CS 530: Developing User Interfaces

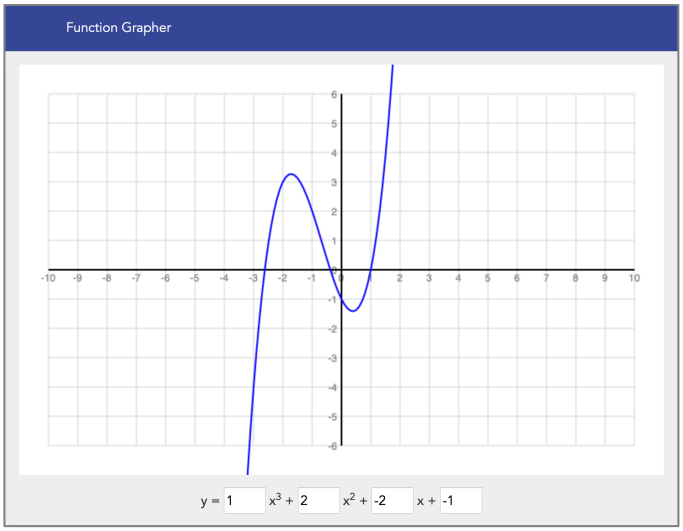
Assignment 3

Goals

This assignment allows you to practice drawing a graphical view using JavaScript and an HTML canvas, combining simple interactive HTML elements to provide data for the graphical view.

Assignment

This assignment asks you to develop a function grapher that takes parameters for cubic equations and graphs the results on a grid. The final product for this assignment will look like this:



The page allows the user to interactively change the coefficients in the equation, and whenever the user makes a change, the page automatically redraws the graph to show what the cubic equation would look like on the graph.

Implementation Setup

To start this assignment, first please download the **a3.zip** package that comes with the assignment. The directory structure contains an HTML file **index.html** that, by double-clicking, you can view in your browser. (We will not be using Flask or Python for this assignment.) The rest of the package contains various files you need for this assignment: a skeletal JS file, **grapher.js**, where you will implement your grapher; and a basic CSS file, **grapher.css**, which you can use to add styles if needed. Besides these three files, you should not need to modify any of the given files for this assignment.

Grapher

The core task in this assignment is to implement the "Grapher" component. The **grapher.js** file contains the start of a Grapher object to flesh out with the necessary functionality.

To construct the grapher, one possible code structure is to separate the creation of the grapher vs. the building process by which the components are added to the page. For example, if your **index.html** file contains a call like the one below, the initial **Grapher()** call creates the object, and then the **build()** method adds the necessary HTML components to **div** defined by the identifier **#grapher-wrapper**. This strategy enables you to better control within **index.html** where the components will be inserted in the page. (It also facilitates creating multiple versions of the same object — not needed for this assignment, but sometimes a useful benefit.)

```
new Grapher().build('#grapher-wrapper');
```

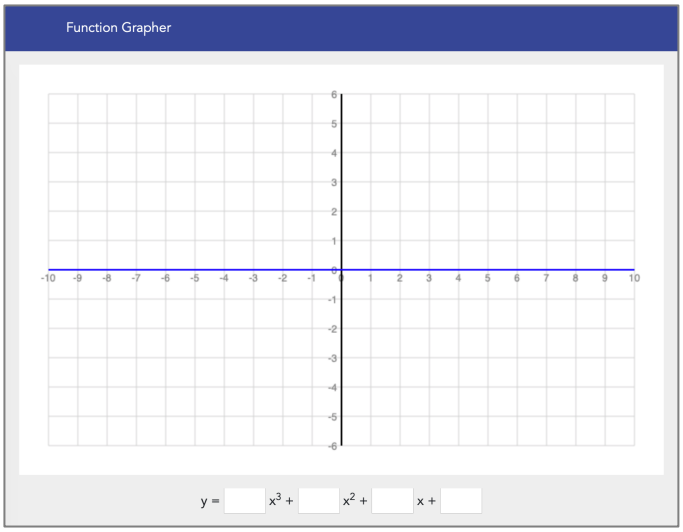
Your **build()** method, as mentioned, adds the various HTML components (using jQuery) to the parent component. This includes the **canvas** where the graph will be drawn, and the equation and input controls needed for the equation coefficients.

Your graph should be drawn to look as much like the examples here as possible. It should have 20 grid cells across and 12 grid cells down, and each grid cell is 35 pixels wide and high. The main axes should be drawn in black with a line width of 2, and the other grid lines in light gray with a line width of 1. The grid labels should be drawn in gray 12px Helvetica.

