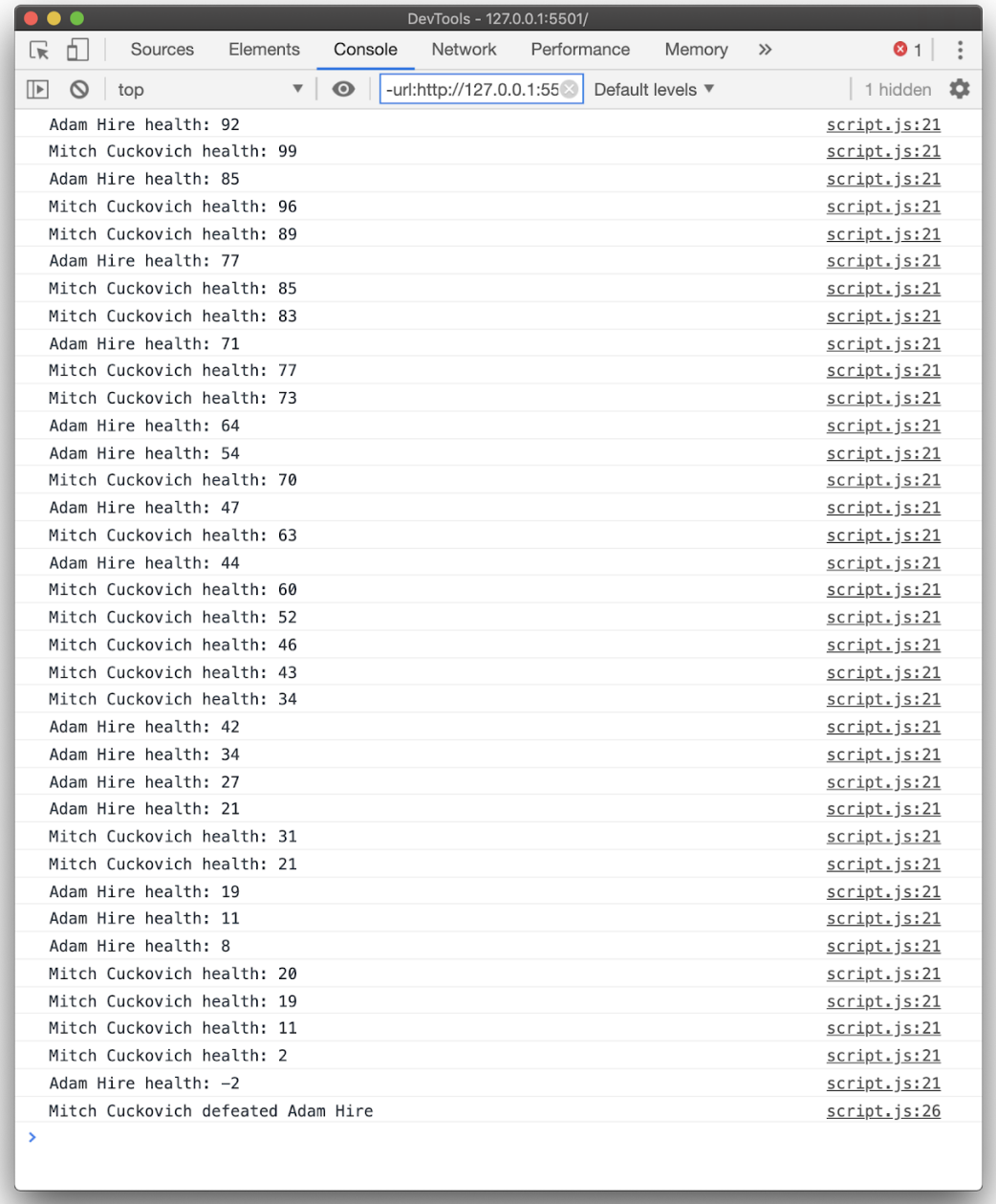
JAVASCRIPT LAB 2 - GAME

Task: This lab will focus on three ways of writing out functions: function declaration, function expression, and arrow functions. The goal is to properly log statements to the console by using a mixture of the aforementioned topics. While this lab explicitly asks you to use certain functions, it is worth mentioning that each example **could** be written using any of the three methods for defining functions. You will only need to construct an `index.html`, `app.js` and `index.js` file. Final output example:



Build Specifications:

- Upload a **ZIP file** to the LMS with the functions located in a file called *app.js* and the calls to those functions in *index.js*.
- Declare an arrow function named `randomDamage` that has no parameters and returns a random integer between 1 and 10.
- Declare an arrow function named `chooseOption` that has two parameters named `opt1` and `opt2`. `chooseOption` does two things:
    - Declares and initializes a variable named `randNum` to either a 0 or 1, randomly.
    - Returns `opt1` if randNum is equal to 0 otherwise return `opt2` . (Use a ternary operator)
- Declare a function expression named `attackPlayer` that has one parameter named `health` which returns a number equal to `health` minus the product of the `randomDamage` function.
- Declare an arrow function named `logHealth` that has two parameters named `player` and `health` which has a console.log method to state the following message: "`player` health: `health`".
- Declare an arrow function named `logDeath` that has two parameters named `winner` and `loser` which has a console.log method to state the following message: "`winner` defeated `loser`"
- Declare an arrow function named `isDead` that has one parameter named `health` which returns a boolean value of true or false based on the following condition: `health` <= 0;
- Declare a function declaration named `fight` that has four parameters.
    - Parameters:
        - `player1` - this will hold the name of the first player
        - `player2` - this will hold the name of the second player
        - `player1Health` - this will hold the health of the first player
        - `player2Health` - this will hold the health of the second player
    - Within the `fight` function, write a while loop that loops while true
        - Declare and initialize a variable named `attacker` equal to the `chooseOption` function with `player1` and `player2` as arguments.
        - Has an if statement that is triggered when `attacker` is equal to `player1`.
            - Set `player2Health` equal to the product of `attackPlayer` with `player2Health` as its argument.
            - Calls the `logHealth` function with `player2` and `player2Health` as its arguments.
            - Has an if statement that runs if the product of `isDead`, with `player2Health` as an argument, equates to true. If true:
                - Call the `logDeath` function with `player1` and `player2` as arguments.
                - Break
        - Has an else statement that:

        - Sets `player1Health` equal to the `attackPlayer` function with `player1Health` as its argument.
        - Call the `logHealth` function with `player1` and `player1Health` as its arguments.
        - Has an if statement that runs if the product of `isDead` with `player1Health` as an argument equates to true. If true:
            - Call the `logDeath` function with `player2` and `player1` as arguments.
            - Break

- Lastly, call the `fight` function with the required four parameters. You pick the names and starting health. For example: player1: "Mitch", player2: "Adam", player1Health: 100, player2Health: 100.

Tests: Same as build specifications.