📘 Day 5: JavaScript Objects

---

1. What is an Object?

- An **object** is a collection of **key–value pairs**.
- Each **key** is called a **property name** (like `name`, `age`, `city`).
- Each **value** can be:
- a string, number, boolean, array, another object, or even a **function** (called a **method** when inside an object).

📌**Syntax:**

```
let objectName = {
  key1: value1,
  key2: value2,
  key3: value3
};
```

🧱**Example:**

```
let car = {
  brand: "Toyota",
  model: "Camry",
  year: 2022
};
```

---

2. Object Literals ``

- The most common way to create an object is using **object literal notation** `{}`.

```
let person = {
  name: "Umesh",
  age: 21,
  city: "Bengaluru"
};
```

---

3. Properties and Methods

- **Properties** → variables inside objects.
- **Methods** → functions inside objects.

```
let student = {
  name: "Rahul",
  age: 20,
  city: "Delhi",

  // method (function inside object)
  greet: function() {
    console.log("Hello, my name is " + this.name);
  }
};

console.log(student.name);    // 🔔 Rahul
student.greet();              // 🔔 Hello, my name is Rahul
```

🔑Here `this.name` refers to the `name` property of the same object.

---

4. Accessing Values

We can access object properties in two ways:

(a) Dot notation (``)

```
console.log(student.age);  // 🔔 20
```

(b) Bracket notation (``)

- Useful when the key is **dynamic** or has spaces.

```
console.log(student["city"]);   // 🔔 Delhi

let key = "name";
console.log(student[key]);      // 🔔 Rahul
```

---

5. Updating Properties Dynamically

You can add new properties or update existing ones anytime.

```
let person = {
  name: "Umesh",
  age: 21
};

// Add new property
person.city = "Bengaluru";

// Update existing property
person.age = 22;

console.log(person);
// 🔔 { name: 'Umesh', age: 22, city: 'Bengaluru' }
```

---

6. Adding a Method to Print Full Details

```
let person = {
  name: "Umesh",
  age: 21,
  city: "Bengaluru",

  printDetails: function() {
    console.log(`Name: ${this.name}, Age: ${this.age}, City: ${this.city}`);
  }
};

person.printDetails();
// 🔔 Name: Umesh, Age: 21, City: Bengaluru
```

---

7. Mini Exercise (Your Example)

```
let person = {
  name: "Umesh",
  age: 21
};

// Add new property dynamically
person.city = "Bengaluru";

// Access using dot notation
console.log(person.name, person.city);   // 🔔 Umesh Bengaluru
```

```
// Add a method
person.printDetails = function() {
  console.log(`Name: ${this.name}, Age: ${this.age}, City: ${this.city}`);
};

// Call the method
person.printDetails();
// 🔔 Name: Umesh, Age: 21, City: Bengaluru
```

---

8. Deleting Properties in JavaScript Objects

The `delete` Keyword

- JavaScript allows you to **remove properties** from an object using the `delete` operator.

📌**Syntax:**

```
delete objectName.propertyName;
delete objectName["propertyName"];
```

Example:

```
let person = {
  name: "Umesh",
  age: 21,
  city: "Bengaluru"
};

console.log(person);
// 🔔 { name: 'Umesh', age: 21, city: 'Bengaluru' }

// Delete the property
delete person.city;

console.log(person);
// 🔔 { name: 'Umesh', age: 21 }
```

Deleting a Non-Existing Property

```
let student = { name: "Rahul" };
```

```
delete student.age;   // doesn't exist

console.log(student);
// 🔔 { name: 'Rahul' }
```

---

9. Full Example with Add, Update, Delete

```
let person = {
  name: "Umesh",
  age: 21
};

// Add a property
person.city = "Bengaluru";

// Update a property
person.age = 22;

// Delete a property
delete person.city;

// Add method
person.printDetails = function() {
  console.log(`Name: ${this.name}, Age: ${this.age}, City: ${this.city}`);
};

console.log(person);
// 🔔 { name: 'Umesh', age: 22, printDetails: [Function] }

person.printDetails();
// 🔔 Name: Umesh, Age: 22, City: undefined (since deleted)
```

🔗Summary Notes

1. Objects = key-value pairs.
2. Properties = variables, Methods = functions inside objects.
3. Access → dot `.` or bracket `[]` notation.
4. Objects are dynamic → add, update, delete anytime.
5. Use `this` to refer to properties inside methods.
6. Use `delete` keyword to remove properties.