14 December 2025 01:13 PM

Fundamentals

1. Node.js Basics
2. Main Features of Node.js
3. Project setup & Modules
4. Top Build in Modules

Express

5. Express Basics
6. Middleware
7. Types of Middleware
8. Routing - I
9. Routing - II
10. Template Engines

REST API

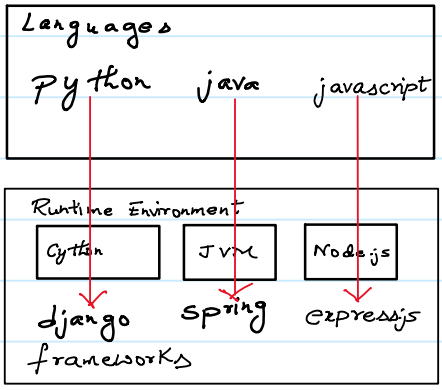
11. REST API Basics
12. HTTP Method & Status Code
13. CORS, Serialization, Deserialization, Others
14. Authentication & Authorization
15. Error Handling & Debugging

Node.js Basics

Q What is Node.js ?

⇒ Node is neither a language nor a framework

⇒ Node / Nodejs is a runtime

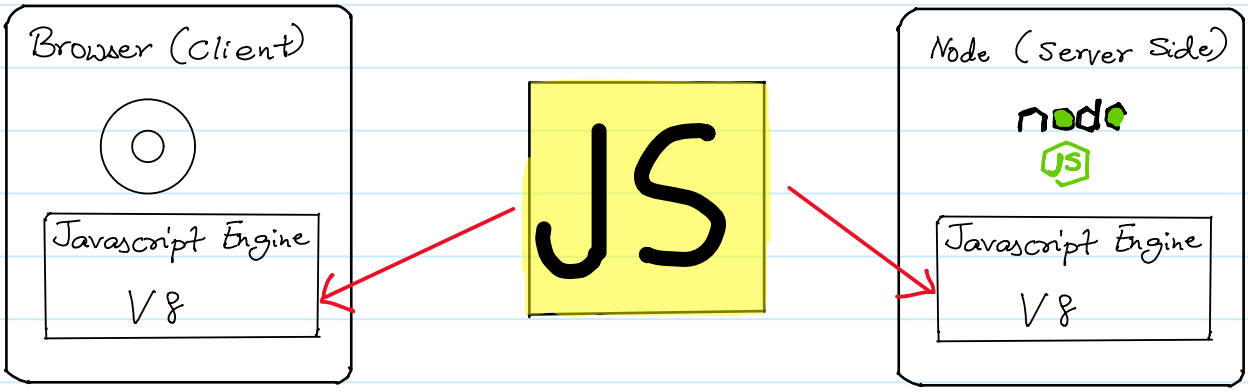


environment for executing javascript code on the server.

Q2. How node is a runtime environment on server side? What is V8 ?

⇒ Browser execute Javascript on the client side, and similarly, Node.js executes Javascript on the server side.

⇒ V8 is a Javascript engine for the Javascript language.



Q3. What is the difference between Runtime environment & Framework ?

⇒ Runtime environment: primarily focuses on providing the necessary infrastructure for code executing including services like memory management and I/O operations.

Framework : Primarily focuses on simplifying the development process by offering a structured set of tools, libraries, and best practices.

Q4. What is the difference between Node.js & Express?

⇒ Node.js is a runtime environment that allows the execution of Javascript code server side.

⇒ Express.js is a framework built on top of Node.js

⇒ It is designed to simplify the process of building web applications and API's by providing a set of features like simple routing system, middleware support etc

