Day 3: Loops & Iteration

لوپس اور دہرانا

> **Quote of the Day:** *"The difference between try and triumph is just a little umph!"* - Mariam Nafees, Pakistani Software Engineer at Google

---

📋 Today's Learning Goals (آج کے اہداف)

By the end of today, you will:

- [ ] Understand WHY we need loops and when to use each type
- [ ] Write `for` loops to repeat actions a specific number of times
- [ ] Use `while` and `do-while` loops for condition-based repetition
- [ ] Control loop flow with `break` and `continue`
- [ ] Debug and avoid infinite loops
- [ ] Build a Cricket Score Counter that tracks runs and wickets

Time Breakdown (کل وقت: 150 منٹ)

- 🕐 7:00-7:05 PM (5min): Daily standup & yesterday's review
- 🕐 7:05-8:05 PM (60min): Concept learning (3× Pomodoro)
- 🕐 8:05-8:50 PM (45min): Hands-on practice
- 🕐 8:50-9:25 PM (35min): Challenge project
- 🕐 9:25-9:30 PM (5min): Quiz & reflection

🎯 What We're Building Today

Cricket Score Counter - A live scoring system that:

- Simulates a T20 innings (6 overs)
- Tracks runs, boundaries, and wickets
- Generates ball-by-ball commentary
- Calculates strike rate automatically

Why This Matters for Your Career: Loops are the foundation of ALL programming. From showing Facebook posts in your feed to processing 10,000 customer orders at Daraz, loops make repetition automatic. Every internship coding test will have loop questions. Master this, and you'll be ahead of 60% of applicants!

🧠 سمجھنا (Understanding): What Are Loops?

The Real-World Analogy

Imagine you're at a biryani stall and the cook needs to serve 50 customers:

❌ Without Loops (Manual Way):

```
Serve customer 1
Serve customer 2
Serve customer 3
...write this 50 times!
```

✅ With Loops (Smart Way):

```
While (customers are waiting):
    Serve next customer
```

Or during Ramadan, counting Tasbih:

You don't say "Subhan Allah" and write it 33 times manually. You:

1. Start counting from 1
2. Say "Subhan Allah"
3. Move to next count
4. Repeat until you reach 33
5. Stop

This is EXACTLY what a loop does in code!

Why Does This Matter?

In real apps:

- **JazzCash:** Loops through your transaction history (100s of entries)
- **Careem:** Checks available drivers nearby (loop through GPS coordinates)
- **Foodpanda:** Displays all restaurants (loop through database)
- **Your Cricket App:** Updates score every ball (loop through 120 balls in T20)

Without loops, you'd have to write the same code hundreds or thousands of times!

The Mental Model

Think of loops as **automation in code**:

```
🔄 LOOP = Repeat + Condition + Action

1. START: Where do I begin?
2. CONDITION: When should I stop?
3. ACTION: What do I do each time?
4. UPDATE: How do I move forward?
5. STOP: When condition becomes false
```

Three Types of Loops (like three ways to wait in line):

1. `for` **loop** = "I know EXACTLY how many times" (Like knowing there are 6 balls in an over)
2. `while` **loop** = "Keep going UNTIL something happens" (Like waiting until your number is called)
3. `do-while` **loop** = "Do it once, THEN check if continue" (Like trying biryani once, then deciding if you want more)

📚 Building Block #1: The `for` Loop

What is a `for` Loop? (کیا ہے؟)

A `for` loop repeats code a **specific number of times** when you know in advance how many iterations you need.

Urdu Concept: جب آپ کو پتہ ہو کہ کتنی بار دہرانا ہے

Use When:

- Processing array elements (we'll learn arrays soon!)
- Counting from X to Y
- Repeating actions a fixed number of times

The Anatomy of a `for` Loop

```javascript
for (initialization; condition; update) {
    // Action to repeat
}
```

Let's break this down like assembling a motorcycle:

javascript

```javascript
for (let i = 0; i < 5; i++) {
    console.log("Round number: " + i);
}

// Let's decode each part:
// let i = 0      → START: Begin counting at 0 (initialization)
// i < 5          → CONDITION: Keep going while i is less than 5
// i++            → UPDATE: After each round, add 1 to i (i = i + 1)
// {...}          → ACTION: What to do each round
```

Flow in Urdu:

1. شروع کرو i = 0 سے (Start from 0)
2. چیک کرو: کیا i < 5؟ (Check: Is i less than 5?)
3. اگر ہاں، تو اندر کا کام کرو (If yes, do the action inside)
4. i اضافہ کرو 1 میں (Add 1 to i)
5. واپس قدم 2 پر جاؤ (Go back to step 2)

