Inheritance in function constructor

In JavaScript, you can achieve inheritance in function constructors using prototype chaining. This involves linking the prototype of one constructor function to another constructor function.

```
<script>
    // Parent constructor function
    function Vehicle(type) {
        this.type = type;
    }

    // Method added to the prototype of Vehicle constructor
    Vehicle.prototype.start = function () {
        return 'The ' + this.type + ' starts.';
    };

    // Child constructor function inheriting from Vehicle
    function Car(type, brand) {
        Vehicle.call(this, type); // Call the Vehicle constructor within the Car constructor
        this.brand = brand;
    }

    // Linking Car's prototype to Vehicle's prototype for inheritance
    Car.prototype = Object.create(Vehicle.prototype);
    Car.prototype.constructor = Car;

    // Method specific to Car
    Car.prototype.drive = function () {
        return 'The ' + this.brand + ' ' + this.type + ' is driving.';
    };

    // Creating an instance of Car
    const myCar = new Car('sedan', 'Toyota');

    console.log(myCar.start()); // Output: The sedan starts.
    console.log(myCar.drive()); // Output: The Toyota sedan is driving.
</script>
```

Explanation:

1. **Vehicle** is the parent constructor function defining the **type** property and has a method **start** on its prototype.
2. **Car** is the child constructor function inheriting from **Vehicle**. **Vehicle.call(this, type)** is used within the **Car** constructor to call the **Vehicle** constructor, allowing **Car** instances to inherit the **type** property from **Vehicle**.
3. **Object.create(Vehicle.prototype)** is used to link the **Car**'s prototype to a new object created from **Vehicle**'s prototype, establishing the prototype chain for inheritance.

4. **Car.prototype.constructor** is set to **Car** to ensure **Car** instances correctly point back to the **Car** constructor.
5. **Car** prototype gets a new method **drive** specific to **Car** instances.

Now, instances of **Car** can access both the **start** method inherited from **Vehicle** and the **drive** method specific to **Car**. This demonstrates inheritance in function constructors through prototype chaining in JavaScript.