Core JavaScript Mastery Guide

Perfect 👍 You want me to **convert the whole provided theory into structured notes** with:

- 🔑 Important points highlighted
- 📌 Key definitions simplified
- 🧑‍💻 Examples provided whenever there's a program

Let's go step by step and create **comprehensive notes on JavaScript Objects**.

📘 JavaScript Objects – Complete Notes

🔹 1. What is an Object?

- A **non-primitive data type**.
- Stores **data as key-value pairs**.
- Keys → always **strings** (called **property names**)
- Values → can be **any data type** (number, string, array, object, function).
- If the value is a **function**, that property is called a **method**.
- Objects are **unordered collections** (no guaranteed order of properties).

✨ Example:

```js
const car = {
  brand: "Toyota",
  model: "Corolla",
  year: 2022,
  drive: function() {
    return "Driving...";
  }
};

console.log(car.brand);   // Toyota
console.log(car.drive()); // Driving...
```

🔹 2. JavaScript as OOP

JavaScript supports **Object-Oriented Programming** through:

1. **Encapsulation** → Group data + methods inside an object.
2. **Abstraction** → Hide internal implementation (e.g., `Math.random()`).
3. **Inheritance** → One class/object can use another's properties/methods.
4. **Polymorphism** → A function behaves differently based on input/context.

🔹 3. Object Properties

- Properties = **variables attached to objects**.
- Can be accessed with **dot (.)** or **bracket []** notation.

```js
const person = { name: "Anoop", age: 21 };
console.log(person.name);    // Anoop
console.log(person["age"]);  // 21
```

Adding new property:

```js
person.city = "Delhi";
console.log(person.city); // Delhi
```

🔹 4. Object Methods

- A **function inside an object** is called a method.
- Methods use `this` **keyword** to refer to the current object.

```js
const student = {
  name: "Ravi",
  marks: 85,
  greet: function() {
    return "Hello, " + this.name;
  }
};
```

```
console.log(student.greet()); // Hello, Ravi
```

🔹 5. Ways to Create Objects

(a) Object Literal `{}`

```js
const book = {
  title: "JS Guide",
  author: "Mohtashim",
  pages: 355
};
console.log(book.title); // JS Guide
```

(b) Using `new` Operator

```js
let employee = new Object();
employee.name = "Rahul";
employee.role = "Developer";

console.log(employee.name); // Rahul
```

(c) Constructor Function

```js
function Book(title, author) {
  this.title = title;
  this.author = author;
}
const myBook = new Book("Perl", "Mohtashim");
console.log(myBook.title);  // Perl
```

(d) `Object.create()` Method

- Creates new object from a prototype.

```js
 const proto = { greet: function(){ return "Hi"; } };
const obj = Object.create(proto);
obj.name = "Anoop";

console.log(obj.greet()); // Hi
```

(e) ES6 Classes

```js
 class Car {
  constructor(brand, model) {
    this.brand = brand;
    this.model = model;
  }
}
const car1 = new Car("Toyota", "Innova");
console.log(car1.brand); // Toyota
```

🔹 6. Adding Methods to Objects

```js
 function Book(title, author) {
  this.title = title;
  this.author = author;
  this.addPrice = function(price) {
    this.price = price;
  };
}

let myBook = new Book("Perl", "Mohtashim");
```

```js
myBook.addPrice(100);
console.log(myBook.price); // 100
```

🔹 7. `with` Keyword (⚠ Rarely Used)

- Makes object properties accessible without writing object name.

```js
let car = { brand: "Honda", model: "Civic" };

with(car) {
  console.log(brand); // Honda
  console.log(model); // Civic
}
```

🔹 8. Native (Built-in) Objects

Some built-in objects in JavaScript:

- **Number, Boolean, String, Array, Date, Math, RegExp, Symbol, Set, Map, WeakSet, WeakMap, Reflect, TypedArray**

Example:

```js
let today = new Date();
console.log(today.getFullYear()); // Current year
```

🔹 9. Object Methods

✅ Static Methods (called on `Object` directly)

- `Object.assign()` → copy properties
- `Object.create()` → create object with prototype
- `Object.entries()` → array of [key, value] pairs
- `Object.keys()` → array of keys
- `Object.values()` → array of values

- `Object.freeze()` → makes object immutable
- `Object.seal()` → prevents adding/removing properties
- `Object.is()` → strict comparison

```js
const user = { name: "Anoop", age: 21 };
console.log(Object.keys(user));   // ["name", "age"]
console.log(Object.values(user)); // ["Anoop", 21]
```

✅ Instance Methods (called on objects)

- `hasOwnProperty()` → check property existence
- `isPrototypeOf()` → check prototype chain
- `propertyIsEnumerable()` → check if property is enumerable

```js
const car = { brand: "BMW" };
console.log(car.hasOwnProperty("brand")); // true
```

🔹 10. Object Property

- `constructor` → refers to function/class that created the object.

```js
let num = 42;
console.log(num.constructor); // ƒ Number()
```

🎯 Interview Key Points

- Difference between **function** and **method**?
  👉 Function = standalone, Method = inside object (uses `this`).
- Ways to create objects in JS?
  👉 Literal, `new Object()`, Constructor function, `Object.create()`, Class.

- What is `Object.freeze()` vs `Object.seal()`?
  👉 Freeze → cannot change anything. Seal → can modify but not add/remove properties.
- Difference between `==` and `Object.is()`?
  👉 `==` does type conversion, `Object.is()` does strict equality check.

Would you like me to **create a mindmap-style flow (stepwise)** for **object creation methods** (literal → constructor → Object.create → class) so you can **visualize it in one page for revision**?