Day 7: Objects & JSON

آبجیکٹس اور JSON

> **Quote of the Day:** *"Objects are the foundation of JavaScript. Master them, and you master the language."* - Kyle Simpson
>
> *"Objects JavaScript۔ انہیں سیکھ لو، زبان سیکھ لی۔" کی بنیاد ہیں*

🎉 WEEK 1 MILESTONE! (ہفتہ 1 مکمل!)

Congratulations! You've completed Week 1 of your JavaScript journey!

What you've mastered:

- ✅ Variables & Data Types
- ✅ Operators & Conditionals
- ✅ Loops & Iteration
- ✅ Functions & Scope
- ✅ Arrays & Array Methods
- 🎯 **Today:** Objects & JSON (the final piece!)

After today, you'll have a COMPLETE foundation in JavaScript!

📋 Today's Learning Goals (آج کے اہداف)

By the end of today, you will:

- ☐ Create and manipulate objects confidently
- ☐ Understand dot vs bracket notation
- ☐ Work with nested objects (objects inside objects)
- ☐ Convert between JavaScript objects and JSON
- ☐ Build a Restaurant Menu System with complex data structures

Time Breakdown (کل وقت: 150 منٹ)

- 🕐 7:00-7:05 PM (5min): Standup - Show your Product Manager!
- 🕐 7:05-8:05 PM (60min): Understanding objects (3× Pomodoro)
- 🕐 8:05-8:50 PM (45min): Practice with object operations
- 🕐 8:50-9:25 PM (35min): Build Restaurant Menu System
- 🕐 9:25-9:30 PM (5min): Quiz & reflection

🎯 What We're Building Today

Today you'll create a **Restaurant Menu System** - a complex data structure with categories, items, prices, and functions to manage orders, calculate bills, filter vegetarian items, and apply discounts!

Why This Matters for Your Career:
Objects are EVERYWHERE in real applications:

- **User profiles:** `{name, email, password, preferences}`
- **Product details:** `{id, name, price, images[], reviews[]}`
- **API responses:** ALL data from servers comes as objects!
- **Database records:** Every row is an object

Objects organize related data together - they're the heart of programming!

🧠 سمجھنا (Understanding): What Are Objects?

The Real-World Analogy

Scenario: Student ID Card (طالب علم کا کارڈ)

Imagine your university ID card:

```
┌──────────────────────────┐
│ SUPERIOR UNIVERSITY      │
├──────────────────────────┤
│ Name:     Hassan Ali     │
│ Roll No:  ADPCS-2024-101 │
│ Program:  Computer Sci   │
│ CGPA:     3.45           │
│ Semester: 2              │
└──────────────────────────┘
```

This is an object!

- Each piece of info has a **label** (Name, Roll No, etc.)
- Each label has a **value** (Hassan Ali, ADPCS-2024-101, etc.)
- All related info is **grouped together**

In JavaScript:

```
const student = {
    name: "Hassan Ali",
    rollNo: "ADPCS-2024-101",
    program: "Computer Science",
    cgpa: 3.45,
    semester: 2
};
```

Arrays vs Objects

Array = Shopping bag with items (numbered list)

```
const bag = ["Apple", "Banana", "Orange"];
// Access by position: bag[0], bag[1], bag[2]
```

Object = Labeled drawers in a cabinet (named properties)

```
const person = {
    name: "Ali",
    age: 20,
    city: "Lahore"
};
// Access by name: person.name, person.age, person.city
```

Why Objects Matter

Without objects - messy separate variables:

```
const studentName = "Hassan";
const studentRoll = "101";
const studentCGPA = 3.45;
const studentProgram = "CS";
// 100 students? 400 variables! 😱
```

With objects - organized data:

```
const student = {
    name: "Hassan",
    rollNo: "101",
    cgpa: 3.45,
    program: "CS"
};
// Everything related to Hassan in ONE place!
```

The Mental Model

Think of an object like a **passport** (پاسپورٹ):

```
Property (Key) : Value
_____________________
Name           : Hassan Ali
CNIC           : 35202-1234567-8
Date of Birth  : 15-Jan-2005
Place of Birth : Lahore
```

Each property has a **key** (label) and a **value** (data)!

📚 Building Block #1: Creating & Accessing Objects

What is an Object? (کیا ہے؟)

Urdu Analogy: Think of an object like your **phone contacts** (رابطہ):

```
Contact: Ali
├─ Number: 0300-1234567
├─ Email: ali@email.com
├─ City: Lahore
└─ Relation: Friend
```

All info about Ali in one "object"!