Your First Example

javascript

```javascript
// THINKING: We want to print numbers 1 to 5 (like counting customers)

for (let customer = 1; customer <= 5; customer++) {
    console.log("Serving customer number: " + customer);
}

// OUTPUT:
// Serving customer number: 1
// Serving customer number: 2
// Serving customer number: 3
// Serving customer number: 4
// Serving customer number: 5

// EXPLANATION OF EACH LINE:
// let customer = 1    → Start with customer 1 (not 0, because customers start at 1!)
// customer <= 5       → Continue while customer is 5 or less
// customer++          → After serving, move to next customer (add 1)
```

Visual Execution (دماغ میں کیسے چلتا ہے):

```
Round 1: customer = 1, check 1 <= 5? YES → Print → customer becomes 2
Round 2: customer = 2, check 2 <= 5? YES → Print → customer becomes 3
Round 3: customer = 3, check 3 <= 5? YES → Print → customer becomes 4
Round 4: customer = 4, check 4 <= 5? YES → Print → customer becomes 5
Round 5: customer = 5, check 5 <= 5? YES → Print → customer becomes 6
Round 6: customer = 6, check 6 <= 5? NO → STOP
```

Common Mistakes

❌ **Wrong:** Starting confusion

```javascript
for (let i = 1; i < 5; i++) {  // This runs 4 times (1,2,3,4)
    console.log(i);
}
```

✅ **Right:** Understanding boundaries

```javascript
for (let i = 1; i <= 5; i++) {  // This runs 5 times (1,2,3,4,5)
    console.log(i);
}
```

💡 **Why:** `<` means "less than" (not including 5), `<=` means "less than or equal" (including 5)

❌ **Wrong:** Infinite loop (never stops!)

```javascript
for (let i = 1; i <= 5; i--) {  // i keeps getting smaller!
    console.log(i);  // 1, 0, -1, -2, -3... NEVER STOPS!
}
```

✅ **Right:** Correct direction

```javascript
```

```
for (let i = 1; i <= 5; i++) {  // i increases toward 5
    console.log(i);
}
```

💡 **Why:** Your update (`i++` or `i--`) must move toward your condition. If going UP to 5, use `i++`. If going DOWN to 0, use `i--`.

❌ **Wrong:** Forgetting semicolons

```
javascript

for (let i = 0 i < 5 i++) {  // SYNTAX ERROR!
    console.log(i);
}
```

✅ **Right:** Proper syntax

```
javascript

for (let i = 0; i < 5; i++) {  // Semicolons separate the 3 parts
    console.log(i);
}
```

Check Your Understanding

- ☐ Can you explain to a friend in Urdu what `i++` does? (i میں 1 کا اضافہ)
- ☐ What happens if you change `i < 5` to `i < 10`? (More iterations!)
- ☐ Why would you use `for` instead of writing 5 console.logs? (Cleaner, easier to change)

📚 Building Block #2: Counting Patterns

Going Backwards (الٹا گنتی)

```
javascript
```

```
// Countdown like a rocket launch: 5, 4, 3, 2, 1, Blast off!

for (let count = 5; count >= 1; count--) {
    console.log(count);
}
console.log("🚀 Blast off!");

// THINKING PROCESS:
// Start at 5 (let count = 5)
// Go while count is 1 or more (count >= 1)
// Each time, SUBTRACT 1 (count--)

// TODO: Now you try - count backwards from 10 to 1
for (let i = ___; i >= ___; i___) {
    console.log(i);
}
```

Skipping Numbers (کر چھوڑ کچھ)

```
javascript

// Print only EVEN numbers from 0 to 10 (0, 2, 4, 6, 8, 10)

for (let num = 0; num <= 10; num = num + 2) {  // Jump by 2 each time!
    console.log(num);
}

// OR using modulo (remainder):
for (let num = 0; num <= 10; num++) {
    if (num % 2 === 0) {  // If num divided by 2 has remainder 0
        console.log(num);
    }
}

// TODO: Print only ODD numbers from 1 to 9
for (let num = ___; num <= 9; num = num + ___) {
    console.log(num);
}
```

Nested Loops (لوپ کے اندر لوپ)

Real-world example: Printing a 5×5 grid of stars (like a biscuit pattern)

```javascript
// THINKING: Each row needs 5 stars
// We need 5 rows
// So: Outer loop = rows, Inner loop = stars per row

for (let row = 1; row <= 5; row++) {          // 5 rows
    let line = "";  // Empty string to build each line

    for (let star = 1; star <= 5; star++) {    // 5 stars per row
        line = line + "* ";
    }

    console.log(line);
}

// OUTPUT:
// * * * * *
// * * * * *
// * * * * *
// * * * * *
// * * * * *
```

Check Your Understanding

- [ ] Can you make a 3×3 grid? (Change both numbers to 3)
- [ ] How would you print a triangle pattern? (Inner loop condition depends on row)
- [ ] What's the difference between `i++` and `i = i + 2`? (Jump size)

🧱 Building Block #3: The `while` Loop

What is a `while` Loop? (کیا ہے؟)

A `while` loop repeats code **while a condition is true**. You use it when you DON'T know exactly how many times to repeat.

