'use strict';

// Selecting Elements:

// Score Selector:

const score0El = document.querySelector('#score--0');

const score1El = document.getElementById('score--1');

// Player Selection

const player0El = document.querySelector('.player--0');

const player1El = document.querySelector('.player--1');

const current0El = document.getElementById('current--0');

const current1El = document.getElementById('current--1');

let activePlayer = 0;

// Dice Selectors:

const diceEl = document.querySelector('.dice');

const btnNew = document.querySelector('.btn--new');

const btnRoll = document.querySelector('.btn--roll');

const btnHold = document.querySelector('.btn--hold');

// Current Score

let scores = [0, 0];

let currentScore = 0;

let playing = true; // Game state variable

// Resetting to Starting Point

const init = function () {

scores = [0, 0];

currentScore = 0;

activePlayer = 0;

playing = true;

// Reset the UI

score0El.textContent = 0;

score1El.textContent = 0;

current0El.textContent = 0;

current1El.textContent = 0;

diceEl.classList.add('hidden');

player0El.classList.remove('player--winner');

player1El.classList.remove('player--winner');

player0El.classList.add('player--active');

player1El.classList.remove('player--active');

};

// Initialize the game on page load

init();

const switchPlayer = function () {

document.getElementById(`current--${activePlayer}`).textContent = 0;

activePlayer = activePlayer === 0 ? 1 : 0;

currentScore = 0;

player0El.classList.toggle('player--active');

player1El.classList.toggle('player--active');

};

// Rolling Dice Functionality:

btnRoll.addEventListener('click', function () {

if (playing) {

// Only allow action if game is active

// Generate Random Roll

const dice = Math.trunc(Math.random() \* 6) + 1;

// Display Dice

diceEl.classList.remove('hidden');

diceEl.src = `dice-${dice}.png`;

// Check for rolled 1: if true, switch to next player

if (dice !== 1) {

// Add dice to current score

currentScore += dice;

document.getElementById(`current--${activePlayer}`).textContent =

currentScore;

} else {

// Switch to Next Player

switchPlayer();

}

}

});

btnHold.addEventListener('click', function () {

if (playing) {

// Only allow action if game is active

// 1. Add current score to active player's score

scores[activePlayer] += currentScore;

document.getElementById(`score--${activePlayer}`).textContent =

scores[activePlayer];

// 2. Check if player's score >= 20

if (scores[activePlayer] >= 20) {

// Finish the game

playing = false; // Set game state to inactive

diceEl.classList.add('hidden');

document

.querySelector(`.player--${activePlayer}`)

.classList.add('player--winner');

document

.querySelector(`.player--${activePlayer}`)

.classList.remove('player--active');

} else {

// 3. Switch to the next player

switchPlayer();

}

}

});

// New Game

btnNew.addEventListener('click', init);

**1. The Pig Game**

* **Overview**: The Pig Game is a simple turn-based dice game where two players compete to reach a set score (e.g., 100 points) by accumulating points from rolling a dice. Players can roll the dice to add to their current score or hold to add the current score to their total. Rolling a 1 switches the turn to the next player without adding the current score.
* **Game Elements**:
  + **Players**: Two players take turns to play.
  + **Dice**: A six-sided die is used to determine the points for each roll.
  + **Scores**: Each player has a current score for their turn and a total score.
  + **Winning Condition**: The first player to reach or exceed the target score wins the game.

**2. Rolling the Dice**

* **Description**: Rolling the dice generates a random number between 1 and 6, which is added to the player's current score unless a 1 is rolled, in which case the player loses their turn.
  + **Random Roll**:

const dice = Math.trunc(Math.random() \* 6) + 1;

* + - **Explanation**: Math.random() generates a random decimal between 0 and 1. Multiplying by 6 and truncating gives a range from 0 to 5, so adding 1 shifts the range to 1 through 6.
  + **Updating the UI**:

diceEl.classList.remove('hidden');

diceEl.src = `dice-${dice}.png`;

* + - **Explanation**: The diceEl element's classList.remove('hidden') makes the dice image visible. The src attribute is dynamically updated to show the correct dice face based on the rolled number.

**3. Switching the Active Player**

* **Description**: When a player rolls a 1, they lose their current score, and the active player switches. The switchPlayer function handles this logic.
  + **Switching Logic**:

const switchPlayer = function () {

document.getElementById(`current--${activePlayer}`).textContent = 0;

activePlayer = activePlayer === 0 ? 1 : 0;

currentScore = 0;

player0El.classList.toggle('player--active');

player1El.classList.toggle('player--active');

};

* + - **Explanation**:
      * **getElementById**: This method selects the element by its ID and resets the current player's score to 0.
      * **activePlayer**: A ternary operator is used to toggle the active player between 0 and 1.
      * **classList.toggle('player--active')**: This method switches the visual indication of which player is active.

**4. Holding Current Score**

* **Description**: The player can choose to hold their score, which adds their current score to their total score. If the player's total score reaches the winning condition, the game ends.
* **Key Code Elements**:
  + **Holding Logic**:

btnHold.addEventListener('click', function () {

if (playing) {

scores[activePlayer] += currentScore;

document.getElementById(`score--${activePlayer}`).textContent = scores[activePlayer];

if (scores[activePlayer] >= 20) {

playing = false;

diceEl.classList.add('hidden');

document.querySelector(`.player--${activePlayer}`).classList.add('player--winner');

document.querySelector(`.player--${activePlayer}`).classList.remove('player--active');

} else {

switchPlayer();

}

}

});

* + - **Explanation**:
      * **getElementById**: Used to update the total score for the active player.
      * **Ending the Game**: The game state is set to inactive (playing = false) if the player wins, and the UI is updated to reflect this with the player--winner class.
      * **classList.add() and classList.remove()**: These methods are used to add or remove CSS classes to indicate the game’s end and which player has won.

**5. Resetting the Game**

* **Description**: The game can be reset at any time by clicking the "New Game" button, which calls the init function to reset all game variables and the UI.
* **Key Code Elements**:
  + **Reset Function**:

const init = function () {

scores = [0, 0];

currentScore = 0;

activePlayer = 0;

playing = true;

score0El.textContent = 0;

score1El.textContent = 0;

current0El.textContent = 0;

current1El.textContent = 0;

diceEl.classList.add('hidden');

player0El.classList.remove('player--winner');

player1El.classList.remove('player--winner');

player0El.classList.add('player--active');

player1El.classList.remove('player--active');

};

btnNew.addEventListener('click', init);

* + - **Explanation**:
      * **init Function**: Resets all game state variables and updates the UI to reflect a new game.
      * **UI Updates**: All scores are reset to 0, the dice is hidden, and the active player is set to Player 1.