How to Create Objects

```
// THINKING: Object literal syntax (most common)

// Method 1: Create with properties
const person = {
    name: "Ali",
    age: 20,
    city: "Lahore",
    isStudent: true
};

// Method 2: Empty object (add properties later)
const product = {};

// Method 3: Using new Object() (less common)
const car = new Object();
```

Object Syntax Rules

```
const student = {
    // key: value,
    name: "Hassan",      // String value
    age: 20,             // Number value
    isActive: true,      // Boolean value
    courses: ["CS", "Math"],  // Array value
    address: {           // Object value (nested!)
        city: "Lahore",
        area: "DHA"
    }
};

// Last property has NO comma (optional)
```

Accessing Object Properties

Method 1: Dot Notation (preferred)

```
const person = {
    name: "Ali",
    age: 20,
    city: "Lahore"
};

console.log(person.name);  // "Ali"
console.log(person.age);   // 20
console.log(person.city);  // "Lahore"
```

Method 2: Bracket Notation (when needed)

```
console.log(person["name"]);  // "Ali"
console.log(person["age"]);   // 20

// Use when property name is in a variable
const prop = "city";
console.log(person[prop]);    // "Lahore"

// Use when property has spaces/special chars
const obj = {
    "first name": "Ali",  // Has space - use brackets!
    "user-age": 20
};
console.log(obj["first name"]);  // "Ali"
```

Adding & Updating Properties

```
// THINKING: Objects are mutable (can be changed)

const student = {
    name: "Hassan",
    rollNo: "101"
};

// Add new property
student.cgpa = 3.45;
student.semester = 2;

console.log(student);
// {name: "Hassan", rollNo: "101", cgpa: 3.45, semester: 2}

// Update existing property
student.semester = 3;  // Changed from 2 to 3

console.log(student.semester);  // 3
```

Deleting Properties

```
const user = {
    name: "Ali",
    email: "ali@email.com",
    tempPassword: "12345"
};

// Remove property
delete user.tempPassword;

console.log(user);
// {name: "Ali", email: "ali@email.com"}
```

Your First Example

```
// TODO: Create a product object
const product = {
    name: "Laptop",
    price: _____,
    brand: _____,
    inStock: _____
};

// TODO: Access properties
console.log("Product:", product._____);
console.log("Price:", product._____);

// TODO: Add warranty property
product._____ = "2 years";

// TODO: Update price (10% increase)
product.price = product.price _____ 1.1;

console.log(product);
```

Common Mistakes

❌ Wrong:

```
const person = {
    name = "Ali",  // Using = instead of :
    age = 20
};
```

✅ Right:

```
const person = {
    name: "Ali",  // Use : not =
    age: 20
};
```

❌ **Wrong:**

```
const person = {name: "Ali"};
console.log(person.Name);  // Capital N - wrong!
```

✅ **Right:**

```
const person = {name: "Ali"};
console.log(person.name);  // Exact spelling matters!
```

Check Your Understanding

- ☐ What's the syntax for creating an object?
- ☐ What's the difference between dot and bracket notation?
- ☐ Can you add properties after creating an object?
- ☐ Are property names case-sensitive?

📚 Building Block #2: Object Methods

What are Object Methods? (کیا ہیں؟)

Urdu Analogy: Think of a **mobile phone** (موبائل):

```
Properties (Data):
├─ brand: "Samsung"
├─ color: "Black"
└─ storage: "128GB"

Methods (Actions):
├─ makeCall()
├─ sendMessage()
└─ takePicture()
```

Methods are functions inside objects!

Creating Methods

```
// THINKING: Methods are object properties that are functions

const calculator = {
    // Properties
    brand: "Casio",
    model: "FX-991",

    // Methods
    add: function(a, b) {
        return a + b;
    },

    subtract: function(a, b) {
        return a - b;
    },

    // Shorthand syntax (modern)
    multiply(a, b) {
        return a * b;
    }
};

// Call methods
console.log(calculator.add(5, 3));       // 8
console.log(calculator.subtract(10, 4)); // 6
console.log(calculator.multiply(6, 7));  // 42
```

The `this` Keyword

```
// THINKING: 'this' refers to the object itself

const student = {
    name: "Hassan",
    rollNo: "101",
    marks: 85,

    // Method using 'this'
    displayInfo() {
        console.log("Name:", this.name);
        console.log("Roll No:", this.rollNo);
        console.log("Marks:", this.marks);
    },

    hasPassed() {
        return this.marks ⩾ 50;
    }
};

student.displayInfo();
// Output:
// Name: Hassan
// Roll No: 101
// Marks: 85

console.log(student.hasPassed());  // true
```

Your First Example

```
// TODO: Create a bank account object

const bankAccount = {
    accountNo: "123456789",
    balance: 10000,

    // TODO: Method to deposit money
    deposit(amount) {
        this.balance _____ amount;
        console.log(`Deposited: Rs. ${amount}`);
        console.log(`New Balance: Rs. ${this.balance}`);
    },

    // TODO: Method to withdraw money
    withdraw(amount) {
        if (amount _____ this.balance) {
            console.log("Insufficient balance!");
        } else {
            this.balance _____ amount;
            console.log(`Withdrawn: Rs. ${amount}`);
            console.log(`New Balance: Rs. ${this.balance}`);
        }
    },

    // TODO: Method to check balance
    checkBalance() {
        console.log(`Current Balance: Rs. ${this._____}`);
    }
};

// Test the methods
bankAccount.deposit(5000);
bankAccount.withdraw(3000);
bankAccount.checkBalance();
```

Check Your Understanding

- ☐ What's a method?
- ☐ What does `this` refer to?
- ☐ Can objects have both properties and methods?

