Node.js - Interview Questions

Dear readers, these **Node.JS Interview Questions** have been designed specially to get you acquainted with the nature of questions you may encounter during your interview for the subject of **Node.JS**. As per my experience good interviewers hardly plan to ask any particular question during your interview, normally questions start with some basic concept of the subject and later they continue based on further discussion and what you answer:

What is Node.js?

Node.js is a web application framework built on Google Chrome's JavaScript Engine(V8 Engine).

Node.js comes with runtime environment on which a Javascript based script can be interpreted and executed (It is analogus to JVM to JAVA byte code). This runtime allows to execute a JavaScript code on any machine outside a browser. Because of this runtime of Node.js, JavaScript is now can be executed on server as well.

Node.js also provides a rich library of various javascript modules which eases the developement of web application using Node.js to great extents.

Node.js = Runtime Environment + JavaScript Library

What do you mean by Asynchronous API?

All APIs of Node.js library are aynchronous that is non-blocking. It essentially means a Node.js based server never waits for a API to return data. Server moves to next API after calling it and a notification mechanism of Events of Node.js helps server to get response from the previous API call.

What are the benefits of using Node.js?

Following are main benefits of using Node.js

- **Asynchronous and Event Driven**All APIs of Node.js library are asynchronous that is non-blocking. It essentially means a Node.js based server never waits for a API to return data. Server moves to next API after calling it and a notification mechanism of Events of Node.js helps server to get response from the previous API call.
- **Very Fast** Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but highly Scalable** − Node.js uses a single threaded model with event looping. Event mechanism helps server to respond in a non-bloking ways and makes server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and same program

can services much larger number of requests than traditional server like Apache HTTP Server.

- **No Buffering** − Node.js applications never buffer any data. These applications simply output the data in chunks.

Is it free to use Node.js?

Yes! Node.js is released under the MIT license and is free to use.

Is Node a single threaded application?

Yes! Node uses a single threaded model with event looping.

What is REPL in context of Node?

REPL stands for Read Eval Print Loop and it represents a computer environment like a window console or unix/linux shell where a command is entered and system responds with an output. Node.js or Node comes bundled with a REPL environment. It performs the following desired tasks.

- **Read** − Reads user's input, parse the input into JavaScript data-structure and stores in memory.
- **Eval** − Takes and evaluates the data structure
- **Print** − Prints the result
- **Loop** − Loops the above command until user press ctrl-c twice.

Can we evaluate simple expression using Node REPL

Yes.

What is the difference of using var and not using var in REPL while dealing with variables?

Use variables to store values and print later. if var keyword is not used then value is stored in the variable and printed. Whereas if var keyword is used then value is stored but not printed. You can use both variables later.

What is the use of Underscore variable in REPL?

Use _ to get the last result.

```
C:\Nodejs_WorkSpace>node
> var x = 10
undefined
```

```
> var y = 20
undefined
> x + y
30
> var sum = _
undefined
> console.log(sum)
30
undefined
>
```

What is npm?

npm stands for Node Package Manager. npm provides following two main functionalities:

- Online repositories for node.js packages/modules which are searchable on search.nodejs.org
- Command line utility to install packages, do version management and dependency management of Node.js packages.

What is global installation of dependencies?

Globally installed packages/dependencies are stored in **<user-directory>**/npm directory. Such dependencies can be used in CLI (Command Line Interface) function of any node.js but can not be imported using require() in Node application directly. To install a Node project globally use -g flag.

```
C:\Nodejs_WorkSpace>npm install express -g
```

What is local installation of dependencies?

By default, npm installs any dependency in the local mode. Here local mode refers to the package installation in node_modules directory lying in the folder where Node application is present. Locally deployed packages are accessible via require(). To install a Node project locally following is the syntax.

```
C:\Nodejs_WorkSpace>npm install express
```

How to check the already installed dependencies which are globally installed using npm?

