

Working with Text

Overview

In this lab, you'll implement a web page that draws quadratic equations such as the following:

$$y = 3x^2 + 6x - 3$$

All of the graphics are complete – you will display the text adornments for the web page. If time permits, you will also explore the use of web fonts.

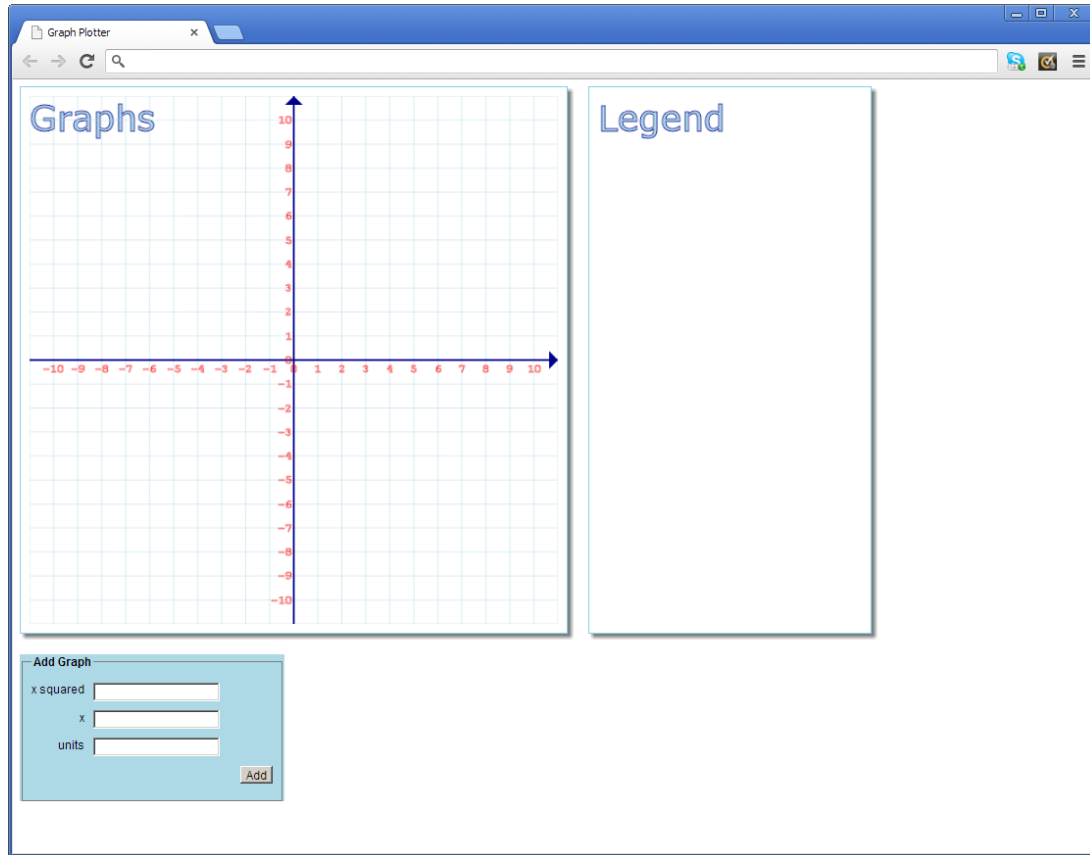
Roadmap

There are 4 exercises in this lab, of which the last exercise is "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Displaying simple text
2. Configuring text placement
3. Measuring text
4. Using Web Fonts (if time permits)

Familiarization

Open a browser window and browse to `GraphPlotter.html` in the *Solutions* folder. The web page appears as follows:



The web page contains two `<canvas>` elements and a `<form>` element:

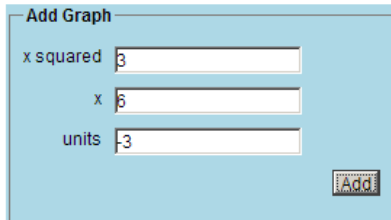
- The first canvas displays graphs for quadratic equations.
- The second canvas displays the quadratic equations textually in a legend.
- The form at the bottom allows the user to add another quadratic equation.

The following page describes how to use the web page.