- ☐ How do you call a method?

📚 Building Block #3: Nested Objects

What are Nested Objects? (کیا ہیں؟)

Urdu Analogy: Think of **address** (پتہ):

```
House Address:
├─ Street: "Main Boulevard"
├─ Area: "DHA Phase 5"
└─ City Details:  ← Nested Object!
    ├─ Name: "Lahore"
    ├─ Province: "Punjab"
    └─ Country: "Pakistan"
```

Objects can contain other objects!

How Nested Objects Work

```
// THINKING: Objects inside objects

const student = {
    name: "Hassan",
    rollNo: "101",

    // Nested object
    address: {
        street: "Main Boulevard",
        area: "DHA Phase 5",
        city: "Lahore",
        zipCode: "54000"
    },

    // Another nested object
    contact: {
        phone: "0300-1234567",
        email: "hassan@email.com"
    }
};

// Access nested properties
console.log(student.address.city);      // "Lahore"
console.log(student.address.area);      // "DHA Phase 5"
console.log(student.contact.phone);     // "0300-1234567"

// Chain multiple levels
console.log(student.address.city);  // Dot notation through levels
```

Objects with Arrays

```
const restaurant = {
    name: "Bundu Khan",
    location: "Lahore",

    // Array of strings
    specialties: ["Biryani", "Karahi", "BBQ"],

    // Array of objects!
    menu: [
        { item: "Chicken Biryani", price: 450 },
        { item: "Mutton Karahi", price: 1200 },
        { item: "Beef Seekh", price: 350 }
    ]
};

// Access array items
console.log(restaurant.specialties[0]);  // "Biryani"

// Access nested object in array
console.log(restaurant.menu[0].item);    // "Chicken Biryani"
console.log(restaurant.menu[0].price);   // 450
```

Your First Example

```
// TODO: Create a complex student object

const student = {
    name: "Fatima",
    rollNo: "102",

    // TODO: Add nested address object
    address: {
        street: _____,
        city: _____,
        country: _____
    },

    // TODO: Add array of courses (objects)
    courses: [
        { code: "CS101", name: "Programming", credits: _____ },
        { code: "MATH201", name: "Calculus", credits: _____ }
    ]
};

// TODO: Access nested data
console.log("City:", student.address._____);
console.log("First course:", student.courses[0]._____);

// TODO: Calculate total credits
let totalCredits = 0;
student.courses.forEach(course => {
    totalCredits _____ course.credits;
});
console.log("Total credits:", totalCredits);
```

Check Your Understanding

- ☐ Can objects contain other objects?
- ☐ How do you access nested properties?
- ☐ Can arrays contain objects?
- ☐ Can objects contain arrays?

📚 Building Block #4: JSON (JavaScript Object Notation)

What is JSON? (کیا ہے؟)

Urdu Analogy: Think of JSON like **written instructions** (لکھی ہوئی ہدایات):

```
You can't send a cooked biryani through text message!
But you can send the RECIPE (text format)

Similarly:
- JavaScript object = Actual data (in memory)
- JSON = Text format of data (to send/store)
```

JSON converts objects to text so you can save/send them!

Why JSON Matters

```javascript
// JavaScript Object (in code)
const user = {
    name: "Ali",
    age: 20
};

// JSON String (text format - can be saved/sent)
const jsonString = '{"name":"Ali","age":20}';

// APIs send data as JSON text!
// localStorage saves data as JSON text!
// Files store data as JSON text!
```

Converting: Object ↔ JSON

```
// THINKING: Two main operations

// 1. Object → JSON (stringify)
const person = {
    name: "Hassan",
    age: 20,
    city: "Lahore"
};

const jsonText = JSON.stringify(person);
console.log(jsonText);
// Output: '{"name":"Hassan","age":20,"city":"Lahore"}'
// This is TEXT (string), not an object!

// 2. JSON → Object (parse)
const jsonString = '{"name":"Ali","age":22,"city":"Karachi"}';

const personObj = JSON.parse(jsonString);
console.log(personObj);
// Output: {name: "Ali", age: 22, city: "Karachi"}
// This is an OBJECT, not text!

// Now you can use it
console.log(personObj.name);  // "Ali"
console.log(personObj.city);  // "Karachi"
```

