Question 2: What is Node.js?

Answer

Node.js is a server-side JavaScript platform which is built on Google Chrome’s JavaScript V8 engine. It is an open source and cross platform application to develop server side and networking applications. Anyone can develop the Node.js application by writing code in JavaScript and it can be run on Microsoft Windows, Linux, or OS X.

Node.js is not a new language and also it is not just a framework built on JavaScript. It is built on Chrome's JavaScript Runtime, so the code is written and executed very similarly to the browser.

Features of Node.js

Following are some important features of Node.js.

Fast in Code execution

It is very fast in code execution.

Highly scalable

Node.js uses a single thread model for event looping. Events respond to the server without blocking other operations. This makes Node.js highly scalable. Traditional servers create limited threads to handle requests and Node.js creates a single thread that provides service to much larger numbers of requests.

Event Driven and Asynchronous

All API of Node.js are asynchronous. It means that the server is moving the next API call without waiting for returning data of previous requests.

No Buffering

Node.js never buffers any data.

Many of us are confused about Node.js. It is not a web server like Apache. Node.js provides a new way to execute our code. It is JavaScript runtime. Node.js provides the facility to create an HTTP server and we can host our application on the same.

==>

NodeJS is javascript runtime environment that is built on chrome to enable lightweight and effective performance of server-side and networking applications in javascript.

Question 3: What are the key features of Node.js?

Answer

Here are the following key features of Node.js:

Real time Data intensive.

Example - Multiplayer Games, Stock Trading, Chat App etc.

Highly scalable servers for Web Applications.

Example - E-Commerce, Social Media, REST API etc.

Builds fast and scalable network Applications.

Example - Proxy Server, Backend web services, HTTP Web Server etc.

Node.js main features

Node.js version 6 focuses on performance improvements, increased reliability and better security for its 3.5 million users.

Event driven and Asynchronous.

Node.js is emerging as a universal platform used for Web Applications, IoT, mobile, Enterprise Application development and micro service architectures.

Full stack JavaScript developers can use it for front end, back end, mobile and IoT projects.

Node.js v6 comes equipped with v8 JavaScript engine 5.0, which has improved ECMAScript 2015 (ES6) support. Ninety-three percent of ES6 features are also now supported in the Node.js v6 release.

Very Fast.

Single Threaded but highly Scalable.

Question 4: What are the advantages of Node.js?

Answer

Node.js is an open source, cross-platform runtime environment for server-side and networking applications. Node.js applications are written in JavaScript and can be run within the Node.jsruntime on OS X, Microsoft Windows, Linux, FreeBSD, NonStop and IBM.

Advantages

Web development is done in a dynamic language (JavaScript) on a VM that is incredibly fast (V8). It is much faster than Ruby, Python, or Perl.

Ability to handle thousands of concurrent connections with minimal overhead on a single process.

JavaScript is perfect for event loops with first class function objects and closures. People already know how to use it this way having used it in the browser to respond to user-initiated events.

Many people already know JavaScript, even people who do not claim to be programmers. It is arguably the most popular programming language.

Using JavaScript on a web server as well as the browser reduces the impedance mismatch between the two programming environments that can communicate data structures via JSON that work the same on both sides of the equation. Duplicate form validation code can be shared between server and client and so on.

Question 5: Explain REPL in Node.js.

Answer

The REPL stands for Read Eval Print Loop, which is a simple program that accepts the commands, evaluates the commands, and prints their results. It represents a computer environment like Unix/Linux shell or a window console in which we can enter the command and the system responds with output. REPL performs the following tasks:

READ

Read the input from user, parse it into JavaScript data structure and store it in memory.

EVAL

Execute the data structure.

RINT

Print the result

LOOP

Loop the command until user presses Ctrl+C two time.

==>

REPL - Read, Eval, Print, Loop.

It is an environment to input commands to perform REPL tasks. Each of the task is associated with its respective operations.

Question 6: What is Closure?

Answer

Closure is a function defined within another scope that has access to all the variables within the outer scope. A Closure allows us a free environment for the outer function to access the inner functions and inner variables without any scope restrictions.