The form at the bottom of the web page invites you to enter the coefficients for a quadratic equation. For example, imagine you want to plot the following equation:

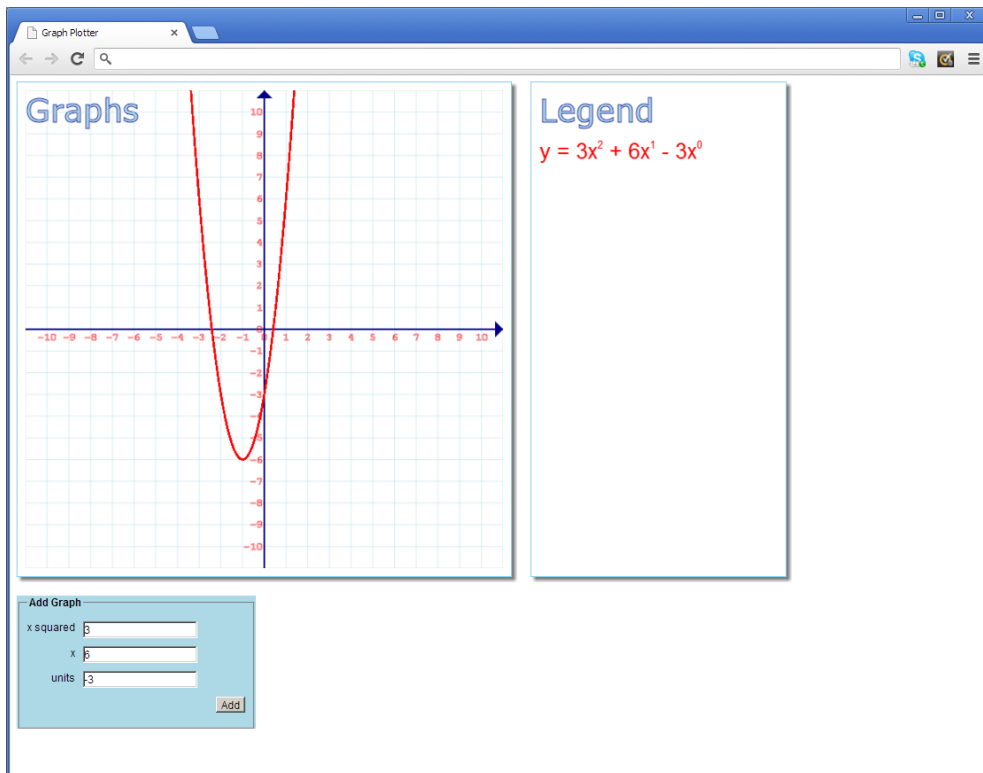
$$y = 3x^2 + 6x - 3$$

To plot this quadratic equation, enter the following values in the form:



A form titled "Add Graph" with three input fields: "x squared" containing the value 3, "x" containing the value 6, and "units" containing the value -3. There is an "Add" button at the bottom right of the form.

When you click *Add*, the web page plots the graph in the first canvas and displays the equation textually in the second canvas:



You can add any number of quadratic equations. The first canvas plots all the equations graphically, and the second canvas shows all the equations textually. Each equation is displayed in a different colour (red, orange, gold, green, blue, indigo, and violet). If you add more than 7 equations, the colour scheme wraps around to red again 😊.

Open File Explorer and open `GraphPlotter.html` in the text editor. Notice that the web page already has two `<canvas>` elements named `graphCanvas` and `legendCanvas`, plus a `<form>` element where the user can add another graph.

All the code for the web page is located in `GraphPlotter.js`. Open this file in the text editor and take a look at the existing code and comments. Note the following global definitions:

- The `Graph` object has properties named `a`, `b`, and `c` to hold the coefficients of x squared, x , and units. There's also a `colour` property, so that every `Graph` object knows what colour to display itself.
- The `CELL_SIZE` variable holds the width of each cell on the grid, in pixels.
- The `w` and `h` variables hold the width and height of the graph canvas, in pixels.
- The `graphs` variable will hold an array of all the `Graph` objects.
- The `legendContext` and `graphContext` variables hold the 2d contexts for the two `<canvas>` elements.

Now take a look at the following functions (all of which are complete), to get a feel for how we've implemented the web page. There are some interesting canvas techniques on show here:

- `onLoad()`
- `initLegendCanvas()`
- `initGraphCanvas()`
- `drawGrid()`
- `drawAxes()`
- `onAddGraph()`
- `drawGraphCurve()`
- `xCoord()` and `yCoord()`

Exercise 1: Displaying simple text

In `GraphPlotter.js`, locate the `drawHeaderText()` function. The purpose of this function is to display a header message at the top of a canvas. Add code where indicated by the `TODO 1` comment, to draw the header text with the following kind of appearance:



