**1. What is Node.js? Where can you use it?**

Node.js is a server-side scripting based on Google’s V8 JavaScript engine. It is used to build scalable programs especially web applications that are computationally simple but are frequently accessed.

2. Why use Node.js?

Node.js makes building scalable network programs easy. Some of its advantages include:

* It is generally fast
* It almost never blocks
* It offers a unified programming language and data type
* Everything is asynchronous
* It yields great concurrency

3. What are the features of Node.js?

Node.js is a single-threaded but highly scalable system that utilizes JavaScript as its scripting language. It uses asynchronous, event-driven I/O instead of separate processes or threads. It is able to achieve high output via single-threaded event loop and non-blocking I/O.

4. How else can the JavaScript code below be written using Node.Js to produce the same output?

console.log("first");  
setTimeout(function(){  
 console.log("second");  
},0);  
console.log("third");

Output:

first  
third  
second

In Node.js version 0.10 or higher, setImmediate(fn) will be used in place of setTimeout(fn,0) since it is faster. As such, the code can be written as follows:

console.log("first");  
setImmediate(function(){  
    console.log("second");  
});  
console.log("third");

6. Why is Node.js Single-threaded?

Node.js is single-threaded for async processing. By doing async processing on a single-thread under typical web loads, more performance and scalability can be achieved as opposed to the typical thread-based implementation.

7. Explain callback in Node.js.

A callback function is called at the completion of a given task. This allows other code to be run in the meantime and prevents any blocking.  Being an asynchronous platform, Node.js heavily relies on callback. All APIs of Node are written to support callbacks.

8. What is callback hell in Node.js?

Callback hell is the result of heavily nested callbacks that make the code not only unreadable but also difficult to maintain. For example:

query("SELECT clientId FROM clients WHERE clientName='picanteverde';", function(id){  
  query("SELECT \* FROM transactions WHERE clientId=" + id, function(transactions){  
 transactions.each(function(transac){  
      query("UPDATE transactions SET value = " + (transac.value\*0.1) + " WHERE id=" + transac.id,function(error){  
 if(!error){  
console.log("success!!");  
}else{  
console.log("error");  
}  
 });  
});  
});  
});

10. Explain the role of REPL in Node.js.

As the name suggests, REPL (Read Eval print Loop) performs the tasks of – Read, Evaluate, Print and Loop. The REPL in Node.js is used to execute ad-hoc Javascript statements. The REPL shell allows entry to javascript directly into a shell prompt and evaluates the results. For the purpose of testing, debugging, or experimenting, REPL is very critical.

11. Name the types of API functions in Node.js.

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

* Blocking functions - In a blocking operation, all other code is blocked from executing until an I/O event that is being waited on occurs. Blocking functions execute synchronously

Forexample:  
constfs=require('fs');  
const data = fs.readFileSync('/file.md'); // blocks here until file is read  
console.log(data);  
// moreWork(); will run after console.log

The second line of code blocks the execution of additional JavaScript until the entire file is read. moreWork () will only be called after Console.log

* Non-blocking functions - In a non-blocking operation, multiple I/O calls can be performed without the execution of the program being halted.  Non-blocking functions execute asynchronously.
* For example:
* constfs=require('fs');  
  fs.readFile('/file.md',(err,data)=>{  
   if(err)throwerr;  
   console.log(data);  
  });  
  // moreWork(); will run before console.log
* Since fs.readFile () is non-blocking, moreWork () does not have to wait for the file read to complete before being called. This allows for higher throughput.

12. Which is the first argument typically passed to a Node.js callback handler?

Typically, the first argument to any callback handler is an optional error object. The argument is null or undefined if there is no error.

Error handling by a typical callback handler could be as follows:

Function

callback(err,results){  
    // usually we'll check for the error before handling results  
 if(err){  
// handle error somehow and return  
 }  
    // no error, perform standard callback handling  
}

13. What are the functionalities of NPM in Node.js?

NPM (Node package Manager) provides two functionalities:

* Online repository for Node.js packages
* Command line utility for installing packages, version management and dependency management of Node.js packages

14. What is the difference between Node.js and Ajax?

Node.js and Ajax (Asynchronous JavaScript and XML) are the advanced implementation of JavaScript. They all serve completely different purposes.

Ajax is primarily designed for dynamically updating a particular section of a page’s content, without having to update the entire page.

Node.js is used for developing client-server applications.

15. Explain chaining in Node.js.

Chaining is a mechanism whereby the output of one stream is connected to another stream creating a chain of multiple stream operations.

16. What are “streams” in Node.js? Explain the different types of streams present in Node.js.

Streams are objects that allow reading of data from the source and writing of data to the destination as a continuous process.

There are four types of streams.

* to facilitate the reading operation
* to facilitate the writing operation
* to facilitate both read and write operations
* is a form of Duplex stream that performs computations based on the available input

17. What are exit codes in Node.js? List some exit codes.

Exit codes are specific codes that are used to end a “process” (a global object used to represent a node process).

Examples of exit codes include:

* Unused
* Uncaught Fatal Exception
* Fatal Error
* Non-function Internal Exception Handler
* Internal Exception handler Run-Time Failure
* Internal JavaScript Evaluation Failure

18. What are Globals in Node.js?

Three keywords in Node.js constitute as Globals. These are:

* Global – it represents the Global namespace object and acts as a container for all otherobjects.
* Process – It is one of the global objects but can turn a synchronous function into an async callback. It can be accessed from anywhere in the code and it primarily gives back information about the application or the environment.
* Buffer – it is a class in Node.js to handle binary data.
* 19. What is the difference between AngularJS and Node.js?
* Angular.JS is a web application development framework while Node.js is a runtime system.
* 20. Why is consistent style important and what tools can be used to assure it?
* Consistent style helps team members modify projects easily without having to get used to a new style every time. Tools that can help include Standard and ESLint.