Bouncing Ball Game using Web Technologies

Aim:
Make a Ball bouncing game, with 3 balls and the color of the changing on every tap.

Introduction:
The Ball Bouncing Game project is an interactive and entertaining web-based game developed using HTML, CSS, and JavaScript. The objective of the game is to provide users with an engaging experience by allowing them to interact with bouncing balls on a grass green background. When a ball is clicked, it bounces up to a maximum height of 350 pixels and changes its color midway through the animation.

The project aims to showcase the capabilities of web technologies in creating simple yet enjoyable games. By combining HTML for structure, CSS for visual styling, and JavaScript for interactivity, the game offers a seamless user experience. The intuitive gameplay mechanics of clicking on the balls to make them bounce and the added visual element of color change make the game both easy to understand and visually appealing.

This project can serve as a source of entertainment, a stress-relieving tool, or a way to pass the time. It demonstrates the potential of web technologies in creating interactive experiences and highlights the importance of creativity and user engagement in game development. Whether played on a desktop computer, laptop, or mobile device, the Ball Bouncing Game promises a delightful and immersive experience for users of all ages.

Technologies Used:
The project utilizes the following technologies:

1. **HTML**: HTML, which stands for HyperText Markup Language, is a fundamental technology used for structuring web page elements and providing content. In the context of the ball bouncing game, HTML is responsible for creating the layout and structure of the game components. It defines the different sections, such as the game container, instruction panel, balls, and game description. HTML tags and attributes are used to define the structure and hierarchy of the elements on the web page.

2. **CSS**: CSS, short for Cascading Style Sheets, is used to style the web page elements in terms of appearance, layout, and presentation. In the ball bouncing game project, CSS is utilized to define the visual aspects of the game, including colors, sizes, positions, and animations. CSS rules and selectors are employed to target specific elements and apply styling properties to achieve the desired visual design. For example, CSS is used to set the background color, dimensions, positions, and outline of the game container, balls, and other components. It also defines the animation properties for the bounce effect.

3. **JavaScript**: JavaScript is a programming language used to add interactivity and implement the game logic of the ball bouncing game. In this project, JavaScript is responsible for handling user interactions and controlling the behavior of the balls. Event listeners are used to detect when a ball is clicked, triggering the bounce animation and color change. JavaScript functions are implemented to calculate the ball's position, initiate the animation, and modify the ball's style properties. Additionally, JavaScript is utilized to generate random colors for the balls during the animation, adding an element of unpredictability and visual interest to the game.

Features:
The ball bouncing game offers the following features:

1. **Interactive Gameplay**: Users can click on any of the three balls to make them bounce.

2. **Bounce Animation**: The balls animate in a linear manner, bouncing up to a maximum height of 350 pixels.

3. **Color Change**: During the animation, the color of the ball changes to a random color, adding visual appeal to the game.

4. **Responsive Design**: The game layout adapts to different screen sizes, making it accessible on various devices.

Implementation Details:
The game is implemented using HTML, CSS, and JavaScript. Here is an overview of the code structure and functionality:

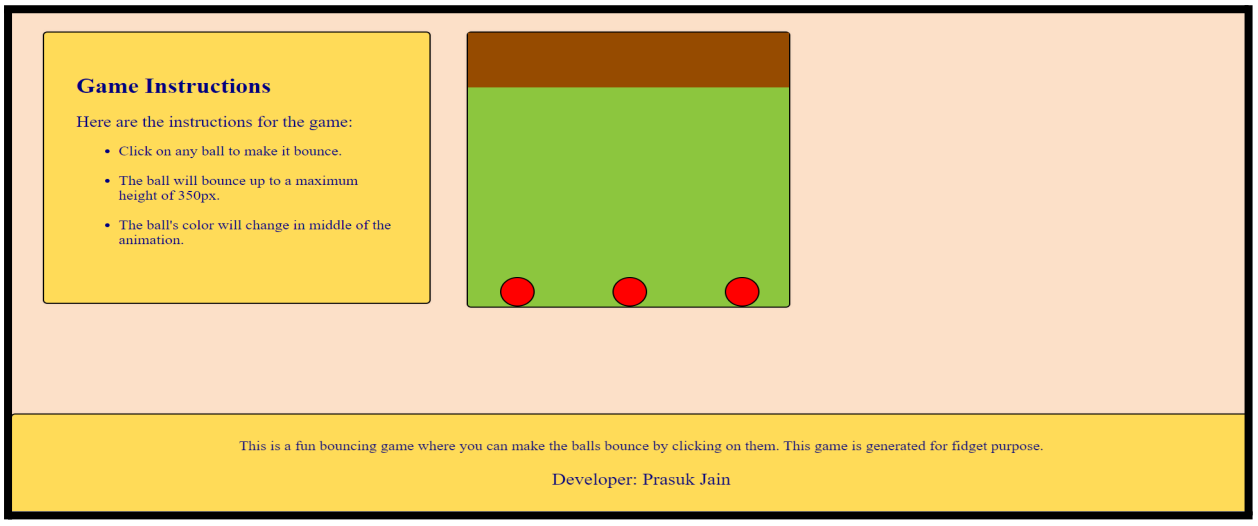
1. **HTML**: The HTML code in the ball bouncing game project defines the structure of the web page. It consists of various div elements that represent different sections and components of the game. These div elements include:
    - **Game container**: The div with the id "game-container" represents the area where the game is displayed. It has a fixed width and height and is positioned relative to the page.
    - **Instruction panel**: The div with the id "instruction-panel" contains the game instructions. It is positioned absolute and appears at the top left corner of the game container.
    - **Balls**: The divs with the ids "ball1", "ball2", and "ball3" represent the three bouncing balls in the game. They are positioned absolutely at the bottom of the game container.