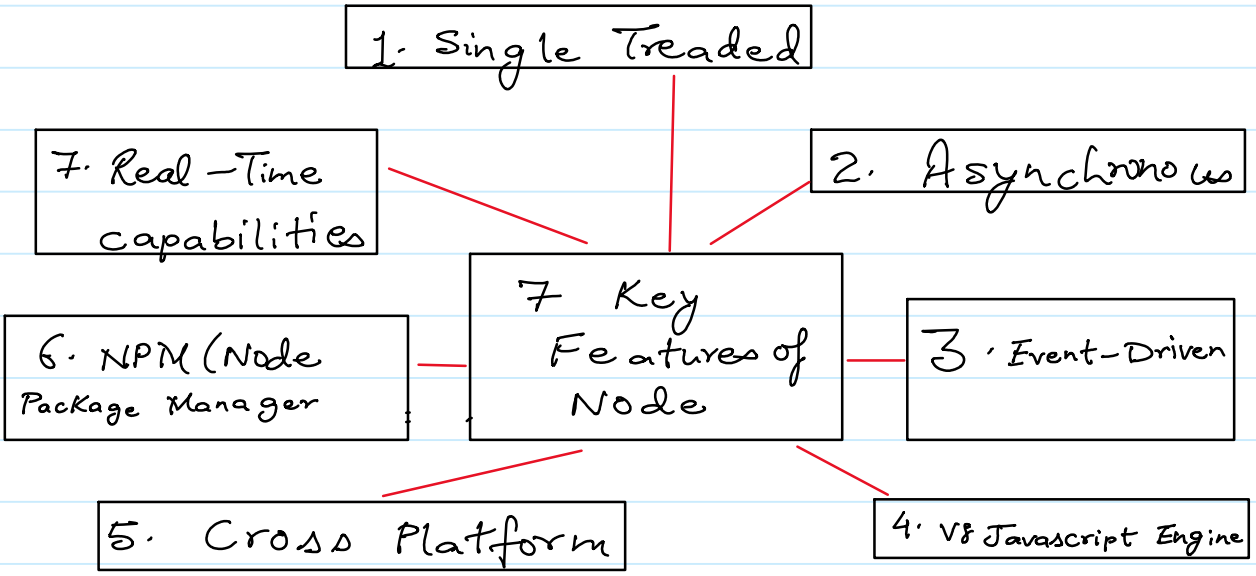
Q. What is the differences between Client-Side (Browser) & Server Side (Node.js)?

	Client-Side (Browser)	Server-Side (Node.js)
Environment location	Runs on the user's web browser	Runs on the server
Primary Languages	HTML, CSS, Javascript	Javascript
Document/window/Navigator/ Event Objects	Yes	No
Request/Response/Server/ Database Object	No	Yes
Responsibilities	Handles UI display, interactions, and client-side logic	Handles business logic, data storage/access, authentication, authorization etc.

