1. What is Node JS ?

* Open source , cross-paltform backend JS runtime environment that runs on V8 engine and executes JS code out side a web browser.

Advantages :

* + Scalability
  + Single programming language
  + Freedom to create cross platform App
  + High performance
  + Cross platform development
  + Extensibility
  + Caching
  + Improved App response Time
  + Simple to Learn
  + Large community Support
  + Web app development speed

Diagram

Description automatically generated

1. V8 ?

* Open source project created by Google
* Execute the Javascript code outside the browser
* Helps the node to handle the aspects of concurrency

1. Libuv?

* Allows to perform I/O operations whether its networking or file operations.
* All TCP level connectivity and file system operations are performed by this library

1. Library modules in Node JS

* http ,fs, crypto,path are very consistent API

1. Node JS components

* Node CLI
* NPM
* Package.json
* Node Modules
* Development tools and frameworks

1. Node Modules ?

* Npm
* Request
* http
* bower
* io
* fs

1. Event Loop in Node JS works in 6 phases

* Timers
* I/O Callbacks
* Preparation / idle phase
* I/O Polling
* setImmediate() callback execution
* close events

<https://heynode.com/tutorial/how-event-loop-works-nodejs/>

1. Node Package Manager

* Command line tool that installs updates and uninstall Nodejs packages in your application.
* It’s a online repository for open source Node Js pacakages.

1. Node js better than other frameworks

* NodeJs is a asynchronous and single threaded architecture
* NodeJs reacts to events and sends events to database
* Able to handle concurrent connections with high throughput on single thread.
* Lightweight and efficient
* Perfect for data intensive real time application that runs across distributed devices.

1. Control flow in Node JS

* Order in which statement or function is executed.
* IO operation is non- blocking in NodeJS not like python (wait for network operations )
* Instead it registers a callback to the event loop and pass control back to node
* New node process and delegates the CPU intensive task to process.
* Need persistent connection between server and client . realtime logging application ,web servers.
* Network and disk operations are async
* Lot of concurrent client application and each request needs fewer CPU go for Node JS
* Event loop as central dispatcher that routes the request to C++ and returns result to JS

1. Libuv ?

* C Library built specially for Node JS
* Full – featured event loop backed by epoll, kqueue , IOCP ,event ports
* Asynchronous TCP and UDP protocols
* Asynchronous DNS resolution ,fila and file system operation
* File system events
* Child process
* Thread pool
* Signal handling
* Threading and synchronization primitives

1. Node JS has 2 threads

* Event loop and k Workers
* Event loop is responsible for JS callback and non – blocking I/O and
* Worker executes task corresponding to C++
* Both threads work on no more than one activity at atime.

1. Promise.all , Promise.race,Promise.allSettled.
2. Timing features of NodeJS

* setTimeout()
  + Scehduled to execute after designated amount of milliseconds
  + setTimeout(callback , delay , args )
* setImmediate()
  + Used to execute at end of code execution
* setInterval()
  + used to call a function at specified interval

1. Advantage of using promises over call back?
2. Fork in Node JS?
3. Why nodeJs is single threaded
4. WAP for server node returns Hello World?
5. Types of API function in Node JS
6. What is REPL ?
7. List 2 arguments that async.queue tasks as input.
8. Purpose of node.exports?
9. Purpose of ESLint?
10. Callback hell?
11. Event loop in Node JS ?
12. Difference between process.nextTick() and setImmediate()
13. How NodeJs overcome problem of blocking I/O Operation
14. How to use async/ await in Node JS ?
15. Node JS Streams (Writable , Readable , Duplex, Transform)
16. Node JS Buffers
17. Node JS middle wares
18. Reactor Pattern in Node JS (Reactor / Handler)
19. Why to separate the Express app and Server
20. Why google uses the V8 engine ?
21. Exist codes on Node JS
22. Explain concept of Node JS
23. Event Emmiter in Node JS
24. Enhance Node JS performance using clustering
25. What is thread pool and which library is used to implement
26. What is WASI in nodeJS. Why it is needed – Web Assembly System Interface
27. How worker thread differs from clusters
28. Measure the duration of async operation
29. Measure the performance of aync operation
30. Default scopes of Node JS application
31. Use of underscore variable in REPL session
32. Node Object Methods

|  |  |  |
| --- | --- | --- |
|  | Method | Description |
| 1 | appendChild() | Appends a new child node to the end of the list of children of a node |
| 2 | cloneNode() | Clones a node |
| 3 | compareDocumentPosition() |  |
| 4 | getFeature(feature,version) |  |
| 5 | getUserData(key) |  |
| 6 | has Attributes() |  |
| 7 | hasChildNodes() |  |
| 8 | insertBefore() |  |
| 9 | isDefaultNamespace(URI) |  |
| 10 | isEqualNode |  |
| 11 | lookupNamespaceURI() |  |
| 12 | Normalize() |  |
| 13 | isSameNode() |  |
| 14 | setUserData(key,data,handler) |  |
| 15 | Replacechild() |  |
| 16 | removeChild() |  |
| 17 | lookupPrefix() |  |
| 18 |  |  |
|  |  |  |

1. How to check equality of 2 modes in Node JS