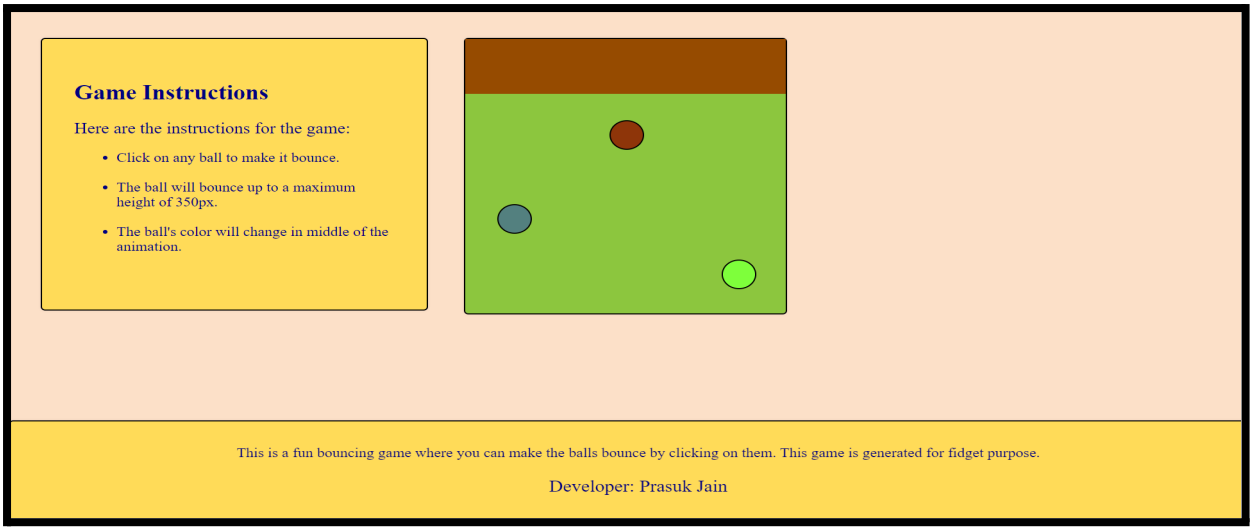
2. **CSS**: The CSS code in the ball bouncing game project defines the styling rules for different elements. It specifies the visual appearance, dimensions, positions, colors, and animations of various components. Some of the CSS rules used in the project include:
    - **Background color**: The background color of the game container and other sections is defined using the "background-color" property.

- **Dimensions**: The width and height of the game container, balls, and other elements are specified using the "width" and "height" properties.
- **Positions**: The positioning of the instruction panel, balls, and other elements is controlled using properties like "position", "top", "left", and "bottom".
- **Colors**: The background color and text color of different elements are set using the "color" and "background-color" properties.
- **Animations**: The bounce animation of the balls is defined using the "@keyframes" rule, specifying the intermediate positions at different animation keyframes.

