1. What do you understand by a test pyramid?

A test pyramid basically is a diagram that describes the ratio of how many unit tests, integration tests, and end-to-end test are required to be written for the successful development of the project.

2. What is an error-first callback in Node.js?

Error-first callbacks in Node.js are used to pass errors and data. The very first parameter you need to pass to these functions has to be an error object while the other parameters represent the associated data. Thus you can pass the error object for checking if anything is wrong and handle it. In case there is no issue, you can just go ahead and with the subsequent arguments.

```
var myPost = new Post({title: 'edureka'});
myPost.save(function(err,myInstance){
if(err){
//handle error and return
}
//go ahead with `myInstance`
});
```

3. Explain the purpose of module.exports?

A module in Node.js is used to encapsulate all the related codes into a single unit of code which can be interpreted by shifting all related functions into a single file. For example, suppose you have a file called greet.js that contains the two functions as shown below:

```
module.exports = {
greetInHindi: function(){
return "NAMASTE";
},
greetInKorean: function(){
return "ANNYEONGHASEYO";
}};
```

As you can see module.exports provide two functions which can be imported in another file using below code:

```
var eduGreets = require ("./greet.js");
eduGreets.greetInHindi() //NAMASTE
eduGreets.greetInKorean() //ANNYEONGHASEYO
```

4. What do you understand by Reactor Pattern in Node.js?

Reactor Pattern in Node.js is basically a concept of non-blocking I/O operations. This pattern provides a handler that is associated with each I/O operation and as soon as an I/O request is generated, it is then submitted to a *demultiplexer*. This demultiplexer is a notification interface which is capable of handling concurrency in non-blocking I/O mode. It also helps in collecting each and every request in the form of an event and then place each event in a queue. Thus resulting in the generation of the Event Queue. Simultaneously, we have our event loop which iterates the events present in the Event Queue.

5. What's the difference between 'front-end' and 'back-end' development?

Front-end Development	Back-end Development
1. Uses mark up and web languages like HTML, CSS, JavaScript	1. Uses programming and scripting languages like Python, Ruby, Perl, etc.
2. Based on asynchronous requests and AJAX	2. Based on Server Architecture
3. Better Accessibility	3. Enhanced Security
4. Used for SEO	4. Used for Backup

6. What are LTS releases of Node.js?

LTS stands Long Term Support version of Node.js that receives all the critical bug fixes along with security updates and performance improvements. These versions are supported for at least 18 months and mainly focus on stability and security. The modifications done to the LTS versions are restricted to the bug fixes, security upgrade, npm, and documentation updates, performance improvement, etc.

7. List down the major security implementations within Node.js?

Major security implementations in Node.js are:

1. Authentications
2. Error Handling

8. What do you understand by callback hell?

Callback Hell is also known as the Pyramid of Doom. It is a pattern caused by intensively nested callbacks which are unreadable and unwieldy. It typically contains multiple nested callback functions which in turn make the code hard to read and debug. It is caused by improper implementation of the asynchronous logic.

```
async_A(function(){
async_B(function(){
async_C(function(){
async_D(function(){
....
});
});
});
});
```

9. Explain libuv.

Libuv is a multi-platform support library of Node.js which majorly is used for asynchronous I/O. It was primarily developed for Node.js, with time it is popularly practiced with other systems like as Luvit, pyuv, Julia, etc. Libuv is basically an abstraction around libev/ IOCP depending on the platform, providing users an API based on libev. A few of the important features of libuv are:

- **Full-featured event loop backed**
- **File system events**
- **Asynchronous file & file system operations**
- **Asynchronous TCP & UDP sockets**
- **Child processes**

10. Explain the concept of middleware in Node.js?

In general, middleware is a function receives the Request and Response objects. In other words, in an application's request-response cycle these functions have access to various request & response objects along with the next function of the cycle. The next function of middleware is represented with the help of a variable, usually named next. Most commonly performed tasks by the middleware functions are:

- **Execute any type of code**

- Update or modify the request and the response objects
- Finish the request-response cycle
- Invoke the next middleware in the stack

11. Explain the concept of URL module.

The URL module of Node.js provides various utilities for URL resolution and parsing. It is a built-in module that helps in splitting up the web address into a readable format:

```
var url = require('url');
```

For example:

```
var url = require('url');
var adrs =
'http://localhost:8082/default.htm?year=2019&amp;amp;month=april';
var q = url.parse(adr, true);
console.log(q.host); //returns 'localhost:8082'
console.log(q.pathname); //returns '/default.htm'
console.log(q.search); //returns '?year=2019 and month=april'
var qdata = q.query; //returns an object: { year: 2019, month: 'april' }
console.log(qdata.month); //returns 'april'
```

12. What do you understand by ESLint?

ESLint is an open source project initially developed by Nicholas C. Zakas in 2013 which aims to provide a linting utility for JavaScript through a plug. Linters in Node.js are good tools for searching certain bug classes, especially those which are related to the variable scope.

