ES6 Classes and inheritance

🏫 ES6 Classes and Inheritance in JavaScript – Explained in Detail

ES6 introduced the `class` syntax in JavaScript, providing a cleaner, more elegant way to work with **objects** and **inheritance**, while still using the underlying prototype-based system.

🔷 What is an ES6 Class?

A `class` is a blueprint for creating objects with **shared structure** and **behavior**.

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }

  greet() {
    console.log(`Hi, I'm ${this.name} and I'm ${this.age} years old.`);
  }
}

const user = new Person("Abhi", 25);
user.greet(); // Hi, I'm Abhi and I'm 25 years old.
```

📌 Key Components of a Class

Keyword	Description
`constructor`	Special method for initializing object properties
Method	Defined without `function` keyword
`new`	Used to create instances of the class

🔷 Inheritance with `extends`

Use the `extends` keyword to create a **subclass** that inherits from a **parent class**.

```
class Student extends Person {
  constructor(name, age, course) {
    super(name, age); // Call parent constructor
    this.course = course;
  }

  study() {
    console.log(`${this.name} is studying ${this.course}`);
  }
}

const student = new Student("Abhi", 25, "JavaScript");
student.greet(); // Hi, I'm Abhi and I'm 25 years old.
student.study(); // Abhi is studying JavaScript
```

✅ `super` Keyword

- `super()` is used to call the **constructor** of the parent class.
- You can also use `super.methodName()` to call **methods** from the parent.

🔷 Class Fields (Public and Private)

✅ Public Fields

```
class Car {
  brand = "Toyota"; // public field
  showBrand() {
    console.log(this.brand);
  }
}
```

✅ Private Fields (ES2022+)

```
class BankAccount {
  #balance = 1000;

  getBalance() {
    return this.#balance;
  }
}
```

You cannot access `#balance` directly from outside the class.

🔷 Static Methods and Properties

- Belong to the class itself, not instances.

```
class MathUtils {
  static square(x) {
    return x * x;
  }
}

console.log(MathUtils.square(5)); // 25
```

🔷 Overriding Methods

A subclass can override a method from the parent class:

```
class Animal {
  speak() {
    console.log("Animal speaks");
  }
}

class Dog extends Animal {
```

```
  speak() {
    console.log("Dog barks");
  }
}

const dog = new Dog();
dog.speak(); // Dog barks
```

🔷 Class Expressions

You can define classes as **expressions**, useful in IIFEs or assignments.

```
const MyClass = class {
  sayHello() {
    console.log("Hello");
  }
};
```

✅ Summary

Concept	Explanation
`class`	Blueprint for creating similar objects
`constructor()`	Initializes object data
`extends`	Creates subclass from a parent class
`super()`	Calls parent constructor or method
Static method	Called on class, not instance
Private fields	Use `#fieldName` to restrict outside access

Would you like a comparison of **class vs constructor function** or an example using **multiple levels of inheritance**?