JSON Rules

```
// THINKING: JSON is stricter than JavaScript objects

// ✅ Valid JSON
{
    "name": "Ali",          // Keys MUST have quotes
    "age": 20,              // Numbers OK
    "isStudent": true,      // Booleans OK
    "courses": ["CS", "Math"]  // Arrays OK
}

// ❌ Invalid JSON
{
    name: "Ali",            // Missing quotes on key!
    'age': 20,              // Single quotes not allowed!
    isStudent: true,        // Missing quotes on key!
    getValue: function() { return 5; }  // Functions not allowed!
}
```

Real-World Usage

```
// THINKING: Saving to localStorage

const userData = {
    username: "hassan123",
    email: "hassan@email.com",
    preferences: {
        theme: "dark",
        language: "Urdu"
    }
};

// Save to localStorage (must be JSON string)
localStorage.setItem("user", JSON.stringify(userData));

// Load from localStorage (convert back to object)
const saved = localStorage.getItem("user");
const loadedUser = JSON.parse(saved);

console.log(loadedUser.username);  // "hassan123"
console.log(loadedUser.preferences.theme);  // "dark"
```

Your First Example

```
// TODO: Practice JSON conversion

const product = {
    id: 123,
    name: "Laptop",
    price: 75000,
    specs: {
        ram: "16GB",
        storage: "512GB SSD"
    }
};

// TODO: Convert to JSON
const jsonString = JSON._____(product);
console.log("JSON:", jsonString);
console.log("Type:", typeof jsonString);  // string

// TODO: Convert back to object
const productObj = JSON._____(jsonString);
console.log("Object:", productObj);
console.log("Type:", typeof productObj);  // object

// TODO: Access nested property
console.log("RAM:", productObj.specs._____);
```

Common Mistakes

❌ Wrong:

```
const obj = {name: "Ali"};
localStorage.setItem("data", obj);  // Saves [object Object]
```

✅ Right:

```
const obj = {name: "Ali"};
localStorage.setItem("data", JSON.stringify(obj));
```

❌ **Wrong:**

```
const json = '{"name":"Ali"}';
console.log(json.name);  // undefined! It's a string!
```

✅ **Right:**

```
const json = '{"name":"Ali"}';
const obj = JSON.parse(json);
console.log(obj.name);  // "Ali" ✅
```

Check Your Understanding

- ☐ What is JSON?
- ☐ What's the difference between an object and JSON?
- ☐ When do you use JSON.stringify()?
- ☐ When do you use JSON.parse()?

💻 Practice Session: Object Mastery

🎯 Practice Goal

By the end of this section, you'll create and manipulate complex objects!

Exercise 1: Student Profile (ہم ساتھ کریں)

Scenario: Create comprehensive student data

Starter Code:

```
// TODO: Create student object
const student = {
    // Basic info
    name: "Hassan Ali",
    rollNo: "ADPCS-2024-101",
    semester: 2,

    // TODO: Add contact info (nested object)
    contact: {
        phone: _____,
        email: _____
    },

    // TODO: Add array of courses
    courses: [
        { code: "CS101", name: "Programming", grade: "A" },
        { code: "MATH201", name: "Calculus", grade: _____ }
    ],

    // TODO: Add method to calculate CGPA
    calculateCGPA() {
        // A=4, B=3, C=2, D=1
        // Implement calculation logic
    }
};

// Test
console.log(student.name);
console.log(student.contact.email);
console.log(student.courses[0].name);
```

Exercise 2: Product Catalog (اب آپ)

Problem: Create product with all details

Requirements:

- ☐ Product has id, name, price, category
- ☐ Nested specifications object
- ☐ Array of reviews (objects)

- ☐ Method to calculate average rating

Starter Code:

```
const product = {
    id: 1001,
    name: "Gaming Laptop",
    price: 125000,
    category: "Electronics",

    // TODO: Add specs
    specifications: {
        brand: _____,
        ram: _____,
        storage: _____
    },

    // TODO: Add reviews array
    reviews: [
        { user: "Ali", rating: 5, comment: "Excellent!" },
        { user: "Sara", rating: _____, comment: _____ }
    ],

    // TODO: Method to get average rating
    getAverageRating() {
        let total = 0;
        this.reviews.forEach(review => {
            total _____ review.rating;
        });
        return total / this.reviews.length;
    }
};

console.log(product.getAverageRating());
```

Don't Look Below Until You Try! ⬇️

Hints (if stuck):

▶ Stuck on average calculation?

```
getAverageRating() {
    let total = 0;
    this.reviews.forEach(review => {
        total += review.rating;
    });
    return total / this.reviews.length;
}
```