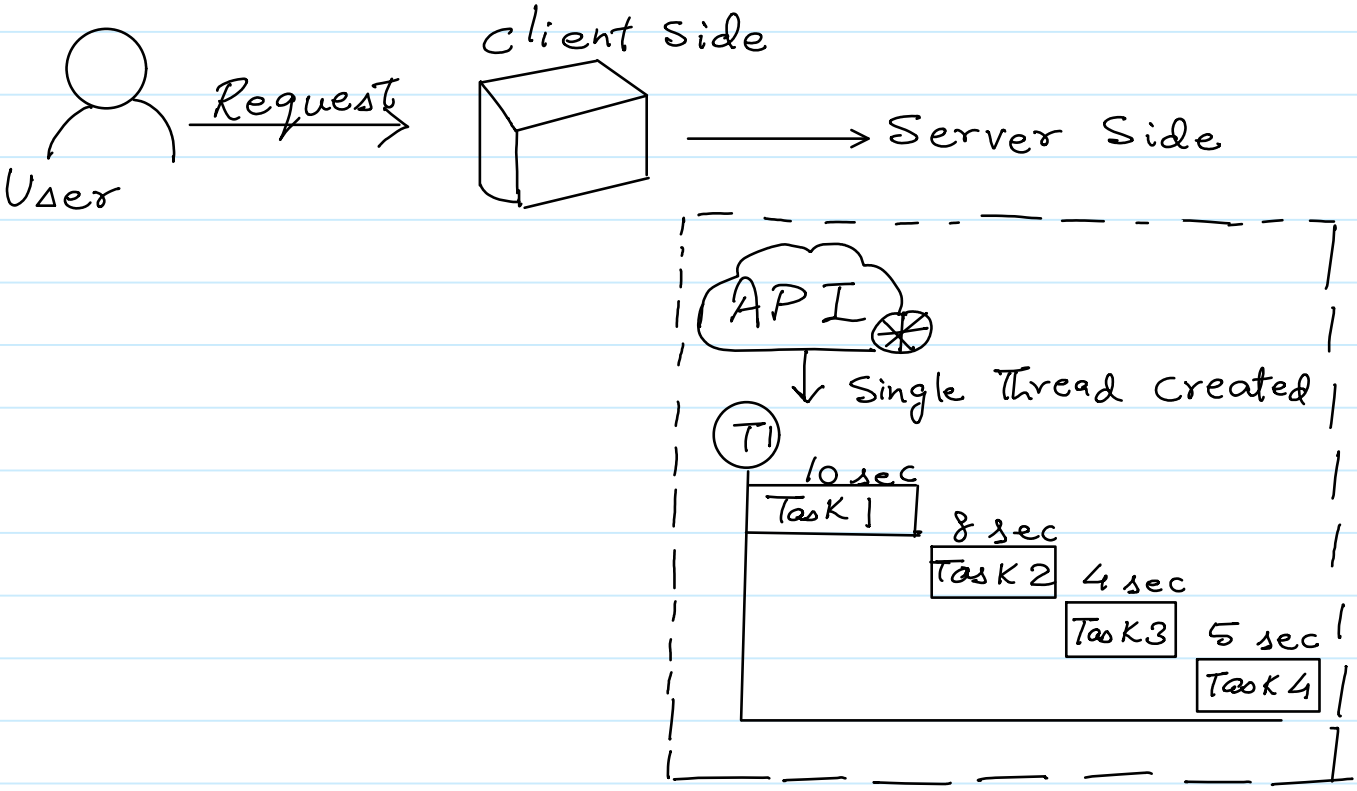
Main Features of Node.js

Q. What are the 7 Main Features of Node.js ?

⇒

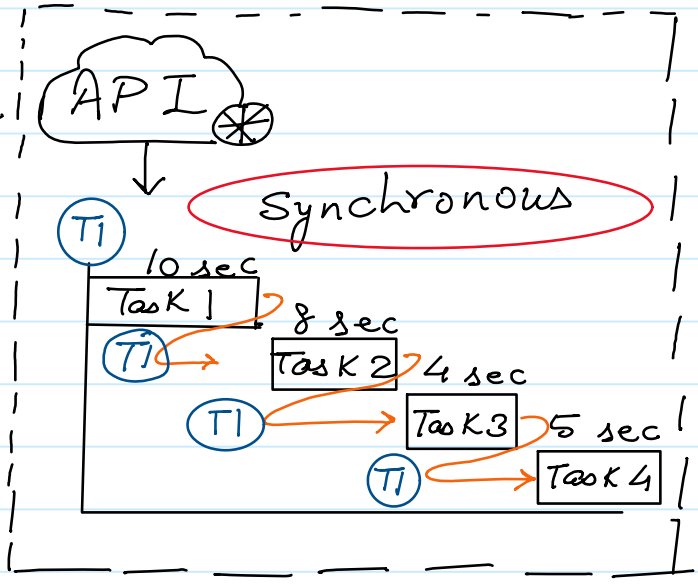


Q. What is Single Treaded Programming ?



Q. What is Synchronous Programming ?

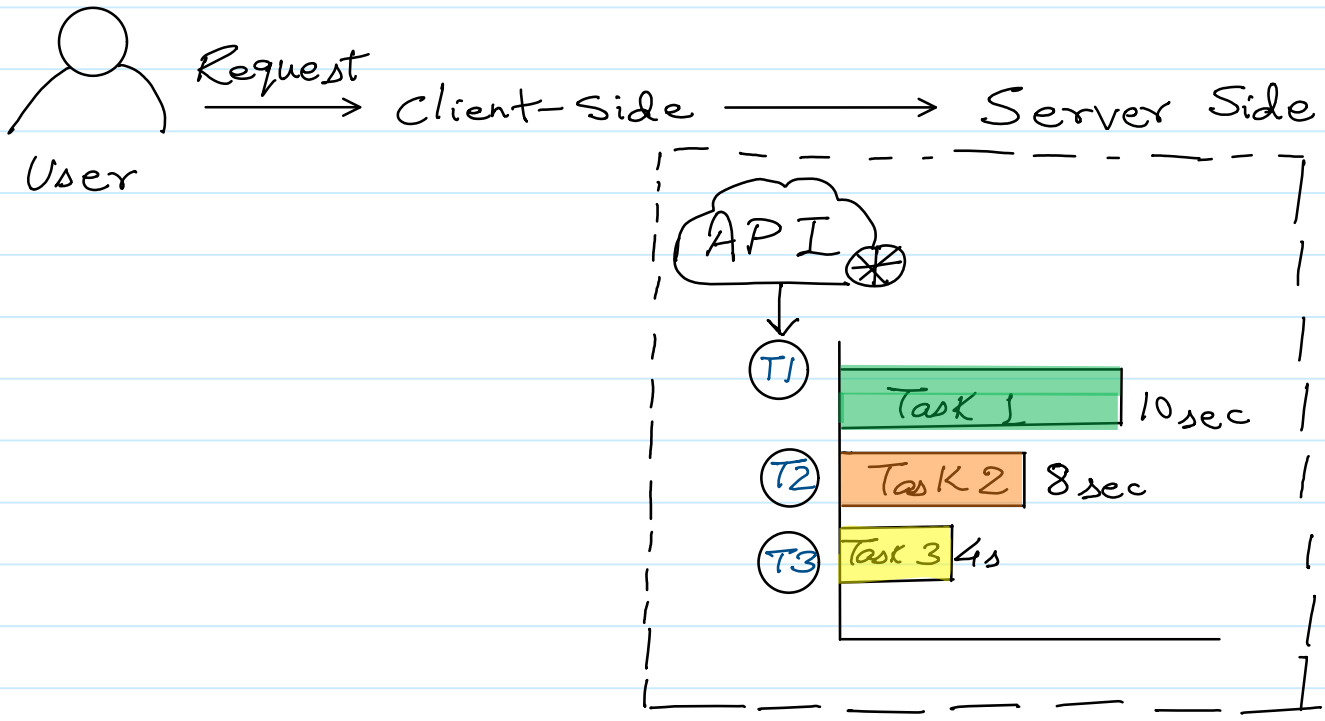
⇒ In a synchronous program each task is performed one after the other, and the program waits for each operation to complete before moving on to the next one.



