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

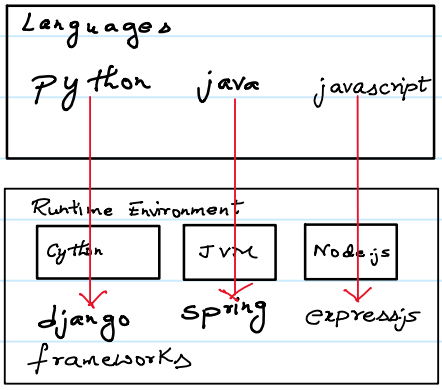
11. REST API Basics
12. HTTP Method & Status Code
13. CORS, Serialization, Deserialization, Others
14. Authentication & Authorization
15. Error Handling & Debugging

Node.js Basics

Q What is Node.js ?

⇒ Node is neither a language nor a framework

⇒ Node / Nodejs is a runtime

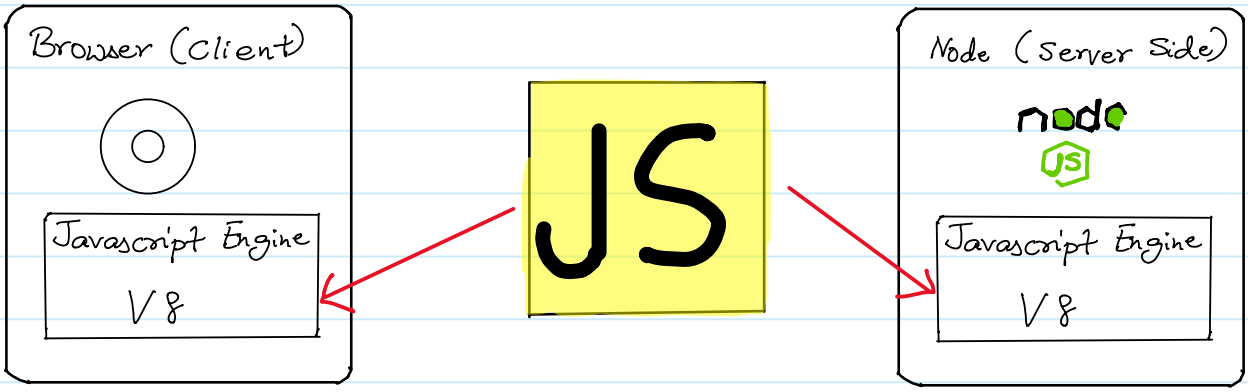


environment for executing javascript code on the server.

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

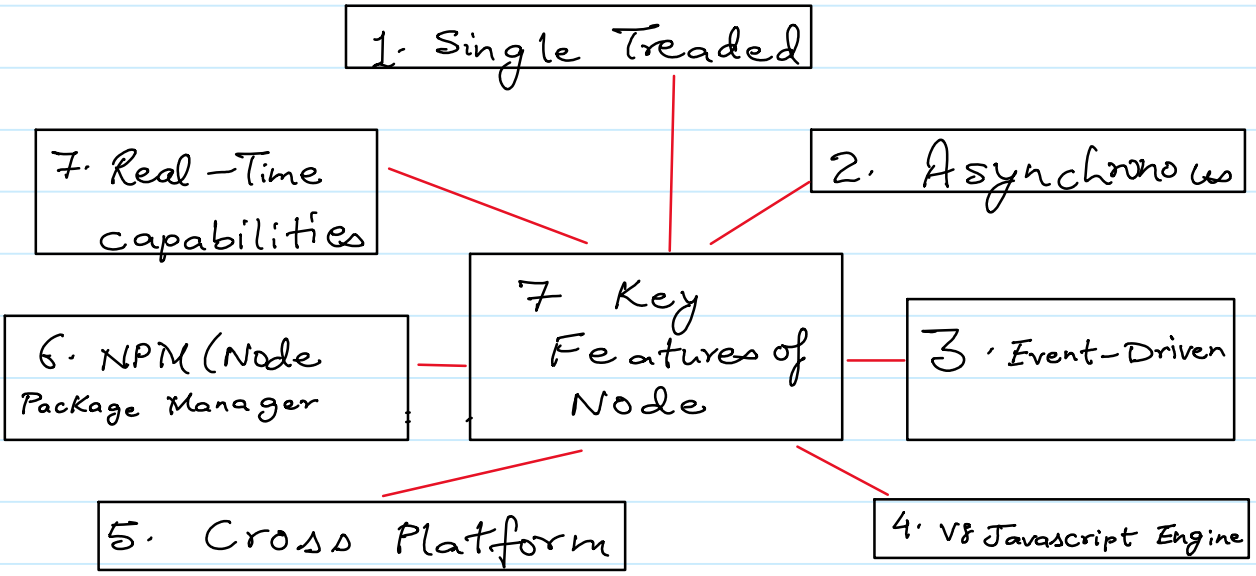
Q. What is the differences between Client-Side (Browser) & Server Side (Node.js)?