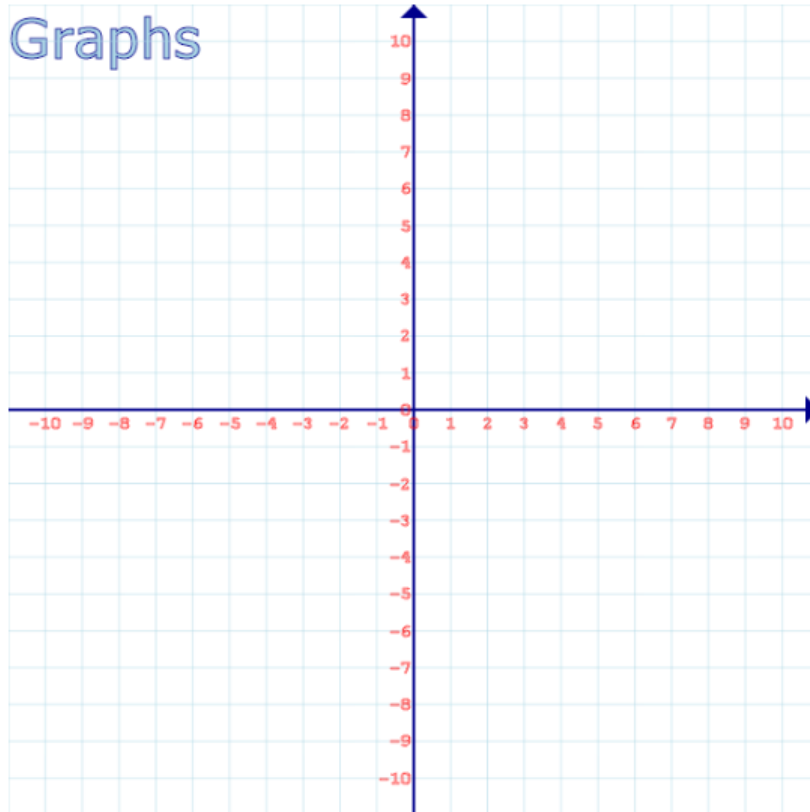
Hints:

- Choose a big sans-serif font, e.g. 30pt Verdana.
- Set the fill style to light blue.
- Set the stroke style to dark blue.
- Set the text base line to 'top', and then fill/stroke the text at position 0, 0 in the specified canvas context. The 'top' baseline ensures that the top of the text will be anchored at this coordinate. You might like to experiment with different values for the text base line, to see the effect.

Open the *Start* web page in the browser. Verify the header text appears correctly in both the graph canvas and the legend canvas.

Exercise 2: Configuring text placement

Locate the `drawLabels()` function. The purpose of this function is to display the number labels on the x and y axes. Add code where indicated by the `TODO 2` comment, to draw the labels as shown here:



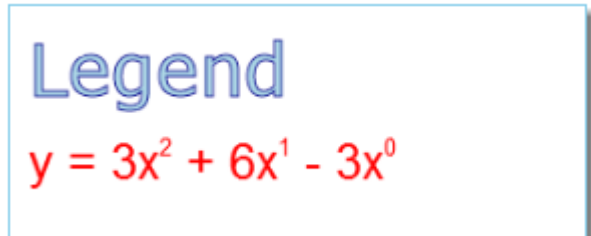
Hints:

- Choose a small font, set the stroke style to red, and set the line width to 0.5.
- To draw the labels on the x axis (i.e. the horizontal axis), loop through the x values from -10 to +10 inclusive. For each value of x, call the `xCoord()` helper function to convert the logical x value into a pixel-based coordinate. Then display the x value as text on the canvas, just beneath the x axis (think about the text base line and the text alignment, to achieve exactly the effect shown above).
- To draw the labels on the y axis (i.e. the vertical axis), loop through the y values from -10 to +10 inclusive. For each value of y, call the `yCoord()` helper function to convert the logical y value into a pixel-based coordinate. Then display the y value as text on the canvas, just to the left of the y axis (think about the text base line and the text alignment again).

Refresh the *Start* web page in the browser, and verify the labels appear correctly.

Exercise 3: Measuring text

In this exercise you'll write code to display a quadratic equation textually in the legend canvas. For example:



To display an equation, you have to take the following things into consideration:

- You must split the text into several distinct parts, so that you can display the power subscripts ², ¹, ⁰ in a smaller font.
- You must set the text base line to 'top', to ensure the power subscripts really are displayed at the top level of the text.
- You must measure each separate part of the text, so you know where to place the next part.

To encapsulate the task of drawing a text snippet, we've written a stub function named `drawLegendTextSnippet()`. Locate this function at the bottom of the file, and implement it according to the TODO 3a comments.

Then locate the `drawGraphEquation()` function. The function receives a `Graph` parameter, which represents the graph equation to display. For the example in the screenshot above, the `Graph` object would have the following properties:

- `a`: 3
- `b`: 6
- `c`: -3
- `colour`: 'red'

Add code where indicated by the TODO 3b comment, to draw the graph equation textually as per the screenshot.

Refresh the *Start* web page in the browser. Enter an equation and click *Add*. Verify the equation text is displayed correctly in the legend canvas, in red. Add another equation and verify it's displayed correctly too, in orange. What happens if you enter negative coefficients... does the equation display correctly?

Exercise 4 (If Time Permits): Using Web Fonts

Add some static text to the web page, and explore the use of web fonts. There are various public web sites that provide web font services – download some free web fonts and integrate them into the web page by using font linking.