Closure is the local variable for a function, kept alive after the function has returned; or it's a stack-frame that is not deallocated when the function returns. A Closure is an inner function that has access to the outer function variable's scope chain. The Closure has 3 scope chains; the first one is to access to its own scope (variables defined within the curly brackets) and the second one is to access the outer function's variables and the third one is to access the global variables.

Points to remember about Closures,

Whenever you use a function inside another function, a Closure is used.

Closure in JavaScript is like keeping a copy of all the local variables.

A Closure is created just on entry to a function and the local variables are added to that Closure.

A new set of local variables are kept every time a function with a Closure is called.

Closure is the term for both the functions along with the variables that are captured.

Question 7: What is NPM? What is the need of NPM in Node.js?

Answer

NPM stands for Node.js Package Management. It comes with Node.js platform and allows us to install various packages for Node.js. This package manages and supports various commands to install and remove the modules. Here one important note is we require either package.json file or the node\_modules folder to install modules locally.

One of the best things about npm is that it locally stores all dependencies. For example, if module X uses module A version 1.0, and module Y uses module A version 1.5, then both X and Y will have their own local copies of module A.

==>

Node Package Manager. It is used to provide command line environment to install and manage NodeJS Packages, and NodeJS repositories that can be accessed at search.nodejs.org

==>

It works as an Online repository for node.js packages/modules which are present at <nodejs.org>.

It works as Command line utility to install packages, do version management and dependency management of Node.js packages.

Question 8: Explain event loop architecture of NodeJS.

Answer

When a request arrives, it is queued up into the EventLoop with a JavaScript Closure that includes the event (request, response) and corresponding callback.

The event is then given to the one worker thread if the job seems to take a long time to complete, that is from a C++ Thread Pool, handled by the libuv library.

Once the job is done, the corresponding callback is fired to return back the response to the Main thread.

The Event Loop returns the response to the client.

Question 9: Explain Node Package Manager.

Answer

The Node Package Manager is a Command line utility. It allows you to find, install, remove, publish and do a lot of other things related to Node Packaged Modules. The Node package Manager provides a link between the Node Package Registry and development environment. Node Package Manager lists some of the Command line options.

Option Description Example

Search Finds Module packages in the repository --> npm search express

Install Installs a package using a package.json --> npm install

Install -g Install a package in a globally accessible location --> npm installexpress –g

remove Removes a module --> npm remove express

pack Packages the module defined by the package.json file into a .tgz file npm pack

view Displays module details npm view express

publish Publish the moduledefined by a package.json file to the registry --> npm publish

unpublish Unpublishes a module you have published --> npm unpublish MyModule

owner Allows you to add, remove,and list owners of a package in the repository --> npm add neerajMyModule

npmrmneerajMyModule

npm Is MyModule

Question 10: Explain Process Object in Node.js.

Answer

The process object is the global object in Node. It can be accessed from anywhere; it is an instance of EventEmitter. Each Node.js has a set of built-in functionalities, accessible through the global process object. The process object provides the standard input/output (stdio) streams stdin, stdout and stderr (as in C/C++) as in the following:

stdin

A readable stream for reading input from the user.

stdout

A writable stream, either synchronously or asynchronously.

stderr

A blocking synchronous writable stream intended for error messages.

The stdout or non-blocking functions are: console.log, console.info, util.puts, util.print and Stderr. The blocking functons are: console.warn, console.error, util.debug and process.stdin (a readable stream for getting user input).

Properties of Process Object

process.title

By default a process title is NODE but you can change it.

Question 11: Explain some concepts in Node.js.

Answer

Node.js uses some of the following concepts,

Debugger

Statement inserting debugger; in the code of java script it will help to enable break point.

Modules

Node.js supports simple module loading system.

Console

Console module provides debugging feature that is similar in Java script console by browser.

Streaming

By various objects in node js it is abstract interface implemented.

Cluster

It allows us to create child process easily that shares server port.

DNS

DNS module contains functions.

Add-ons

It is a dynamically linked shared object.

Domain

It provides a way to handle multiple different IO operations as a single group.

Buffer

It is similar to array of integers but corresponds to fixed-sized.

Global

It is available for all modules.

Net

It provides asynchronous network wrapper.

Call backs

It is called when given task will be completed.

Error handling

It supports various types of categories error.

Crypto

It provides cryptographic functionality that includes a set of wrappers for Open SSL’s hash.