	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.

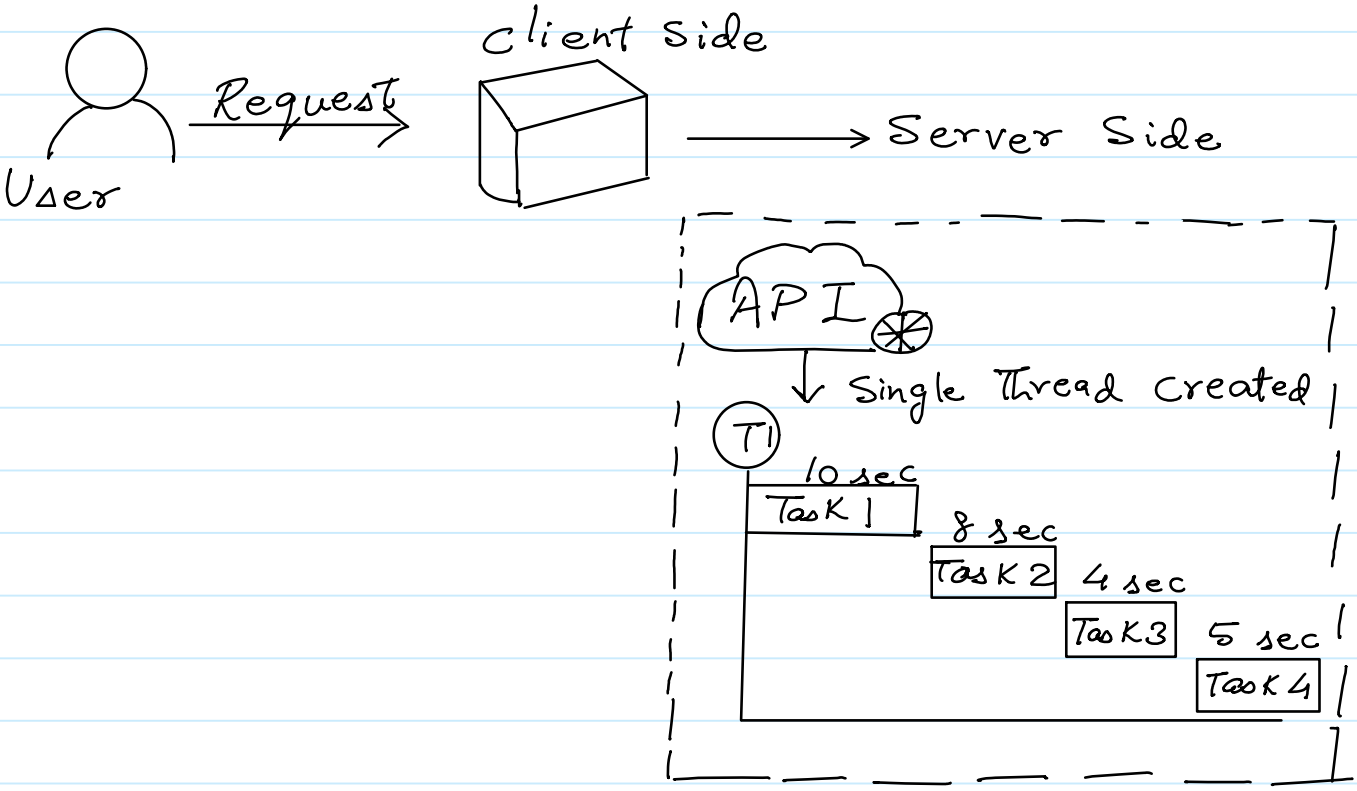
Main Features of Node.js

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

⇒

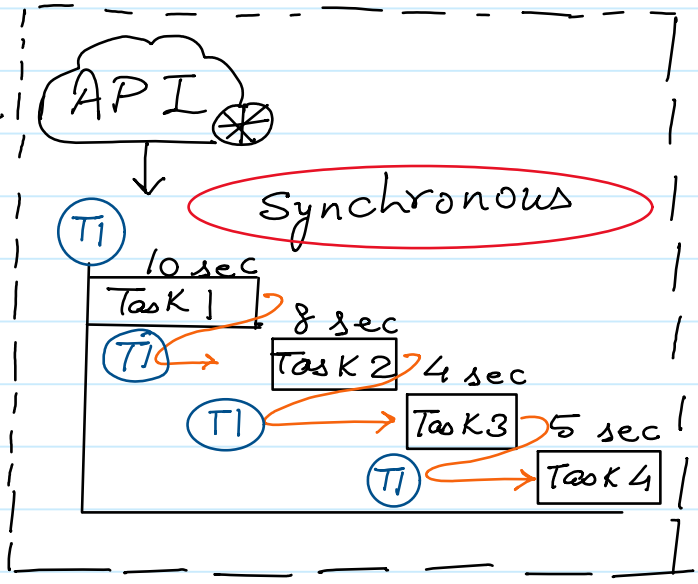


Q. What is Single Treaded Programming ?