Urdu Concept: جب تک شرط سچ ہے، دہراتے رہو

Use When:

- Waiting for user input
- Reading a file until end
- Game loops (play until player loses)

- Don't know iteration count in advance

The Anatomy of a `while` Loop

```javascript
while (condition) {
    // Action to repeat
    // Don't forget to UPDATE the condition!
}
```

Your First Example

```javascript
// SCENARIO: Waiting for correct password (like ATM)

let password = "";

while (password !== "1234") {  // Keep asking until correct
    password = prompt("Enter password:");

    if (password !== "1234") {
        console.log("❌ Wrong password! Try again.");
    }
}

console.log("✅ Access granted!");

// THINKING PROCESS:
// 1. Start with wrong password (empty)
// 2. Check: Is password NOT equal to "1234"? If yes, continue
// 3. Ask user for password
// 4. Check if wrong, show error
// 5. Go back to step 2
// 6. When password is "1234", condition becomes false, loop exits
```

Real Example: Counting Unknown Quantities

```javascript
```

```
// Count how many times you can subtract 100 from 1000 (like counting 100-rupee notes)

let money = 1000;
let notes = 0;

while (money >= 100) {  // While we have at least 100 rupees
    money = money - 100;  // Take out one 100-rupee note
    notes = notes + 1;    // Count it
    console.log("Note #" + notes + " taken. Remaining: Rs." + money);
}

console.log("Total notes: " + notes);

// OUTPUT:
// Note #1 taken. Remaining: Rs.900
// Note #2 taken. Remaining: Rs.800
// ...
// Note #10 taken. Remaining: Rs.0
// Total notes: 10
```

Common Mistakes

❌ DANGER: Infinite Loop!

javascript

```
let count = 0;

while (count < 5) {
    console.log(count);
    // FORGOT to update count!!! Loop never stops!
}
```

✅ **Right:** Always update the condition

javascript

```
let count = 0;

while (count < 5) {
    console.log(count);
    count++;  // CRITICAL: This makes the loop eventually stop
}
```

💡 **Why:** If the condition never becomes false, the loop runs forever and crashes your browser!

❌ **Wrong:** Condition never true

```javascript
let num = 10;

while (num > 20) {  // 10 is NOT > 20, so loop never runs!
    console.log(num);
    num++;
}
```

✅ **Right:** Check your initial condition

```javascript
let num = 10;

while (num <= 20) {  // 10 IS <= 20, loop will run
    console.log(num);
    num++;
}
```

Check Your Understanding

- [ ] What's the main difference between `for` and `while`? (for = known count, while = unknown)
- [ ] Why must you update the condition variable inside the loop? (To eventually stop)
- [ ] When would you choose `while` over `for`? (When you don't know how many iterations)

📚 Building Block #4: The `do-while` Loop

What is a `do-while` Loop? (کیا ہے؟)

A `do-while` loop **does the action first**, THEN checks the condition. It ALWAYS runs at least once.

Urdu Concept: پہلے کام کرو، پھر چیک کرو کہ دوبارہ کرنا ہے یا نہیں

Real-world analogy:

- Trying biryani at a new restaurant - you taste it ONCE, then decide if you want more
- Playing one round of a game - you play ONCE, then ask "Play again?"

The Anatomy of a `do-while` Loop

```javascript
do {
    // Action to repeat (runs AT LEAST ONCE)
} while (condition);
```

Your First Example

```javascript
// SCENARIO: Restaurant menu - show menu at least once

let choice = "";

do {
    console.log("=== MENU ===");
    console.log("1. Biryani");
    console.log("2. Karahi");
    console.log("3. Exit");

    choice = prompt("Enter your choice:");

    if (choice === "1") {
        console.log("🍛 Biryani ordered!");
    } else if (choice === "2") {
        console.log("🍗 Karahi ordered!");
    }

} while (choice !== "3");  // Keep showing menu until user chooses Exit

console.log("Thank you! آپ کا شکریہ");

// THINKING: Menu must show AT LEAST ONCE, even if user wants to exit
```

