Name: Nyasha Makaya

CodePost email: nmakaya@caltech

HW 2 Written Component

As noted at the start of the HW 2 Part B spec, there will be a small written component for you to answer. This is designed to make sure you understand the concepts of asynchronous JS, events, the DOM, and time management, as well as help you be aware of common bugs (we recommend not doing it last). **You are allowed to ask questions about these on Discord #hw2, or utilize weekend OH with EI to go over answers.**

The first two questions are optional, but encouraged to help (and thorough answers in this document may be considered to bump up the overall grade).

Grading will be based on demonstration of understanding; if you're unsure, ask! These are designed to help identify things students miss that will ultimately save time in the long run. Particularly strong answers/extensions are eligible for points lost somewhere else in HW 2A/B.

Bug List (Optional)

This part is optional, but provided first to encourage you to keep track of bugs that come up (and reference later). It also helps us know what students most commonly run into in this assignment! Add any bugs that come up, how you identified them, and anything you learned from the process of debugging. An example is started for you.

1. Negative time shown after 0:00 (-0:001 was shown on my page after 1 minute passed). <more details...>
2. The ingredient-list card keeps on responding after I set it to disabled

The DOM and Page Lifecycle (Optional)

What is the purpose of defer in the script tag? Why don't we use it for utils.js?

- If you didn't attend lecture when EI demo'd this and don't know the answer, try setting a breakpoint in init() and remove defer temporarily in your HTML. Step through, and you should see an error that you wouldn't get with defer (assuming your Part A was finished). Take a look at the Elements pane; what do you notice about the HTML?
- The purpose of defer is to tell the browser to run the javascript after the HTML has already been loaded/parsed.

1. Anonymous Functions with function() { } vs. Arrow Functions with () => { }

El introduced the module pattern with anonymous functions. Later, we learned about the `this` keyword and how to use it to simplify callback functions and event listeners. Arrow functions are popular "syntactic sugar" for anonymous functions in ES6, and you'll see them quite often in modern JS tutorials (they are popular for short functions, especially those passed to higher-order functions like map, reduce, filter, and forEach). We will continue to use these as we start working with APIs, especially when learning Node.js/Express. However, the two forms are not exactly the same.

Subtle bugs come up when using arrow functions as callbacks with `this`; take a look at the slides, and briefly identify the key difference between the two types of anonymous functions.

Answer:

While not required, we encourage you to think about _why_ the JS language designers (for ES6) chose to have this behavior. Also note that one form isn't necessarily better than the other, as long as you're aware of when this different behavior will have an impact in your program.

Managing Delayed Events with "Timers"

The unit on timers introduced the timerId variable (usually a module-global) to help implement most JS programs that involve delayed events. This question tests your understanding of the purpose of this variable and appropriate use.

Remember that this is a variable and a convention we expect you to use. The following questions should be answered using the lecture materials (in other words, assume timerId refers to the variable throughout the slides/code).

a. What is the purpose of timerId?

It is for keeping track of the timer so that when the time reaches 0, we can destroy the timer.
If we do not destroy the timer, it will keep on counting down in the background, which will result In issues in our HTML

b. What is the datatype of timerId? If you don't know, El showed this in lecture when using the console and the supplementary lecture recording will be useful to Review.

The datatype of timerId is number

c. What is the purpose of initializing timerId = null?
This ensures that we do not clear the interval before actually initializing it, because then , the value of timerId is null and the program will throw an error. It also ensures that we only have one timer running at a time, so that if value of timerId is not null, then there is another timer on it

d. Do we always need timerId when using setTimeout/setInterval?
If so, why?
If not: provide (and briefly justify)

- 1 example where it should be used
- 1 example (that still uses setTimeout/setInterval) where it is unnecessary

We always need a timerId when using setInterval. This is because setInterval calls the timer repeatedly forever, and we will need to clear it to stop it, so we need to know the timerId, or else it will run forever

However setTimeout only calls the timer once, so even if we do not get the timerId, the process stops on its own after the first call, so we do need always need a timerId when dealing setTimeout

e. If we have a timerId in our program, is it always needed to have a case to check if it's null? If so, why? If not, what is an example where it's unnecessary to check?

In the case where the timer is initialized within the scope of the function, there is no need to check if it is null, as there are no other functions accessing the timerId or editing it

f. Debugging: If you call setTimeout in the Chrome Inspector console, it will evaluate to a value. Suppose it evaluates to 500 but you expect 2. **Assuming this is a bug, what does this tell you? Justify your answer with an example**, either from lecture, OH, or a different example.

It tells you that maybe you put your time in terms of seconds instead of Milliseconds when calling setTimeout

g. Debugging: In the video, you'll see the count-down of the game time in the game view. It's important to always do the "3 click" (and a few more) test when working with UI and timers. Suppose you click to start a game, the game ends (and the displayed time says 0:00), you click to go back to the menu, then click to start a new game. On the new game view, you see an expected 3:00 for a 3-minute game. However, the displayed time decrements twice as fast (after 30 seconds, it shows 2:00 instead of 2:30). This is a classic bug when working with timers.

What does this tell you?
It tells you that the initial interval wasn't cleared, in fact, it kept running, and when a new game was created, the interval time for the new game was added to the interval time of the old value To make it twice as fast.

It may also means there are two instances of the timer running: and they are both decrementing the time, making it move twice as fast

How do you know, and how could you debug this?
Make sure that when you click back to menu, or when the game ends, you clear the interval using the timerId, and set timerId to null, then check before you start a new game that timerId is null

Reflection

a. What is one (or more) feature you could think of adding to this game? Be creative, but we're looking for you to stretch your thinking and make connections to the DOM and events in your answer. Consider game variants, different UI elements, storing (or retrieving) data on a server, etc. 1-3 sentences is fine, or you can add an updated wireframe for possible additional points.

A stack of all the burgers you have made: Instead of the burger disappearing when it is made, it is instead reduced in size, pushed to the left, and the burgers keep getting stacked to the left, using the wrap around flex functionality. That way the player can get to see how many orders they have completed so far, visually.

b. How did you like this new assignment? Is there anything you'd like to let us know to keep in mind for future terms (or even HW3)?

It was fun, except for the writing part

c. What did you find most helpful when working through this assignment?
The console was very helpful when debugging the code.

d. What form of feedback would you like the grader to prioritize when grading? Anything they should know about what you intentionally didn't prioritize (if applicable)? No need to go in depth, but this will help us keep grading efficient.

- Functionality of the game