Q. What is Synchronous Programming ?

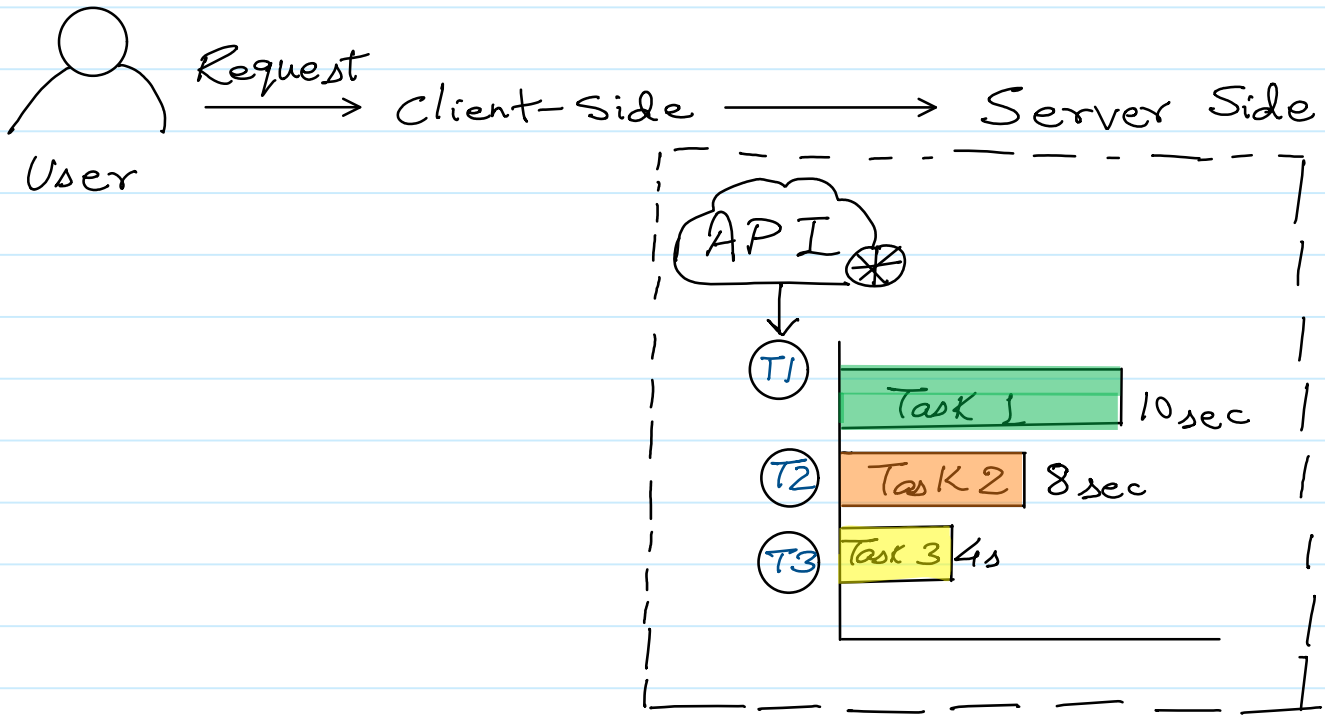
⇒ In a synchronous program each task is performed one after the other, and the program waits for each operation to complete before moving on to the next one.