Difference Between `while` and `do-while`

```javascript
```

```
// Example where it matters:

let age = 25;

// Using WHILE (checks first)
while (age < 18) {
    console.log("You are a minor");
    age++;
}
// OUTPUT: Nothing! (25 is not < 18, so never runs)

// Using DO-WHILE (does first, then checks)
do {
    console.log("You are a minor");
    age++;
} while (age < 18);
// OUTPUT: "You are a minor" (runs once, then checks and stops)
```

Check Your Understanding

- [ ] When would you use `do-while` instead of `while`? (When action must happen at least once)
- [ ] What's the semicolon after the condition? (Required syntax for do-while)
- [ ] Can you think of a real app scenario? (Login screen, game menu, ATM interface)

📚 Building Block #5: Loop Control - `break` and `continue`

What Are `break` and `continue`? (کیا ہیں؟)

`break` = Stop the loop immediately and exit (like emergency exit button) `continue` = Skip this iteration and jump to next one (like "next customer please")

Urdu:

- `break` = لوپ سے باہر نکل جاؤ
- `continue` = اگلے راؤنڈ پر جاؤ

Using `break`

```
javascript
```

```
// SCENARIO: Searching for a specific item in Daraz search results

for (let item = 1; item <= 100; item++) {
    console.log("Checking item #" + item);

    if (item === 7) {
        console.log("✅ Found the perfect laptop!");
        break;  // Stop searching, we found it!
    }
}

console.log("Search finished");

// OUTPUT:
// Checking item #1
// Checking item #2
// Checking item #3
// Checking item #4
// Checking item #5
// Checking item #6
// Checking item #7
// ✅ Found the perfect laptop!
// Search finished

// WITHOUT BREAK, it would check all 100 items even after finding it!
```

Using `continue`

javascript

```
// SCENARIO: Printing only valid customer IDs (skip negative ones)

for (let id = -2; id <= 5; id++) {
    if (id < 0) {
        continue;  // Skip negative IDs, go to next iteration
    }

    console.log("Processing customer #" + id);
}

// OUTPUT:
// Processing customer #0
// Processing customer #1
// Processing customer #2
// Processing customer #3
// Processing customer #4
// Processing customer #5

// IDs -2 and -1 were SKIPPED (not processed)
```

Real-World Example: Filter and Process

javascript

```
// SCENARIO: Processing mobile numbers, skip invalid ones

let numbers = ["0300-1234567", "invalid", "0321-9876543", "", "0333-5555555"];

for (let i = 0; i < numbers.length; i++) {
    let number = numbers[i];

    // Skip empty or invalid entries
    if (number === "" || number === "invalid") {
        console.log("⚠️ Skipping invalid entry");
        continue;  // Jump to next iteration
    }

    // If we reach here, number is valid
    console.log("✅ Processing: " + number);

    // Check if it's Jazz (0300)
    if (number.startsWith("0300")) {
        console.log("📱 This is a Jazz number");
        break;  // Found Jazz number, stop processing
    }
}

// OUTPUT:
// ✅ Processing: 0300-1234567
// 📱 This is a Jazz number
```

Visual Comparison

javascript

```
// WITHOUT continue:
for (let i = 1; i <= 5; i++) {
    if (i === 3) {
        // Do nothing for 3
    } else {
        console.log(i);
    }
}
// Output: 1, 2, 4, 5

// WITH continue (cleaner):
for (let i = 1; i <= 5; i++) {
    if (i === 3) {
        continue;  // Skip 3
    }
    console.log(i);
}
// Output: 1, 2, 4, 5 (same result, cleaner code)
```

Check Your Understanding

- [ ] What does `break` do to a loop? (Exits immediately)
- [ ] What does `continue` do to a loop? (Skips to next iteration)
- [ ] When would you use `break`? (Found what you're looking for, error condition)
- [ ] When would you use `continue`? (Skip invalid data, filter items)

💻 Practice Session: Loops عمل در (Loops in Action)

🎯 Practice Goal

By the end of this section, you'll be comfortable writing all three types of loops and controlling their flow.