⇒ Synchronous programming focuses on the order of execution in a sequential manner, while single-threaded programming focuses on the single thread.

Q. What is Multi Threaded Programming ?

⇒



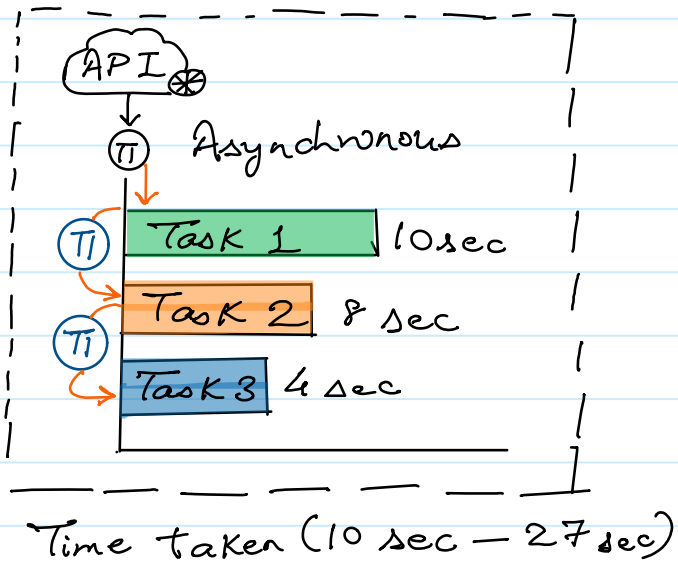
Q. What is Asynchronous Programming ?

⇒ In Node.js, asynchronous flow can be achieved by it's single-threaded, non-blocking, and event-driven architecture.

⇒ In Node.js, if there are 4 tasks (task1, task 2, task 3, task 4) to be completed for an event. Then below steps are executed :

1) First thread T1 will be created.
2) Thread T1 initiates Task 1, but it won't wait for Task 1 to complete. Instead, T1 proceeds to initiate Task 2, then Task 3 and Task 4 (This asynchronous execution allows T1 to efficiently handle multiple task concurrently.
3) Whenever Task 1 completes, an event is emitted.
4) Thread T1, being event-driven, promptly respond to this

event, interrupting it's current task and delivering the result of Task



Q. What is the difference between Synchronous & Asychronous programming ?

Synchronous programming	Asynchronous programming
1. In synchronous programming, tasks are executed one after another in a sequential manner.	1. In asynchronous programming, tasks can start, run, and complete in parallel.
2. Each task must complete before the program moves on to the next task.	2. Tasks can be executed indepe-ndently of each other.
3. Execution of code is blocked until a task is finished.	Asynchronous operations are typically non-blocking
4. Synchronous operations can lead to blocking and unresponsiveness	It enables better concurrency and responsivenes

Q. What are Events, Event Emitter, Event Queue, Event Loop & Event Driven ?

⇒ **Event**: Signals that something has happened in a program

Event Emitter: Create or emit events.

Event Queue: Events emitted queued (stored) in event queue

Event Loop: The event loop picks up event from the event queue and executes them in the order they were added.

Event Driven Architecture: It means operations in Node are drive or based by events.