1. Differentiate between JavaScript and Node.js.

Features	JavaScript	Node.js
Type	Programming Language	Interpreter and environment for JavaScript
Utility	Used for any client-side activity f a web application	Used for accessing or performing any non-blocking operation of any operating system
Running Engine	Spider monkey (FireFox), JavaScript ,Core (Safari), V8 (Google Chrome),etc.	V8 (Google Chrome)

2. What Is Node.js?

Node.js is an extremely powerful framework developed on Chrome's V8 JavaScript engine that compiles the JavaScript directly into the native machine code. It is a lightweight framework used for creating server-side web applications and extends JavaScript API to offer usual server-side functionalities. It is generally used for large-scale application development, especially for video streaming sites, single page application, and other web applications.

3. List down the major benefits of using Node.js?

Features	Description
Fast	Node.js is built on Google Chrome's V8 JavaScript Engine which makes its library very fast in code execution
Asynchronous	Node.js based server never waits for an API to return data thus making it asynchronous
Scalable	It is highly scalable because of its event mechanism which helps the server to respond in a non-blocking way
Open Source	Node.js has an extensive open source community which has contributed in producing some excellent modules to add additional capabilities to Node.js applications
No Buffering	Node.js applications simply output the data in chunks and never buffer any data

4. What is the difference between Angular and Node.js?

Angular	Node.js
1. It is an open source web application development framework	1. It is a cross-platform run-time environment for applications
2. It is written in TypeScript	2. It is written in C, C++ and JavaScript languages
3. Used for building single-page client-side web applications	3. Used for building fast and scalable server-side networking applications
4. Angular itself is a web application framework	4. Node.js has many different frameworks like Sails.js, Partial.js, and Express.js, etc.
5. Ideal for creating highly active and interactive web apps	5. Ideal for developing small size projects
6. Helpful in splitting an app into MVC components	6. Helpful in generating database queries
7. Suitable for developing real-time applications	7. Suitable in situations where something faster and more scalable is required

5. Why Node.js is single threaded?

Node.js uses a single threaded model in order to support async processing. With async processing, an application can perform better and is more scalable under web loads. Thus, Node.js makes use of a single-threaded model approach rather than typical thread-based implementation.

6. How do Node.js works?

Node.js is a virtual machine that uses JavaScript as its scripting language and runs on a v8 environment. It works on a single-threaded event loop and a non-blocking I/O which provides high rate as it can handle a higher number of concurrent requests. Also, by making use of the ‘HTTP’ module, Node.js can run on any stand-alone web server.

7. Where Node.js can be used?

Node.js can be used to develop:

- Real-Time Web Applications
- Network Applications
- Distributed Systems
- General Purpose Applications

8. How many types of API functions are there in Node.js?

There are two types of API functions in Node.js:

- Asynchronous, non-blocking functions
- Synchronous, blocking functions

9. What is the difference between Asynchronous and Non-blocking?

Asynchronous	Non-blocking
Asynchronous means not synchronous. Using these we can make asynchronous HTTP requests not wait for the server to respond. These functions continue to respond to the request for which it has already received the server response.	Non-blocking functions are used in regards with I/O operations. They immediately respond with whatever data is available and keeps on running as per the requests. In case, any answer couldn't be retrieved then the API returns immediately with an error.

10. What is package.json?

The package.json file in Node.js is the heart of the entire application. It is basically the manifest file that contains the metadata of the project where we define the properties of a package.

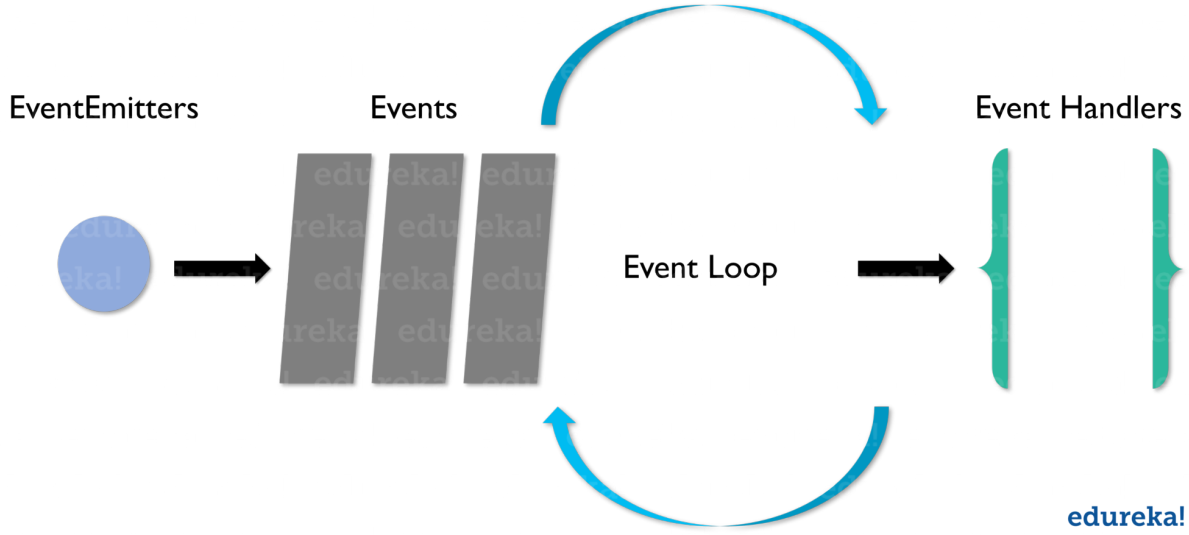
11. What do you understand by Event-driven programming?

Event-driven programming is an approach that heavily makes use of events for triggering various functions. An event can be anything like a mouse click, key

press, etc. When an event occurs, a call back function is executed that is already registered with the element. This approach mainly follows the publish-subscribe pattern. Because of event-driven programming, Node.js is faster when compared to other technologies.

12. What is an *Event loop* in Node.js and how does it work?

An event loop in Node.js handles all the asynchronous callbacks in an application. It is one of the most important aspects of Node.js and the reason behind Node.js have non-blocking I/O. Since Node.js is an event-driven language, you can easily attach a listener to an event and then when the event occurs the callback will be executed by the specific listener. Whenever functions like setTimeout, http.get, and fs.readFile are called, Node.js executed the event loop and then proceeds with the further code without waiting for the output. Once the entire operation is finished, Node.js receives the output and then executes the callback function. This is why all the callback functions are placed in a queue in a loop. Once the response is received, they are executed one by one.



13. Explain REPL in the context of Node.js.

REPL in Node.js stands for Read, Eval, Print, and Loop. It represents a computer environment such as a window console or Unix/Linux shell where any command

can be entered and then the system can respond with an output. Node.js comes bundled with a REPL environment by default. REPL can perform the below-listed tasks:

- **Read: Reads the user's input, parses it into JavaScript data-structure and then stores it in the memory.**
- **Eval: Receives and evaluates the data structure.**
- **Print: Prints the final result.**
- **Loop: Loops the provided command until *CTRL+C* is pressed twice.**

14. List down the tasks which should be done asynchronously using the event loop?

Below is the list of the tasks which must be done asynchronously using the event loop:

- **I/O operations**
- **Heavy computation**
- **Anything requiring blocking**

15. List down the steps using which "Control Flow" controls the function calls in Node.js?

1. **Control the order of execution**
2. **Collect data**
3. **Limit concurrency**
4. **Call the next step in the program**