Exercise 1: Guided Practice - Counting Prayers (ہم ساتھ کریں)

Scenario: You want to count Tasbih after prayer (33 times each)

Starter Code:

```
javascript
```

```
// TODO Step 1: Count "SubhanAllah" 33 times using a for loop
// HINT: Start from 1, go up to 33

for (let count = ___; count <= ___; count___) {
    console.log("SubhanAllah - " + count);
}

// TODO Step 2: Now count "Alhamdulillah" 33 times
// HINT: Similar to above

for (let count = ___; count <= ___; count___) {
    console.log(_____________________);
}

// TODO Step 3: Finally count "Allahu Akbar" 34 times
// HINT: Change the condition

for (let count = ___; count <= ___; count___) {
    console.log(_____________________);
}

console.log("✅ Tasbih completed! 100 times total");
```

Test Your Code:

```
javascript

// Expected output: Should print each phrase the correct number of times
// Total count should be: 33 + 33 + 34 = 100
```

Exercise 2: Pattern Printing (اب آپ)

Problem: Print a multiplication table for any number (like the back of your notebook!)

Requirements:

- [ ] Ask user for a number (use `prompt` or hard-code it)
- [ ] Print table from 1 to 10
- [ ] Format: "5 × 1 = 5"

Thinking Framework:

1. First, think: What number do I want the table for?

2. Then, think: What are the steps? (Loop 1 to 10, multiply, display)
3. Finally, think: How do I test? (Check if 5×10 = 50)

Don't Look Below Until You Try! ↓

Hints (if stuck):

- Stuck on loop structure? Think: `for (let i = 1; i <= 10; i++)`
- Need to multiply? Use the `*` operator
- Want nice formatting? Use: `number + " × " + i + " = " + (number * i)`

Exercise 3: Even Numbers Challenge

Problem: Print all EVEN numbers from 2 to 20, but SKIP 10

Requirements:

- [ ] Use a for loop
- [ ] Only print even numbers
- [ ] Use `continue` to skip 10

Thinking Framework:

1. How do I check if a number is even? (Hint: `num % 2 === 0`)
2. When should I use `continue`? (When num equals 10)
3. What should the output look like? (2, 4, 6, 8, 12, 14, 16, 18, 20)

Don't Look Below Until You Try! ↓

Solution Approach (not full code!):

```javascript
for (let num = 2; num <= 20; num = num + 2) {  // Jump by 2 for even numbers
   if (num === ___) {  // What number to skip?
      continue;
   }
   console.log(___);
}
```

Exercise 4: Password Validator with While Loop

Problem: Keep asking for password until user enters "Pakistan123"

Requirements:

- [ ] Use a while loop
- [ ] Show "Wrong password" message
- [ ] Exit when correct password entered
- [ ] Bonus: Limit to 3 attempts

Thinking Framework:

1. What condition keeps the loop running? (password is wrong)
2. How do I get user input? (prompt function or predefined variable for testing)
3. How do I track attempts? (counter variable)

Starter Code:

```javascript
let password = "";
let attempts = 0;
const correctPassword = "Pakistan123";

while (password !== correctPassword && attempts < 3) {
    // TODO: Ask for password
    // TODO: Increment attempts
    // TODO: Check if wrong and show message
}

// TODO: Check if password correct or attempts exceeded
```

Exercise 5: Sum Calculator

Problem: Calculate sum of numbers from 1 to 100

Requirements:

- [ ] Use any loop type
- [ ] Add all numbers: 1+2+3+...+100
- [ ] Print final sum

Expected Output: 5050

Thinking Framework:

1. What variable holds the running sum? (Start at 0)
2. How do I add each number? (sum = sum + i)
3. Where do I print the sum? (After loop ends)

Don't Look Below Until You Try! ⬇️

Hints:

- Start sum at 0
- Each iteration: add current number to sum
- Loop from 1 to 100

🚀 آج کا چیلنج (Today's Challenge)

Project: Cricket Score Counter

کرکٹ سکور کاؤنٹر

The Problem: You're building a simple cricket innings simulator. Pakistan is playing a T20 match and you need to track the score ball-by-ball for 6 overs (36 balls total). Each ball can result in 0-6 runs, with occasional wickets.

What You're Building: A JavaScript program that:

1. Simulates 6 overs (6 balls each = 36 balls total)
2. Generates random runs (0-6) for each ball
3. Tracks boundaries (4s and 6s)
4. Counts wickets
5. Stops if team gets all out (10 wickets)
6. Calculates final score and strike rate

Success Criteria:

- [ ] Loops through all 6 overs correctly
- [ ] Generates realistic ball outcomes (0-6 runs)

- [ ] Tracks total runs, 4s, 6s, and wickets
- [ ] Stops early if 10 wickets fall (use break)
- [ ] Shows ball-by-ball commentary
- [ ] Calculates strike rate: (total runs / balls faced) × 100
- [ ] No console errors

Phase 1: Planning (سوچیں پہلے)

Before writing code, answer these questions:

1. What data do I need to track?

```
// Think about:
- Current over number (1 to 6)
- Current ball in over (1 to 6)
- Total runs scored
- Number of 4s hit
- Number of 6s hit
- Wickets fallen
- Total balls faced
```

2. What loops do I need?

```
// Think about:
- Do I need nested loops? (Yes! Overs inside balls)
- Outer loop: overs (1 to 6)
- Inner loop: balls per over (1 to 6)
```

3. When do I use break?

```
// Think about:
- When should the match end early?
- What's the maximum wickets before all out?
```

Planning Checkpoint:

- [ ] I've written down my approach in Urdu/English
- [ ] I've identified what variables I need
- [ ] I understand I need nested loops (overs → balls)
- [ ] I know when to use `break` (10 wickets)

Phase 2: Foundation (بنیاد)

Starter Code:

```
html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cricket Score Counter</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            max-width: 800px;
            margin: 50px auto;
            padding: 20px;
            background: linear-gradient(135deg, #1e3c72 0%, #2a5298 100%);
            color: white;
        }
        h1 {
            text-align: center;
            color: #4CAF50;
        }
        #scorecard {
            background: rgba(255,255,255,0.1);
            padding: 20px;
            border-radius: 10px;
            margin: 20px 0;
        }
        .ball {
            padding: 5px;
            margin: 2px 0;
            background: rgba(0,0,0,0.3);
            border-left: 3px solid #4CAF50;
        }
        .wicket {
            border-left-color: #f44336;
            font-weight: bold;
        }
        .boundary {
            border-left-color: #FFC107;
        }
        .final-score {
            font-size: 24px;
            text-align: center;
            margin: 20px 0;
            padding: 20px;
```

```
            background: rgba(76, 175, 80, 0.3);
            border-radius: 10px;
        }
    </style>
</head>
<body>
    <h1>🏏 Pakistan Innings Simulator</h1>
    <div id="scorecard"></div>
    <div id="finalScore"></div>

    <script>
        // TODO Step 1: Initialize all tracking variables
        let totalRuns = 0;
        let wickets = 0;
        let fours = 0;
        let sixes = 0;
        let ballsFaced = 0;

        // Get the scorecard div to display results
        const scorecard = document.getElementById('scorecard');
        const finalScoreDiv = document.getElementById('finalScore');

        // TODO Step 2: Create outer loop for overs (1 to 6)
        for (let over = 1; over <= 6; over++) {

            // Display over header
            scorecard.innerHTML += `<h3>Over ${over}</h3>`;

            // TODO Step 3: Create inner loop for balls (1 to 6)
            for (let ball = 1; ball <= 6; ball++) {

                // TODO Step 4: Generate random runs (0-6)
                // HINT: Math.floor(Math.random() * 7) gives 0-6
                let runs = _____;

                // TODO Step 5: Randomly decide if it's a wicket (10% chance)
                // HINT: Math.random() < 0.1 means 10% chance
                let isWicket = _____;

                // TODO Step 6: If wicket, handle it
                if (isWicket && wickets < 10) {
                    // Increment wickets
                    // Display wicket message
                    // Check if all out (10 wickets), then break
```

```
            } else {
                // TODO Step 7: If not wicket, add runs

                // TODO Step 8: Check if boundary (4 or 6)

                // TODO Step 9: Display ball result

            }

            // TODO Step 10: Increment balls faced

            // TODO Step 11: Check if all out, break outer loop too

        }

        // Break outer loop if all out
        if (wickets >= 10) {
            break;
        }
    }

    // TODO Step 12: Calculate strike rate
    // Strike Rate = (Total Runs / Balls Faced) × 100
    let strikeRate = _____;

    // TODO Step 13: Display final scorecard
    finalScoreDiv.innerHTML = `
        <div class="final-score">
            <h2>📊 Final Score</h2>
            <p>Pakistan: ${totalRuns}/${wickets} in ${ballsFaced} balls</p>
            <p>Boundaries: ${fours} fours, ${sixes} sixes</p>
            <p>Strike Rate: ${strikeRate.toFixed(2)}</p>
        </div>
    `;
</script>
</body>
</html>
```

Phase 3: Milestones (سنگ میل)

Milestone 1: Basic Loop Structure Working ✅

- [ ] Outer loop runs 6 times (overs)
- [ ] Inner loop runs 6 times per over (balls)
- [ ] Can see over headers in output

Test: Console.log each over and ball number

Milestone 2: Random Run Generation ✅

- [ ] Each ball generates a random number 0-6
- [ ] Runs are added to total
- [ ] Can see run totals increasing

Test: Check if total runs increase after each ball

Milestone 3: Wicket Logic Working ✅

- [ ] Wickets are randomly generated (about 10% of balls)
- [ ] Wicket counter increases correctly
- [ ] Match stops at 10 wickets (use `break`)

Test: Watch if innings ends before 36 balls sometimes

Milestone 4: Boundary Tracking ✅

- [ ] 4s are counted separately
- [ ] 6s are counted separately
- [ ] Boundaries are highlighted differently

Test: Count manually and verify totals

Milestone 5: Commentary & Display ✅

- [ ] Each ball shows what happened
- [ ] Wickets are displayed prominently
- [ ] Final scorecard shows all stats

Test: Read through commentary - makes sense?

