Inheritance in function constructor

In JavaScript, inheritance in function constructors is achieved using a combination of constructor functions and the prototype chain.

- **Constructor Functions:**

  A constructor function is a function used to create and initialize objects. It serves as a blueprint for creating objects with a specific structure.

```
function Animal(name) {
  this.name = name;
}

// Creating an instance of Animal
const animal = new Animal('Lion');
```

- **Prototype Chain:**

  JavaScript uses a prototype chain to link objects. Each object has a prototype, which can be another object. Properties and methods defined on the prototype are shared among all instances of the constructor function.

```
// Adding a method to the Animal prototype
Animal.prototype.makeSound = function () {
  console.log('Some generic sound');
};

// Now, the animal instance can use the makeSound method
animal.makeSound(); // Outputs: Some generic sound
```

- **Inheriting from Another Constructor:**

  To achieve inheritance, you can create a new constructor function that calls the parent constructor and inherits its properties. You also need to set the prototype of the child constructor to an instance of the parent constructor.

```
function Dog(name, breed) {
  // Call the parent constructor
  Animal.call(this, name);
  this.breed = breed;
}

// Set the prototype of Dog to an instance of Animal
Dog.prototype = Object.create(Animal.prototype);

// Adding a method specific to Dog
Dog.prototype.bark = function () {
  console.log('Woof! Woof!');
};

// Creating an instance of Dog
const dog = new Dog('Buddy', 'Golden Retriever');

// Both Animal and Dog methods are available
dog.makeSound(); // Outputs: Some generic sound
dog.bark();      // Outputs: Woof! Woof!
```