## Questions

### Biography:

How old are you?

* 25

Have you used JavaScript before?

* Yes

Have you ever written asynchronous code? (async, await, promise, then, …)

* Yes, but not sure how good his understanding is

From a scale of 1-10 rate your understanding of how asynchronous code works in JavaScript?

* 6-7

### Reading the code:

***Now pass the code example to the user. Give them enough time to read and understand the code.***

Did you fully understand the functionality of the provided code?

* APIs - easy, async
* planTrip
  + Understands the await on the fetchDestinationDetails
  + Wasn’t sure about .includes() at first, but got it after
  + Rest of it - all good
* Pretty much a perfect understanding of the code

Describe the control flow of the code using the line numbers

* Line 37 - Executes planTrip
* Lines 38 onwards - .then() or the .catch() get executed after planTrip is finished
  + The .then() kind of screwed him up a little bit at first, since he uses async/await more
* Line 22 - fetches the destination, waits for the result
* Lines 25-27 - checks if the variable has the destination, and returns if it does
* Lines 29-31 - fetches the other things one at a time
* Lines 33-34 - console log the results, and return them in an object
* Lines 38-40 - the stuff inside the .then() will be “populated” with the result of the planTrip()
  + The tripDetails part is only available inside that block

What would the overall runtime be? (Give a minimum in ms)

* Would just be an accumulation of the 4 runtimes
* Since you need to wait for it all to finish

Can you think of any changes you can make to the code that will reduce the overall runtime?

* Can’t just remove all the awaits
  + Because you’re not going to get all the things “populated in time”
  + Was concerned that there would be a “race condition” if you remove the awaits, and some of the results aren’t returned by the time you do the console log on Line 33
    - Doesn’t have a deep understanding of how promises work, based on this
* **I clarified that the goal is to get the same result as before, then he knew what to do:**
  + Can add them all into this “promise queue”
    - Promise.all()
    - But thinks it won’t work out..
    - Not sure in this case if it would optimize
    - Because it would need to wait for all 4 to come back
    - Then, realized after reading the code again that it will work
      * Realized that the other 3 calls are dependent on the first one, and the other 3 are independent of each other
  + First one needs to be awaited for
    - Notices they all rely on the first call
    - 29-31, can run in “promise synchronization”
      * Can add these into array of Promises, call Promise.all()
      * All 3 run at the same time
      * The one that finishes last (the 600ms) is what determines how long this part will take

### Our Program:

***Now show the expected results of the example that our program produces to the user.***

Were there any changes that the user did not come up with that the program did?

* The if statement check
  + **Forgot about that**
  + But understands that usually you check for a “null” case and return early

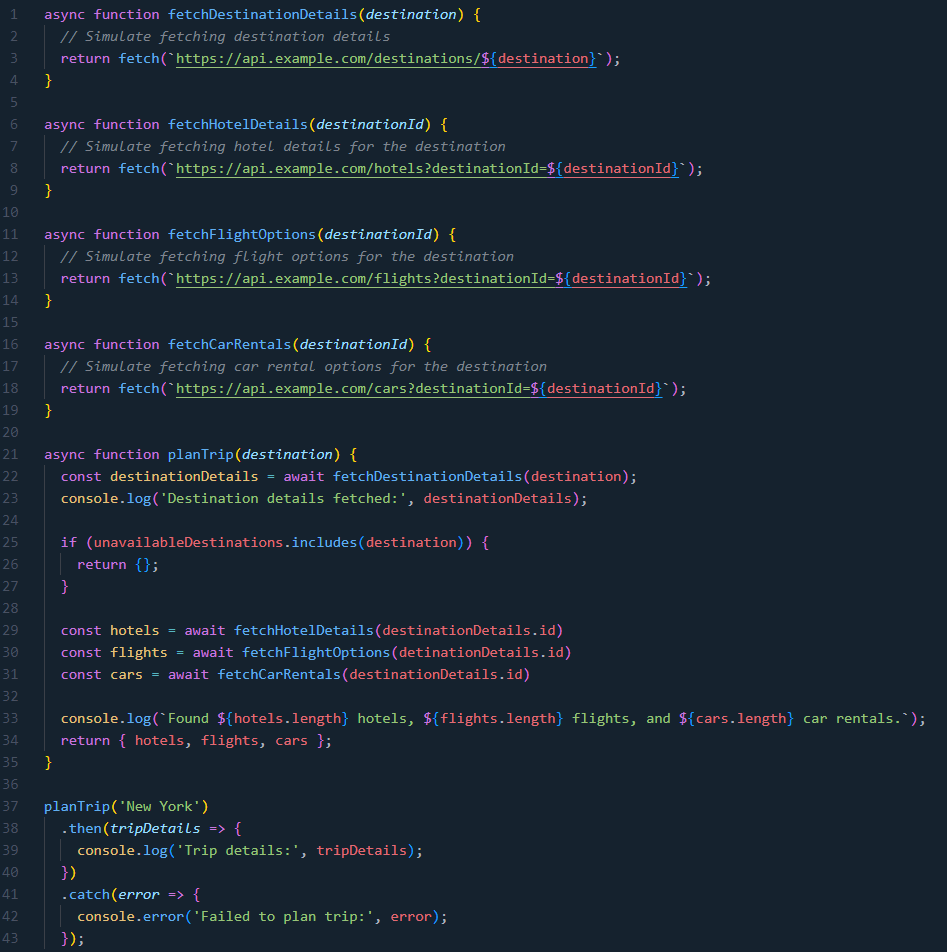
Were there any changes that the program did not come up with but the user did?

* Nope

Is the program output easy to understand? If not, provide any feedback

* Fairly straightforward, easy
  + But thinks that some people that are more beginner won’t understand Promise.all()

## Code



Assume that fetchDestinationDetails takes 1000ms, fetchhotelDetails takes 500ms, fetchFlightOptions takes 400ms, and fetchCarRentals takes 600ms.

What our program would suggest doing:



In the inefficient example the total runtime would be at least (1000ms + 500ms + 400ms + 600ms) = 2500ms as we’re fetching hotel details, flight options, and car rentals one at a time. By using Promise.all we’re taking advantage of fetching the data in parallel reducing the total runtime to at least (1000ms + max(500ms, 400ms, 600ms)) = 1600ms. The program would also suggest to move the if statement in planTrip to the beginning as it’s blocked from running as it waits for fetchDestinationDetails even though it doesn’t need the results from that async call to do the check.