Single player Dice game

Html code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="./style.css">
</head>
<body>
    <!-- The main container for the dice rolling game -->
    <div class="container">
        <!-- A div to hold the dice images -->
        <div class="dice-wrapper">
            <img id="die-1" src="images/1.png" alt="die-1">
            <img id="die-2" src="images/2.png" alt="die-2">
        </div>
        <!-- A paragraph to display the total -->
        <p id="total">text here</p>
        <!-- A button that triggers the roll() function when clicked -->
        <button onclick="roll()">ROLL DICE</button>
    </div>

    <!-- Including the JavaScript file -->
    <script src="./script.js"></script>
</body>
</html>
```

Explanation:

1. `<!DOCTYPE html>`: This declaration specifies the document type and version of HTML being used.

2. `<html lang="en">`: The opening tag for the HTML document. The `lang` attribute specifies the language of the content.

3. `<head>`: This section contains metadata about the document, such as character encoding and the document's title.

4. `<meta charset="UTF-8">`: Specifies the character encoding of the document as UTF-8, which is a widely used character encoding for supporting various languages and characters.

5. `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Sets the initial scale and width of the viewport, which is important for responsive design on different devices.

6. `<title>Document</title>`: Sets the title of the document, which is displayed in the browser's title bar or tab.

7. `<link rel="stylesheet" href="./style.css">`: Links an external CSS stylesheet for styling the content.

8. `<body>`: The main content of the web page is contained within the `<body>` tag.

9. `<div class="container">`: A container for the entire game content.

10. `<div class="dice-wrapper">`: A container for the dice images.

11. `<img id="die-1" src="images/1.png" alt="die-1">`: Displays the first dice image. The `id` attribute provides a unique identifier for JavaScript manipulation, and the `src` attribute specifies the image source.

12. `<img id="die-2" src="images/2.png" alt="die-2">`: Displays the second dice image.

13. `<p id="total">text here</p>`: A paragraph element to display the total or result of the dice roll. The initial content is "text here."

14. `<button onclick="roll()">ROLL DICE</button>`: A button that, when clicked, triggers the `roll()` function. The `onclick` attribute specifies the JavaScript function to execute.

15. `<script src="./script.js"></script>`: Links an external JavaScript file for scripting functionality.

Overall, this HTML code sets up the structure of a simple dice rolling game interface, including dice images, a total display, and a button to roll the dice. The JavaScript function `roll()` (defined in the linked `script.js` file) is called when the "ROLL DICE" button is clicked.

Javascript code:

```
// An array of dice image filenames
let images = [
    "1.png",
    "2.png",
    "3.png",
    "4.png",
    "5.png",
    "6.png",
];

// Select all the img elements on the page (the dice images)
let dice = document.querySelectorAll("img");

// Function that simulates rolling the dice
function roll() {
    // Add a "shake" class to each dice image to create a shaking animation
    dice.forEach(function(die){
        die.classList.add("shake");
    });

    // After a delay of 1000ms (1 second), execute the following code
    setTimeout(function(){
        // Remove the "shake" class from each dice image to stop the shaking
animation
        dice.forEach(function(die){
            die.classList.remove("shake");
        });

        // Generate random values for the two dice
        let dieOneValue = Math.floor(Math.random() * 6);
        let dieTwoValue = Math.floor(Math.random() * 6);

        // Display the random values in the console
        console.log(dieOneValue, dieTwoValue);

        // Update the src attribute of the dice images to show the rolled
values
        document.querySelector("#die-1").setAttribute("src", "images/" +
images[dieOneValue]);
        document.querySelector("#die-2").setAttribute("src", "images/" +
images[dieTwoValue]);

        // Calculate the total of the two dice values and display it in the
"total" paragraph
        document.querySelector("#total").innerHTML = "Your roll is " +
((dieOneValue + 1) + (dieTwoValue + 1));
    },
```

```
    1000 // Delay of 1000ms (1 second)
    );
}

// Initial roll when the script is first loaded
roll();
```

Explanation:

1. The `images` array contains the filenames of the dice images ("1.png" to "6.png").

2. `document.querySelectorAll("img");` selects all `<img>` elements on the page and stores them in the `dice` variable.

3. The `roll()` function simulates rolling the dice:

- It adds the "shake" class to each dice image, creating a shaking animation effect.

- After a 1-second delay (`setTimeout`), it removes the "shake" class to stop the shaking animation.

- It generates random values (`dieOneValue` and `dieTwoValue`) for the two dice using `Math.random()` and `Math.floor()` functions.

- The random values are displayed in the console.

- The `src` attributes of the dice images are updated to show the rolled values.

- The sum of the two dice values is calculated and displayed in the "total" paragraph.

4. The last line `roll();` calls the `roll()` function initially when the script is first loaded, so the dice are rolled and values are displayed immediately.

Overall, this JavaScript code controls the behavior of the dice rolling game. It handles the animation, generates random values, updates the UI, and calculates the total value of the rolled dice.

Style.css code:

