Mathsoc Programming Workshop: Week 3

Introduction

This week will primarily be a catch-up session, but for those of you who get up to speed, we will be learning how to animate our 2d drawings!

For this week's program, you will be creating new blank files and using the ever helpful copy/paste method of programming. This means that you will have to have completed the previous week's exercises before you get started. If at any stage you get stuck you are advised to ask others, the workshop helpers, or look for relevant materials online.

Setup

For these exercises, you will be creating a new workspace and importing your previous weeks work as projects within that workspace.

1. Go to the Cloud9 dashboard and click on *Create New Workspace*. Give your workspace a name which doesn't refer to it being week 3 (you'll see why later). Under *choose a template*, select HTML5, then click *create*.

2. You will be presented with two template files: `hello-world.html` and `README.md`. Delete both of these by right-clicking their entries on the left-hand panel. Now look at the top of this panel and see that there is a folder with the name you gave the workspace: right-click on this and create three *New Folders* named `week1`, `week2` and `week3`. These folders are *projects within the workspace*.

3. Now open the workspace you used in week 2. Highlight all of the files in the left-hand panel and download them (right-click). Extract them from the zipped folder (both layers) to some location on your computer (they can also be downloaded individually if your computer can't handle .gz format). Now go to your new workspace, highlight the `week2` folder and click on *File – Upload Local Files...* to upload the files you downloaded.

4. You can repeat this for week 1 if you wish.

5. Right-click on the `week3` folder and create two *New Files* called `make-stuff-happen.html` and `make-stuff-happen.js`. Open both these files - note they are completely blank. Now open `function-plotting.html` and `function-plotting.js` from the `week2` folder, copy/paste everything from them into the new files, and close them.

6. You now need to edit `make-stuff-happen.html` so that it makes sense:

   - Change the title to say Week 3.
   - Change the line:
     `<script src="function-plotting.js"></script>`
     to say
     `<script src="make-stuff-happen.js"></script>`

You can now right-click and preview 'make-stuff-happen.html' and it should correctly run the copied code from last week's exercises (that is, draw a pretty graph).

Program 5: Moving Pictures

The goal of these exercises is to create some animated drawings.

1. Look at `make-stuff-happen.js`. This is where you'll be doing everything this week. Delete everything in the file except for the function `doSomeDrawing()` and the lines

```
$(function() {
    doSomeDrawing();
});
```

You can always open the code from previous weeks whenever you need to refer to it.

2. These lines are executed when the html page is loaded. At the moment they simply call the function `doSomeDrawing` once and then end. We want to create some animation, so we want to *call a function repeatedly over time*. You might think to do this by using a loop, however in this particular situation that will not work. Fortunately, Javascript has a built-in function, `setInterval(callback, ms)`, which tells the computer to run the function `callback` every `ms` milliseconds. Replace the line

   ```
   doSomeDrawing();
   ```

   with

   ```
   setInterval(doSomeDrawing, 30);
   ```

3. This will now run the function doSomeDrawing every 30 milliseconds (about 30 times per second) while the page is open. Note that the function `setInterval` is not *calling* `doSomeDrawing` - rather doSomeDrawing is being passed in as a parameter, and is being called elsewhere in the underlying Javascript.

   You'll now want to get your code drawing something again, so refer back to last week's code and edit the function `doSomeDrawing` until it draws something of your choice onto the canvas. You will want to keep the lines

   ```
   var canvas = $("#canvas")[0];
   var ctx = canvas.getContext("2d");
   ```

   at the top of the function, and replace the rest of it with your new code.

4. In order to animate your drawing, it will need to change over time - that is, it will need to vary with some parameter, say `t`. But `doSomeDrawing` isn't being called by your own code anymore, so adding a parameter to the function definition won't work. You need to use *Global Variables!*

   At the very top of the code - above all the function definitions - put the line

   ```
   var t = 0;
   ```

   and make your drawing code depend on `t` in some way. `t` doesn't vary yet, but test out your changes to make sure the code at least works when `t` is zero.

5. Now add the line

   ```
   t += 1;
   ```

   to the end of the `doSomeDrawing` function and watch what happens when the code is executed.

6. Something very strange? Yes - the canvas isn't being erased before being drawn on again, so things will just get smeared all over the place. To fix this, put the line

   ```
   ctx.clearRect(0, 0, canvas.width, canvas.height);
   ```

   before your code for drawing. Now look at what happens when you tweak the value you increase `t` by each frame and the number of milliseconds between each frame.

Exercise

See if you can display a ball bouncing off the bottom edge of the canvas. Bonus points for:

- creating a separate ball-drawing function which takes the radius and x and y values as parameters.
- using a `for`-loop as the basis of said function. `Math.cos(x)` and `Math.sin(x)` will be helpful.
- making the ball lose momentum with each bounce, so that it tends towards a resting state.

Next Week

We will be combining interactivity with our drawings and animations, as well as possibly looking into the uses of object-orientation.