⇒ 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 ?

⇒



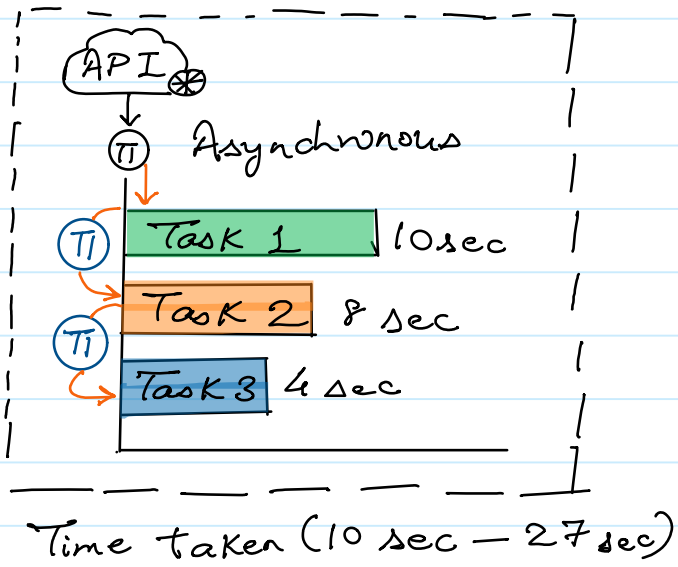
Q. What is Asynchronous Programming ?

⇒ 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

14 December 2025 08:56 PM

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.

Q. What are the main features & advantages of Nodejs ?

	Features	Advantages
1.	Asynchronous	Enables handling **multiple concurrent requests & non blocking execution** of thread.
2.	V8 JS Engine	Built on the V8 JS engine from google Chrome, Node.js **execute code fast.**
3.	Event-Driven Architecture	**Efficient handling** events Great for real time applications like chat applications, gaming applications (using web sockets) where bidirectional communication is required.

4. Cross Platform : Supports deployment on various operating systems, enhancing flexibility

5. Javascript : Coding in JS language therefore no need to learn a new language

: Suitable for building scalable applications that can handle increased loads

Q. What are the disadvantages of node ? When to use and when not to use Node ?

When to use Node.js ?

☆ Ideal for **real-time** applications like chat applications, online gaming, and collaborative tools due to its event-driven Architecture

☆ Excellent for building **lightweight and scalable RESTful APIs** that handle a large number of Concurrent connections.

☆ Well-suited for building **microservices-based architectures**, enabling modular and scalable systems.

When not to use Node.js ?

☆ **CPU-Intensive Tasks :** Avoid for applications that involve heavy CPU processing (Image/Video Processing, Data Encryption/Decryption) as Node.js may not provide optimal performance in such scenarios because it single threaded and for heavy Computation multi-threaded is better.

16 December 2025 08:36 PM

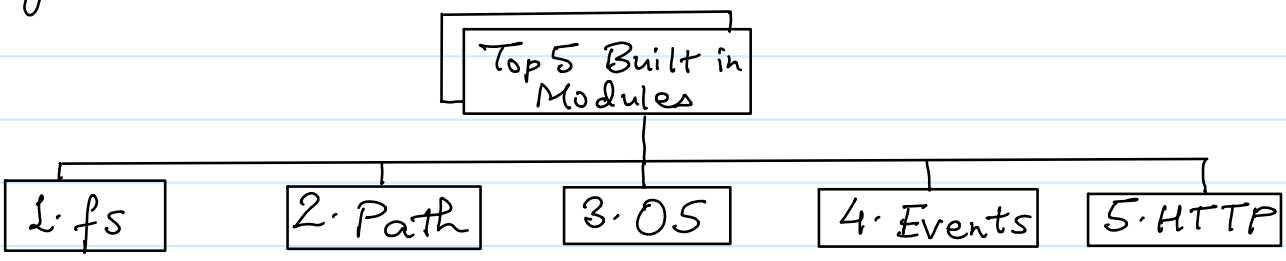
Q. What is the role of package.json file in Node?