Exercise 3: Restaurant Data

Problem: Model a restaurant with menu

```
const restaurant = {
    name: "Bundu Khan",
    location: "Lahore",
    rating: 4.5,

    // TODO: Menu categories
    menu: {
        starters: [
            { name: "Seekh Kabab", price: _____ },
            { name: "Chicken Tikka", price: _____ }
        ],
        mains: [
            { name: "Biryani", price: _____ },
            { name: "Karahi", price: _____ }
        ]
    },

    // TODO: Method to find item
    findItem(itemName) {
        // Search in all categories
        // Return item or null
    },

    // TODO: Method to calculate bill
    calculateBill(orderItems) {
        // orderItems = ["Biryani", "Seekh Kabab"]
        // Calculate total price
    }
};
```

🚀 آج کا چیلنج (Today's Challenge)

Project: Restaurant Menu System

ریسٹورنٹ مینیو سسٹم

The Problem:
You're building a complete restaurant management system! Create a detailed menu with

categories, items with properties, functions to add items, find vegetarian options, calculate bills, apply discounts, and search!

What You're Building:
A comprehensive menu system with complex nested data structures

Success Criteria:

- ☐ Menu has multiple categories
- ☐ Each item has name, price, isVeg, spiceLevel
- ☐ Can add new items to categories
- ☐ Can find all vegetarian items
- ☐ Can calculate bill from order array
- ☐ Can apply discount for orders > Rs. 2000
- ☐ Can search for items by name
- ☐ Professional display formatting

Phase 1: Planning (سوچیں پہلے)

Before coding, answer:

1. **What structure for menu?**

```
{
    restaurantName: "...",
    categories: {
        appetizers: [...],
        mains: [...],
        desserts: [...],
        drinks: [...]
    }
}
```

2. **What structure for each item?**

```
{
    name: "Chicken Biryani",
    price: 450,
    isVeg: false,
    spiceLevel: 3  // 1-5 scale
}
```

3. **What functions needed?**
    - addItem(category, item)
    - findVegetarian()
    - calculateBill(orderArray)
    - applyDiscount(total)
    - searchItem(name)
    - displayMenu()

Planning Checkpoint:

- ☐ I understand the data structure
- ☐ I know how to work with nested objects
- ☐ I have test cases ready

Phase 2: Foundation (بنیاد)

Starter Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Restaurant Menu System</title>
</head>
<body>
    <h1>🍽️ Restaurant Menu System</h1>
    <h2>ریسٹورنٹ مینیو سسٹم</h2>
    <h3>Press F12 to see menu management!</h3>

    <script>
        // ==================================
        // RESTAURANT MENU SYSTEM
        // By: [Your Name]
        // Date: [Today's Date]
        // ==================================

        console.log("🍽️ ریسٹورنٹ مینیو سسٹم");
        console.log("==============================\n");

        // ========== MENU DATA STRUCTURE ==========

        const restaurant = {
            name: "Lahori Flavors",
            location: "Gulberg, Lahore",
            rating: 4.5,

            menu: {
                // TODO: Appetizers category
                appetizers: [
                    { name: "Chicken Seekh Kabab", price: 350, isVeg:
false, spiceLevel: 3 },
                    { name: "Paneer Tikka", price: 300, isVeg: true,
spiceLevel: 2 },
                    { name: "Spring Rolls", price: 250, isVeg: true,
spiceLevel: 1 }
                ],

                // TODO: Main courses
                mains: [
                    { name: "Chicken Biryani", price: 450, isVeg: false,
spiceLevel: 4 },
                    { name: "Mutton Karahi", price: 1200, isVeg: false,
spiceLevel: 5 },
                    { name: "Dal Makhani", price: 380, isVeg: true,
spiceLevel: 2 },
```

```
                    { name: "Vegetable Pulao", price: 320, isVeg: true,
spiceLevel: 1 }
                ],

                // TODO: Desserts
                desserts: [
                    { name: "Gulab Jamun", price: 150, isVeg: true,
spiceLevel: 0 },
                    { name: "Kheer", price: 180, isVeg: true, spiceLevel: 0
},
                    { name: "Ras Malai", price: 200, isVeg: true,
spiceLevel: 0 }
                ],

                // TODO: Drinks
                drinks: [
                    { name: "Lassi", price: 120, isVeg: true, spiceLevel: 0
},
                    { name: "Fresh Lime", price: 100, isVeg: true,
spiceLevel: 0 },
                    { name: "Rooh Afza", price: 80, isVeg: true,
spiceLevel: 0 }
                ]
            },

            // ========== METHODS ==========

            // Method 1: Display full menu
            displayMenu() {
                console.log(`\n📋 ${this.name.toUpperCase()} MENU`);
                console.log("=".repeat(50));

                // TODO: Loop through each category
                for (let category in this.menu) {
                    console.log(`\n${category.toUpperCase()}:`);
                    console.log("-".repeat(30));

                    this.menu[category].forEach((item, index) => {
                        const vegIcon = item.isVeg ? "🌿" : "🍖";
                        const spice = "🌶️".repeat(item.spiceLevel);

