Knights of JavaScript

Objective

Create a fully functional browser-based turn-based game, in which players will battle against each other, defeat each other, and ultimately come out victorious.

List of Criteria

Here is a list of the final criteria what you as the developer will need to fix / features you will need to implement:

- When changing the styling, use only **classList()** methods: https://developer.mozilla.org/en-US/docs/Web/API/Element/classList
  - The classes you create should be found in your CSS code.
- Create a function that tracks whose turn it is, and have that reflect on the 'Attack' button.
  - The button of the player whose turn it is should be grey, and the button of the player being attacked should be red (or not grey). The buttons should then switch in appearance at the end of each player's turn.
- Create a visual effect when each player is attacking / being attacked (see **attackPlayerTwo()** for reference)
- When a player's health has reached 0 or lower, ensure that their health remains 0, and create a visual indicating that the game is over, as well as which player won
- Add a background image to **<body>**
- Comment on each line of code with an explanation in your own words what is happening at that current line.
  - You are already provided with comments for the starter code and do not need to replace any of those comments. Instead, only add comments for your new code.

How You Will Be Graded

- All of the above criteria has been met
- No bugs / errors
- Smooth gameplay when the players attack, their health reduces, and they switch turns
- UI is seamless and effectively communicates to the player what is the present state of the game.

Extra credit

These are features that will help create a FULLY realized gameplay experience. These features will not replace any of the above criteria, as in all of the above criteria needs to be met first, before extra credit can be awarded. It can, however, increase your fully passing score and work as a portfolio piece should you complete them.

- Change the game font
- Animate player 2
- Add a sound effect to both players when a player is attacking / taking damage
- Add background music
- Create a restart feature at the end of the game to reset players back to full health

Instructions

You will be given a template project with the basic mechanics needed for Player 1 to function correctly. You will be expected to then take the starter code and replicate the same logic for Player 2, ensuring there are no bugs or errors in your browser console when the user plays the game. You will be using HTML, CSS, and JavaScript for this to work. This game should be fully-functional, bug / error free, and an engaging experience for users.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Document</title>
</head>
<body>
    <h2 id="title">Fight!</h2>
    <div id="gameOverScreen">
        <h2>GAME OVER!</h2>
        <h3 id="winningPlayer"></h3>
    </div>
    <div id="playerTurn">
        <h2><span id="playerName">Player 1</span>'s turn!</h2>
    </div>
    <div id="fightingSpace">
        <div class="player" id='playerOne'>
            <h2>Player 1</h2>
            <div class="healthSection">
                <span>Health: </span><span id="playerOneHealth">100</span>
            </div>
            <div class="playerSpace">
                <img id="playerOneSprite" class="idle" src="./images/R_Idle.png" alt="">
                <span>Image by <a
                        href="https://www.freepik.com/free-vector/set-warrior-different-postures_1354461.htm#query=robot%20sprite&position=
                        Freepik</a></span>
                <button id="playerOneAttack" class="inactive" onclick="attackPlayerOne()">Attack!</button>
            </div>
        </div>
        <div class="player" id='playerTwo'>
            <h2>Player 2</h2>
            <div class="healthSection">
                <span>Health: </span><span id="playerTwoHealth"> 100</span>
            </div>
            <div class="playerSpace">
                <img id="playerTwoSprite" class="idle" src="./images/L_Idle.png" alt="">
                <span>Image by <a
                        href="https://www.freepik.com/free-vector/set-warrior-different-postures_1354461.htm#query=robot%20sprite&position=
                        Freepik</a></span>
                <button id="playerTwoAttack" class="active" onclick="attackPlayerTwo()">Attack!</button>
            </div>
        </div>
        <div><audio id="SFX_PlayerDamage" src="audio/SFX_PlayerDamage.wav"></audio></div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

