Private fields and methods

🔐 Private Fields and Methods in JavaScript – Explained in Detail

JavaScript (starting from **ES2022**) provides native support for **private fields** and **methods** in classes using the `#` syntax. These are not accessible outside the class definition, enforcing true encapsulation.

📌 Why Use Private Fields and Methods?

- To **hide implementation details**.
- Prevent external code from **modifying or accessing** internal values directly.
- Improve **code robustness** and **maintainability**.

🔷 Syntax: Private Fields

```
class Counter {
  #count = 0; // private field

  increment() {
    this.#count++;
    console.log(this.#count);
  }
}

const c = new Counter();
c.increment(); // 1
c.increment(); // 2

console.log(c.#count); // ❌ SyntaxError: Private field '#count' must be declar
ed in an enclosing class
```

> #count is not accessible outside the class.

🔷 Syntax: Private Methods

```
class Greeter {
  #getMessage(name) {
    return `Hello, ${name}!`;
  }

  greet(name) {
    console.log(this.#getMessage(name));
  }
}

const greeter = new Greeter();
greeter.greet("Abhi"); // Hello, Abhi

// greeter.#getMessage("Abhi"); // ❌ SyntaxError: Cannot access private met
hod
```

🧠 Key Characteristics

Feature	Description
Prefix	Private fields and methods use `#`
Scope	Only accessible **inside** the class body
Inheritance	Private members are **not inherited** by child classes
Encapsulation	Stronger than `var`, `_name`, or closure-based "privacy" hacks

🔶 Real-World Example

```
class BankAccount {
  #balance = 0;

  deposit(amount) {
```

```
    if (amount > 0) this.#balance += amount;
  }

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

const account = new BankAccount();
account.deposit(500);
console.log(account.getBalance()); // 500
console.log(account.#balance);     // ❌ Error
```

🧪 Private vs Protected vs Public (in other languages)

Visibility	Symbol	Access Level
Public	none	Accessible anywhere
Private	`#`	Only within class
Protected*	N/A JS	Not directly supported

> *JavaScript does not have a built-in protected keyword like Java or C++. You can simulate it with naming conventions or symbols.

⚠️ Notes and Limitations

- Private members **cannot be dynamically accessed** (`this['#name']` is invalid).
- `#` syntax is **not polyfillable** for older browsers (use Babel or stick to closures).
- Do **not** confuse private class fields with the convention `_name`, which is only by convention—not enforced privacy.

✅ Summary

Concept	Description

Private field	Declared with `#`, accessible only in class
Private method	Declared with `#`, callable only in class
Access from outside	Not allowed – throws syntax errors
Benefits	True encapsulation, secure internal logic

Let me know if you'd like a practical project using private fields/methods (e.g., a `UserAuth` class or `TodoManager`).