                        console.log(`${index + 1}. ${item.name}
${vegIcon}`);
                        console.log(`   Rs. ${item.price} ${spice}`);
                    });
                }
                console.log("\n" + "=".repeat(50));
            },
```

```
            // Method 2: Add new item
            addItem(category, itemData) {
                // TODO: Check if category exists
                if (!this.menu[category]) {
                    console.log(`❌ Category '${category}' does not
exist!`);
                    return false;
                }

                // TODO: Add item to category
                this.menu[category]._____(itemData);

                console.log(`✅ Added '${itemData.name}' to ${category}`);
                return true;
            },

            // Method 3: Find all vegetarian items
            findVegetarian() {
                console.log("\n🌱 VEGETARIAN OPTIONS");
                console.log("-".repeat(30));

                const vegItems = [];

                // TODO: Loop through all categories
                for (let category in this.menu) {
                    this.menu[category].forEach(item => {
                        if (item._____) {
                            vegItems.push({
                                category: category,
                                ...item
                            });
                        }
                    });
                }

                vegItems.forEach((item, index) => {
                    console.log(`${index + 1}. ${item.name}
(${item.category})`);
                    console.log(`   Rs. ${item.price}`);
                });

                console.log(`\nTotal vegetarian items:
${vegItems.length}`);
                return vegItems;
            },

            // Method 4: Search items by name
            searchItem(searchTerm) {
                console.log(`\n🔍 SEARCH: "${searchTerm}"`);
```

```
                console.log("-".repeat(30));

                const results = [];

                // TODO: Search in all categories
                for (let category in this.menu) {
                    this.menu[category].forEach(item => {
                        if
(item.name.toLowerCase().includes(searchTerm.toLowerCase())) {
                            results.push({
                                category: category,
                                ...item
                            });
                        }
                    });
                }

                if (results.length === 0) {
                    console.log("No items found");
                } else {
                    results.forEach(item => {
                        console.log(`• ${item.name} (${item.category}) -
Rs. ${item.price}`);
                    });
                }

                return results;
            },

            // Method 5: Calculate bill
            calculateBill(orderItems) {
                console.log("\n💰 CALCULATING BILL");
                console.log("-".repeat(30));

                let subtotal = 0;
                const foundItems = [];

                // TODO: Find each ordered item and add to bill
                orderItems.forEach(itemName => {
                    let found = false;

                    for (let category in this.menu) {
                        const item = this.menu[category].find(i => i.name
=== itemName);
                        if (item) {
                            foundItems.push(item);
                            subtotal _____ item.price;
                            console.log(`✓ ${item.name} - Rs.
${item.price}`);
```

```
                    found = true;
                    break;
                }
            }

            if (!found) {
                console.log(`X ${itemName} - Not found`);
            }
        });

        console.log("-".repeat(30));
        console.log(`Subtotal: Rs. ${subtotal}`);

        // TODO: Apply discount if > 2000
        let discount = 0;
        if (subtotal > 2000) {
            discount = subtotal * 0.1;  // 10% discount
            console.log(`Discount (10%): - Rs. ${discount}`);
        }

        const tax = subtotal * 0.05;  // 5% tax
        console.log(`Tax (5%): + Rs. ${tax}`);

        const total = subtotal - discount + tax;
        console.log("-".repeat(30));
        console.log(`TOTAL: Rs. ${total.toFixed(2)}`);

        return {
            subtotal,
            discount,
            tax,
            total
        };
    },

    // Method 6: Get items by spice level
    getBySpiceLevel(level) {
        console.log(`\n🌶️ ITEMS WITH SPICE LEVEL ${level}`);
        console.log("-".repeat(30));

        const items = [];

        for (let category in this.menu) {
            this.menu[category].forEach(item => {
                if (item.spiceLevel === level) {
                    items.push({ category, ...item });
                }
            });
        }
```

```
            items.forEach(item => {
                console.log(`• ${item.name} (${item.category})`);
            });

            return items;
        }
    };

    // ========== TESTING THE SYSTEM ==========

    console.log("🎯 TESTING RESTAURANT MENU SYSTEM\n");

    // Test 1: Display full menu
    restaurant.displayMenu();

    // Test 2: Find vegetarian items
    restaurant.findVegetarian();

    // Test 3: Search functionality
    restaurant.searchItem("Biryani");
    restaurant.searchItem("Tikka");

    // Test 4: Add new item
    console.log("\n➕ ADDING NEW ITEM");
    restaurant.addItem("mains", {
        name: "Butter Chicken",
        price: 550,
        isVeg: false,
        spiceLevel: 3
    });

    // Test 5: Calculate bill
    const order1 = ["Chicken Biryani", "Paneer Tikka", "Lassi"];
    restaurant.calculateBill(order1);