Use the following command −

```
C:\Nodejs_WorkSpace>npm ls -g
```

What is Package.json?

package.json is present in the root directory of any Node application/module and is used to define the properties of a package.

Name some of the attributes of package.json?

Following are the attributes of Package.json

- **name** − name of the package
- **version** − version of the package
- **description** − description of the package
- **homepage** − homepage of the package
- **author** − author of the package
- **contributors** − name of the contributors to the package
- **dependencies** − list of dependencies. npm automatically installs all the dependencies mentioned here in the node_module folder of the package.
- **repository** − repository type and url of the package
- **main** − entry point of the package
- **keywords** − keywords

How to uninstall a dependency using npm?

Use following command to uninstall a module.

```
C:\Nodejs_WorkSpace>npm uninstall dependency-name
```

How to update a dependency using npm?

Update package.json and change the version of the dependency which to be updated and run the following command.

```
C:\Nodejs_WorkSpace>npm update
```

What is Callback?

Callback is an asynchronous equivalent for a function. A callback function is called at the completion of a given task. Node makes heavy use of callbacks. All APIs of Node are written is such a way that they supports callbacks. For example, a function to read a file may start reading file and return the control to execution environment immidiately so that next instruction can be

executed. Once file I/O is complete, it will call the callback function while passing the callback function, the content of the file as parameter. So there is no blocking or wait for File I/O. This makes Node.js highly scalable, as it can process high number of request without waiting for any function to return result.

What is a blocking code?

If application has to wait for some I/O operation in order to complete its execution any further then the code responsible for waiting is known as blocking code.

How Node prevents blocking code?

By providing callback function. Callback function gets called whenever corresponding event triggered.

What is Event Loop?

Node js is a single threaded application but it support concurrency via concept of event and callbacks. As every API of Node js are asynchronous and being a single thread, it uses async function calls to maintain the concurrency. Node uses observer pattern. Node thread keeps an event loop and whenever any task get completed, it fires the corresponding event which signals the event listener function to get executed.

What is Event Emmitter?

EventEmitter class lies in **events** module. It is accessibly via following syntax −

```
//import events module
var events = require('events');

//create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

When an EventEmitter instance faces any error, it emits an 'error' event. When new listener is added, 'newListener' event is fired and when a listener is removed, 'removeListener' event is fired.

EventEmitter provides multiple properties like **on** and **emit**. **on** property is used to bind a function with the event and **emit** is used to fire an event.

What is purpose of Buffer class in Node?

Buffer class is a global class and can be accessed in application without importing buffer module. A Buffer is a kind of an array of integers and corresponds to a raw memory allocation outside the V8 heap. A Buffer cannot be resized.

What is Piping in Node?

Piping is a mechanism to connect output of one stream to another stream. It is normally used to get data from one stream and to pass output of that stream to another stream. There is no limit on piping operations. Consider the above example, where we've read test.txt using readerStream and write test1.txt using writerStream. Now we'll use the piping to simplify our operation or reading from one file and writing to another file.

Which module is used for file based operations?

fs module is used for file based operations.

```
var fs = require("fs")
```

Which module is used for buffer based operations?

buffer module is used for buffer based operations.

```
var buffer = require("buffer")
```

Which module is used for web based operations?

http module is used for web based operations.