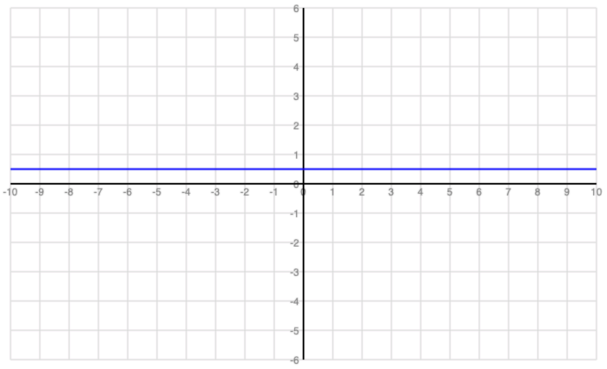
The graphed line should be drawn in blue with a line width of 2. To draw the line, please keep it as simple as possible. Specifically, you should not try to plot out an actual cubic spline, but rather sample the points across the grid: create a list of (x, y) points for the function such that x lies in the range of [-10,10], and by incrementing x by a small amount (for example, 0.1), you will have enough sampled points to create a nice graph. Note that simple straight lines between the points are sufficient (the small sampling will help the curves look smooth).

For the input boxes, any change to any input should cause the graph to be redrawn and updated. You can use the jQuery **on('keyup', fn)** method for this purpose. If an input box is blank or contains anything other than a valid floating-point number, its value should be assumed to be 0 for purposes of graphing.

Below are some sample graphs to compare with your own.