3. **JavaScript**: The JavaScript code in the ball bouncing game project handles the game logic and user interactions. It uses event listeners to detect clicks on the balls and triggers the bounce animation. The key functionalities implemented in JavaScript include:
    - **Event listeners**: The "click" event listeners are attached to each ball, enabling the detection of user clicks on the balls.
    - **Bounce animation**: When a ball is clicked, the JavaScript code updates the position of the ball using the "style" property, making it bounce up to a maximum height.
    - **Color change**: The JavaScript code generates a random color using the "generateRandomColor()" function and changes the background color of the ball midway through the animation.
    - **Timing**: The "setTimeout()" function is used to control the timing of the animation and color change, ensuring smooth transitions and synchronized effects.

Screenshots of the Game:





HTML code:

```
<!DOCTYPE html>
<html>
<head>
<title>Bouncing game</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<div id="instruction-panel">
   <h1>Game Instructions</h1>
   <p>Here are the instructions for the game:</p>
   <ul>
      <li>Click on any ball to make it bounce.</li>
      <li>The ball will bounce up to a maximum height of 350px.</li>
      <li>The ball's color will change in middle of the animation.</li>
   </ul>
</div>

<div id="game-container">
   <div class="wall"></div> <!-- Wall added -->
   <div id="ball1" class="ball"></div>
   <div id="ball2" class="ball"></div>
   <div id="ball3" class="ball"></div>
</div>
<script src="script.js"></script>
<div id="game-description">
<p style="font-size: 24px;">This is a fun bouncing game where you can make the balls bounce by clicking on them. This game is generated for fidget purpose.</p>
<p style="font-size: 30px;">Developer: Prasuk Jain(96)</p>
</div>
</body>
</html>
```

CSS code:

```
body {
    margin: 0;
    padding: 0;
    background-color: #FCE0C8; /* Skin color */
}
#game-container {
    width: 500px;
    height: 500px;
    position: relative;
    margin: 50px 0px 0px 700px;
    border-radius: 5px;
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
    outline: 2px solid black; /* Add black outline */
    background-color: #8CC63E; /* Grass green */
    display:inline-block;
}
.wall {
    width: 100%;
    height: 100px;
    position: absolute;
    top: 0;
    background-color: #964B00; /* Fox Brown */
}
.ball {
    width: 50px;
    height: 50px;
    border-radius: 50%;
    position: absolute;
    bottom: 0;
    background-color: red;
    animation: none;
    border: 2px solid black; /* Add black outline */
}
#instruction-panel {
    position: absolute;
    top: 50px;
    left: 50px;
    width: 500px;
    padding: 50px;
    background-color: #FFDB58;
    border-radius: 5px;
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
    outline: 2px solid black; /* Add black outline */
    display: inline-block;
}
```

```
#instruction-panel h1 {
    font-size: 38px;
    color: #000080;
    margin-bottom: 10px;
}
#instruction-panel p {
    font-size: 28px;
    color: #000080;
    margin-bottom: 10px;
}
#instruction-panel ul {
    font-size: 24px;
    color: #000080;
    margin-left: 26px;
}
#game-description {
    position: fixed;
    bottom: 0;
    left: 0;
    width: 100%;
    padding: 20px;
    background-color: #FFDB58;
    border-radius: 5px;
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
    outline: 2px solid black; /* Add black outline */
    color: #000080;
    font-size: 24px;
    text-align: center;
}
#ball1 {
    left: 50px;
}
#ball2 {
    left: 225px;
}
#ball3 {
    left: 400px;
}
@keyframes bounce {
    0% {
        bottom: 0;
    }
    50% {
        bottom: 350px;
    }
    100% {
```

```
        bottom: 0;
    }
}
```

JavaScript Code:

```
var balls = document.getElementsByClassName("ball");

for (var i = 0; i < balls.length; i++) {
    balls[i].addEventListener("click", bounceBall);
}

function bounceBall() {
    var ball = this;

    var posY = parseInt(ball.style.bottom) || 0;
    var maxHeight = 350; // Maximum height for the ball to bounce

    if (posY < maxHeight) {
        ball.style.animation = "bounce 2s linear";
        ball.style.bottom = (posY + maxHeight) + "px";
    }

    setTimeout(function() {
        ball.style.animation = "";
        ball.style.bottom = posY + "px";
    }, 2000);

    setTimeout(function() {
        var randomColor = generateRandomColor();
        ball.style.backgroundColor = randomColor;
    }, 1000); /*change ball colour after it mid of animation*/
}

function generateRandomColor() {
    var letters = "0123456789ABCDEF";
    var color = "#";

    for (var i = 0; i < 6; i++) {
        color += letters[Math.floor(Math.random() * 16)];
    }

    return color;
}
```