Question 12: What is Cluster Server Object in Node.JS?

Answer

A single instance of Node runs in a single thread. The cluster module allows you to create child processes that all share server ports. When you call server.listen() in a worker, it serializes the argument and passes the request to the master process. If the master process already has a listening server matching the worker's requirement, then it passes the handle to the server.

If it does not already have a listening server matching that requirement then it will create one and pass the handle to the worker.

Cluster Events

fork

online

listening

disconnect

Fork: the fork event is fired when the master attempts to fork a new child; it receives a worker object.

Online: the online event is fired when the master receives a notification that the child is fully bound; it receives a worker object.

Listening: when the worker performs an action that requires a listen() call such as HTTP server. The event emits two arguments, the first one is a worker object, and the address object contains the address, port and address type value of the connection.

Disconnect: called whenever a child is disconnected from the cluster; that can happen either through a process or exit event.

Question 13: Why do we use Node.js and how does Node.js work?

Answer

Node.js is specially designed for building fast, efficient, scalable network applications and uses an event-driven, non-blocking I/O model to maximize efficiency. The callback function is what Node calls a listener function and it is called by the server whenever a new request comes in.

How it works

The following describes how Node.js works,

You use your web browser to make a request for "about.html" on a Node.js web server.

The Node server accepts your request and calls a function to retrieve that file from disk.

While the Node server is waiting for the file to be retrieved, it services the next web request.

When the file is retrieved, there is a callback function inserted into the Node server's queue.

The Node server executes that function that in this case would render the HTML page and send it back to your browser.

Question 14: Explain Process.nextTick() function in Node.js.

Answer

The Process.nextTick() function typically runs before any other I/O events fire. As we know, every node application runs on a single thread, in other words only one task or event is processed by Node's event loop. In an event loop, a queue of callbacks are processed by Node on every Tick of the event loop. On the next loop around the event loop calls these callbacks.

The Process.nextTick() function defers the function until a completely new stack. You can call as many functions as you want to in the current stack. When the event loop is looking for a new event to execute the nextTick function will be in the queue and will execute an entire new stack. The Process.nextTick() function defers the execution of an action until the next pass around the event loop.

Question 16: What Is NodeJS Module?

Answer

A module encapsulates related code into a single unit of code. When creating a module, this can be interpreted as moving all related functions into a file. Node modules run in their own scope so that they do not conflict with other modules. Node relatedly provides global access to to help facilitate module interoperability. The primary 2 items that we are concerned with here are require and exports. You require other modules that you wish to use in your code and your module exports anything that should be exposed publicly. Let's see an example: save the below code as test.js,

Question 17: What is Cluster Worker Object in Node.js?

Answer

A Worker Object is a script that runs in a background process and does not block the other code processes attached to the page. A Worker Object contains all the public information and methods about a worker. The master can use "cluster.workers." If a worker throws an exception and does not handle the exception itself, the exception will create an event that you can listen for as an onerror event.

Worker.id

A unique id is provided to each worker. This unique id resides in the "id". While a worker is alive, on the basis of the unique id we can get the indexes from the cluster.workers.

Worker.process

Workers are created using "child\_process.fork()", the returned object from this function is stored as ".process". Child nodes are entire new instances of V8. Allow at least 30ms startup and 10mb for each new node. A ChildProcess class is not intended to be used directly; use the "spawn()" or "fork()"methods to create child process instances.

Methods of Worker Class

Worker

send(message,[sendHandle])

Worker.kill()

This function kills the workers.

Worker

Disconnect()

For more visit the following link,

Question 20: What are the main components of Node.js?

Answer

The main components of NodeJs are APIs, a V8 Engine and Libuv.

Libuv Library

Libuv is a multi-platform support library for asynchronous I/O. It was developed for Node.js using C and C++. But it's also used by Mozilla's Rust language, Luvit, Julia, pyuv and others.

This libuv library is the main part for I/O related operations like reading files and interacting with the OS.

You can check it out on GitHub for more information about the libuv library.

V8 Engine

From Google: “V8 is Google's open-source high-performance JavaScript engine, written in C++ and used in Google Chrome, the open source browser from Google. It implements ECMAScript as specified in ECMA-262, 3rd edition and runs on Windows XP and Vista, Mac OS X 10.5+, and Linux systems that use IA-32, ARM or MIPS processors. V8 can run standalone, or can be embedded into any C++ application.”