⇒ The package.json file contains project metadata (information about the project). For example, project name, version, description, author, license, scripts to run, build, the project etc.

Q. What are Modules in Node? What is the difference between a function & Module?

⇒ A module contains a specific functionality that can be easily reused within a Node.js application.
⇒ Ideally in Node.js, a javascript file can be treated as a Module
⇒ A module is a broader concept that encapsulates functionality, while a function is a specific set of instructions within that module
⇒ Modules can contain multiple functions and variables.

Q. What are the Top 5 built in modules commonly used in node projects?



20 December 2025 06:16 PM

Q. Explain the role of fs module? Name some functions of it ?

```
// fs-example.js
Const fs = require("fs");
// Reading the Contents of a file asynchronously
fs.readFile("fs.txt", "utf8", (err, data) => {
    if (err) {
        return;
    }
    Console.log("File Contents:", data);
});
// writing to a file asynchronously
Const contentToWrite = "some Content";
fs.writeFile("fs.txt", contentToWrite, "utf8"
    , (err) => {
    if (err){
        return;
    }
    Console.log("write operation Completed");
});
```

□ fs (File System) module in Node provides a set of methods for interacting with the file system

Q Explain the role of fs module? Name some functions of it ?

⇒ 7 Main functions of fs module are :

1. fs.readFile() Read the content of the file specified.

2. fs.writeFile() Writes data to the specified file, creating the file if it does not exist.

3. fs.appendFile() Append data to a file. If the file does not exist, it is created.

4. fs.unlink() Delete the specified file.

5. fs.readdir() Read the content of a directory

6. fs.mkdir() Create a new directory.

7. fs.rmdir() Removes the specified directory.

Q. Explain the role of path module? Name some functions of it?

- Path module provides utilities for joining, resolving, parsing, formatting, normalizing, and manipulating paths.

```
// path-example.js

const path = require("path");
// Joining path segments
const fullPath = path.join('/docs', 'file.txt');
console.log(fullPath); // /docs/file.txt
// Parsing Path
const parsedPath = path.parse('/docs/file.txt')
console.log(parsedPath);
/* output: { root: '/', dir: '/docs', base: 'file.txt',
           ext: '.txt', name: 'file' } */
```

20 December 2025 06:50 PM

```
// Getting the directory name of a path

Const dirName = path.dirname('/folder/file.txt');

// Getting the file extension of a path

Const fileExtension = path.extname('/folder/file.txt');

// Parsing a path into an object with its components

Const pathObj = path.parse('/folder/file.txt');
```

Q. Explain the role of OS module? Name some functions of it

- The OS module in Nodejs provides a set of methods for interacting with the operating system.
- Operating system can be used by developers for building cross-platform applications or performing system-related tasks in Nodejs applications.

```
const os = require('os');
// 1. Fetch and print the operating system Information
console.log(os.type());
// 2. Fetch current user information
console.log(os.userInfo());
// 3. Fetch and print system uptime in hours
console.log(os.uptime() / 3600);
// 4. Fetch and print total system memory in GB
console.log(os.totalmem() / (1024 ** 3));
// 5. Fetch and print free system memory in GB
console.log(os.freemem() / (1024 ** 3));
// 6. Fetch and print CPU core information
console.log(os.cpus());
// 7. Fetch and print system architecture
console.log(os.arch());
// 8. Fetch and print system hostname
console.log(os.hostname());
// 9. Fetch and print system platform
console.log(os.platform());
// 10. Fetch and print home directory of the current user
console.log(os.homedir());
```