    // Test 6: Large order with discount
    const order2 = ["Mutton Karahi", "Chicken Biryani", "Dal Makhani",
"Gulab Jamun", "Lassi"];
    restaurant.calculateBill(order2);

    // Test 7: Get spicy items
    restaurant.getBySpiceLevel(5);

    // ========== CONVERT TO/FROM JSON ==========

    console.log("\n📤 JSON CONVERSION TEST");
    console.log("-".repeat(30));
```

```
        // Convert menu to JSON
        const menuJSON = JSON.stringify(restaurant.menu, null, 2);
        console.log("Menu converted to JSON (first 200 chars):");
        console.log(menuJSON.substring(0, 200) + "...");

        // Save to localStorage (simulated)
        console.log("\n✅ Menu can be saved to localStorage");
        console.log("✅ Menu can be sent to API");

        console.log("\n" + "=".repeat(50));
        console.log("🎉 WEEK 1 COMPLETE! CONGRATULATIONS!");
        console.log("=".repeat(50));

    </script>
</body>
</html>
```

Phase 3: Milestones (سنگ میل)

Milestone 1: Data Structure Works ✅

- ☐ All categories populated
- ☐ All items have correct properties
- ☐ Menu displays correctly
- Test: Can you see all 4 categories?

Milestone 2: Search & Filter Works ✅

- ☐ Can find vegetarian items
- ☐ Can search by name
- ☐ Can filter by spice level
- Test: Search "Biryani" finds item?

Milestone 3: Order System Works ✅

- ☐ Can calculate bill correctly
- ☐ Discount applies when > Rs. 2000

- ☐ Tax calculated correctly
- Test: Order 3 items, verify total

Milestone 4: JSON Conversion Works ✅

- ☐ Can convert to JSON
- ☐ Can parse back to object
- ☐ Properties accessible after parsing
- Test: Save and load menu

Debugging Guide (اگر پھنس جائیں)

Problem: Cannot access nested property

- ☐ Check: Are you using correct property names?
- ☐ Check: Did you chain dots correctly? `obj.prop1.prop2`
- ☐ Add: console.log(obj) to see structure

Problem: this is undefined

- ☐ Check: Are you using arrow functions? (Use regular function)
- ☐ Check: Are you calling method correctly? `obj.method()`

Problem: JSON.parse() gives error

- ☐ Check: Is the string valid JSON?
- ☐ Check: Are keys in double quotes?
- ☐ Use: JSON validator online to check

Common Logic Issues:

```
// ❌ WRONG: Forgot to check if property exists
const price = restaurant.menu.appetizers[0].cost;
// If 'cost' doesn't exist → undefined!

// ✅ RIGHT: Check first
const item = restaurant.menu.appetizers[0];
const price = item.price || 0;
```

Extension Challenges (بونس چیلنج)

If you finish early:

🌟 Level 1: Advanced Features

```
// Get most expensive item
getMostExpensive() {
    let max = 0;
    let expensive = null;
    // Find across all categories
}

// Generate daily special (random item with discount)
generateSpecial() {
    // Pick random item, apply 20% discount
}
```

🌟🌟 Level 2: Order Management

```
// Track order history
const orders = [];

placeOrder(orderItems, customerName) {
    const bill = this.calculateBill(orderItems);
    orders.push({
        customer: customerName,
        items: orderItems,
        total: bill.total,
        timestamp: new Date()
    });
}

// Get sales report
getSalesReport() {
    // Total sales, popular items, etc.
}
```

🌟🌟🌟 Level 3: Full Restaurant System

```
// Add tables and reservations
const restaurant = {
    // ... existing code ...

    tables: [
        { number: 1, capacity: 4, isAvailable: true },
        { number: 2, capacity: 2, isAvailable: false }
    ],

    reservations: [],

    bookTable(tableNumber, customerName, time) {
        // Implementation
    }
};
```

📝 Daily Quiz (5 منٹ کا ٹیسٹ)

Instructions: Answer WITHOUT looking at notes!

1. How do you create an object?

- A) `const obj = []`
- B) `const obj = {}`
- C) `const obj = ()`
- D) `const obj = ""`

▶ See Answer (Try first!)

Answer: B - `const obj = {}`. Curly braces `{}` create an object. Square brackets `[]` create an array.

---

2. What's the difference between dot and bracket notation?

- A) No difference
- B) Dot is faster
- C) Bracket allows dynamic property names
- D) Bracket is deprecated

▶ See Answer (Try first!)

Answer: C - Bracket notation allows dynamic property names (in variables) and properties with spaces/special characters. `obj[variableName]` vs `obj.propertyName`

---

3. What does JSON.stringify() do?

- A) Creates an object
- B) Converts object to string

- C) Converts string to object
- D) Deletes properties

▶ See Answer (Try first!)

Answer: B - Converts JavaScript object to JSON string (text format). Used for saving/sending data. Opposite of JSON.parse().

---

4. What will this output?