```css
/* Set the background color of the body to yellow and make it cover the entire viewport height */
body {
    background-color: yellow;
    height: 100vh;
}

/* Apply default styling to all elements */
* {
    padding: 0;
    margin: 0;
    box-sizing: border-box;
}

/* Style for the main container of the game */
.container {
    width: 400px;
    padding: 50px;
    position: absolute;
    /* Center the container both horizontally and vertically */
    transform: translate(-50%, -50%);
    top: 50%;
    left: 50%;
    background-color: black;
    /* Add shadow to create a sense of elevation */
    box-shadow: 0 15px 35px;
    border-radius: 8px;
    display: flex;
    justify-content: center;
    flex-direction: column;
    align-items: center;
    /* Set the font family for the text inside the container */
    font-family: cursive;
}

/* Style for the container that holds the dice images */
.dice-wrap {
    width: 90%;
    display: flex;
    /* Distribute space around the dice images */
    justify-content: space-around;
}

/* Style for all images on the page */
img {
    padding: 10px;
    /* Add rounded corners to images */
```

```
    border-radius: 17px;
}

/* Styling for paragraph elements */
p {
    font-size: 16px;
    color: whitesmoke;
    /* Add vertical margin above and below the paragraph */
    margin: 30px 0;
    font-weight: 500;
}

/* Styling for buttons */
button {
    background-color: rgb(161, 23, 23);
    border-radius: 2px;
    border: none;
    outline: none;
    color: whitesmoke;
    width: 150px;
    /* Add spacing between letters */
    letter-spacing: 1px;
    padding: 15px 0;
}

/* Apply animation class to create shaking effect */
.shake {
    animation: shake 0.5s infinite;
}

/* Define keyframes for the shake animation */
@keyframes shake {
    0% {
        transform: rotate(8deg);
    }
    50% {
        transform: rotate(-8deg);
    }
    100% {
        transform: rotate(8deg);
    }
}
```

1. **Background and Viewport Height**:

   - `body`: Sets the background color of the entire webpage to yellow.

   - `height: 100vh;`: Makes the body's height cover the entire viewport height, ensuring the background color fills the screen.

2. **Universal Styling**:

   - `*`: Resets padding, margin, and box-sizing for all elements to ensure a consistent base styling.

3. **Main Container Styling**:

   - `.container`: Styles the main game container with the following properties:

     - `width: 400px;`: Sets the width of the container.

     - `padding: 50px;`: Adds padding around the content inside the container.

     - `position: absolute;`: Positions the container absolutely within its containing element.

     - `transform: translate(-50%, -50%);`: Centers the container both horizontally and vertically using a negative translate transformation.

     - `top: 50%; left: 50%;`: Positions the container at the center of the screen.

     - `background-color: black;`: Sets the background color of the container to black.

     - `box-shadow: 0 15px 35px;`: Adds a shadow to create a sense of elevation.

     - `border-radius: 8px;`: Rounds the corners of the container.

     - `display: flex; justify-content: center; flex-direction: column; align-items: center;`: Uses flexbox to center content vertically and horizontally, and arranges content in a column.

     - `font-family: cursive;`: Sets the font family for the text inside the container.

4. **Dice Images Container Styling**:

   - `.dice-wrap`: Styles the container that holds the dice images with the following properties:

     - `width: 90%;`: Sets the width of the container to 90% of its parent's width.

     - `display: flex; justify-content: space-around;`: Uses flexbox to evenly distribute space around the dice images.

5. **Image Styling**:

- `img`: Styles all images on the page with the following properties:

  - `padding: 10px;`: Adds padding around the images.

  - `border-radius: 17px;`: Rounds the corners of the images.

6. **Paragraph Styling**:

- `p`: Styles paragraph elements with the following properties:

  - `font-size: 16px; color: whitesmoke;`: Sets font size and text color.

  - `margin: 30px 0;`: Adds vertical margin above and below the paragraph.

  - `font-weight: 500;`: Sets the font weight of the text.

7. **Button Styling**:

- `button`: Styles button elements with the following properties:

  - `background-color: rgb(161, 23, 23);`: Sets the background color of the button.

  - `border-radius: 2px; border: none; outline: none;`: Adds rounded corners and removes borders and outlines.

  - `color: whitesmoke;`: Sets the text color of the button.

  - `width: 150px;`: Sets the width of the button.

  - `letter-spacing: 1px;`: Adds spacing between letters.

  - `padding: 15px 0;`: Adds padding to the top and bottom of the button.

8. **Shake Animation Styling**:

- `.shake`: Applies the `shake` animation to create a shaking effect on dice images.

- `@keyframes shake`: Defines the keyframes for the `shake` animation:

  - Defines three keyframes (0%, 50%, 100%) with different degrees of rotation to create the shaking effect.

This CSS code is responsible for styling the dice rolling game interface, including background, container layout, font styles, button appearance, and the animation for shaking the dice images during rolling.