Object Masterclass Session

Instructor: Pramod Kumar Jena

Session Outline

---

1. Introduction to Objects in JavaScript

What is an Object?

- **Definition**: In JavaScript, an object is a collection of key-value pairs. Each key is a unique identifier, and each value can be any data type (string, number, array, function, another object).
- **Structure**: Objects are structured as `{ key: value }` pairs, making them ideal for organizing related data.

Why Use Objects?

- **Organized Data**: Group related information together (e.g., a user's profile data).
- **Access by Key**: Retrieve values using keys, improving readability and organization.

Real-Life Examples:

1. **User Profile**: An object can store a user's details, such as name, age, and location.
2. **E-commerce Product**: An object can represent a product with properties like `price`, `name`, and `category`.
3. **Library Book**: Each book can be an object containing properties like `title`, `author`, and `ISBN`.

---

2. Creating and Accessing Objects

Creating Objects:
Object Literal Notation:

```
let person = {
    name: "Alice",
    age: 30,
```

```
    occupation: "Engineer"
};
```

Using `new Object()`:

```
let car = new Object();
car.brand = "Toyota";
car.year = 2020;
```

Accessing Properties:
Dot Notation:

```
console.log(person.name); // "Alice"
```

Bracket Notation:

```
console.log(person["age"]); // 30
```

Example:

Define a book object and access its properties.

```
let book = {
    title: "To Kill a Mockingbird",
    author: "Harper Lee",
    year: 1960
};
console.log(book.title); // "To Kill a Mockingbird"
```

---

3. Adding, Modifying, and Deleting Properties

Adding Properties:

```
person.location = "New York";
```

Modifying Properties:

```
person.age = 31;
```

Deleting Properties:

```
delete person.occupation;
```

Example:

Update a library book's availability status:

```
book.available = true;
book.available = false; // Changing availability
delete book.year; // Remove publication year
```

---

4. Nested Objects and Arrays in Objects

Nested Objects:

Objects can contain other objects as properties.

```
let employee = {
    name: "Bob",
    job: {
        title: "Developer",
        department: "IT"
    }
};
console.log(employee.job.title); // "Developer"
```

Arrays within Objects:

Objects can also contain arrays as properties.

```
let student = {
    name: "Chris",
    subjects: ["Math", "Science", "History"]
};
console.log(student.subjects[1]); // "Science"
```

Real-Life Example:

A shopping cart item structure in an e-commerce website:

```
let cartItem = {
    productId: "A123",
    productName: "Laptop",
    price: 1200,
    quantity: 2,
    seller: {
        name: "ElectroMart",
        rating: 4.5
    },
    reviews: ["Great product!", "Worth the price!"]
};
console.log(cartItem.seller.name); // "ElectroMart"
```

---

5. Iterating Over Object Properties

Using `for...in` Loop:

Iterate over keys in an object.

```
for (let key in person) {
    console.log(key, person[key]);
}
```

Using `Object.keys()`, `Object.values()`, and `Object.entries()`:

`Object.keys()`: Returns an array of keys.

```
console.log(Object.keys(person)); // ["name", "age", "location"]
```

`Object.values()`: Returns an array of values.

```
console.log(Object.values(person)); // ["Alice", 31, "New York"]
```

`Object.entries()`: Returns an array of key-value pairs.

```
console.log(Object.entries(person)); // [["name", "Alice"], ["age",
31], ["location", "New York"]]
```

---

6. Advanced Object Features

6.1 `this` Keyword

Contextual Reference: Refers to the current object within a method.

```
let user = {
    name: "Charlie",
    greet: function() {
        console.log("Hello, " + this.name);
    }
};
user.greet(); // "Hello, Charlie"
```

6.2 Object Shorthand and Computed Properties

Shorthand for properties when variable names match property names.

```
let title = "Designer";
let employee = { title };
```

Computed Properties:

```
let propName = "age";
let person = { [propName]: 25 };
```

6.3 Destructuring Objects

Extract multiple properties in a single line.

```
let { name, age } = person;
```

6.4 Object Methods (`assign`, `freeze`, `seal`)

`Object.assign()`: Merge objects.

```
let fullProfile = Object.assign({}, person, employee);
```

`Object.freeze()`: Prevent modification.

```
Object.freeze(person);
```

`Object.seal()`: Allow modification but no addition/removal.

```
Object.seal(employee);
```

---

7. Practical Examples and Exercises (1 Hour)

Real-Life Examples:

Inventory Management System:

```
let product = {
    id: "P001",
```

```
    name: "Phone",
    stock: 30,
    restock(amount) {
        this.stock += amount;
    }
};
product.restock(20); // Updates stock to 50
```

Order Tracking System:

```
let order = {
    orderId: "ORD123",
    items: ["Laptop", "Charger"],
    total: 1200,
    isDelivered: false,
    deliver() {
        this.isDelivered = true;
    }
};
order.deliver();
```

Exercises:

1. **Create a Library System**:
    - Define a `Book` object with properties like title, author, ISBN, and availability.
    - Add a method to `borrow` and `return` the book.
2. **Implement a Simple Banking System**:
    - Define a `BankAccount` object with properties like `accountNumber`, `balance`, and `accountHolder`.
    - Add methods to `deposit` and `withdraw` funds.
3. **E-commerce Cart**:
    - Create an `Item` object to represent a product in the cart.
    - Create a `Cart` object to manage items, including methods to add, remove, and calculate total cost.
4. **Student Grades**:
    - Define a `Student` object with a `grades` array.
    - Add a method to calculate the `averageGrade`.

---

8. Interview-Style Questions and Patterns

Question Examples:

1. **Object Merging**:
    - Explain how to merge two objects and handle conflicts.
2. **Nested Object Access**:
    - Describe how to safely access deeply nested properties.

Pattern Challenge:

1. **Pattern 1: Organization Chart**:
    - Use nested objects to represent a company's hierarchy (departments, teams, employees).
2. **Pattern 2: JSON Parsing**:
    - Create an object from a JSON string and access properties dynamically.