❖ Node – JS Introduction: -

1. Write an essay on the history and evolution of Node.js, discussing its architecture and key features.

- Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to build scalable and high-performance applications. It has transformed the way JavaScript is used, extending its capabilities beyond the browser and enabling server-side development.
- Node.js was created by Ryan Dahl in 2009 as a response to the limitations of traditional web servers
- Dahl introduced Node.js as a non-blocking, event-driven runtime that could efficiently manage asynchronous operations
- Node.js is built on a highly efficient and scalable architecture that distinguishes it from traditional server-side technologies.

1. Event-Driven, Non-Blocking I/O Model

- Unlike traditional web servers, which use multiple threads to handle concurrent requests, Node.js uses an event-driven, single-threaded model.
- The Event Loop continuously listens for events and executes callbacks asynchronously, reducing blocking operations and improving efficiency.

2. V8 JavaScript Engine

- Node.js uses the V8 engine, developed by Google for Chrome, to compile JavaScript directly into machine code, resulting in faster execution.

3. Asynchronous Programming

- Node.js relies heavily on asynchronous programming using callbacks, Promises, and async/await, ensuring that applications remain non-blocking and scalable.

4. Libuv Library

- Node.js uses the libuv library to manage asynchronous I/O operations efficiently, including file system access, network operations, and timers.

5. Built-in Modules

- Node.js provides a rich set of built-in modules, such as fs (file system), http (server handling), and os (operating system utilities), reducing the need for external dependencies.

2. Compare Node.js with traditional server-side technologies like PHP and Java

Node JS	PHP	Java
High due to asynchronous execution	Moderate, synchronous by default	High, optimized with JVM
High (suitable for microservices & real-time apps)	Moderate (good for traditional web apps)	High (suitable for enterprise applications)
Real-time applications (chat apps, streaming, APIs)	CMS (WordPress), eCommerce	Large-scale enterprise applications
Easy for JS developers	Easy to moderate	Steep (strict typing, complex setup)
Lightweight, deployable on cloud & serverless	Traditional hosting, LAMP stack	Requires dedicated servers, JVM-based