Mini Project Report - 09

Master of Computer Application – General
Semester – III

Sub: Web Technologies

Topic: Class and Constructors

By
Name: SANDRA B
Reg no.: 24110222500001

Faculty Name: VEERA RAGHAV K

Faculty Signature: _____

Department of Computer Application
Alliance University
Chandapura - Anekal Main Road, Anekal
Bengaluru - 562 106

August 2025

Mini Project Report: Car Class using JavaScript

1. Introduction :

JavaScript is a dynamic and versatile programming language widely used in **web development**. Initially, JavaScript followed a **prototype-based object system**, which was often confusing for developers coming from languages like Java, Python, or C++.

With the release of **ECMAScript 6 (ES6) in 2015**, JavaScript introduced the **class keyword**, which simplified object-oriented programming by providing a cleaner and more familiar syntax.

A **class** in JavaScript is essentially a template for creating objects. It bundles data (properties) and behavior (methods) into one structure.

In this project, a Car class is created to represent a real-world car object. Properties such as name, color, and price are stored, and a method called details() is used to print them.

This project not only demonstrates how to write a class but also shows the importance of **OOP principles** such as **abstraction, encapsulation, and reusability** in programming.

---

2. Objectives :

- To explore **ES6 features of JavaScript** (Classes & Constructors).
- To apply the **OOP paradigm** in JavaScript programming.
- To create a **real-world model (Car)** using class-based approach.
- To demonstrate **data encapsulation** using properties and methods.
- To develop a **basic mini-project** that can be extended further.

OOP plays a vital role in modern programming. Its benefits are:

1. **Modularity:** Code is divided into classes and objects, making it organized.
2. **Reusability:** Classes can be reused in different projects.
3. **Scalability:** Applications can grow easily without rewriting code.
4. **Encapsulation:** Data is protected inside objects.
5. **Abstraction:** Hides complexity and exposes only what is needed.
6. **Inheritance & Polymorphism (Future Scope):** Extend existing classes and modify behaviors.

Example:

- A Car class can be reused to create multiple car objects (BMW, Audi, Tesla) with different attributes.
- This avoids writing duplicate code.

---

3. Theory of Classes and Constructors in JavaScript

3.1 Class in JavaScript

- Introduced in **ES6**.
- Defined using the class keyword.
- Contains **properties** and **methods**.

3.2 Constructor

- A special method called automatically when an object is created.
- Used to initialize object properties.

3.3 Methods

- Functions inside a class.
- Define the behavior of objects.

3.4 Object Creation

- Objects are instances of classes.
- Created using the new keyword.

Syntax:

```
class ClassName {
  constructor(parameters) {
     this.property = value;
  }
  methodName() {
     // code
  }
}
var obj = new ClassName(arguments);
obj.methodName();
```

---

4. Algorithm / Flow

1. Start program.
2. Define a class Car.
3. Inside class, create a constructor(name, color, price) to assign values.
4. Define a method details() to display values.
5. Create an object obj of the Car class.
6. Pass arguments "BMW", "Red", 4300000".
7. Call obj.details().
8. Console prints car details.
9. End program.

---

5. Code Implementation

```
// Define a class Car
class Car {
   constructor(name, color, price) {   // constructor
      this.name = name;
      this.color = color;
      this.price = price;
   }
   details() {   // normal method
      console.log("Name: " + this.name);
      console.log("Color: " + this.color);
      console.log("Price: ₹" + this.price);
   }
}
// Creating an object of Car class
var obj = new Car("BMW", "Red", 4300000);
// Calling method
obj.details();
```

---

6. Explanation of Code

- class Car { ... } → Defines a class Car.
- constructor(name, color, price) → Initializes properties when an object is created.
- this.name = name; → Assigns the parameter value to the class property.
- details() → Prints object details.
- var obj = new Car("BMW", "Red", 4300000); → Creates an object with values.
- obj.details(); → Calls the method to display car details.