```
var http = require("http")
```

fs module provides both synchronous as well as asynchronous methods.

true.

What is difference between synchronous and asynchronous method of fs module?

Every method in fs module have synchronous as well as asynchronous form. Asynchronous methods takes a last parameter as completion function callback and first parameter of the callback function is error. It is preferred to use asynchronous method instead of synchronous method as former never block the program execution where the latter one does.

Name some of the flags used in read/write operation on files.

flags for read/write operations are following:

- **r** − Open file for reading. An exception occurs if the file does not exist.
- **r+** − Open file for reading and writing. An exception occurs if the file does not exist.

- **rs** − Open file for reading in synchronous mode. Instructs the operating system to bypass the local file system cache. This is primarily useful for opening files on NFS mounts as it allows you to skip the potentially stale local cache. It has a very real impact on I/O performance so don't use this flag unless you need it. Note that this doesn't turn fs.open() into a synchronous blocking call. If that's what you want then you should be using fs.openSync()
- **rs+** − Open file for reading and writing, telling the OS to open it synchronously. See notes for 'rs' about using this with caution.
- **w** − Open file for writing. The file is created (if it does not exist) or truncated (if it exists).
- **wx** − Like 'w' but fails if path exists.
- **w+** − Open file for reading and writing. The file is created (if it does not exist) or truncated (if it exists).
- **wx+** − Like 'w+' but fails if path exists.
- **a** − Open file for appending. The file is created if it does not exist.
- **ax** − Like 'a' but fails if path exists.
- **a+** − Open file for reading and appending. The file is created if it does not exist.
- **ax+'** − Like 'a+' but fails if path exists.

What are streams?

Streams are objects that let you read data from a source or write data to a destination in continuous fashion.

How many types of streams are present in Node.

In Node.js, there are four types of streams.

- **Readable** − Stream which is used for read operation.
- **Writable** − Stream which is used for write operation.
- **Duplex** − Stream which can be used for both read and write operation.
- **Transform** − A type of duplex stream where the output is computed based on input.

Name some of the events fired by streams.

Each type of Stream is an **EventEmitter** instance and throws several events at different instance of times. For example, some of the commonly used events are:

- **data** − This event is fired when there is data is available to read.
- **end** − This event is fired when there is no more data to read.
- **error** − This event is fired when there is any error receiving or writing data.
- **finish** − This event is fired when all data has been flushed to underlying system

What is Chaining in Node?

Chanining is a mechanism to connect output of one stream to another stream and create a chain of multiple stream operations. It is normally used with piping operations.

How will you open a file using Node?

Following is the syntax of the method to open a file in asynchronous mode:

```
fs.open(path, flags[, mode], callback)
```

Parameters

Here is the description of the parameters used:

- **path** − This is string having file name including path.
- **flags** − Flag tells the behavior of the file to be opened. All possible values have been mentioned below.
- **mode** − This sets the file mode (permission and sticky bits), but only if the file was created. It defaults to 0666, readable and writeable.
- **callback** − This is the callback function which gets two arguments (err, fd).

How will you read a file using Node?

Following is the syntax of one of the methods to read from a file:

```
fs.read(fd, buffer, offset, length, position, callback)
```

This method will use file descriptor to read the file, if you want to read file using file name directly then you should use another method available.

Parameters

Here is the description of the parameters used −

- **fd** − This is the file descriptor returned by file fs.open() method.
- **buffer** − This is the buffer that the data will be written to.
- **offset** − This is the offset in the buffer to start writing at.
- **length** − This is an integer specifying the number of bytes to read.
- **position** − This is an integer specifying where to begin reading from in the file. If position is null, data will be read from the current file position.
- **callback** − This is the callback function which gets the three arguments, (err, bytesRead, buffer).

How will you write a file using Node?

Following is the syntax of one of the methods to write into a file:

```
fs.writeFile(filename, data[, options], callback)
```

This method will over-write the file if file already exists. If you want to write into an existing file then you should use another method available.

Parameters

Here is the description of the parameters used:

- **path** − This is string having file name including path.
- **data** − This is the String or Buffer to be written into the file.
- **options** − The third parameter is an object which will hold {encoding, mode, flag}. By default encoding is utf8, mode is octal value 0666 and flag is 'w'
- **callback** − This is the callback function which gets a single parameter err and used to to return error in case of any writing error.

How will you close a file using Node?

Following is the syntax of one of the methods to close an opened file:

