Let's break down the script.js code line-by-line focusing on data flow and logic:

// Initial player credits (250 shown in HTML)

let credits = 250;

// Stores all lottery boards with their numbers and status

let boards = [];

**1. Starting Point (Credits & Boards)**

* We start with 250 credits
* boards array will store our lottery tickets

function updateCredits() {

document.getElementById("credits").innerText = credits;

}

**2. Credit Display Updater**

* Simple function that updates the credit display whenever called
* Used whenever credits change (purchases or winnings)

function purchaseBoards() {

// Get user input for number of boards

const boardCount = parseInt(document.getElementById("boardCount").value);

// Validation checks

if (invalidInput) {

alert("Please enter valid number");

return;

}

// Calculate cost and check funds

const cost = boardCount \* 25;

if (credits < cost) {

alert("Not enough credits!");

return;

}

// Deduct credits and update display

credits -= cost;

updateCredits();

// Clear previous boards and create new ones

const container = document.getElementById("boardsContainer");

container.innerHTML = "";

boards = [];

// Create new boards

for (let i = 0; i < boardCount; i++) {

boards.push({

id: i,

numbers: [], // Will store selected numbers

finalized: false // Can't change after finalizing

});

container.appendChild(createBoardElement(i));

}

// Disable draw button until all boards finalized

document.getElementById("startDrawBtn").disabled = true;

}

**3. Board Purchase Flow**

1. User inputs number of boards
2. System checks valid number and sufficient credits
3. Deducts cost from credits
4. Creates new board objects and their HTML elements
5. Resets draw button state

function createBoardElement(boardId) {

// Create container div

const boardDiv = document.createElement("div");

// Create 20 number buttons (1-20)

const numberGrid = document.createElement("div");

for (let num = 1; num <= 20; num++) {

const numBtn = document.createElement("button");

numBtn.onclick = () => selectNumber(boardId, num, numBtn);

// ...

}

// Create finalize button

const finalizeBtn = document.createElement("button");

finalizeBtn.onclick = () => finalizeBoard(boardId);

// ...

return boardDiv;

}

**4. Board Creation**

* Dynamically creates HTML elements for each board
* Each number button triggers selectNumber()
* Finalize button triggers finalizeBoard()

function selectNumber(boardId, num, button) {

const board = boards.find(b => b.id === boardId);

// Prevent selection if finalized

if (board.finalized) return;

// Toggle number selection

if (number not selected) {

if (less than 6 selected) {

add to board.numbers

highlight button

}

} else {

remove from board.numbers

unhighlight button

}

// Enable finalize button only when 6 numbers selected

document.getElementById(`finalize-${boardId}`).disabled = board.numbers.length !== 6;

}

**5. Number Selection Logic**

* Tracks selected numbers in board.numbers array
* Limits to 6 numbers per board
* Updates button visual state
* Controls finalize button availability

function finalizeBoard(boardId) {

const board = boards.find(b => b.id === boardId);

// Lock the board

board.finalized = true;

// Disable all number buttons

// ...

// If all boards finalized, enable draw button

if (all boards finalized) {

document.getElementById("startDrawBtn").disabled = false;

}

}

**6. Finalizing Boards**

* Prevents further changes to numbers
* Checks if all boards are ready
* Enables the draw button when all set

function startDraw() {

// Generate 6 unique random numbers

const winningNumbers = [];

while (winningNumbers.length < 6) {

const rand = Math.floor(Math.random() \* 20) + 1;

if (!winningNumbers.includes(rand)) {

winningNumbers.push(rand);

}

}

// Display winning numbers

// ...

calculateWinnings(winningNumbers);

}

**7. Draw Execution**

* Generates 6 unique random numbers (1-20)
* Displays them to the user
* Triggers winnings calculation

function calculateWinnings(winningNumbers) {

// Winnings table: index = matches, value = prize

const winningsTable = [0, 0, 10, 50, 200, 500, 1000];

boards.forEach((board, index) => {

// Count matching numbers

const matches = board.numbers.filter(num =>

winningNumbers.includes(num)

).length;

// Award credits based on matches

credits += winningsTable[matches];

// Display results

// ...

});

updateCredits();

}

**8. Winnings Calculation**

* Compares each board's numbers with winning numbers
* Uses predefined prize table (0 for 0-1 matches, 10 for 2, etc.)
* Updates total credits
* Shows results for each board

**Key Data Flow:**

1. Credits start at 250
2. Purchasing boards reduces credits
3. Selected numbers stored in board objects
4. Draw creates winning numbers
5. Comparison calculates winnings
6. Credits updated with winnings

**Important Logic:**

* Cannot spend more credits than available
* Exactly 6 numbers per board required
* Numbers must be unique in winning draw
* Visual feedback for selections and finalization
* Progressive enabling of features (finalize -> draw)