Name of Student: Samyak S Lamsoge		
Roll Number: 29		**LAB Assignment Number: 4**
Title of LAB Assignment: Create an application to demonstrate Node.js Functions-timer function		
DOP: 26-09-2023		**DOS 29-09-2023**
CO Mapped: **CO1**	**PO Mapped:** **PO3, PO5, PSO1, PSO2**	**Signature:**

Practical No: 4

Aim: Create an application to demonstrate Node.js Functions-timer function (displays every 10 seconds)

Description:
Node.js Functions - Timer Function

In Node.js, timer functions are essential tools for scheduling and executing code at specified intervals. These timer functions are a fundamental part of Node.js's asynchronous and event-driven architecture, making them useful for various tasks, including periodic data fetching, task automation, real-time updates, and more. Among these timer functions, one of the most commonly used is the `setInterval` function for creating timer-based functionality.

Key Concepts:

1. `setInterval` Function:

- The `setInterval` function is a built-in Node.js feature that allows you to repeatedly execute a specified function at a fixed time interval.
- It takes two arguments: the function to execute and the time interval in milliseconds.
- For example, to execute a function every 5 seconds, you can use `setInterval(myFunction, 5000)`.

2. `setTimeout` Function:

- While `setInterval` repeatedly executes a function at intervals, the `setTimeout` function executes a function once after a specified delay in milliseconds.
- It is often used for tasks that require a one-time delay.

3. Use Cases:

- **Periodic Tasks**: Timer functions are commonly used for tasks that need to occur at regular intervals, such as sending heartbeats, checking for updates, or refreshing data.
- **Real-time Applications**: In real-time applications like chats or online games, timer functions can be used to update and synchronize data among clients.
- **Task Scheduling**: Node.js timers are helpful for scheduling tasks, enabling automation of repetitive processes.

4.Clearing Timers:

- To stop a timer created with `setInterval` or `setTimeout`, you can use the `clearInterval` or `clearTimeout` functions, respectively.
- Clearing timers is important to prevent memory leaks and ensure efficient resource management in Node.js applications.

5.Asynchronous Nature:

- Timer functions in Node.js are non-blocking and asynchronous, allowing you to perform other tasks while waiting for the specified time interval to elapse.

6.Edge Cases:

- It's important to consider edge cases and potential issues when working with timers, such as timer drift (small variations in timing) and ensuring that the timer's function does not block the event loop.

Q.1 Create an application to demonstrate Node.js Functions-timer function(displays every 10 seconds)

Code:

Timer.js

```
let count = 0; let
interval;

function timerFunction() {
  console.log("This is message number " + (count + 1) + " Printed after 10 seconds"); count++;
}

module.exports = {
  startTimer: function() {
    interval = setInterval(timerFunction, 10000);
  },
  stopTimer: function() {
    clearInterval(interval);
    console.log("Timer stopped after 10 Iterations");
  }
};
```

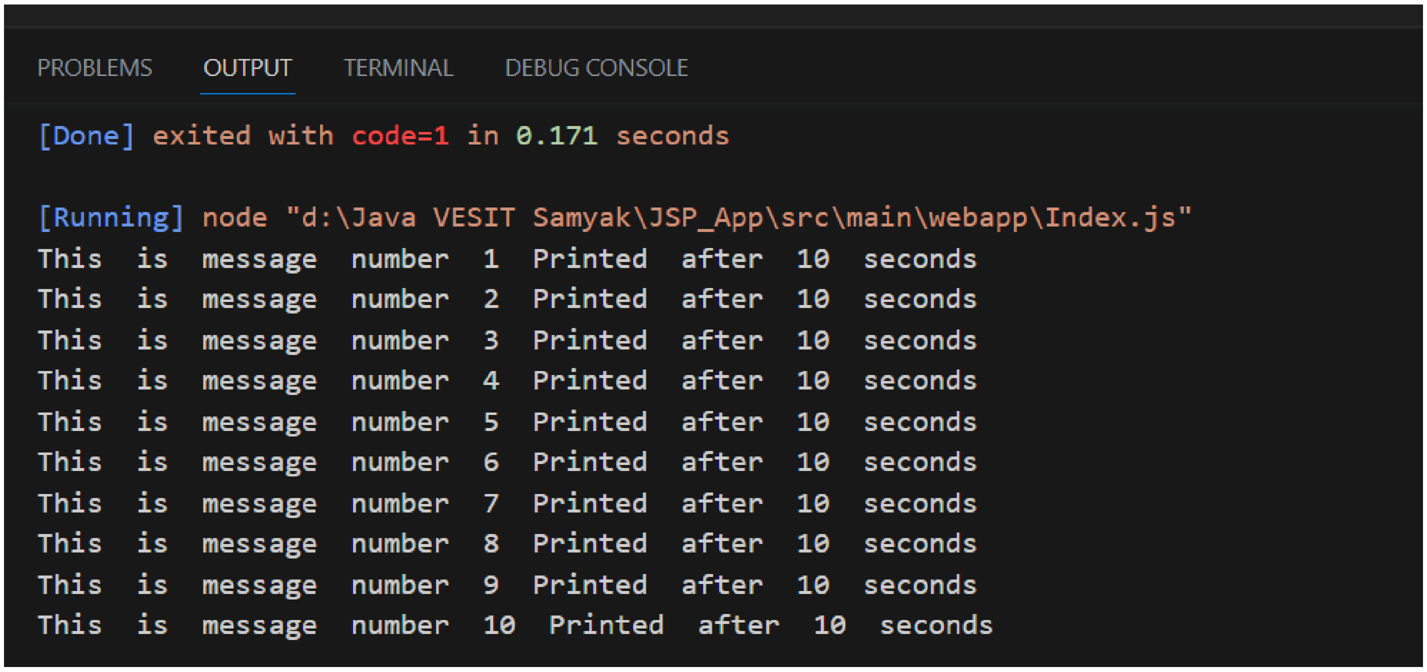
Index.js

```
const timer = require('./timer');

timer.startTimer();

// Using setTimeout to stop the timer after 10 iterations setTimeout(timer.stopTimer, 110000);
```

Output:


PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
[Done] exited with code=1 in 0.171 seconds

[Running] node "d:\Java VESIT Samyak\JSP_App\src\main\webapp\Index.js"
This  is  message  number  1  Printed  after  10  seconds
This  is  message  number  2  Printed  after  10  seconds
This  is  message  number  3  Printed  after  10  seconds
This  is  message  number  4  Printed  after  10  seconds
This  is  message  number  5  Printed  after  10  seconds
This  is  message  number  6  Printed  after  10  seconds
This  is  message  number  7  Printed  after  10  seconds
This  is  message  number  8  Printed  after  10  seconds
This  is  message  number  9  Printed  after  10  seconds
This  is  message  number  10  Printed  after  10  seconds
```

Conclusion:

Node.js timer functions, particularly `setInterval` and `setTimeout`, are powerful tools for adding time- based functionality to your applications. They are instrumental in tasks that require periodic execution, synchronization, and automation. However, developers should be mindful of potential edge cases and handle timers carefully to ensure optimal performance and reliability in their Node.js applications.
Properly managed timers can enhance the responsiveness and efficiency of your code, making them a fundamental aspect of Node.js development.