13. For Node.js, why Google uses V8 engine?

Google uses V8 as it is a Chrome runtime engine that converts JavaScript code into native machine code. This, in turn, speeds up the application execution and response process and give you a fast running application.

14. Explain the working of the control flow function.

In Node.js, the control flow function is basically the code that is executed between the asynchronous function calls. Below are the steps that must be followed for executing it:

1. Firstly, the order of execution must be controlled.

2. Then, the required data need to be collected.
3. Next, the concurrency must be limited.
4. Once done, the next step of the program has to be invoked.

15. List down the two arguments that async.queue takes as input?

Below are the two arguments that async.queue takes as input:

1. Task Function
2. Concurrency Value

16. Differentiate between spawn() and fork() methods in Node.js?

In Node.js, the spawn() is used to launch a new process with the provided set of commands. This method doesn't create a new V8 instance and just one copy of the node module is active on the processor. When your child process returns a large amount of data to the Node you can invoke this method.

Syntax:

```
child_process.spawn(command[, args][, options])
```

Whereas, the fork() in Node.js is a special instance of spawn() that executes a new instance of the V8 engine. This method simply means that multiple workers are running on a single Node code base for various task.

Syntax:

```
child_process.fork(modulePath[, args][, options])
```

17. What do you understand by global objects in Node.js?

In Node.js, Globals are the objects which are global in nature and are available in all the modules of the application. You can use these objects directly in your application, rather than having to include them explicitly. The global objects can be modules, functions, strings, object, etc. Moreover, some of these objects can be in the module scope instead of global scope.

18. Explain the concept of stub in Node.js.

In Node.js, stubs are basically the programs or functions that are used for stimulating the module or component behavior. During any test cases, stubs provide the canned answers of the functions.

19. How assert works in Node.js?

In Node.js, assert is used to write tests. It only provides feedback only when any of the running test cases fails. This module gives you a set of assertion tests which are then used for testing invariants. It is basically used internally by

Node.js but using require('assert') code, it can be used in other applications as well.

```
var assert = require('assert');
function mul(a, b) {
return a * b;
}
var result = mul(1,2);
assert( result === 2, 'one multiplied by two is two');
```

20. Define the concept of the test pyramid. Explain the process to implement them in terms of HTTP APIs.

The test pyramid is basically a concept that is developed by Mike Cohn. According to this, you should have a higher number of low-level unit tests as compared to high-level end-to-end tests that running through a GUI.

In terms of HTTP APIs it may be defined as:

- A higher number of low-level unit tests for each model
- Lesser integration tests to test model interactions
- Lesser acceptance tests for testing actual HTTP endpoints

21. Explain the purpose of ExpressJS package?

Express.js is a framework built on top of Node.js that facilitates the management of the flow of data between server and routes in the server-side applications. It is a lightweight and flexible framework that provides a wide range of features required for the web as well as mobile application development. Express.js is developed on the middleware module of Node.js called *connect*. The connect module further makes use of http module to communicate with Node.js. Thus, if you are working with any of the connect based middleware modules, then you can easily integrate with Express.js.

22. Differentiate between process.nextTick() and setImmediate()?

In Node.js, process.nextTick() and setImmediate(), both are functions of the Timers module which help in executing the code after a predefined period of time. But these functions differ in their execution. The process.nextTick function waits for the execution of action till the next pass around in the event loop or once the event loop is completed only then it will invoke the callback function. On the other hand, setImmediate() is used to execute a callback method on the next cycle of the event loop which eventually returns it to the event loop in order to execute the I/O operations.

23. Explain the usage of a buffer class in Node.js?

Buffer class in Node.js is used for storing the raw data in a similar manner of an array of integers. But it corresponds to a raw memory allocation that is located outside the V8 heap. It is a global class that is easily accessible can be accessed in an application without importing a buffer module. Buffer class is used because pure JavaScript is not compatible with binary data. So, when dealing with TCP streams or the file system, it's necessary to handle octet streams.

24. How does Node.js handle the child threads?

In general, Node.js is a single threaded process and doesn't expose the child threads or thread management methods. But you can still make use of the child threads using spawn() for some specific asynchronous I/O tasks which execute in the background and don't usually execute any JS code or hinder with the main event loop in the application. If you still want to use the threading concept in your application you have to include a module called ChildProcess explicitly.

25. Explain stream in Node.js along with its various types.

Streams in Node.js are the collection of data similar to arrays and strings. They are objects using which you can read data from a source or write data to a destination in a continuous manner. It might not be available at once and need not to have fit in the memory. These streams are especially useful for reading and processing a large set of data. In Node.js, there are four fundamental types of streams:

1. ***Readable :*** **Used for reading large chunks of data from the source.**
2. ***Writeable :*** **Use for writing large chunks of data to the destination.**
3. ***Duplex :*** **Used for both the functions; read and write.**
4. ***Transform :*** **It is a duplex stream that is used for modifying the data.**