What is Node.js?

Node.js is an open source server environment. It runs on the various platforms like (Windows, Linux, Unix, Mac OS X etc.). It allows to use JavaScript on the server. It is normally used to build the backend of the application and is highly scalable.

Why Node.js?

**Real-time Applications**: Node.js is particularly well-suited for building real-time applications like chat applications, collaboration tools, and streaming services.

**JSON and API Integration**: Since JSON is a common format for data interchange, Node.js, which uses JavaScript, naturally integrates well with JSON. This makes it particularly effective for building RESTful APIs and handling JSON data efficiently.

**High Performance**: Node.js is built on the V8 JavaScript engine from Google, which compiles JavaScript directly to machine code, making it extremely fast. Its single-threaded event loop model also helps in handling a large number of simultaneous connections with high throughput.

**JavaScript Everywhere**: Node.js allows developers to use JavaScript for both frontend and backend development. This unification of the programming language across the stack simplifies development, reduces context switching, and allows for code reuse between the client and server sides.

**Asynchronous and Event-Driven**: Node.js operates on a non-blocking, event-driven architecture, making it efficient and scalable for I/O-intensive tasks. This allows it to handle multiple connections simultaneously, making it ideal for real-time applications such as chat applications, online gaming, and live updates.

### [**What is the difference between Node.js and JavaScript?**](https://www.geeksforgeeks.org/difference-between-node-js-and-javascript)

| [Node.js](https://www.geeksforgeeks.org/node-js-introduction) | [JavaScript](https://www.geeksforgeeks.org/introduction-to-javascript) |
| --- | --- |
| Server-side runtime environment | Client-side scripting language |
| Allows running JavaScript code on server | Primarily used for web development |
| Built on Chrome’s V8 JavaScript engine | Runs in a web browser’s JavaScript engine |
| Enables building scalable network applications | Executes code within a browser environment |
| Provides access to file system and network resources | Limited to browser APIs and capabilities |
| Supports event-driven, non-blocking I/O operations | Executes in a single-threaded event loop |
| Used for building backend APIs, servers, and applications | Utilized for creating interactive web pages and client-side logic |

### **Is Node.js single-threaded?**

Yes,[Node.js is single-threaded](https://www.geeksforgeeks.org/why-node-js-is-a-single-threaded-language) by default. However, it utilizes event-driven architecture and non-blocking I/O operations to handle multiple concurrent requests efficiently, enabling scalability and high performance in applications.

### **What kind of API function is supported by Node.js?**

* **Synchronous:** These API functions are used for blocking code.
* **Asynchronous:** These API functions are used for non-blocking code.

### **What is the difference between Synchronous and Asynchronous functions?**

| **Feature** | **Synchronous Functions** | **Asynchronous Functions** |
| --- | --- | --- |
| Execution Blocking | Blocks the execution until the task completes. | Does not block the execution; allows other tasks to proceed concurrently. |
| Waiting for Completion | Executes tasks sequentially; each task must complete before the next one starts. | Initiates tasks and proceeds with other operations while waiting for completion. |
| Return Value | Returns the result immediately after completion. | Typically returns a promise, callback, or uses event handling to handle the result upon completion. |
| Error Handling | Errors can be easily caught with try-catch blocks. | Error handling is more complex and often involves callbacks, promises, or async/await syntax. |
| Usage Scenario | Suitable for simple, sequential tasks with predictable execution flow. | Ideal for I/O-bound operations, network requests, and tasks requiring parallel processing. |

### **What is a module in Node.js?**

In Node.js Application, a [Module](https://www.geeksforgeeks.org/what-are-modules-in-node-js) can be considered as a block of code that provide a simple or complex functionality that can communicate with external application. Modules can be organized in a single file or a collection of multiple files/folders. Modules are useful because of their reusability and ability to reduce the complexity of code into smaller pieces. Some examples of modules are. http, fs, os, path, etc.

### **What is npm and its advantages?**

npm (Node Package Manager) is the default package manager for Node.js. It allows developers to discover, share, and reuse code packages easily. Its advantages include dependency management, version control, centralized repository, and seamless integration with Node.js projects.

### **What is middleware?**

[Middleware](https://www.geeksforgeeks.org/middleware-in-express-js) is the function that works between the request and the response cycle. Middleware gets executed after the server receives the request and before the controller sends the response.

### **What do you mean by event loop in Node.js?**

The[event loop](https://www.geeksforgeeks.org/node-js-event-loop) in Node.js is a mechanism that allows it to handle multiple asynchronous tasks concurrently within a single thread. It continuously listens for events and executes associated callback functions.

### **What is control flow in Node.js?**

Control flow in Node.js refers to the sequence in which statements and functions are executed. It manages the order of execution, handling asynchronous operations, callbacks, and error handling to ensure smooth program flow.

