IP Assignment 4b

Roll No: 71 Batch : T14

Aim : Write a Program on inheritance, Iterators and Generators.

Theory :

1.Inheritance

Inheritance is a fundamental concept in object-oriented programming that allows a new class (subclass or child class) to inherit properties and methods from an existing class (superclass or parent class). In JavaScript, inheritance is achieved using the class syntax and the extends keyword.

1. Constructor: The constructor is a special method that gets called when an object is created from a class. It's used to initialize object properties and perform setup tasks. When a subclass is created, its constructor can call the parent class constructor using the super() method.
2. Property: Properties are class members that hold values. They are defined within the class and can be accessed and modified using dot notation.
3. Object: Objects are instances of classes that have their own set of properties and methods.
4. Extends: The extends keyword is used to create a subclass that inherits from a superclass.
5. Super: The super keyword is used to call methods or access properties from the parent class within the child class.
6. Instance: Instances of a class can be passed to other functions or classes, allowing you to work with the object's properties and methods.
7. Static: Static members (properties or methods) belong to the class itself rather than instances. They cannot be accessed from derived classes.

Iterators:

Iterators are a fundamental concept in JavaScript introduced with ES6 to simplify the process of iterating over sequences of values, such as arrays or other data structures. They provide a standardized way to access and loop through the elements of these sequences. The key component of an iterator is its next() method, which returns an object containing the next

value in the sequence and a boolean flag indicating whether the iteration is done.

JavaScript iterators are used to loop over a sequence of values. They are commonly used in

various looping constructs to simplify the process of iterating through elements without

manually managing index variables

An iterator object implements a next() function that returns an object with two properties:

- value: The next value in the sequence.
- done: A boolean indicating if the sequence has been fully consumed

User-Defined Iterators:

JavaScript also allows you to define your own iterators for custom objects or data structures by implementing the iterator interface (next() and done properties) on an object.

Syntax : [Symbol].iterator()

Generators :

Prior to ES6, JavaScript functions followed a "run-to-completion" model, which meant that once a function started executing, it would run until it completed without the ability to pause and resume its execution. ES6 introduced a powerful feature called generators, which allows functions to pause their execution and later resume it from where they left off. This feature is particularly useful when dealing with asynchronous code, iterators, and managing complex flows.

A generator is defined using a function that is prefixed with an asterisk * and contains one or more **yield** statements. The yield keyword is used to pause the generator's execution and return a value to the caller. The function's execution can be resumed later using the next() method.

- Passing Arguments to Generators: You can also pass arguments to generator functions when using the next() method. The value passed to next() will be used as the result of the yield expression where the generator was paused.
- Returning Values from Generators: Generators can also be terminated with a return statement, which will make the generator immediately finish and return an object with the value property set to the value passed to return.

Conclusion :

Implemented Iterators, generators and inheritance. Iterators simplify looping and data traversal

Generators provide elegant solutions to asynchronous programming challenges