then *exit execution*. Your final version should still support breakpoints and stepping.

6. You may assume the user will not hit the "Compile", "Run" and "Continue" buttons while the PL/0 interpreter is running.

7. Optionally, you may include any (interesting) PL/0 code you create. Be sure to describe it in the body of your submission message.