JavaScript – Lab 6: Battle Game (Functions)

Assigned: 10/19/22

Due: 10/24/22

Completed: 10/21/22

<https://docs.google.com/document/d/1IZ_xKVrmKeKvqMqhsKCib48lfwNHXuqtIhtR442fJOw/preview>

**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.

**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.

**JavaScript:**

let randomDamage = () => {

    return Math.floor(Math.random() \* 10) + 1

};

let chooseOption = (opt1, opt2) => {

    let randNum = Math.random();

    return randNum < 0.5 ? opt1 : opt2;

};

let attackPlayer = (health) => {

    return health - randomDamage();

}

let logHealth = (player, health) => {

    console.log(`${player} health: ${health}`);

    return;

};

let logDeath = (winner, loser) => {

    console.log(`${winner} defeated ${loser}`);

    return;

};

let isDead = (health) => {

    return health <= 0 ? true : false;

};

function fight(player1, player2, player1Health, player2Health) {

    while (true) {

        let attacker = chooseOption(player1, player2);

        if (attacker === player1) {

            player2Health = attackPlayer(player2Health);

            logHealth(player2, player2Health)

            if (isDead(player2Health)) {

                logDeath(player1, player2);

                break;

            }

        } else {

            player1Health = attackPlayer(player1Health);

            logHealth(player1, player1Health)

            if (isDead(player1Health)) {

                logDeath(player2, player1);

                break;

            }

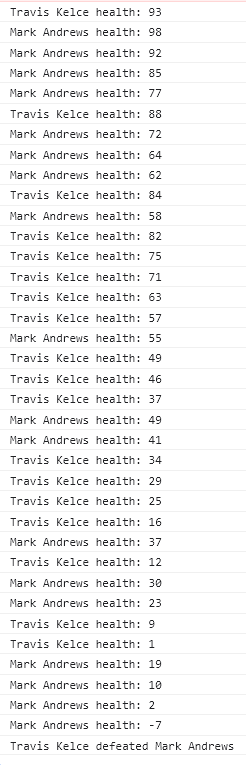
        }

    }

}

fight("Travis Kelce", "Mark Andrews", 100, 100);

**Fight #1: Travis Kelce wins!**



**Fight #2: Mark Andrews wins!**

Table

Description automatically generated

**Testing with Jasmine (HTML):**

<!DOCTYPE html>

<html>

<head>

  <meta charset="utf-8">

  <title>Jasmine Spec Runner v3.5.0</title>

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/jasmine/3.5.0/jasmine.css">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/jasmine/3.5.0/jasmine.min.js"></script>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/jasmine/3.5.0/jasmine-html.min.js"></script>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/jasmine/3.5.0/boot.min.js"></script>

  <script>

    jasmine.getEnv().configure({random: false});

  </script>

  <!-- include source files here... -->

  <script src="../src/script.js"></script>

  <!-- include spec files here... -->

  <script src="./lab2.spec.js"></script>

</head>

<body>

</body>

</html>

Text

Description automatically generated

**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](https://docs.google.com/document/d/1NBCxNmPD42O2U7_lcKLOCT9WOWRSInH91TO3kHrjOHg/preview)