---

7. Execution Environment

- IDE Used: **Visual Studio Code (VS Code)**
- Language: **JavaScript (ES6)**
- Runtime: **Node.js**
- Steps:
    1. Save file as car.js.
    2. Open terminal in VS Code.
    3. Run:
    4. node car.js

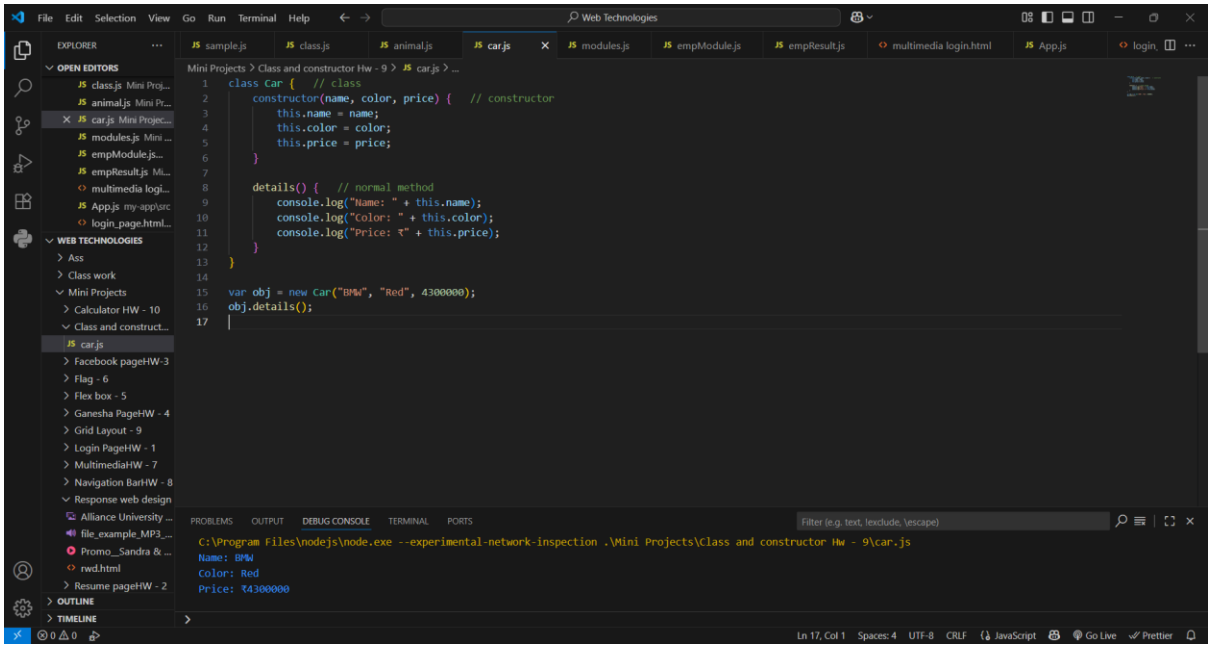
---

8. Output

Name: BMW

Color: Red

Price: ₹4300000



---

9. Result

- Successfully created a **JavaScript Class with Constructor**.
- Implemented Car class with attributes: name, color, price.
- Displayed details using a method.
- Output matched the expected values.

---

10. Advantages of Classes in JavaScript

1. Simplifies **object creation**.
2. Makes code **clean and modular**.
3. Encourages **code reusability**.
4. Useful in **real-world modeling**.
5. Works similar to **Java/Python OOP**, helping beginners transition.

---

11. Applications of JavaScript Classes in Real Life

- **E-commerce Websites** → Products as objects (name, price, category).
- **Banking Systems** → Account as class, methods for transactions.
- **Game Development** → Characters as objects with attributes like power, speed.
- **Car Showroom Apps** → Cars as objects with details (as shown in this project).

---

12. Conclusion

This mini project successfully demonstrates **Object-Oriented Programming in JavaScript** using **Classes and Constructors**.

The project showed how to:

- Define a class in JavaScript.
- Use constructors for initialization.
- Create and use objects.
- Implement methods to display values.

By completing this project, I gained confidence in **ES6 features**, improved my understanding of **OOP in JavaScript**, and learned how to represent real-world objects through programming.
