NodeJS Training - Day 3

By : LAKSHMIKANT DESHPANDE (M.Tech)

Node.js OS Interaction

- Day 2 Recap
- What is OS Module?
- Retrieving System Information
- Retrieving User Information
- Managing Environment Variables
- Working with File System
- REST (REpresentational State Transfer)
- Creating Web Server

Session 1

- Day 2 Recap
  - Exception/Error Handling
  - Modules (CommonJS & ES6)
  - EventEmitter
  - Reading and Writing Streams
  - Pipe()
  - Duplex Stream
  - Transform

Session 1 contd….

- Live Assignment on first two days topics

Session 2

- Core Modules in detail
- OS Module

Session 3

- File System

Session 4

- REST (REpresentational State Transfer)
- Creating Web Server

Live Assignments

- Gaming Zone: Player Profile Manager
- Bank: Account Management System
- E-commerce: Product Inventory System
- Edutech: Student Records System

OS Module

The **“os”** module in Node.js provides a variety of operating system-related utility methods and properties. This module is especially useful for interacting with the underlying operating system, retrieving system information, managing environment variables, and working with the file system.

Example :

```
const os = require('os');

console.log('OS Type:', os.type());

console.log('Platform:', os.platform());

console.log('Architecture:', os.arch());
```

File System

The **fs** (file system) module in Node.js provides a comprehensive suite of functions and methods for interacting with the file system.

This includes reading and writing files, creating and deleting directories, watching for changes, and handling various file system events.

File System Operations

- Reading Files
- Writing Files
- Appending to Files
- Deleting Files
- Working with Directories
- Watching Files and Directories
- File and Directory Statistics
- Copying Files
- Renaming Files and Directories

REST (REpresentational State Transfer)

REST (REpresentational State Transfer) is an architectural style for designing networked applications. It leverages standard HTTP methods and status codes to create a stateless, scalable, and efficient way to communicate between clients and servers.

Key Principles of REST

- **Statelessness**
    - Each request from a client to the server must contain all the information needed to understand and process the request. The server does not store any client context between requests.

Key Principles of REST contd….

- **Client-Server Architecture**
  - The client and server are separate entities. The client is responsible for the user interface and user experience, while the server handles data storage and business logic.
- **Uniform Interface**
  - A consistent and standardized way to interact with resources. This includes using HTTP methods (GET, POST, PUT, DELETE) and standard status codes.

Key Principles of REST contd…

- **Resource-Based**
  - Resources (e.g., users, products) are identified by URIs (Uniform Resource Identifiers). Each resource can be represented in various formats (e.g., JSON, XML).
- **Layered System**
  - The architecture is composed of multiple layers, with each layer having a specific role (e.g., client, server, cache). This allows for scalability and separation of concerns.
- **Code on Demand (Optional)**
  - Servers can extend client functionality by transferring executable code (e.g., JavaScript). This is an optional principle and is not commonly used.

Creating Web Server

- The http module provides the functionality needed to create an HTTP server, handle requests, and send responses.
- Creating a web server with the http module in Node.js is simple and allows you to handle various HTTP requests, serve static files, and manage errors effectively. This approach provides a foundation for understanding the core concepts of web servers, which can then be extended using more advanced frameworks like Express.js.