If you are interested in learning more about the V8 engine, please visit here.

APIs (NodeJS Core Libs)

The NodeJs APIs are nothing but functions to do something upon your request. By default the NodeJSapis are asynchronous in nature but still you can use NodeJS APIs synchronously.

Question 21: Explain EventEmitter Object in NodeJS.

Answer

To access the EventEmitter class, use,

require('events').EventEmitter

API methods are accessed through an instance of the EventEmitter class. For this you need to create an object of the EventEmitter class, then you can access the method. For example,

varobj=new EventEmitter();

Using that you can access the API method via an EventEmitter instance. Let's see how in the following:

obj.emitEvent();

In Node we do not have the DOM so we use the EventEmitter class. This class has mainly two methods, "on" and "emit". I have given the example and sample use of these methods in the NodeJS environment.

EventEmitter Class

The event module contains the EventEmitter class. The Util package provides the way to inherit from one class to another class. The "EventEmitter" class allows us to listen for events and assign actions to run when those events occur. The EventEmitter class is based on a publish/subscribe model because we can subscribe to events and then publish them.

When we use theEventEmitter class we need to require an events module in the program, for example,

varev=require("events");

The ev object will then have a single property, which is the EventEmitter class itself.

Objecs that emit events are instances of "events.EventEmitter." You can access them using,

require("events")

Functions can be attached to objects, to be executed when an event is emitted; these function are listeners. Inside a listener function, "this" refers to the "EventEmitter" that the listener was attached to.

Features of EventEmitter class,

This class is for managing events.

Can be extended to provide event functionality in other classes.

It emits events.

It listens to and handles those events.

Something that drives the two, causing events to be emitted.

EventEmitter Methods

emitobj.on(event, listener)

emitobj.emit(event,[arg1],[arg2],..[argN])

emitobj.once(event, listener)

emitobj.removeListener(event, listener)

Question 22: Explain Event-Driven Programming in Node.js.

Answer

Event-Driven Programming is the term where the flow of the code is determined by events (click, load and so on). It's one of the basic milestones of today's popular programming languages, such as C#, Java and many more; I won't dwell on all of them here. In Node.js and moreover in any kind of JavaScript project, you'll be using or have used event-driven processes. Whether it's a page on load or a button click event, this is something you have done, whether you knew it or not.

Question 23: What are Request and Cheerio in Node.js NPM?

Answer

Request and Cheerio are our npm packages. Cheerio doesn’t try to emulate a full implementation of the DOM. It specifically focuses on the scenario where you want to manipulate an HTML document using jQuery-like syntax. As such, it compares to jsdom favorably in some cases, but not in every situation.

Cheerio itself doesn’t include a mechanism for making HTTP requests, and that’s something that can be tedious to handle manually. It’s a bit easier to use a module called request to facilitate requesting remote HTML documents. Request handles common tasks like caching cookies between multiple requests, setting the content length on POSTs, and generally makes life easier.

Question 24: Explain Karma And Jasmine in Node.js.

Answer

Karma

Karma is a tool made on top of NodeJS to run JavaScript test cases. This is not a testing framework like Jasmine or Mocha or Chai etc. It only allows us to run JavaScript test cases written using testing frameworks like Jasmine.

Karma runs JavaScript test cases against real browsers through Command Line Interface (CLI) rather than virtual browser and DOM. Since DOM rendering across browsers vary, for correctness of the test report it uses real browsers. It can configure what browsers need to be used while running Karma.

Jasmine

Jasmine is a BDD (Behavior Driven Development) Javascript testing framework which provides building blocks to write JavaScript unit test cases. It does not require any other JavaScript Framework. As stated in official documentation of Jasmine 2.0.0,

“Jasmine is a behavior-driven development framework for testing JavaScript code. It does not depend on any other JavaScript frameworks. It does not require a DOM. And it has a clean, obvious syntax so that you can easily write tests.”

Here is a code snippet which describes the basic structure of Jasmine Unit Test Spec,

describe("simple test", function()

{

beforeEach(function(){

});

afterEach(function(){

});

it("a is a string", function(){

})

})

Question 26: Describe the Node.js processing model.

