@JavaScriptsMagic

Inheritance In Js

JavaScriptsMagic.com

What Is Inheritance?

Inheritance Is A Key Concept In Object-Oriented Programming That Allows One Class To Inherit Properties And Methods From Another Class.

```
class Animal {
    constructor(name) {
        this.name = name;
    }
    speak() {
        return `${this.name} Makes A Sound`;
    }
}
```

Creating A Subclass Using extends

Use The extends Keyword To Create A Subclass That Inherits From A Parent Class.

```
class Dog extends Animal {
    speak() {
        return `${this.name} Barks`;
    }
}
const dog = new Dog('Buddy');
console.log(dog.speak());
// Buddy Barks
```

The Subclass Dog Inherits The Properties Of Animal And Adds Its Own Method.

Using The super() Method

Use super() To Call The Constructor And Methods Of The Parent Class.

```
class Cat extends Animal {
    constructor(name, color) {
        super(name);
        this.color = color;
    }
    describe() {
        return `${this.name} Is A ${this.color} Cat`;
    }
}
const cat = new Cat('Whiskers', 'Black');
console.log(cat.describe());

// Whiskers Is A Black Cat
```

super() Helps You Access Parent Class Properties And Methods.

Method Overriding In Inheritance

Method Overriding Allows A Subclass To Provide A Specific Implementation For A Method That Is Already Defined In Its Parent Class.

```
class Bird extends Animal {
    speak() {
        return `${this.name} Chirps`;
    }
}
const bird = new Bird('Tweety');
console.log(bird.speak());
// Tweety Chirps
```

Inheritance With Function Constructors (ES5)

Before ES6, Inheritance Was Achieved Using Function Constructors And prototype.

```
function Person(name) {
    this.name = name;
}
Person.prototype.greet = function() {
    return `Hello, ${this.name}`;
};

function Employee(name, role) {
    Person.call(this, name);
    this.role = role;
}
Employee.prototype = Object.create(Person.prototype);

const emp = new Employee('Alice', 'Developer');
console.log(emp.greet());  // Hello, Alice
```

Benefits Of Using Inheritance

- Code Reusability: Reuse Existing Code For New Classes.
- Ease Of Maintenance: Simplifies Code Updates.
- Clear Structure: Creates A Logical Hierarchy Of Classes.