Transforming Drainage Project Algorithm

Overview & Purpose

The Transforming Drainage Project's algorithm is the backbone of the application. It is what utilizes the 30+ years of data gathered by researchers along with user inputs to produce the graph outputs.

Implementation

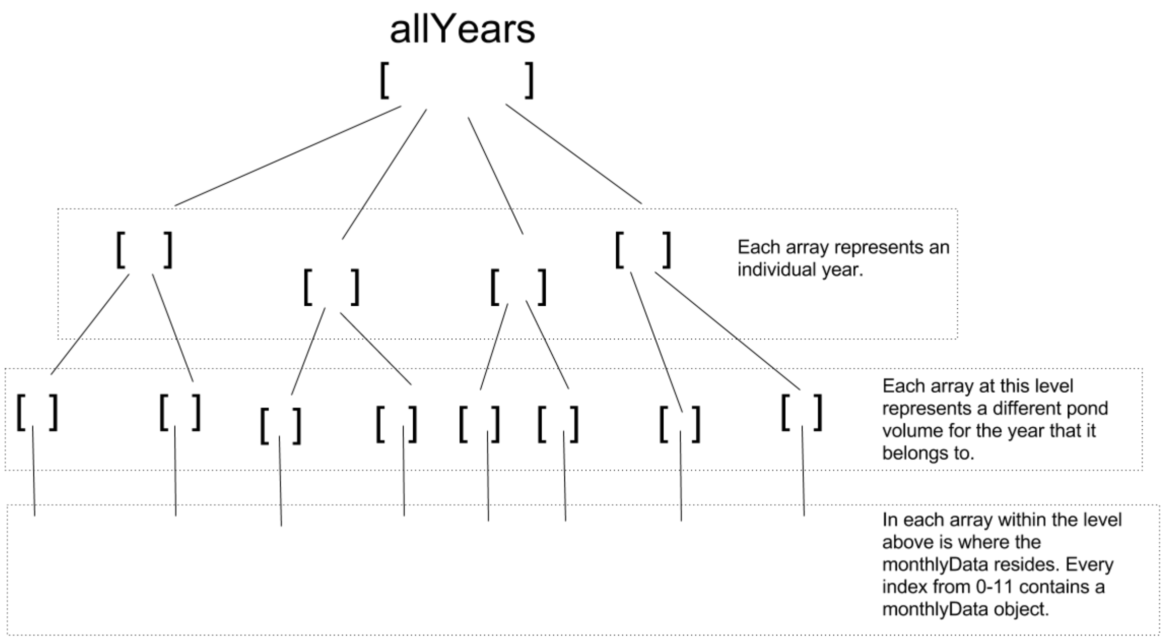
The current implementation of TDPAlg is located within /utils/TDPAlg.js. The code is all run on the server and is only ever run once per calculation. Because of this the algorithm must respond to the client with enough data to produce every possible graph the user could select. Details on how this works below in *Dataflow*. TDPAlg is only called from server.js (located in root folder) where all user inputs are passed into the function TDPAlg.calc. The function responds with an object that contains an array with all graph information, an array of all pond volumes calculated using users increment value, and the corresponding year to the first row of data from the database.

Workflow

TDPAlg starts by querying the database with the passed in location argument. Once this is done the calculations begin a loop starting at smallest pond volume and going until largest pond volume(both specified by user). Inside of this loop is another loop that goes through every row in the database (each row represents a day). Inside of this loop is where daily values are calculated and once the loop finishes the daily values are added to its corresponding year, increment, and month.

Dataflow

The data structure allYears, which contains all graphing information requires some understanding before using. I will explain its structure the best I can here and provide a figure below for better understanding. First, every entry in allYears array represents a single year. So, allYears[0] would represent the year 1980 if the year 1980 was the earliest year of data we had in the database. allYears would continue in order from there: 1981, 1982, etc. Now, at each of those years is another array, so allYears[0] = []. Again, this array indirectly represents the year 1980 (the reason TDPAlg.calc returns the initial year is because the years are never explicitly stored inside of allYears). Now, what is in the array stored in allYears[0]? The array stored at allYears[0] contains more arrays all of which represent a different increment. Quick review: allYears[0] = [], allYears[0][i] would be the data from the year 1980 (assuming 1980 is the first year in database) and the data at i inside of the 1980 array would be data for the particular increment i. Let's say the user selected a smallest pond volume of 10 with an increment of 2. Then allYears[0][0] would represent the data stored at 1980 (assuming 1980 is the first year in database) at the very first pond volume which in this case is 10. Continuing to the next entry allYears[0][1] would be data stored for the year 1980 for the pond volume of 12. So, what is actually stored at allYears[0][1]? You guessed it another array! This array is always of length 12 because each index has data for every month. All month data is stored within a monthlyData object (definition in TDPAlg.js). Another recap: allYears[0][1][0] (maintaining values from above) would return a monthlyData object for the month of January for the pond volume of 12 in the year 1980.


allYears
[ ]
Each array represents an individual year.
Each array at this level represents a different pond volume for the year that it belongs to.
In each array within the level above is where the monthlyData resides. Every index from 0-11 contains a monthlyData object.