Debugging Log

Logged a total of **2.8 hours** of debugging.

Entry 2

35 minutes spent debugging

Failure

Several query results are getting reversed order on displaying, But the content of results is correct.

Experiments

Experiment 2.1

Question: Why is the displayed order reversed

Steps Taken: Firstly, I clicked the inspect to get into the network panel, and click the current wrong combination of buttons to get what route is the "click on button" routed to. The current one is findByFriend. So the findByFriend function had something wrong. I thought it because the order of element in res.results, which is an array is not correct. So I pop the results out in a temp array, and push them back in.

Result: When I open the network panel and click the button, it told me which route is being routed and what function is called. Turns out it's the findByFriend's result is in inversed order.

Lesson: Need to be careful about the order of elements after push pop operations

Experiment 2.2

Question: Can I actually use a for-loop + subscript instead of while-loop+pop to prevent the bug?

Steps Taken: replace while with for-loop

Result: order is correct now

Lesson: while+pop, and for+subscript are in reverse order

Defect

```
127
```

After a couple push and pop, the order of the element in the result is not what we want.

Types

Type checking **would not** have helped. This is a bug about order of elements in an array, caused by push pop operations, not type

Entry 4

1.4 hours spent debugging

Failure

Every time I click the chemistry building, different friends are displayed. And the friends are not close at all.

Experiments

Experiment 4.1

Question: What caused the output to be different even when clicking on the same building?

Steps Taken: Restarted the server multiple times to see if the first output is the same.

Result: The first output is always the same, though not correct.

Lesson: The issue must be related to something that resets when the server restarts. There's no randomness in the code.

Experiment 4.2

Question: Could it be that the map declared isn't reset every time the query comes in and the function is called?

Steps Taken: Added console logs to check the map.

Result: The map is reset every time the function is called.

Lesson: The problem is not with the map.

Experiment 4.3

Question: Could the results returned from sortBuildingsByPosition be incorrect?

Steps Taken: Console logged the results (a sorted array by distance).

Result: The distance values were NaN.

Lesson: Since the distance was NaN, calculations and comparisons weren't working properly. This might not be the root cause of the bug, but it's an issue that needs fixing.

Experiment 4.4

Question: Are all parameters (x, y, x_2, y_2) successfully passed and parsed in sortBuildingsByPosition?

Steps Taken: Added console.log for coordinates and distance calculation.

Result: All coordinates were passed in correctly, but the distance was still NaN.

Lesson: The bug is in the distance calculation: const distance = (x_2-x)**(x_2-x) + (y_2-y)**(y_2-y);

Experiment 4.5

Question: What's wrong with the distance expression?

Steps Taken: Logged each part of the calculation.

Result: The expression (x_2-x)**(x_2-x) was evaluating to infinity

Lesson: A simple mistake in the formula; it should be const distance = (x_2-x)**2 + (y_2-y)**2;

Experiment 4.6

Question: Why do different persons appear when clicking the same spot, and why are displayed friends getting farther away? Could it be because the ordered array of friends gets popped each query?

Steps Taken: Logged the length of results returned by sortBuildingsByPosition during multiple queries.

Result: All result arrays had a length of 52.

Lesson: The problem isn't with the result array or sortBuildingsByPosition function.

Experiment 4.7

Question: What causes findFriendsByPosition to return different people when querying the same location multiple times, even though the sorted building array is correct?

Steps Taken: Checked for pop() operations on global variables like schedulesByShortName.

Result: No pop operations were found for Map objects.

Lesson: Need to review the entire logic of findFriendsByPosition.

Experiment 4.8

Question: At what steps are the closer buildings changing?

Steps Taken: Logged closer buildings while querying the same location repeatedly.

Result: The closer buildings did change, but previous closer buildings were also logged.

Lesson: The issue isn't with pop operations on sortedBuildings, but might be because the friends recorded with each building are changing.

Experiment 4.9

Question: Is it because the friends related to buildings are changing?

Steps Taken: Realized that when getting related friends from the Map using get(), the returned array was being modified with pop(), which affected the global variable schedulesByShortName.

Result: No additional steps needed, already identified the bug.

Lesson: Global variables should not be modified.

Experiment 4.10

Question: Should I use a for loop with indexing to access the array from map.get(), or copy the global map for each function call?

Steps Taken: Considered that copying a global array would use more time and space, while changing from while+pop to for+[] seemed more efficient.

Result: I changed the while to for loop, and it works, even solved another bug that is the displayed friends is off by one friend who is also in the same building. Because if use pop, the last friend will pop up, and use indexing, the first friend got displayed first. What did you learn from that?

Lesson: When array indexing and for-loops can solve a problem, avoid while-loops and pop() operations.

Defect

```
236, 205+206
```

The first bug is caused by mistakenly calculating the distance, did not clearly understand the operator **'s function, actually I knew what it means, but just too stupid to use it right. The second bug is again a while-loop+pop() bug which modifies the global variable, which happens every single time when I tried to use while and pop in this assignment.

Types

Type checking **would not** have helped. First bug is a calculating bug, second bug is a while+pop bug, nothing related to type.

Entry 5

28 minutes spent debugging

Failure

The findBuildingsByPosition function is definitely not returning the closest buildings. And the buildings being shown is always the three. Since I used a different algorithm than the findFriendsByPosition, I should debug from the very begining, or I can also use the helper function wrote for findFriendsByPosition to make things eadier.

Experiments

Experiment 5.1

Question: Whether the three buildings displayed is from the first for loop.

Steps Taken: log out the firstThree array's elements

Result: Yes they are belong to the first for loop.

Lesson: so the problem is on the second for loop.

Experiment 5.2

Question: what part of the second for loop is not working

Steps Taken: Add console log part by part to see which part didn't work.

Result: The while loop on line 91 is not triggered at all and the distance is not changing at all.

Lesson: The element I got from the global variable buildings is not chaging at all. Therefore, the bug is on line 85.

Experiment 5.3

Question: What makes the line 85 const element = buildings[index]; always return the same value

Steps Taken: I see I was using the index from the previous loop, not the current loop's i, so change it to buildings[i].

Result: while loop triggered, but the server crashed with the error distance is undefined, note that this distance is from the firstThree array's distance property

Lesson: Probably because after pop every element in the firstThree, the while loop still check for next element, so a typical Subscript out of bound

Experiment 5.4

Question: Does add a check if add another condition checks on whether the length of firstThree hit the 0 fixs the bug?

Steps Taken: add an extra condition firstThree.length > 0 && to the while loop

Result: it works, bug fixed!

Lesson: Carful about the bounds for array when accessing it not by using array.length, for example array.length - 1

Defect

```
85,  91
```

on line 85, not using the correct index. on line 91 subscript out of bound

Types

Type checking **would not** have helped. the main bug is a subscript out of bound bug, nothing about type

Edit Log