JavaScript Advanced Functions: First-Class Functions Lab

- Due No Due Date
- Points 1
- Submitting a website url

(https://github.com/learn-co-curriculum/phase-1-first-class-functions-lab) (https://github.com/learn-co-curriculum/phase-1-first-class-functions-lab/issues/new)

Learning Goals

- Assign functions to a variable.
- Store functions in a data structure.
- Write functions that return other functions.
- Pass a function to another function.
- Call a function returned by another function.

Introduction

Functions are a very important part of JavaScript, and you will use them all the time. Without functions, we wouldn't get anything done! In this lab, we'll take a look at how we can use functions as first-class objects to pass them around, store them in variables and data structures, and return them from other functions.

For this lab, you will continue working on functions for our scooter-taxi client, Scuber.

Instructions

Fork and clone this lab into your local environment. Navigate into its directory in the terminal, then run `code .` to open the files in Visual Studio Code.

Using the tests to guide you, create the following deliverables:

- `returnFirstTwoDrivers()` — Declare the variable `returnFirstTwoDrivers` with `const` and assign an anonymous function to it. The assigned function should take one argument, an array containing the names of Scuber's drivers, and return a new array containing the **first** two drivers in the array:

```
returnFirstTwoDrivers(['Antonia', 'Nuru', 'Amari', 'Mo'])
// => ['Antonia', 'Nuru']
```

- `returnLastTwoDrivers()` — Declare a variable with `const` that is assigned an anonymous function. The assigned function should take one argument, an array containing the names of Scuber's drivers, and return a new array containing the **last** two drivers in the array:

```
returnLastTwoDrivers(['Antonia', 'Nuru', 'Amari', 'Mo'])
// => ['Amari', 'Mo']
```

- `selectingDrivers` — This is an array containing two elements: the two functions that we previously defined (`returnFirstTwoDrivers()` and `returnLastTwoDrivers()`).
- `createFareMultiplier()` — This is a higher-order function that takes in one argument, an integer, and returns a function that will multiply a fare for a ride accordingly. For example, if `createFareMultiplier()` receives an argument of `4` , it will return a function that takes in a fare as an argument and quadruples the fare.
- `fareDoubler()` — Declare a variable with `const` and assign a function returned by `createFareMultiplier()` to it. Invoke `createFareMultiplier()` in such a way that the new `fareDoubler()` function accepts a fare as its lone argument and doubles it.
- `fareTripler()` — Declare a variable with `const` and assign a function returned by `createFareMultiplier()` to it. Invoke `createFareMultiplier()` in such a way that the new `fareTripler()` function accepts a fare as its lone argument and triples it.
- `selectDifferentDrivers()` — This function takes two arguments, an array of the names of Scuber's `drivers` and either the `returnFirstTwoDrivers()` or `returnLastTwoDrivers()` function. Based on these two arguments, `selectDifferentDrivers()` will return either the first two drivers or the last two drivers.

When you're done, remember to commit and push your changes up to GitHub, then submit your work to Canvas using CodeGrade.

Resources

- **Wikipedia: First-class function (https://en.wikipedia.org/wiki/First-class_function)**
- **MDN: Functions (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions)**

- **TypeOfNaN: What is a Higher-Order Function? (https://typeofnan.dev/what-is-a-higher-order-function/)**