Outputs

20 December 2025 07:06 PM

```
PS C:\Users\Rayan Ahmad\Desktop\interview-notes> node "c:\Users\Rayan
Ahmad\Desktop\interview-notes\pract.js"
// Windows_NT
/* {
  uid: -1,
  gid: -1,
  username: 'Rayan Ahmad',
  homedir: 'C:\\Users\\Rayan Ahmad',
  shell: null
} */
// 0.8631855555555555
// 7.325794219970703
// 0.9815559387207031
/* [
  {
    model: 'AMD Ryzen 5 5500U with Radeon Graphics         ',
    speed: 2096,
    times: { user: 184562, nice: 0, sys: 279812, idle: 2642531, irq: 40859 }
  },
.........
    {
    model: 'AMD Ryzen 5 5500U with Radeon Graphics         ',
    speed: 2096,
    times: { user: 246125, nice: 0, sys: 141421, idle: 2719281, irq: 6156 }
  }
]
*/
// x64
// LAPTOP-DN3IF1V1
// win32
// C:\Users\Rayan Ahmad
// PS C:\Users\Rayan Ahmad\Desktop\interview-notes>
```

Q. Explain the role of events module? How to handle events in Node ?

```
// import events module
const EventEmitter = require("events");
// create an event emitter object
const eventEmitter = new EventEmitter();
// create an event handler
const eventHandler = () => {
  console.log("Event occurred: Custom event has been
triggered!");
};
// assign the event handler to an event
eventEmitter.on("customEvent", eventHandler);
// trigger the event
eventEmitter.emit("customEvent");
```

1. events module is used to handle events.

2. EventEmitter class of events module is used to register event listeners and emit events.

3. An event listener is a function that will be executed when a particular event occurs.

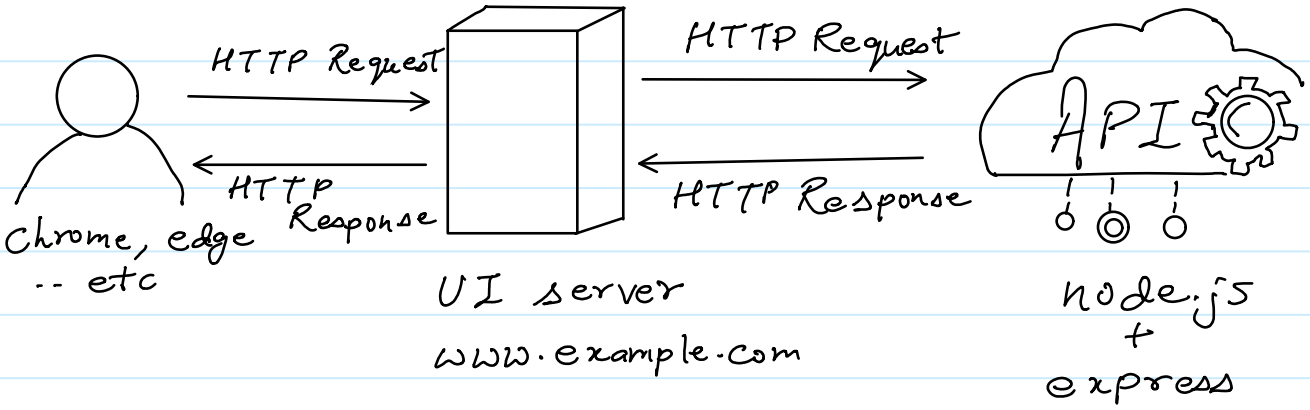
4. On() method is used to register event listeners.

Q. What are Event Arguments ?

```
// events arguments example
const eventHandlerWithArgs = (arg1, arg2) => {
  console.log(`Event occurred with arguments: ${arg1}, ${arg2}`);
}
eventEmitter.on("eventWithArgs", eventHandlerWithArgs);
eventEmitter.emit("eventWithArgs", "Argument 1", "Argument 2");
```

Q. What is the role of http module in node?

- The HTTP module can **create an HTTP server** that listens to server ports and gives a response back to the client.