```
const obj = {a: {b: {c: 5}}};
console.log(obj.a.b.c);
```

- A) undefined
- B) 5
- C) {c: 5}
- D) Error

▶ See Answer (Try first!)

Answer: B - 5. You can chain dots to access nested properties: `obj.a` gets `{b: {c: 5}}`, then `.b` gets `{c: 5}`, then `.c` gets `5`.

---

5. What does `this` refer to in an object method?

- A) The window object
- B) The function itself
- C) The object containing the method
- D) undefined

▶ See Answer (Try first!)

Answer: C - The object containing the method. `this` refers to the object that the method belongs to. So `this.property` accesses that object's property.

Scoring:

- **5/5:** 🎉 Object Master! Week 1 Champion!
- **4/5:** 👏 Excellent! You're ready for Week 2!
- **3/5:** 👍 Good! Review object concepts

- **/5:** 💪 Practice more with objects

🎓 Today's Homework (گھر کا کام)

Required (لازمی):

- ☐ Complete the Restaurant Menu System
- ☐ Test all methods with different data
- ☐ Explain to a family member: "What is an object?" using ID card analogy
- ☐ **CELEBRATE WEEK 1 COMPLETION!** 🎉

Optional (اختیاری):

- ☐ Try the extension challenges
- ☐ Create a "Library Management System" with books (objects)
- ☐ Build a "School Database" with students, courses (nested objects)
- ☐ Make a "Shopping Cart" system similar to Restaurant

For Weekend:

- ☐ Review ALL Week 1 concepts (Variables → Objects)

- ☐ Practice your weakest topic
- ☐ Prepare for Week 2: DOM & Events!

💭 Daily Reflection (روزانہ کی سوچ)

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

مشکل کیا لگا (What I Found Difficult):

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

My Confidence Level (1-10): _____

🎉 WEEK 1 COMPLETE! (ہفتہ 1 مکمل!)

What You've Mastered:

✅ **Day 1:** Variables & Data Types
✅ **Day 2:** Operators & Conditionals
✅ **Day 3:** Loops & Iteration

✅ **Day 4:** Functions & Scope
✅ **Day 6:** Arrays & Methods
✅ **Day 7:** Objects & JSON

Your Progress:

You now have a **COMPLETE** foundation in JavaScript! You can:

- Store and manipulate data (variables, arrays, objects)
- Make decisions (conditions)
- Repeat actions (loops)
- Organize code (functions)
- Structure complex data (nested objects)
- Save/send data (JSON)

This is HUGE! Most people take 2-3 months to learn this. You did it in ONE WEEK! 🚀

🔄 Week 2 Preview

Next week, your code comes to life in the browser!

Week 2: DOM Manipulation & Interactivity

- Day 8: DOM Basics & Selection
- Day 9: Event Handling
- Day 10: Forms & Validation (MILESTONE!)
- Day 11: LocalStorage & Persistence
- Day 12: Dynamic HTML Creation
- Day 13: Styling with JavaScript
- Day 14: Week 2 Review (MILESTONE!)

Get Ready By:

- ☐ Celebrating your Week 1 success! 🎉
- ☐ Resting your brain this weekend
- ☐ Thinking: How does JavaScript make websites interactive?

📚 Resources (اگر مزید پڑھنا ہو)

Free Resources (3G-Friendly):

📖 MDN - Objects

- Link: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects
- Best for: Deep dive into objects

📖 MDN - JSON

- Link: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON
- Best for: Understanding JSON

📹 JavaScript Objects in Urdu

- Search: "JavaScript objects JSON Urdu tutorial"
- Best for: Visual learners

💻 Practice Objects

- Model real things: students, products, restaurants
- Practice nested structures
- Try JSON conversion

CodeSensei's Tip of the Day: 💡

"Objects are everywhere in professional code. Every API response is an object. Every user profile is an object. Every product listing is an object. You've just learned the most important concept in JavaScript. From tomorrow, you'll use objects to build REAL interactive applications. Get excited!"

"Objects ہر جگہ ہیں۔ ہر API response ایک object ہے۔ ہر user profile ایک object ہے۔ آپ نے JavaScript کا سب سے اہم concept سیکھ لیا!"

👥 Team Activity (Monday Standup)

Monday at 7:00 PM, be ready to share:

1. Your Restaurant Menu System demo
2. One thing you're proud of from Week 1
3. One concept you want to review over weekend
4. Your excitement level for Week 2 (1-10)!

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."

🎉 WEEK 1 COMPLETE! YOU DID IT! 🎉

Rest well this weekend. Week 2 starts Monday!

اللہ حافظ! Enjoy your success - you earned it! 🌟