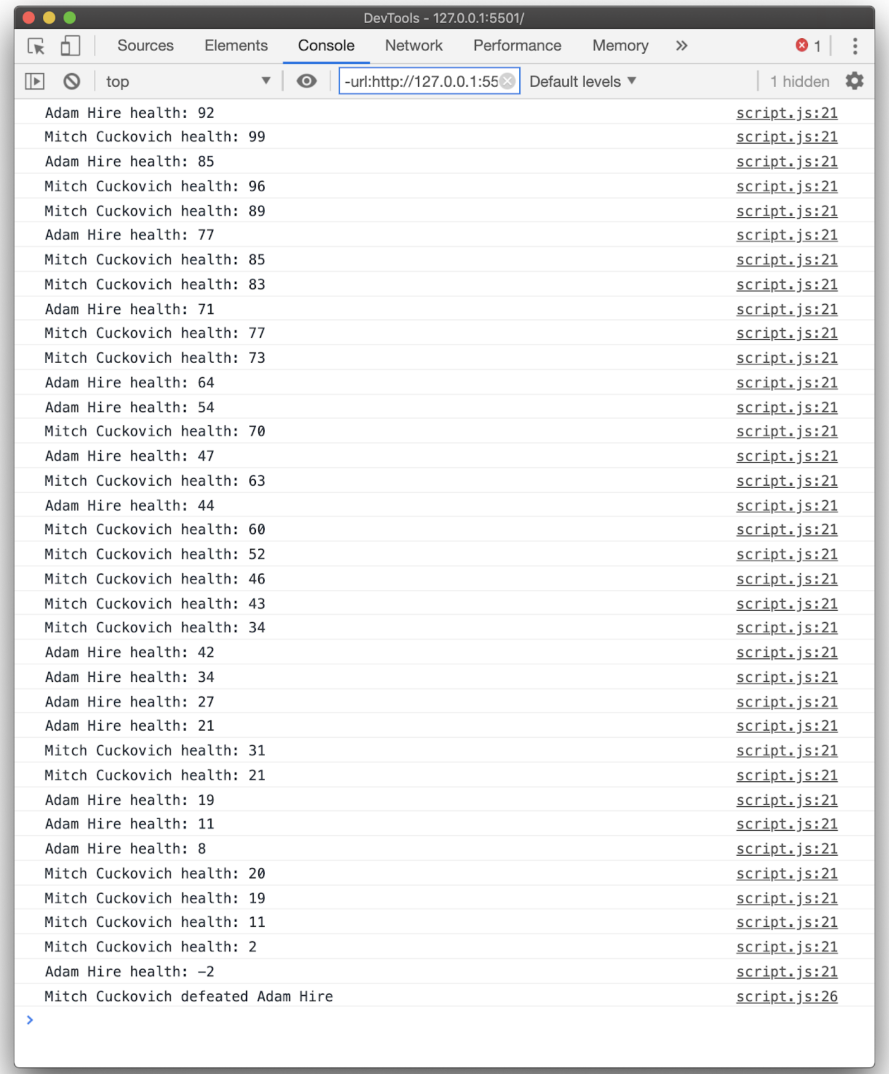
Battle Game

Starter: Use the starter code on GitHub Classroom. The starter includes tests you can run to automatically check some of the requirements. Instructions are in the readme.md file.

Task: 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 the instructions explicitly ask you to use certain functions, it is worth mentioning that each example `could` be written using any of the three methods for defining functions.



```
Adam Hire health: 92                                  script.js:21
Mitch Cuckovich health: 99                            script.js:21
Adam Hire health: 85                                  script.js:21
Mitch Cuckovich health: 96                            script.js:21
Mitch Cuckovich health: 89                            script.js:21
Adam Hire health: 77                                  script.js:21
Mitch Cuckovich health: 85                            script.js:21
Mitch Cuckovich health: 83                            script.js:21
Adam Hire health: 71                                  script.js:21
Mitch Cuckovich health: 77                            script.js:21
Mitch Cuckovich health: 73                            script.js:21
Adam Hire health: 64                                  script.js:21
Adam Hire health: 54                                  script.js:21
Mitch Cuckovich health: 70                            script.js:21
Adam Hire health: 47                                  script.js:21
Mitch Cuckovich health: 63                            script.js:21
Adam Hire health: 44                                  script.js:21
Mitch Cuckovich health: 60                            script.js:21
Mitch Cuckovich health: 52                            script.js:21
Mitch Cuckovich health: 46                            script.js:21
Mitch Cuckovich health: 43                            script.js:21
Mitch Cuckovich health: 34                            script.js:21
Adam Hire health: 42                                  script.js:21
Adam Hire health: 34                                  script.js:21
Adam Hire health: 27                                  script.js:21
Adam Hire health: 21                                  script.js:21
Mitch Cuckovich health: 31                            script.js:21
Mitch Cuckovich health: 21                            script.js:21
Adam Hire health: 19                                  script.js:21
Adam Hire health: 11                                  script.js:21
Adam Hire health: 8                                   script.js:21
Mitch Cuckovich health: 20                            script.js:21
Mitch Cuckovich health: 19                            script.js:21
Mitch Cuckovich health: 11                            script.js:21
Mitch Cuckovich health: 2                             script.js:21
Adam Hire health: -2                                  script.js:21
Mitch Cuckovich defeated Adam Hire                    script.js:26
>
```

Build Specifications:

- 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 result 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.
        - If **`attacker`** is equal to **`player1`**.
            - Set **`player2Health`** equal to the result of **`attackPlayer`** with **`player2Health`** as its argument.
            - Calls the **`logHealth`** function with **`player2`** and **`player2Health`** as its arguments.
            - If the result of **`isDead`** with **`player2Health`** as an argument is 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.
        - If the result of `isDead` with `player1Health` as an argument is 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.

Extended Challenges:
Write some additional functions. Use whatever function style you like. Here are some ideas…

- `getGrade`: This function takes in a number parameter (0 to 100). It returns the corresponding letter grade using the scale: A=90+, B=80+, C=70+, D=60+, F=below 60. Call the function with different numbers and log the results. (Note: there should be no console.log *inside* the function.)
- `prioritize`: This function has two parameters, `urgent` and `important`, both boolean. It returns the priority according to this rule: urgent & important → 1, important not urgent → 2, urgent not important → 3, neither urgent nor important → 4.
- `calculatePay`: This function has two parameters, `wage` and `hours`, both numbers. It calculates and returns a person's weekly pay based on the extended challenge from Life Events