Q. What is the role of createServer() method of http module?

- The createServer() method of the http module in Node.js is used to **create an HTTP server.**

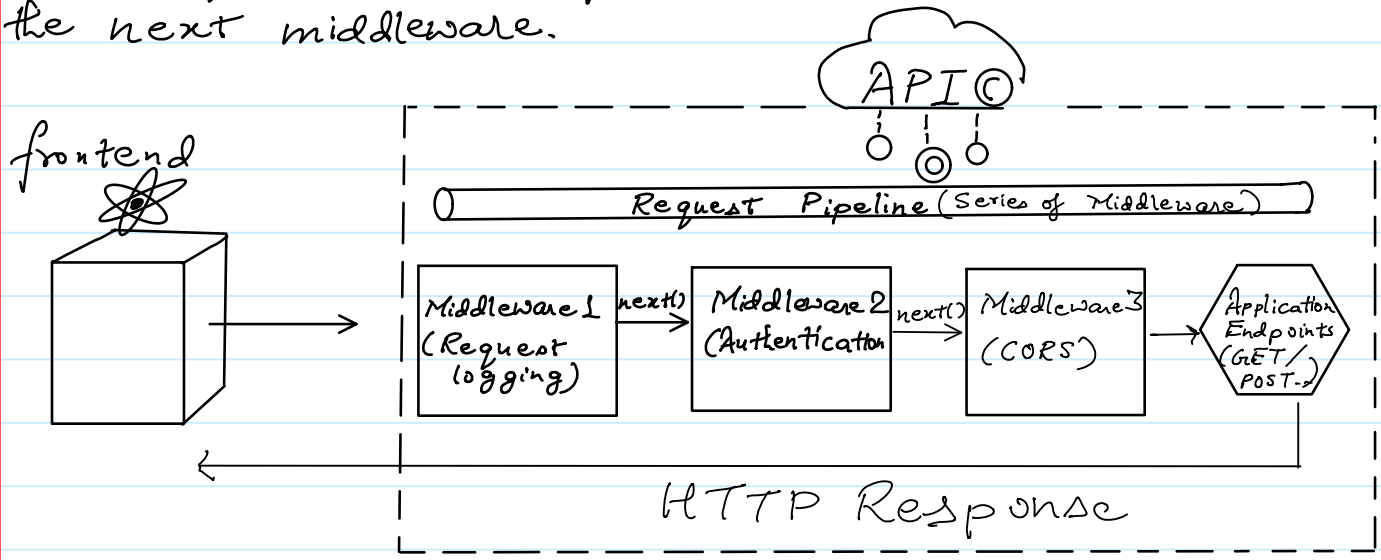
```
// 1. import the http module to create an HTTP server
const http = require("http");
// 2. create an HTTP server
const server = http.createServer((req, res) => {
  // 3. set the response HTTP header with HTTP status and content type
  res.writeHead(200, { "Content-Type": "text/plain" });
  //   Handle incoming http requests
  // 4. send the response body "Hello, World!"
  res.end("Hello, World!\n");
});
// 5. define the port number and hostname
const PORT = 3000;
const HOSTNAME = "localhost";
// 6. make the server listen on the specified port and hostname
server.listen(PORT, HOSTNAME, () => {
  console.log(`Server running at http://${HOSTNAME}:${PORT}/`);
});
// 7. handle server errors
server.on("error", (err) => {
  console.error(`Server error: ${err}`);
});
```

Q. What are the advantages of using Express.js with Node?

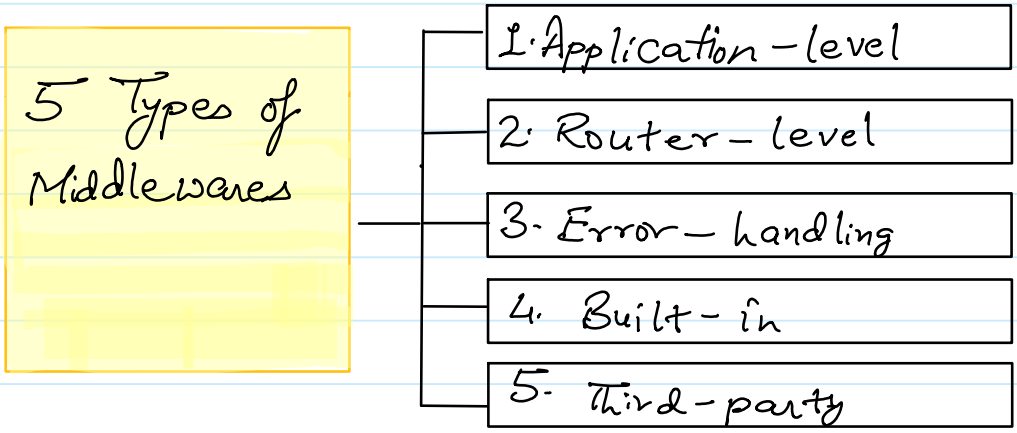
1. Simplified web development
2. Middleware support
3. Flexible routing system {GET, POST, PUT, DELETE}
4. Template Engine Integration. {generate Dynamic HTMLS}

