◆ ১. JavaScript Class কী?

`class` হচ্ছে একটি ব্লুপ্রিন্ট (blueprint), যার মাধ্যমে আমরা অবজেক্ট তৈরি করতে পারি।

ধরো, তোমার কাছে অনেকগুলো ছাত্র আছে। প্রতিটি ছাত্রের নাম, বয়স ইত্যাদি থাকবে। তাহলে `Student` নামে একটা `class` তৈরি করে তুমি সেই অনুযায়ী অবজেক্ট বানাতে পারো।

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

  greet() {
    console.log(`Hello, আমার নাম ${this.name} এবং আমার বয়স
${this.age}`);
  }
}

const person1 = new Person("Rahim", 25);
person1.greet(); // Output: Hello, আমার নাম Rahim এবং আমার
বয়স 25
```

◆ ২. Constructor

`constructor()` হল একটি special method যা class থেকে নতুন object তৈরি করার সময় স্বয়ংক্রিয়ভাবে চলে। এখানে object-এর initial value সেট করা হয়।

✵ **constructor** হলো একটি **বিশেষ ফাংশন**, যেটা **class** এর ভিতরে থাকে।

✵ যখন তুমি **class থেকে object বানাও**, তখন constructor **অটোমেটিক চালু হয়**।

✵ constructor-এর কাজ হলো:
➞ object তৈরি হওয়ার সময় সেটার জন্য **প্রাথমিক মান (value)** সেট করে দেওয়া।

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

➞ `this.name` মানে object-এর `name` প্রপার্টি, এবং `name` হচ্ছে parameter।

◆ ৩. Methods

Class এর ভিতরে আমরা function তৈরি করলে তাকে method বলে। এগুলো object ব্যবহার করে call করা যায়।

```
greet() {
  console.log("Hello!");
}
```

➞ Method call: `objectName.greet();`

◆ ৪. Object তৈরি (Instance)

`class` থেকে তৈরি হওয়া প্রত্যেকটি object কে বলে `instance`।

```
const p1 = new Person("Karim", 30);
```

➞ এখানে `p1` হচ্ছে `Person` ক্লাসের একটি instance।

◆ ৫. `this` কী?

`this` keyword দ্বারা আমরা class এর ভেতরে সেই object-এর data বুঝাই যেটি class থেকে তৈরি হয়েছে।

```
this.name = name;
```

➞ মানে object এর name প্রপার্টি সেট করছি।

◆ ৬. Inheritance (উত্তরাধিকার)

একটি class থেকে অন্য class সব property ও method পেতে পারে — একে inheritance বলে।

```
class Student extends Person {
  constructor(name, age, id) {
    super(name, age); // Parent class এর constructor কল
    this.id = id;
  }

  display() {
```

```
    console.log(`নাম: ${this.name}, বয়স: ${this.age}, আইডি: ${this.id}`);
  }
}

const s1 = new Student("Mehedi", 22, "S101");
s1.display();
```

➡ `extends` দিয়ে inheritance হয়
➡ `super()` দিয়ে parent constructor কল হয়

◆ ৭. Getters and Setters

এই method গুলোর মাধ্যমে object এর property সহজে access বা সেট করা যায়।

```
class Circle {
  constructor(radius) {
    this._radius = radius;
  }

  get radius() {
    return this._radius;
  }

  set radius(value) {
    if (value > 0) {
      this._radius = value;
    }
  }
}

const c = new Circle(5);
console.log(c.radius); // getter use
c.radius = 10;         // setter use
```

◆ ৮. Static Methods

`static` method গুলো class এর সাথে সংযুক্ত থাকে, object এর সাথে নয়।

```
class MathHelper {
  static add(a, b) {
    return a + b;
  }
}
```

```
console.log(MathHelper.add(3, 4)); // 7
```

➞ object দিয়ে `static` method কল করা যায় না

◆ ৯. Private Property (ECMAScript 2022+)

দিয়ে প্রপার্টির আগে দিলে সেটা private হয়, বাইরের কেউ access করতে পারবে না।

```
class BankAccount {
  #balance = 0;

  deposit(amount) {
    this.#balance += amount;
  }

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

◆ ১০. Class Expression

Class কে variable এ store করাও সম্ভব।

```
const MyClass = class {
  constructor(name) {
    this.name = name;
  }
};
```

✓ Summary (সংক্ষেপে)

বিষয়	বর্ণনা
`class`	Object তৈরির ব্লুপ্রিন্ট
`constructor()`	Initial মান সেট করা হয়
`method`	class এর ভিতরের function
`this`	Object কে নির্দেশ করে
`extends`	অন্য class থেকে বৈশিষ্ট্য নেয়
`super()`	parent class এর constructor কল
`getter/setter`	প্রপার্টি read/write control

বিষয়	বর্ণনা
`static`	Object ছাড়াই কল হয়
`#private`	প্রাইভেট property