# Static methods and properties

## 🧲 Static Methods and Properties in JavaScript – Explained in Detail

In JavaScript, **static methods and properties** belong to the **class itself**, not to instances of the class. This is useful for creating **utility functions**, constants, or shared logic that isn't tied to a specific object.

---

## 🔷 What is a Static Method?

A static method is defined on the **class**, not its instances. You call it directly using the class name.

### ✅ Syntax:

```
class MyClass {
  static myMethod() {
    console.log("I am static!");
  }
}

MyClass.myMethod(); // ✅ Works
const obj = new MyClass();
// obj.myMethod(); ❌ Error: obj.myMethod is not a function
```

---

### 🧠 Use Case for Static Methods

Useful for:

- Utility functions (e.g., `Math.random()`)
- Factory functions
- Common calculations or operations

### Example:

```
class MathUtils {
  static add(x, y) {
    return x + y;
  }
}

console.log(MathUtils.add(3, 7)); // 10
```

---

## 🔷 What is a Static Property?

Static properties are variables that belong to the class, not to any object created from it.

### ✅ Syntax (ES2022+):

```
class Company {
  static companyName = "OpenAI";
}

console.log(Company.companyName); // OpenAI
```

Before ES2022, static properties could only be defined **outside** the class:

```
class Company {}
Company.companyName = "OpenAI";
```

---

## 🔷 Combined Example – Static Method & Static Property

```
class Circle {
  static PI = 3.14159;

  static area(radius) {
    return this.PI * radius * radius;
  }
}

console.log(Circle.area(5)); // 78.53975
```

## 🔄 Instance vs Static Comparison

| Feature | Instance Method / Property | Static Method / Property |
|---|---|---|
| Accessed on | Instances ( `new MyClass()` ) | Class itself ( `MyClass` ) |
| Purpose | Behaviors of individual objects | Shared utility logic or constants |
| Memory usage | Separate copy per object | Single shared definition |

## ⚠️ Notes & Gotchas

- Static methods can't access instance properties directly ( `this.property` ) unless an instance is passed in.
- You can still use inheritance with static members.

### Example with Inheritance:

```
class Parent {
  static greet() {
    return "Hello from Parent";
  }
}

class Child extends Parent {}

console.log(Child.greet()); // Hello from Parent
```

## ✅ Summary

| Concept | Description |
|---|---|
| `static method` | Called on class, not on instances |
| `static property` | Belongs to class, not to any instance |
| Use case | Utilities, constants, shared calculations |
| Access syntax | `ClassName.methodName()` or `ClassName.prop` |

---

Let me know if you want a breakdown of **static vs instance** in real-world examples or a **practice project** using static members.