Answer

Node begins by a limited thread pool to receive the client requests. When a request arrives, it’s placed within a stack called “Event Queue”. The thing responsible to catch the request, to process and return it for client is called “Event Loop” that is the heart of Node and it works like a a single-thread. However, if there is a process I/O, the Event Loop stops executing and calls an internal thread pool, done in C++, to execute the process and return for Event Loop when finished. But the great difference that we have noted is that the Event Loop does not wait for the return of Internal Thread Pool, it goes to catch other requests. With this concept Node.js is faster than current web servers, such as IIS or Apache.

Question 31: How can you create HTTPS Server WithNodeJS?

Answer

We use HTTPS when we need secure sessions. A secure session implies that the web browser will encrypt everything you do with a digitally signed certificate. So obviously before you do HTTPS, you need to create certificates. Let's therefore spend some time to see how to create certificates for SSL.

Creating HTTPS server

Just like we do for HTTP, an import to node's HTTP module, for HTTPS we import the HTTPS module. Also, since we need to pass in the certificate and key file, we also need to import the filestream (fs) module to enable Node to read the files. The following is the code to create the HTTPS server.

var https = require('https');

var fs = require('fs');

var options = {

key: fs.readFileSync('hostkey.pem'),

cert: fs.readFileSync('hostcert.pem')

};

https.createServer(options, function (req, res) {

res.writeHead(200);

res.end("hello world\n");

}).listen(8000);

Question 38: What is middleware?

Middleware are just simple functions with three arguments; request, response and next, they act as a middleman; when express receives a request, express passes this request to each middleware in the chain until it receives a response. In general, there are two kinds of middleware, first one being a normal middleware, and the second one an error handling middleware.

Middleware concept is at the core of express.js and very important to understand, almost everything in express.js is a middleware, from body parsers to loggers, from routes to error handlers.

Question 48: What are the key points of Node.js?

Answer

These are the key points about Node.js,

Node.js is nothing but server-side JavaScript.

Single-threaded environment.

Asynchronous event driven server-side JavaScript.

Runs on Google JS ver-8 Engine.

Non blocking I/O and File API.

Highly scalable.

Nodes apps are created.

Achieving Scale and Concurrency

Written in C and C++

A file in node.js is called a Module.

High performance.

Q12) What is error-first callback in Node JS?

In order to check for proper working of the code, we need to verify error free execution. In this regard, error-first callbacks are used, that will send error first, followed by related data to the error.

Q13) What is cluster?

Cluster is a process to handle thread execution load while working with multi-core systems.

Q14) Why node is single threaded?

In order to upgrade the performance well, single thread operations work better than multi-threaded operations.

Q15) How does URL module works.

This module helps URL to parse it into host, pathname, search, query, etc.

Example:

var url = require('url');

var adr = 'https://mindmajix.com/?s=node+js+training';

var q = url.parse(adr, true);

console.log(q.host); //returns 'mindmajix.com'

console.log(q.search); //returns '?s=node+js+training'

Q-10. What Is Callback In Node.Js?

Answer.

We may call “callback” as an asynchronous equivalent for a function. Node.js makes heavy use of callbacks and triggers it at the completion of a given task. All the APIs of Node.js are written in such a way that they support callbacks.

Q-11. What Is Callback Hell?

Answer.

Initially, you may praise Callback after learning about it. Callback hell is heavily nested callbacks which make the code unreadable and difficult to maintain.

Q-12. How To Avoid Callback Hell In Node.Js?

Answer.

Node.js internally uses a single-threaded event loop to process queued events. But this approach may lead to blocking the entire process if there is a task running longer than expected.

Node.js addresses this problem by incorporating callbacks also known as higher-order functions. So whenever a long-running process finishes its execution, it triggers the callback associated. With this approach, it can allow the code execution to continue past the long-running task.

However, the above solution looks extremely promising. But sometimes, it could lead to complex and unreadable code. More the no. of callbacks, longer the chain of returning callbacks would be. Just see the below example.

With such an unprecedented complexity, it’s hard to debug the code and can cause you a whole lot of time. There are four solutions which can address the callback hell problem.

1. Make Your Program Modular.

It proposes to split the logic into smaller modules. And then join them together from the main module to achieve the desired result.