Milestone 6: Strike Rate Calculation ✅

- [ ] Formula: (runs / balls) × 100
- [ ] Displays with 2 decimal places
- [ ] Makes sense (typically 80-200 in T20)

Test: Manually calculate for one innings

Debugging Guide (اگر پھنس جائیں)

Problem: Infinite loop, page freezes

- [ ] Check: Did you increment over and ball counters?
- [ ] Check: Are your loop conditions correct (<=, not just <)?
- [ ] Check: Did you add break for wickets?

Problem: Runs don't add up

- [ ] Check: Are you using `totalRuns = totalRuns + runs`?
- [ ] Check: Are you generating runs correctly (Math.random)?
- [ ] Add console.log to see runs each ball

Problem: Match doesn't stop at 10 wickets

- [ ] Check: Did you add `if (wickets >= 10) break;`?
- [ ] Check: Is break inside both loops (need to break outer loop too)?
- [ ] Check: Are you incrementing wickets correctly?

Problem: Strike rate is NaN or Infinity

- [ ] Check: Are you dividing by 0? (ballsFaced should be > 0)
- [ ] Check: Did you increment ballsFaced each ball?
- [ ] Check: Formula is `(totalRuns / ballsFaced) * 100`

Problem: Display looks messy

- [ ] Check: Are you using innerHTML correctly?
- [ ] Check: Are CSS classes applied (ball, wicket, boundary)?
- [ ] Open browser console (F12) to see errors

Extension Challenges (بونس چیلنج)

If you finish early and want more:

⭐ Level 1: Add Commentary

javascript

```
// Add realistic commentary for each ball
if (runs === 6) {
    commentary = "🎆 SIX! What a shot! The ball is out of the park!";
} else if (runs === 4) {
    commentary = "👏 FOUR! Beautiful timing!";
} else if (runs === 0) {
    commentary = "⚫ Dot ball. Good bowling!";
}
```

⭐⭐ Level 2: Multiple Innings

- Track both teams (Pakistan vs India)
- Second team must chase the target
- Determine winner

⭐⭐⭐ Level 3: Advanced Features

- Add player names (random selection from array)
- Track run rate per over
- Add dot ball counter
- Show required run rate if chasing

⭐⭐⭐⭐ Level 4: Make it Interactive

- Let user click button to simulate each ball
- Add animations for boundaries
- Sound effects for wickets and sixes
- Responsive design for mobile

📝 Daily Quiz (5 منٹ کا ٹیسٹ)

Instructions: Answer without looking at notes. Be honest with yourself!

1. What will this code output?

```javascript
for (let i = 0; i < 3; i++) {
    console.log(i);
}
```

- A) 0 1 2
- B) 0 1 2 3
- C) 1 2 3
- D) Nothing

<details> <summary>See Answer (Try first!)</summary>

Answer: A) 0 1 2

Explanation: Loop starts at 0, continues while i < 3 (so 0, 1, 2 are printed), and stops when i becomes 3 (because 3 is NOT less than 3).

</details>

---

2. Which loop ALWAYS runs at least once?

- A) for loop
- B) while loop
- C) do-while loop
- D) None of the above

<details> <summary>See Answer (Try first!)</summary>

Answer: C) do-while loop

Explanation: do-while checks the condition AFTER running the code block, so it always executes at least once. `while` and `for` check conditions BEFORE running, so they might never run.

</details>

---

3. What does `break` do in a loop?

- A) Skips to the next iteration
- B) Exits the loop immediately
- C) Pauses the loop
- D) Restarts the loop

<details> <summary>See Answer (Try first!)</summary>

Answer: B) Exits the loop immediately

Explanation: `break` terminates the loop and continues with code after the loop. `continue` skips to next iteration. There's no pause or restart.

