**Training Days**

As a seasoned athlete, one of your favorite activities is running marathons. You use a service called Training Days that sends you a message for the event you signed up for and the days you have left to train.

Since you also code, Training Days has asked you to help them solve a problem: The program currently uses the wrong scope for its variables. They know this can be troublesome as their service evolves. In this project you will make Training Days more maintainable and less error-prone by fixing variable scopes.

If you get stuck during this project or would like to see an experienced developer work through it, click “**Get Help**“ to see a **project walkthrough video**.

**Tasks**

**1/12Complete**

Mark the tasks as complete by checking them off

**1.**

Let’s begin by running the **trainingDays.js** file. In the console we can see that the program is broken!

Ideally, the getRandEvent() function selects an event at random. The getTrainingDays() function returns the number of days to train based on the event selected. The logEvent() and logTime() functions print the athlete name, event, and number of days to the console.

But poorly scoped variables are causing errors.

**Expand days scope**

**2.**

To avoid the ReferenceError, declare days within the getTrainingDays function, before the if statement.

Stuck? Get a hint

**3.**

Run the program again: no error, but days is undefined! New days variables are being defined in the scope of each if/else if statement.

Delete the three let‘s within the if/else if statements.

**4.**

Run the program again: fixed! Now the if/else if statements are changing the original days rather than defining a new one.

**Make name global**

**5.**

The log functions–logEvent() and logTime()–use the same name variable. There seems to be a problem with the scoping; we can tell by the amount of duplicate code here! In addition to variables scoped too broadly, duplicate code can indicate that a variable may be scoped too tightly.

Let’s avoid this repetition by adding name as the first parameter for each function.

Stuck? Get a hint

**6.**

Move the name variable to global scope.

Stuck? Get a hint

**7.**

Pass name as the first argument to logEvent() and logTime().

Stuck? Get a hint

**8.**

Check that the program still works! Run it and check the output.

**Make random local**

**9.**

Try the functions for another competitor. Copy and paste this code at the end of the file.

const event2 = getRandEvent();

const days2 = getTrainingDays(event2);

const name2 = 'Warren';

logEvent(name2, event2);

logTime(name2, days2);

**10.**

Run the program. The events are assigned randomly, but Nala and Warren are running the same events!

**11.**

We see that the random variable is defined in the global scope. Each time getRandEvent() is called, it uses the same value.

At the top of the file, move the random variable from the global scope to block scope within the getRandEvent function.

Stuck? Get a hint

**12.**

Well done! Training Days is more maintainable and less error-prone thanks to your work. Run the program a few times to make sure the results are randomized.