Name: Nardine William Boules
Track: Open Source – Alexandria
Intake 44

Day 5 Report

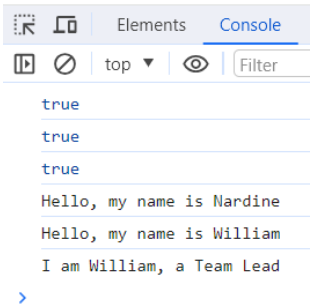
Inheritance in function constructor

In JavaScript, inheritance in function constructors can be achieved using a concept called "prototype chaining." It involves creating a hierarchy of objects where one object inherits properties and methods from another.

Example code:

```
// Person constructor
function Person(name, age) {
    this.name = name;
    this.age = age;
}
// Add a method to the Person prototype
Person.prototype.sayHello = function() {
    console.log(`Hello, my name is ${this.name}`);
};
// Employee constructor inheriting from Person
function Employee(name, age, position) {
    Person.call(this, name, age); // Call the parent constructor
    this.position = position;
}
// Link prototypes to establish inheritance
Employee.prototype = Object.create(Person.prototype);
Employee.prototype.constructor = Employee;

// Add a method to the Employee prototype
Employee.prototype.introduce = function() {
    console.log(`I am ${this.name}, a ${this.position}`);
};

// Example usage
const Nardine = new Person('Nardine', 25);
const William = new Employee('William', 45, 'Team Lead');

console.log(Nardine instanceof Person); // Output: true
console.log(William instanceof Person); // Output: true
console.log(William instanceof Employee); // Output: true

Nardine.sayHello(); // Output: Hello, my name is Nardine
William.sayHello(); // Output: Hello, my name is William
William.introduce(); // Output: I am William, a Team Lead
```

Output



Code Explanation:

1. **Person Constructor:**
    - Defines a Person constructor function that takes name and age as parameters.
    - Sets name and age properties on the instance using this.

2. **Person Prototype Method:**
    - Adds a sayHello method to the Person prototype.
    - This method logs a message containing the person's name to the console.

3. **Employee Constructor Inheriting from Person:**
    - Defines an Employee constructor function that takes name, age, and position as parameters.
    - Calls the Person constructor using Person.call(this, name, age) to set name and age properties on the Employee instance.
    - Sets the position property specific to Employee.

4. **Linking Prototypes for Inheritance:**
    - Establishes the inheritance relationship by linking the Employee prototype to an object created from the Person prototype using Object.create.
    - Sets the constructor property of Employee.prototype back to Employee to maintain proper constructor references.

5. **Employee Prototype Method:**
    - Adds an introduce method to the Employee prototype.
    - This method logs a message containing the employee's name and position to the console.

6. **Example Usage:**
    - Creates instances of Person and Employee using the new keyword.
    - Calls the sayHello method on the instance of class Person.
    - Calls both sayHello and introduce methods on the instance of class Employee.

In summary, this code demonstrates the concept of inheritance through function constructors in JavaScript. The Employee constructor inherits properties and methods from the Person constructor, and instances of both constructors have access to their respective methods.