Project Overview

The *Rock Paper Scissors Game* is a simple yet engaging web-based project that allows a user to play the classic "Rock, Paper, Scissors" game against the computer. It is built using **HTML**, **CSS**, and **JavaScript**, focusing on DOM manipulation, event handling, and basic game logic implementation.

This project demonstrates interactive web development concepts where the browser dynamically updates game outcomes based on user input and computer-generated choices. It highlights the use of core front-end technologies and showcases understanding of **JavaScript logic building** and **UI interactivity**.

Technologies Used

1. **HTML (HyperText Markup Language)** – for creating the structure and layout of the web page.
2. **CSS (Cascading Style Sheets)** – for styling, layout design, and visual presentation.
3. **JavaScript (Vanilla JS)** – for implementing the game logic, handling user interactions, and dynamically updating the game state on the webpage.

User Interface Design

The interface of the game is simple, clean, and user-friendly. It includes:

- A **title section** displaying “Rock Paper Scissors.”
- Three clickable **choice icons** (Rock, Paper, Scissors) each represented by an image.
- A **scoreboard** that tracks both the user’s and computer’s scores.
- A **message container** that provides real-time feedback such as “You Win!”, “You Lost!”, or “Game Draw.”

The design uses consistent colors, rounded images, and hover effects to make the gameplay visually engaging.

Game Functionality

The gameplay follows the traditional *Rock Paper Scissors* rules:

- Rock beats Scissors
- Paper beats Rock
- Scissors beats Paper

When the user clicks on one of the three options, the JavaScript program:

1. **Generates a random choice** for the computer using Math.random().
2. **Compares** the user's selection with the computer's choice.
3. **Determines the outcome** — Win, Lose, or Draw — based on game logic.
4. **Updates the scoreboard** and **displays a message** accordingly.

JavaScript Logic Breakdown

1. Score Tracking

Two variables, userScore and compScore, are initialized to 0. These values update whenever a round ends, depending on whether the user or computer wins.

```
let userScore = 0;

let compScore = 0;
```

2. Computer Choice Generation

The computer randomly selects one of the three choices using Math.random() and Math.floor().

```
const genCompChoice = () => {
  const options = ["rock", "paper", "scissors"];
  const randIdx = Math.floor(Math.random() * 3);
  return options[randIdx];
};
```

3. Game Decision Logic

The core of the program determines whether the round results in a win, loss, or draw:

```
const playGame = (userChoice) => {
  const compChoice = genCompChoice();

  if (userChoice === compChoice) {
    drawGame();
```

```
  } else {
    let userWin = true;
    if (userChoice === "rock") {
      userWin = compChoice === "paper" ? false : true;
    } else if (userChoice === "paper") {
      userWin = compChoice === "scissors" ? false : true;
    } else {
      userWin = compChoice === "rock" ? false : true;
    }
    showWinner(userWin, userChoice, compChoice);
  }
};
```

4. Dynamic Feedback and UI Updates

Messages and background colors change dynamically based on the result:

- Green for a **win**
- Red for a **loss**
- Blue for a **draw**

Example:

```
const showWinner = (userWin, userChoice, compChoice) => {
  if (userWin) {
    userScore++;
    userScorePara.innerText = userScore;
```

```
    msg.innerText = `You win! Your ${userChoice} beats ${compChoice}`;
    msg.style.backgroundColor = "green";
  } else {
    compScore++;
    compScorePara.innerText = compScore;
    msg.innerText = `You lost. ${compChoice} beats your ${userChoice}`;
    msg.style.backgroundColor = "red";
  }
};
```

5. Event Handling

Each choice (Rock, Paper, Scissors) is assigned an event listener that triggers the game when clicked.

```
choices.forEach((choice) => {
  choice.addEventListener("click", () => {
    const userChoice = choice.getAttribute("id");
    playGame(userChoice);
  });
});
```

Features Implemented

- Interactive gameplay with responsive design
- Real-time score updates for both user and computer
- Dynamic messages showing the outcome of each round
- Hover effects and visual feedback on player selection
- Randomized computer choices using JavaScript logic
- Simple, clean, and mobile-friendly UI

Learning Outcomes

Through this project, I gained hands-on experience in:

- Implementing **event-driven programming** using JavaScript
- Practicing **conditional statements** and **logical operators** for decision making
- Using **DOM manipulation** methods to update webpage elements dynamically
- Designing and structuring a responsive web layout using HTML and CSS
- Understanding **randomization techniques** (Math.random())
- Enhancing user experience with **visual feedback and animations**

Future Improvements

To enhance the project further, the following features can be added:

- Adding **sound effects** for win/loss events
- Introducing **difficulty levels** or **time-based challenges**
- Implementing **animations** for transitions and outcomes
- Keeping track of **total matches played** and displaying final results
- Storing the **score history** using browser's Local Storage
- Making the UI more responsive and visually appealing with advanced CSS

Conclusion

The *Rock Paper Scissors Game* project is a perfect example of combining fundamental web technologies to create an interactive and enjoyable application. It demonstrates how core programming concepts like randomization, logic handling, and DOM updates can be applied to build a real-world mini-project.

This project strengthened my understanding of **front-end web development** and provided valuable experience in building interactive user interfaces from scratch.