script.js

```
// these values are set at the beginning
// and then used throughout the game
let gameState = {
    players: 2,
    whoseTurn: 1,
    gameOver: false
}


// function that considers which player's turn it is and then
// changes the UI accordingly
function changePlayer() {
    // if the current player is player 1 at the end of a move
    if (gameState.whoseTurn === 1) {
        let playerTwoHealth = document.getElementById("playerTwoHealth");
        // conversts the innerHTML from string to a number and stores it in a variable
```

```
        let playerTwoHealthNum = Number(playerTwoHealth.innerHTML);
        // reduces by 10
        playerTwoHealthNum -= 10;
        // resets the HTML to the new value
        playerTwoHealth.innerHTML = playerTwoHealthNum;

        // checks if the player has reached 0 health
        if (playerTwoHealthNum <= 0) {
            // ensures health does not dig into the negative
            playerTwoHealth = 0;
            // ends the game
            gameOver();
        }
        else {
            // switch to the next player and change the UI's display / behavior
            gameState.whoseTurn = 2;

            // grabs the 'playerName' element and changes the player's turn display
            let playerName = document.getElementById("playerName");
            playerName.innerHTML = `Player ${gameState.whoseTurn}`;
        }
    }
}

// if a player's health reaches 0 at the end of a turn, the game ends
// and the winner is announced
function gameOver() {
    let title = document.getElementById("title");
    title.style = "display: none;";
    let playerTurnDisplay = document.getElementById("playerTurn");
    playerTurnDisplay.style = "display: none;";

    let winningPlayer = document.getElementById("winningPlayer");
    winningPlayer.innerHTML = `Player ${gameState.whoseTurn} wins!`

    let gameOverScreen = document.getElementById("gameOverScreen");
    gameOverScreen.style = "display: flex; flex-direction: column;";
}

// function that allows the player two attack button to reduce the player two's
// health
function attackPlayerTwo() {
    // compartmentalized function that will switch the player 2 attack button to inactive
    // and player 1 attack button to active using DOM manipulation
    // this also DISABLES the button, meaning they are not interactable
    function changeButtonStatus() {
        let playerTwoAttackButton = document.getElementById("playerTwoAttack");
        playerTwoAttackButton.disabled = true;
        playerTwoAttackButton.classList.add("inactive");
        playerTwoAttackButton.classList.remove("active");

        let playerOneAttackButton = document.getElementById("playerOneAttack");
        playerOneAttackButton.disabled = false;
        playerOneAttackButton.classList.add("active");
        playerOneAttackButton.classList.remove("inactive");
    }

    // commpartmentalized function that changes the player 1's sprite using the array
    // containing multiple images
    function animatePlayer() {
        // an array containing the images using in player one's animation
        // the indices are later used to cycle / "animate" when the player attacks
        let playerOneFrames = [
            "./images/R_Idle.png",
            "./images/R_Attack.png"
        ];

        let playerSprite = document.getElementById("playerOneSprite");
        // function we will call in setTimeout, before the frames change back
        // the idle stance
        // in other words, we set to the attack sprite, wait 3 seconds,
        // then set it back to the idle sprite
        playerSprite.src = playerOneFrames[1];

        // removes the 'idle' class from the player sprite
        playerSprite.classList.remove("idle");
        // adds the 'attack' class to the player sprite
        // ** CHECK THE CSS TO NOTE THE CHANGES MADE **
```

```
        playerSprite.classList.add("attack");

        // grabs the enemy sprite
        let enemySprite = document.getElementById("playerTwoSprite");
        let enemyDamage = document.getElementById("SFX_PlayerDamage");
        // removes the 'idle' class from the enemy sprite
        enemySprite.classList.remove("idle");
        // adds the 'attack' class to the enemy sprite
        // ** CHECK THE CSS TO NOTE THE CHANGES MADE **
        enemySprite.classList.add("damage");
        // sound that plays when enemy takes damage
        enemyDamage.play();

        // the function we will call in the setTimeOut method below
        // after 350 milliseconds
        // this function will execute this block of code
        function changePlayerOneSprite() {
            enemySprite.classList.remove("damage");
            enemySprite.classList.add("idle");

            playerSprite.src = playerOneFrames[0];
            playerSprite.classList.remove("attack");
            playerSprite.classList.add("idle");
        }

        setTimeout(changePlayerOneSprite, 350);
    }

    // for easy reading,
    // we do not include ALL of the above code within this condition
    // instead, we create higher-order functions to keep the code neat and readable
    if (gameState.whoseTurn === 1) {
        animatePlayer();
        changeButtonStatus();
        changePlayer();
    }
}

function attackPlayerOne() {
    if (gameState.whoseTurn === 2) {
        let playerOneHealth = document.getElementById("playerOneHealth");
        let playerOneHealthNum = Number(playerOneHealth.innerHTML);
        playerOneHealthNum -= 10;
        playerOneHealth.innerHTML = playerOneHealthNum;

        if (playerOneHealth <= 0) {
            playerOneHealth = 0;
            gameOver();
        } else {
            changePlayer();
        }
    }
}
```

- In `changePlayer()` refactor the condition to check if it's player 2's turn
  - Using the same logic when it's player 1's turn, create JavaScript logic that will
  1. Generate random damage amount
  2. Check player 2's health (if less than 0, end the game, and if greater than 0, reduce the overall health by the random damage amount
- Create a function called `attackPlayerOne()` after `attackPlayerTwo()`
  - Review the logic in `attackPlayerTwo()`. `attackPlayerOne()` should be accomplishing the same, but while dealing damage to player 1 and having the UI reflect that
  - This will include:
    1. Player 2's attack button should be enabled / active
    2. Player 1's attack button should be disabled / inactive

3. Player 2's animation frames should be the correct files (Idle and Attack)
4. When Player 2 attacks, the player sprite should change from "idle" to "playerTwoAttack"
5. When the enemy is attacked / taking damage, play the `SFX_EnemyDamage`
6. The enemy sprite's status should change from "idle" to `playerOneDamage`
7. Create a function called `changePlayerTwoSprite`
    a. The enemy sprite should return to "idle" and remove "playerOneDamage"
    b. The playerSprite should also return to "idle" and remove "playerTwoAttack"
8. At the end check if it's still player 2's turn, and if it is:
    a. Animate the player
    b. Change the button status
    c. Change the players' turns

style.css

Insert this code in your `style.css`

```
body {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}

#gameOverScreen {
    display: none;
}

#fightingSpace {
    display: flex;
    flex-direction: row;
}

.player {
    display: flex;
    flex-direction: column;
    text-align: center;
    justify-content: center;
    align-items: center;
}

.healthSection {
    display: flex;
    flex-direction: row;
    justify-content: center;
    align-items: center;
    font-size: 24px;
    font-weight: 600;
}

.inactive {
    background-color: grey;
}

.active {
    background-color: red;
}

.idle {
    height: 350px;
    width: 350px;
    background-color: none;
}
```

```
.attack {
    height: 350px;
    width: 360px;
    background: linear-gradient(-90deg, rgba(0,255,0, 0.7), rgba(255,0,0,0) 90.71%); border-top-right-radius: 80%; border-bottom-right-radi
}

.damage {
    height: 350px;
    width: 350px;
    background: linear-gradient(-90deg, rgba(255,0,0,0), rgba(255,0,0,.3) 70.71%);
}

#playerOneHealth,
#playerTwoHealth {
    color: green;
}

.playerSpace {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    gap: 20px;
}

button {
    height: 75px;
    width: 125px;
    background-color: red;
    border: solid 5px grey;
    border-radius: 0.5rem;
    color: white;
    font-weight: 600;
    font-size: 24px;
}
```