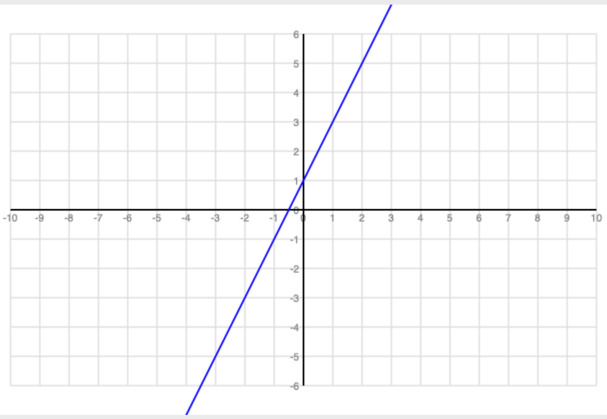


Function Grapher



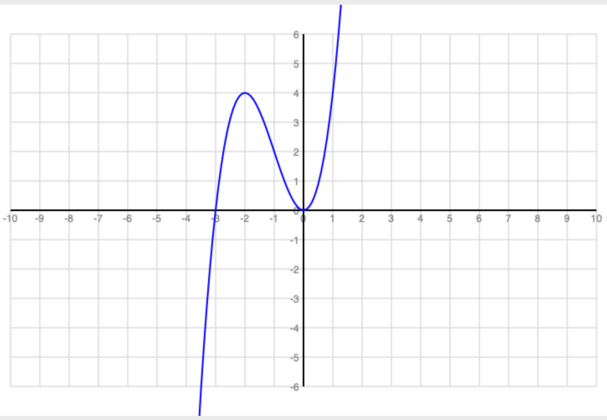
$y =$ $x^3 +$ $x^2 +$ $x +$ 0.5

Function Grapher

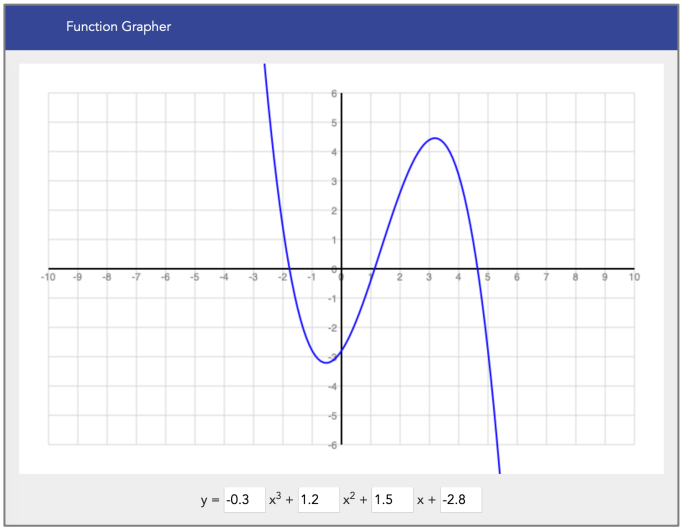


$y =$ $x^3 +$ $x^2 +$ 2 $x +$ 1

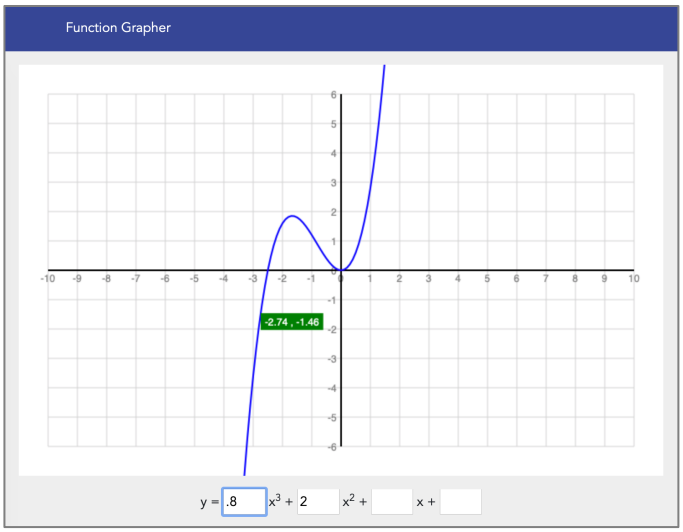
Function Grapher

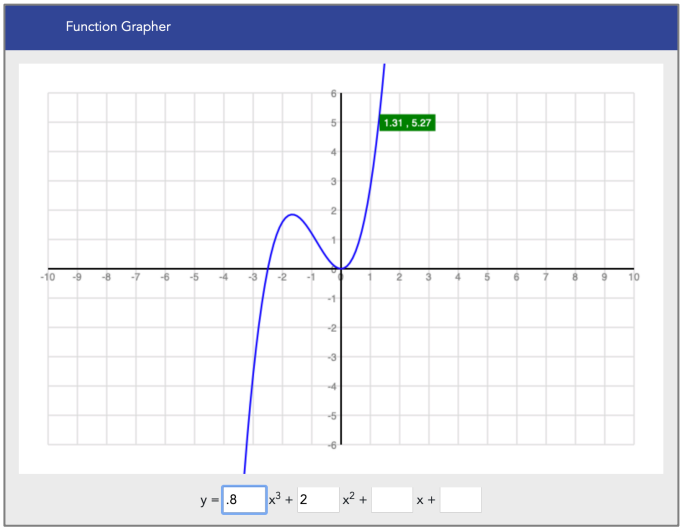


$y =$ 1 $x^3 +$ 3 $x^2 +$ $x +$



As a final piece, we will make the graph interactive: when the user hovers the mouse over the canvas, the mouse's *x* location will be used to highlight a particular data point, showing the *x*,*y* point at that location. At that data point, you should draw a green rectangle where the upper-left corner of the rectangle lies directly on the data point. Inside the rectangle, display the *x*,*y* location in white text. You can use the jQuery **on('mousemove', fn)** command on the canvas element to handle mouse-move events, and within your handler function that takes an **event** argument, **event.offsetX** will give the mouse *x* position. It should be straightforward to calculate the *y* value from *x* (given that you're already doing this to generate the function). Note that the highlight box should be displayed at the mouse's *x* coordinate regardless of the mouse's *y* coordinate (for example, the mouse cursor could be inside the canvas but well above or below the line — that's ok, the highlight box should still appear at the appropriate place on the function line). Below are two examples of highlighted data points.





Documentation

It is expected that all code written for this assignment is properly commented where needed, especially in places where you have made particular choices about data structures and/or algorithms to employ. Also, please add the following identification header to every file you create:

```
# Your Name, Your Email
# CS530: DUI, Assignment [#]
```

(or whatever commenting syntax is appropriate for the file at hand).

Submission

Please submit your files as an attachment for the assignment on Blackboard Learn. Please use a compression utility to compress your files into a single ZIP file (not RAR or any other compression format). The final ZIP file must be submitted electronically using Blackboard—do not email your assignment to a TA or instructor! If you are having difficulty with your Blackboard account, you are responsible for resolving these problems with a TA or someone from IRT before the assignment it due. If you have any doubts, complete your work early so that someone can help you.