Q. What is Middleware in Express.js and when to use them

- A middleware in Express.js is a function that handles HTTP requests, perform operations, and passes control to the next middleware.

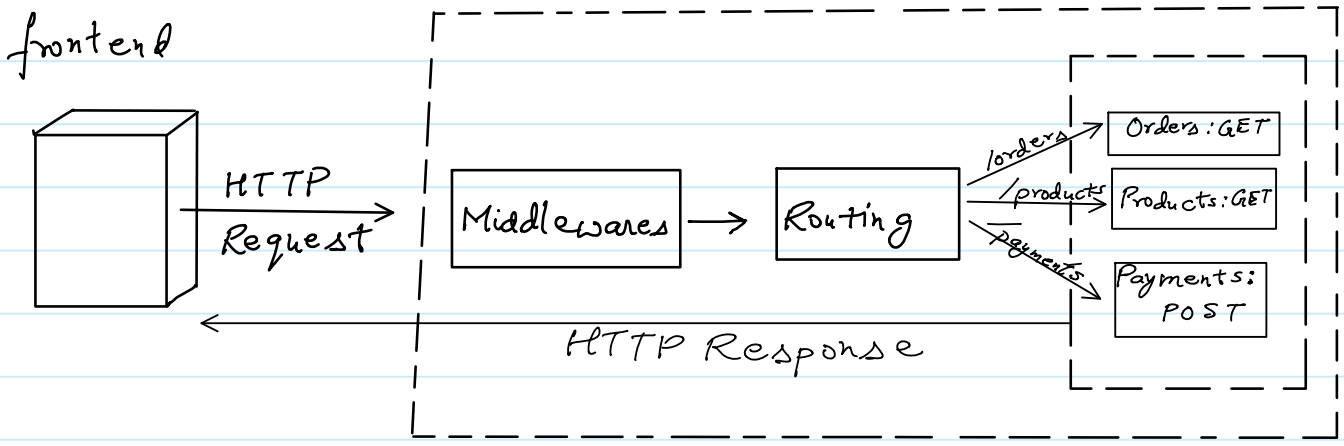


Q. What are the Types of Middleware's in Express.js?



Q. What is Routing in Express.js ?

□ Routing is the process of directing incoming HTTP request to the appropriate handler functions based on the request's method (eg. GET, POST) and the URL path.



Q. What are Top 5 REST guidelines and the advantages of them ?

1. Seperation of Client & Server → The implementation of the client and server must be done independently.
2. Stateless → The server will not store anything about the latest HTTP request the client made.
3. Uniform Interface → Identify the resources by URL
4. Cacheable → The API response should be cacheable to improve performance
5. Layered System → The system should follow layered pattern eg → Model - view - Controller (MVC)

Q. What is the difference between REST API and SOAP API ?

Feature	REST API	SOAP API
Architecture	REST is an architectural style	SOAP (simple object Access Protocol) is a protocol
Protocol	Uses HTTP or HTTPS	Can use various protocols (HTTP, SMTP, etc)
Message Format	Uses lightweight formats like JSON, XML.	Typically uses XML.
State	Stateless	Can be stateful or stateless
Error Handling	Relies on HTTP Status Codes	Defines its own fault mechanism
Performance	Generally lightweight and faster	Can be slower due to XML processing

Q. What are the types of authentication in Node.js ?

5 Types of Authentication

1. Basic (Username, Password)
2. API Key
3. Token-based (JWT)
4. Multi-factor
5. Certificate - based