2. Use Async Mechanism.

It is a widely used Node.js module which provides a sequential flow of execution.

The async module has <async.waterfall> API which passes data from one operation to other using the next callback.

Another async API <async.map> allows iterating over a list of items in parallel and calls back with another list of results.

With the async approach, the caller’s callback gets called only once. The caller here is the main method using the async module.

3. Use Promises Mechanism.

Promises give an alternate way to write async code. They either return the result of execution or the error/exception. Implementing promises requires the use of <.then()> function which waits for the promise object to return. It takes two optional arguments, both functions. Depending on the state of the promise only one of them will get called. The first function call proceeds if the promise gets fulfilled. However, if the promise gets rejected, then the second function will get called.

4. Use Generators.

Generators are lightweight routines, they make a function wait and resume via the yield keyword. Generator functions uses a special syntax <function\* ()>. They can also suspend and resume asynchronous operations using constructs such as promises or <thunks> and turn a synchronous code into asynchronous.

Q-15. What Are Globals In Node.Js?

Answer.

There are three keywords in Node.js which constitute as Globals. These are Global, Process, and Buffer.

Global.

The Global keyword represents the global namespace object. It acts as a container for all other <global> objects. If we type <console.log(global)>, it’ll print out all of them.

An important point to note about the global objects is that not all of them are in the global scope, some of them fall in the module scope. So, it’s wise to declare them without using the var keyword or add them to Global object.

Variables declared using the var keyword become local to the module whereas those declared without it get subscribed to the global object.

Process.

It is also one of the global objects but includes additional functionality to turn a synchronous function into an async callback. There is no boundation to access it from anywhere in the code. It is the instance of the EventEmitter class. And each node application object is an instance of the Process object.

It primarily gives back the information about the application or the environment.

<process.execPath> – to get the execution path of the Node app.

<process.Version> – to get the Node version currently running.

<process.platform> – to get the server platform.

Some of the other useful Process methods are as follows.

<process.memoryUsage> – To know the memory used by Node application.

<process.NextTick> – To attach a callback function that will get called during the next loop. It can cause a delay in executing a function.

Buffer.

The Buffer is a class in Node.js to handle binary data. It is similar to a list of integers but stores as a raw memory outside the V8 heap.

We can convert JavaScript string objects into Buffers. But it requires mentioning the encoding type explicitly.

<ascii> – Specifies 7-bit ASCII data.

<utf8> – Represents multibyte encoded Unicode char set.

<utf16le> – Indicates 2 or 4 bytes, little endian encoded Unicode chars.

<base64> – Used for Base64 string encoding.

<hex> – Encodes each byte as two hexadecimal chars.

Q-18. How Many Types Of Streams Are Present In Node.Js?

Answer.

Stream in Node.js are objects that allow reading data from a source or writing data to a specific destination in a continuous fashion. In Node.js, there are four types of streams.

<Readable> – This is the Stream to be used for reading operation.

<Writable> – It facilitates the write operation.

<Duplex> – This Stream can be used for both the read and write operations.

<Transform> – It is a form of a duplex Stream, which performs the computations based on the available input.

All the Streams, discussed above are an instance of an “EventEmitter” class. The event thrown by the Stream varies with time. Some of the commonly used events are as follows.

<data> – This event gets fired when there is data available for reading.

<end> – The Stream fires this event when there is no more data to read.

<error> – This event gets fired when there is any error in reading or writing data.

<finish> – It fires this event after it has flushed all the data to the underlying system.

Q-19. List And Explain The Important REPL Commands?

Answer.

Following is the list of some of the most commonly used REPL commands.

<.help> – It displays help for all the commands.

<tab Keys> – It displays the list of all the available commands.

<Up/Down Keys> – Its use is to determine what command was executed in REPL previously.

<.save filename> – Save the current REPL session to a file.

<.load filename> – To Load the specified file in the current REPL session.

<ctrl + c> – used to Terminate the current command.

<ctrl + c (twice)> – To Exit from the REPL.

<ctrl + d> – This command perfoms Exit from the REPL.

<.break> – It leads Exitting from multiline expression.

<.clear> – Exit from multiline expression.

Q-26. What Is Chaining Process In Node.Js?

Answer.

It’s an approach to connect the output of one stream to the input of another stream, thus creating a chain of multiple stream operations.