### **What are the main disadvantages of Node.js?**

* **Single-threaded nature:** May not fully utilize multi-core CPUs, limiting performance.
* **NoSQL preference:** Relational databases like MySQL aren’t commonly used.
* **Rapid API changes:**Frequent updates can introduce instability and compatibility issues.

### **What is REPL in Node.js?**

[REPL](https://www.geeksforgeeks.org/node-js-repl-read-eval-print-loop)in Node.js stands for Read, Evaluate, Print, and Loop. It is a computer environment similar to the shell which is useful for writing and debugging code as it executes the code in on go.

### [**How to import a module in Node.js?**](https://www.geeksforgeeks.org/import-and-export-in-node-js)

We use the require module to import the External libraries in Node.js. The result returned by require() is stored in a variable which is used to invoke the functions using the dot notation.

### **What is package.json in Node.js?**

[package.json](https://www.geeksforgeeks.org/node-js-package-json) in Node.js is a metadata file that contains project-specific information such as dependencies, scripts, version, author details, and other configuration settings required for managing and building the project.

### **What is the most popular Node.js framework used these days?**

The most famous Node.js framework used is Express.js as it is highly scalable, efficient, and requires very few lines of code to create an application.

### [**What is web socket?**](https://www.geeksforgeeks.org/web-socket-in-node-js)

Web Socket is a protocol that provides full-duplex (multiway) communication i.e. allows communication in both directions simultaneously. It is a modern web technology in which there is a continuous connection between the user’s browser (client) and the server. In this type of communication, between the web server and the web browser, both of them can send messages to each other at any point in time.

### [**What is a cluster in Node.js?**](https://www.geeksforgeeks.org/how-does-the-cluster-module-work)

Due to a single thread in node.js, it handles memory more efficiently because there are no multiple threads due to which no thread management is needed. Now, to handle workload efficiently and to take advantage of computer multi-core systems, cluster modules are created that provide us the way to make child processes that run simultaneously with a single parent process.

### [**What is body-parser in Node.js?**](https://www.geeksforgeeks.org/body-parser-middleware-in-node-js)

Body-parser is the Node.js body-parsing middleware. It is responsible for parsing the incoming request bodies in a middleware before you handle it. It is an NPM module that processes data sent in HTTP requests.

### **What is callback hell?**

[Callback hell](https://www.geeksforgeeks.org/what-is-callback-hell-in-node-js) is an issue caused due to a nested callback. This causes the code to look like a pyramid and makes it unable to read To overcome this situation we use promises.

### [**What are the three methods to avoid callback hell?**](https://www.geeksforgeeks.org/how-to-avoid-callback-hell-in-node-js)

* Using async/await()
* Using promises
* Using generators

### **What is fork in Node.js?**

Fork is a method in Node.js that is used to create child processes. It helps to handle the increasing workload. It creates a new instance of the engine which enables multiple processes to run the code.

### **What is the difference between setTimeout() and setImmediate() method?**

The setImmediate function is used to execute a particular script immediately whereas the setTimeout function is used to hold a function and execute it after a specified period of time.

### [**Explain the use of timers module in Node.js**](https://www.geeksforgeeks.org/node-js-timers-module)

The Timers module in Node.js contains various functions that allow us to execute a block of code or a function after a set period of time. The Timers module is global, we do not need to use require() to import it.

It has the following methods:

* [setTimeout()](https://www.geeksforgeeks.org/what-is-the-purpose-of-settimeout-function-in-javascript) method
* [setImmediate()](https://www.geeksforgeeks.org/node-js-immediate-timer-class) method
* [setInterval()](https://www.geeksforgeeks.org/what-is-setinterval-in-javascript) method

### **What is event-driven programming in Node.js?**

[Event-driven programming](https://www.geeksforgeeks.org/explain-event-driven-programming-in-node-js) is used to synchronize the occurrence of multiple events and to make the program as simple as possible. The basic components of an Event-Driven Program are:

* A callback function ( called an event handler) is called when an event is triggered.
* An event loop that listens for event triggers and calls the corresponding event handler for that event.

# Node Child Process?

[Node](https://www.geeksforgeeks.org/introduction-to-node-js/) is a tool that uses JavaScript and has many useful parts to it. Normally, it does work with one thread at a time, which means it can handle tasks without waiting. However, when there’s a lot of work to be done, we use the child\_process module to create additional threads. These extra threads can talk to each other using a built-in messaging system.

Four different ways to create child process: -

* [spawn() method](https://www.geeksforgeeks.org/node-js-child-process/#spawn-method)
* [fork() method](https://www.geeksforgeeks.org/node-js-child-process/#fork-method)
* [exec() method](https://www.geeksforgeeks.org/node-js-child-process/#exec-method)
* [execFile() method](https://www.geeksforgeeks.org/node-js-child-process/#execfile-method)

# Difference Between PUT and PATCH Request?

**PUT**: Replaces the entire resource with the new data provided.

**PATCH**: Partially updates the resource with only the data provided, leaving the rest of the resource unchanged.