Objects in depth in javascript

1. What Is a JavaScript Object?

- **Definition**: Objects are collections of key–value pairs. Think of them like labelled boxes storing values.
- **Example**:

  ```
  let person = {
    name: "Alice",
    age: 25
  };
  ```

  Here, `name` and `age` are *properties*.

---

2. Accessing & Modifying Properties

- **Dot notation**:

  ```
  console.log(person.name);  // “Alice”
  person.age = 26;
  ```

- **Bracket notation** (useful when keys are dynamic or invalid identifiers):

  ```
  console.log(person["age"]);
  person["city"] = "Delhi";
  ```

---

3. Property Mutation and Deletion

- **Add new property**:

  ```
  person.country = "India";
  ```

- **Delete a property**:

```
delete person.age;
```

---

4. Nested Objects

- Objects can hold objects as values:

```
let student = {
  name: "Rahul",
  address: {
    city: "Chennai",
    pin: 600001
  }
};
console.log(student.address.city);  // "Chennai"
```

- Useful for modelling real-world structured data.

---

5. Looping Through Object Properties

- `for...in` **loop**:

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

- Loops through all enumerable properties; good for object inspection.

---

6. Checking Properties

- `hasOwnProperty` ensures property is in the object itself:

```
student.hasOwnProperty("name");  // true
```

- `in` **operator** checks own and inherited props:

```
"toString" in student;  // true (inherited)
```

---

7. Object Methods

- Functions inside objects are called methods:

```
let calculator = {
  add(a, b) { return a + b; },
  subtract(a, b) { return a - b; }
};
console.log(calculator.add(5,3));  // 8
```

---

8. `this` Keyword Explained

- Inside a method, `this` refers to the object itself:

```
let circle = {
  radius: 5,
  area() { return Math.PI * this.radius ** 2; }
};
console.log(circle.area());
```

- Clarifies how methods access own data.

---

9. Dynamic Property Names

- Use bracket notation with variables:

```
let key = "score";
let game = {};
game[key] = 100;
```

---

10. Practical Examples & Best Practices

- Real-life example: modelling a user profile.
- **Best practices taught**:
  - Keep objects’ structure shallow.
  - Avoid deleting or mutating objects too often—may lead to confusion.
  - Use methods to encapsulate logic.

📘 Revision Notes (Beginner-Friendly)

```
# JavaScript Object Notes

## 1. Object Structure
let obj = { key: value, ... };

## 2. Access/Change Props
obj.key
obj["key"]

## 3. Add/Delete Props
obj.newProp = val
delete obj.oldProp

## 4. Nested Objects
let n = { inner: { a:1 } };
n.inner.a

## 5. Loop Props
for (let k in obj) { console.log(k, obj[k]); }

## 6. Check Existence
obj.hasOwnProperty("key")
"key" in obj

## 7. Methods & `this`
```

```
let o = {
  x:10,
  getX() { return this.x; }
};

## 8. Computed Props
let dynamic = "score";
obj[dynamic] = 50;
```