```
fs.close(fd, callback)
```

Parameters

Here is the description of the parameters used:

- **fd** − This is the file descriptor returned by file fs.open() method.
- **callback** − This is the callback function which gets no arguments other than a possible exception are given to the completion callback.

How will you get information about a file using Node?

Following is the syntax of the method to get the information about a file:

```
fs.stat(path, callback)
```

Parameters

Here is the description of the parameters used:

- **path** − This is string having file name including path.

- **callback** − This is the callback function which gets two arguments (err, stats) where **stats** is an object of fs.Stats type which is printed below in the example.

How will you truncate a file using Node?

Following is the syntax of the method to truncate an opened file −

```
fs.ftruncate(fd, len, callback)
```

Parameters

Here is the description of the parameters used:

- **fd** − This is the file descriptor returned by file fs.open() method.
- **len** − This is the length of the file after which file will be truncated.
- **callback** − This is the callback function which gets no arguments other than a possible exception are given to the completion callback.

How will you delete a file using Node?

Following is the syntax of the method to delete a file −

```
fs.unlink(path, callback)
```

Parameters

Here is the description of the parameters used:

- **path** − This is the file name including path.
- **callback** − This is the callback function which gets no arguments other than a possible exception are given to the completion callback.

How will you create a directory?

Following is the syntax of the method to create a directory:

```
fs.mkdir(path[, mode], callback)
```

Parameters

Here is the description of the parameters used:

- **path** − This is the directory name including path.
- **mode** − This is the directory permission to be set. Defaults to 0777.
- **callback** − This is the callback function which gets no arguments other than a possible exception are given to the completion callback.

How will you delete a directory?

Following is the syntax of the method to remove a directory:

```
fs.rmdir(path, callback)
```

Parameters

Here is the description of the parameters used:

- **path** − This is the directory name including path.
- **callback** − This is the callback function which gets no arguments other than a possible exception are given to the completion callback.

How will you read a directory?

Following is the syntax of the method to read a directory:

```
fs.readdir(path, callback)
```

Parameters

Here is the description of the parameters used:

- **path** − This is the directory name including path.
- **callback** − This is the callback function which gets two arguments (err, files) where files is an array of the names of the files in the directory excluding '.' and '..'.

What is the purpose of __filename variable?

The __filename represents the filename of the code being executed. This is the resolved absolute path of this code file. For a main program this is not necessarily the same filename used in the command line. The value inside a module is the path to that module file.

What is the purpose of __dirname variable?

The __dirname represents the name of the directory that the currently executing script resides in.

What is the purpose of setTimeout function?

The setTimeout(cb, ms) global function is used to run callback cb after at least ms milliseconds. The actual delay depends on external factors like OS timer granularity and system load. A timer cannot span more than 24.8 days.

This function returns an opaque value that represents the timer which can be used to clear the timer.

What is the purpose of clearTimeout function?

The clearTimeout(t) global function is used to stop a timer that was previously created with setTimeout(). Here t is the timer returned by setTimeout() function.

What is the purpose of setInterval function?

The setInterval(cb, ms) global function is used to run callback cb repeatedly after at least ms milliseconds. The actual delay depends on external factors like OS timer granularity and system load. A timer cannot span more than 24.8 days.

This function returns an opaque value that represents the timer which can be used to clear the timer using the function clearInterval(t).

What is the purpose of console object?

console object is used to Used to print information on stdout and stderr.

What is the purpose of process object?

process object is used to get information on current process. Provides multiple events related to process activities.

What is Next ?

Further you can go through your past assignments you have done with the subject and make sure you are able to speak confidently on them. If you are fresher then interviewer does not expect you will answer very complex questions, rather you have to make your basics concepts very strong.

Second it really doesn't matter much if you could not answer few questions but it matters that whatever you answered, you must have answered with confidence. So just feel confident during your interview. We at tutorialspoint wish you best luck to have a good interviewer and all the very best for your future endeavor. Cheers :-)