</details>

---

4. What's wrong with this code?

```
javascript
```

```
let x = 5;
while (x > 0) {
   console.log(x);
}
```

- A) Nothing wrong
- B) Infinite loop
- C) Syntax error
- D) x is not defined

<details> <summary>See Answer (Try first!)</summary>

Answer: B) Infinite loop

Explanation: x starts at 5 and is always > 0, but we never decrease it! The condition never becomes false, so the loop never stops. Fix: Add `x--;` inside the loop.

</details>

---

5. When should you use a `for` loop instead of a `while` loop?

- A) When you don't know how many iterations
- B) When you know exactly how many iterations
- C) When you need to loop forever
- D) They're exactly the same

<details> <summary>See Answer (Try first!)</summary>

Answer: B) When you know exactly how many iterations

Explanation: Use `for` when you know the count (loop 10 times, process 50 items). Use `while` when you don't know (loop until user enters "quit", read until end of file). `for` loops are cleaner for counting.

</details>

Scoring:

- 5/5: 🎉 Excellent! You've mastered loops!
- 4/5: 👍 Great! Review the one you missed
- 3/5: 😊 Good start - review concepts again tomorrow

- <3/5: 💪 Don't worry! Practice more exercises above

🎓 Today's Homework (گھر کا کام)

Required:

- [ ] Complete the Cricket Score Counter challenge
- [ ] Fix any bugs you found
- [ ] Add at least ONE extension feature
- [ ] Review any concept from today that confused you

Optional:

- [ ] Try building a countdown timer (10 to 0)
- [ ] Create a times table generator (user inputs number, shows table 1-10)
- [ ] Build a number guessing game (1-100, give hints "higher/lower")
- [ ] Help a classmate understand loops in Urdu

Share Your Work:

- Post your Cricket Counter in your class group
- Help debug someone else's code
- Explain one concept you learned today

💭 Daily Reflection (روزانہ کی سوچ)

Take 2 minutes to answer honestly:

What I Learned Today (آج میں نے کیا سیکھا):

Example: I learned that for loops are best when I know the count,
and that I must update the condition in while loops to avoid infinite loops.

What I Found Difficult (مشکل کیا لگا):

Example: Nested loops were confusing at first, especially keeping track
of which variable belongs to which loop.

What I Want to Explore More (مزید کیا سیکھنا ہے):

Example: I want to practice more with break and continue,
and learn how to use loops with arrays.

My Confidence Level (1-10): ____

1-3: Need more practice (completely normal! Ask for help)
4-6: Getting there (review and practice more)
7-8: Comfortable (ready to help others)
9-10: Confident (can explain to friends)

📖 Tomorrow's Preview

Tomorrow we'll learn about **Arrays & Data Structures** (فہرستیں اور ڈیٹا) where you'll:

- Store multiple values in one variable
- Access and modify array elements
- Loop through arrays (combining today's loops!)
- Build a Shopping Cart system

Get Ready By:

- ☐ Make sure your Cricket Counter works
- ☐ Review for loops (you'll use them with arrays!)
- ☐ Think: How would you store a list of 100 student names in code?

📚 Resources (اگر مزید پڑھنا ہو)

Free Resources (3G-Friendly):

- 📖 MDN - Loops and Iteration - Complete reference
- 🎥 JavaScript Loops - Urdu Tutorial - Video explanation
- 💻 W3Schools Loops - Interactive examples
- 📱 Download this lesson as PDF for offline reading

Practice Problems:

- freeCodeCamp JavaScript section (free account)
- Codewars (start with 8kyu problems)
- Your own ideas - think of repetitive tasks and automate them!

CodeSensei's Tip of the Day: 💡

"The best way to master loops is to break them. Yes, really! Create infinite loops on purpose (in a test file), then fix them. Create loops that don't run at all, then fix them. Understanding WHY loops fail teaches you HOW they work. یہ غلطیاں سیکھنے کا حصہ ہیں!"

Common Beginner Mistake: Using `=` instead of `===` in loop conditions

```javascript
// WRONG:
for (let i = 0; i = 5; i++) { } // This assigns 5 to i, not a comparison!

// RIGHT:
for (let i = 0; i < 5; i++) { } // This compares i with 5
```

"کوڈ سیکھنا ایک سفر ہے، منزل نہیں۔ ہر دن ایک قدم آگے۔"
"Learning to code is a journey, not a destination. One step forward every day."

Remember: Every expert programmer started exactly where you are today. The difference is they kept practicing. You're on Day 3 of your 21-day journey. Keep going! 💪PK

Need Help?

- Ask in the class WhatsApp group
- Review this lesson again
- Try the exercises one more time
- Remember: Struggling is learning!

Prepared with ❤️ by CodeSensei for Superior University ADPCS Students
Stay motivated, stay curious, and keep coding!